Yandex.Maps API Reference

Version 2.1.63

13.04.2018

Yandex

Yandex.Maps API Reference. Version 2.1.63. Version
Document build date: 13.04.2018.
This volume is a part of Yandex technical documentation.
Yandex helpdesk site: http://help.yandex.ru
© 2008—2018 Yandex LLC. All rights reserved.

Copyright Disclaimer

Yandex (and its applicable licensor) has exclusive rights for all results of intellectual activity and equated to them means of individualization, used for development, support, and usage of the service Yandex.Maps API Reference. It may include, but not limited to, computer programs (software), databases, images, texts, other works and inventions, utility models, trademarks, service marks, and commercial denominations. The copyright is protected under provision of Part 4 of the Russian Civil Code and international laws.
You may use Yandex.Maps API Reference or its components only within credentials granted by the Terms of Use of Yandex.Maps API Reference or within an appropriate Agreement.
Any infringements of exclusive rights of the copyright owner are punishable under civil, administrative or criminal Russian laws.

Contact information

Yandex LLC
http://www.yandex.com
Phone: +7 495 739 7000
Email: pr@yandex-team.ru
Headquarters: 16 L'va Tolstogo St., Moscow, Russia 119021

Contents

Yandex.Maps API Reference ... 12
Balloon ... 12

behavior ... 17
- behavior.DblClickZoom ... 17
- behavior.Drag ... 18
- behavior.LeftMouseButtonMagnifier ... 20
- behavior.MultiTouch ... 22
- behavior.RightMouseButtonMagnifier ... 23
- behavior.RouteEditor ... 24
- behavior.Ruler ... 27
- behavior.ScrollZoom ... 29
- behavior.storage ... 31

borders ... 32
- borders.load ... 32

Circle ... 34

clusterer ... 44

- **clusterer.addon** ... 44
  - clusterer.addon.balloon ... 44
  - clusterer.addon.hint ... 44
- clusterer.Balloon ... 46
- clusterer.Hint ... 48

Clusterer ... 51
ClusterPlacemark ... 59

collection ... 71
- collection.Item ... 71

Collection ... 74

control ... 79
- control.Button ... 79
- control.FullscreenControl ... 85
- control.GeolocationControl ... 90
- control.ListBox ... 95
- control.ListBoxItem ... 103
- control.Manager ... 107
- control.RouteButton ... 113
- control.RouteEditor ... 117
- control.RoutePanel ... 121
- control.RulerControl ... 124
- control.SearchControl ... 127
- control.storage ... 138
- control.TrafficControl ... 139
- control.TypeSelector ... 145
- control.ZoomControl ... 151

coordSystem ... 155
- coordSystem.cartesian ... 155
- coordSystem.geo ... 155

data ... 157
- data.Manager ... 157

DomEvent ... 161

domEvent ... 163
- domEvent.manager ... 163
- domEvent.MultiPointer ... 166

domEvent.MultiTouch ... 167
domEvent.Pointer ... 168
domEvent.Touch ... 168

event ... 170
event.Group ... 170
event.Manager ... 171
event.Mapper ... 173

Event ... 175
formatter ... 178
geocode ... 179
GeocodeResult ... 181
geolocation ... 186

geometry ... 188

geometry.base ... 188
geometry.base.Circle ... 188

geometry.base.LinearRing ... 190
geometry.base.LinearRing.fromEncodedCoordinates ... 193
geometry.base.LinearRing.toEncodedCoordinates ... 193

geometry.base.LineString ... 194
geometry.base.LineString.fromEncodedCoordinates ... 196
geometry.base.LineString.toEncodedCoordinates ... 196

geometry.base.Point ... 198

geometry.base.Polygon ... 199
geometry.base.Polygon.fromEncodedCoordinates ... 202
geometry.base.Polygon.toEncodedCoordinates ... 203

geometry.base.Rectangle ... 204

geometry.Circle ... 206

geometry.json ... 209
geometry.json.circle ... 209
geometry.json.lineString ... 209
geometry.json.Point ... 210
geometry.json.polygon ... 211
geometry.json.rectangle ... 212

geometry.LineString ... 213
geometry.LineString.fromEncodedCoordinates ... 217
geometry.LineString.toEncodedCoordinates ... 217

geometry.pixel ... 218
geometry.pixel.Circle ... 218
geometry.pixel.LineString ... 219
geometry.pixel.MultiLineString ... 220
geometry.pixel.MultiPolygon ... 222
geometry.pixel.Point ... 224
geometry.pixel.Polygon ... 226
geometry.pixel.Rectangle ... 228

geometry.Point ... 231

geometry.Polygon ... 233
geometry.Polygon.fromEncodedCoordinates ... 238
geometry.Polygon.toEncodedCoordinates ... 238

geometry.Rectangle ... 240

geometryEditor ... 244
geometryEditor.Circle ... 244

geometryEditor.LineString ... 245

geometryEditor.model ... 258
geometryEditor.model.ChildLinearRing ... 258
geometryEditor.model.ChildLineString ... 260
geometryEditor.model.ChildVertex ... 262
geometryEditor.model.Edge ... 265
geometryEditor.model.EdgeGeometry ... 266
geometryEditor.model.RootLineString ... 267
geometryEditor.model.RootPolygon ... 269

geometryEditor.Point ... 271
geometryEditor.Polygon ... 273

geometryEditor.view ... 286
geometryEditor.view.Edge ... 286
geometryEditor.view.MultiPath ... 286
geometryEditor.view.Path ... 287
geometryEditor.view.Vertex ... 288

geoObject ... 289

geoObject.addon ... 289
geoObject.addon.balloon ... 289
geoObject.addon.editor ... 290
geoObject.addon.hint ... 291

geoObject.Balloon ... 292
geoObject.Hint ... 294
geoObject.Sequence ... 297

GeoObject ... 302
GeoObjectCollection ... 317
geoQuery ... 325
GeoQueryResult ... 328

geoXml ... 363
geoXml.load ... 363

getZoomRange ... 365

graphics ... 366

graphics.style ... 366
graphics.style.color ... 366
graphics.style.stroke ... 366

Hint ... 369
Hotspot ... 371

hotspot ... 374

hotspot.layer ... 374

hotspot.layer.addon ... 374
hotspot.layer.addon.balloon ... 374
hotspot.layer.addon.hint ... 375

hotspot.layer.Balloon ... 376
hotspot.layer.Hint ... 378
hotspot.layer.Object ... 380

hotspot.Layer ... 384
hotspot.ObjectSource ... 395

interactivityModel ... 404
interactivityModel.storage ... 404

Interfaces ... 405

IBalloon ... 405
IBalloonLayout ... 406
IBalloonManager ... 409
IBalloonOwner ... 412
IBaseCircleGeometry ... 412
IBaseGeometry ... 414
IBaseLinearRingGeometry ... 415
IBaseLineStringGeometry ... 417
IBasePointGeometry ... 419
IBasePolygonGeometry ... 420
IBaseRectangleGeometry ... 423
IBehavior ... 424
ICanvasTile ... 426
IChild ... 428
IChildOnMap ... 429
ICircleGeometry ... 430
ICircleGeometryAccess ... 432
ICollection ... 435
IContainerPane ... 437
IControl ... 439
IControlParent ... 441
ICoordSystem ... 442
ICopyrightsAccessor ... 446
ICopyrightsProvider ... 447
ICustomizable ... 448
IDataManager ... 449
IDomEvent ... 450
IDomEventEmitter ... 452
IDomTile ... 455
IEvent ... 457
IEventController ... 459
IEventEmitter ... 460
IEventGroup ... 461
IEventManager ... 462
IEventPane ... 465
IEventTrigger ... 468
IEventWorkflowController ... 469
IExpandableControlLayout ... 470
IFreezable ... 473
IGeocodeProvider ... 474
IGeometry ... 476
IGeometryEditor ... 479
IGeometryEditorChildModel ... 480
IGeometryEditorModel ... 482
IGeometryEditorRootModel ... 483
IGeometryJson ... 484
IGeoObject ... 484
IGeoObjectCollection ... 488
IGeoObjectPopupData ... 493
IGeoObjectSequence ... 494
IGroupControlLayout ... 497
IHint ... 500
IHintManager ... 501
IHintOwner ... 503
IHotspot ... 503
IHotspotLayerObject ... 506
IHotspotObjectSource ... 510
IHotspotShape ... 511
IIterator ... 515
ILayer ... 516
ILayout ... 519
ILinearRingGeometryAccess ... 522

ILineStringGeometry ... 527
ILineStringGeometryAccess ... 530
IMapAction ... 535
IMapObjectCollection ... 536
IMapState ... 537
IMultiRouteModelJson ... 538
IMultiRouteParams ... 538
IMultiRouteReferencePoint ... 540
IMultiRouterRouteBalloon ... 541
IOptionManager ... 542
IOverlay ... 545
IPane ... 550
IPanorama ... 552
IPanoramaConnection ... 554
IPanoramaConnectionArrow ... 555
IPanoramaConnectionMarker ... 556
IPanoramaGraph ... 556
IPanoramaGraphEdge ... 557
IPanoramaGraphNode ... 558
IPanoramaMarker ... 558
IPanoramaMarkerIcon ... 559
IPanoramaMarkerIconSet ... 560
IPanoramaTileLevel ... 561
IParentOnMap ... 562
IPixelCircleGeometry ... 563
IPixelGeometry ... 564
IPixelLineStringGeometry ... 566
IPixelMultiLineGeometry ... 567
IPixelMultiPolygonGeometry ... 569
IPixelPointGeometry ... 572
IPixelPolygonGeometry ... 573
IPixelRectangleGeometry ... 576
IPointGeometry ... 578
IPointGeometryAccess ... 579
IPolygonGeometry ... 580
IPolygonGeometryAccess ... 583
IPopup ... 589
IPopupManager ... 592
IPositioningContext ... 596
IProjection ... 597
IPromiseProvider ... 598
IRatioMap ... 599
IRectangleGeometry ... 600
IRectangleGeometryAccess ... 601
IRoutePanel ... 604
ISearchControlLayout ... 607
ISearchProvider ... 610
ISelectableControl ... 612
ISelectableControlLayout ... 614
IShape ... 617
ISuggestProvider ... 619
ISuggestViewLayout ... 620
ITrafficControlLayout ... 621
ITrafficProvider ... 624
ITransportProperties ... 625
IZoomControlLayout ... 626

layer ... 630
layer.storage ... 630

layer.tile ... 630
layer.tile.CanvasTile ... 630
layer.tile.DomTile ... 631

layer.tileContainer ... 634
layer.tileContainer.CanvasContainer ... 634
layer.tileContainer.DomContainer ... 636

Layer ... 639
LayerCollection ... 645

layout ... 650
layout.Image ... 650
layout.ImageWithContent ... 653
layout.PieChart ... 656
layout.storage ... 661

layout.templateBased ... 662
layout.templateBased.Base ... 662

LoadingObjectManager ... 667
Map ... 675

map ... 697

map.action ... 697
map.action.Continuous ... 697
map.action.Manager ... 699
map.action.Single ... 703

map.addon ... 706
map.addon.balloon ... 706
map.addon.hint ... 706

map.Balloon ... 708

map.behavior ... 710
map.behavior.Manager ... 710

map.Container ... 715
map.Converter ... 719
map.Copyrights ... 720
map.GeoObjects ... 723
map.Hint ... 729

map.layer ... 730
map.layer.Manager ... 730

map.margin ... 734
map.margin.Accessor ... 734
map.margin.Manager ... 736

map.pane ... 740
map.pane.Manager ... 740

map.ZoomRange ... 744

MapEvent ... 746
MapType ... 747

mapType ... 750
mapType.storage ... 750

meta ... 751

modules ... 753
modules.define ... 753
modules.isDefined ... 755
modules.require ... 756

Monitor ... 758

multiRouter ... 761

multiRouter.driving ... 761
multiRouter.driving.Path ... 761
multiRouter.driving.PathModel ... 765
multiRouter.driving.Route ... 768
multiRouter.driving.RouteModel ... 772
multiRouter.driving.Segment ... 775
multiRouter.driving.SegmentModel ... 779

multiRouter.Editor ... 783
multiRouter.EditorAddon ... 792

multiRouter.masstransit ... 801
multiRouter.masstransit.Path ... 801
multiRouter.masstransit.PathModel ... 805
multiRouter.masstransit.Route ... 808
multiRouter.masstransit.RouteModel ... 812
multiRouter.masstransit.StopModel ... 814
multiRouter.masstransit.TransferSegment ... 816
multiRouter.masstransit.TransferSegmentModel ... 820
multiRouter.masstransit.TransportSegment ... 822
multiRouter.masstransit.TransportSegmentModel ... 827
multiRouter.masstransit.WalkSegment ... 829
multiRouter.masstransit.WalkSegmentModel ... 833

multiRouter.MultiRoute ... 836
multiRouter.MultiRouteModel ... 844

multiRouter.pedestrian ... 850
multiRouter.pedestrian.Path ... 850
multiRouter.pedestrian.PathModel ... 854
multiRouter.pedestrian.Route ... 857
multiRouter.pedestrian.RouteModel ... 861
multiRouter.pedestrian.SegmentModel ... 863
multiRouter.pedestrian.WalkSegment ... 865

multiRouter.ViaPoint ... 869
multiRouter.ViaPointModel ... 872
multiRouter.WayPoint ... 875
multiRouter.WayPointModel ... 879

ObjectManager ... 883

objectManager ... 895

objectManager.addon ... 895
objectManager.addon.clustersBalloon ... 895
objectManager.addon.clustersHint ... 895
objectManager.addon.objectsBalloon ... 896
objectManager.addon.objectsHint ... 897

objectManager.Balloon ... 898
objectManager.ClusterCollection ... 902
objectManager.Hint ... 908
objectManager.ObjectCollection ... 911
objectManager.OverlayCollection ... 918

option ... 930
option.Manager ... 930
option.presetStorage ... 934

overlay ... 946
overlay.Circle ... 946

overlay.hotspot ... 950
overlay.hotspot.Circle ... 950
overlay.hotspot.Placemark ... 954

overlay.hotspot.Polygon ... 958
overlay.hotspot.Polyline ... 963
overlay.hotspot.Rectangle ... 966

overlay.html ... 972
overlay.html.Balloon ... 972
overlay.html.Hint ... 977
overlay.html.Placemark ... 981
overlay.html.Rectangle ... 987

overlay.Pin ... 993
overlay.Placemark ... 997
overlay.Polygon ... 1002
overlay.Polyline ... 1007
overlay.Rectangle ... 1010
overlay.storage ... 1015

pane ... 1017
pane.EventsPane ... 1017
pane.MovablePane ... 1020
pane.StaticPane ... 1022

panorama ... 1025

panorama.Base ... 1025
panorama.Base.createPanorama ... 1028
panorama.Base.getMarkerPositionFromDirection ... 1030

panorama.createPlayer ... 1031
panorama.isSupported ... 1032
panorama.locate ... 1032
panorama.Manager ... 1032
panorama.Player ... 1035

Panorama ... 1044
Placemark ... 1046
Polygon ... 1055
Polyline ... 1065
Popup ... 1073

projection ... 1076
projection.Cartesian ... 1076
projection.sphericalMercator ... 1077
projection.wgs84Mercator ... 1077

ready ... 1078
Rectangle ... 1079

regions ... 1089
regions.load ... 1089

RemoteObjectManager ... 1091
route ... 1100

router ... 1104

router.addon ... 1104

router.addon.editor ... 1104
router.addon.editor.get ... 1104

router.Editor ... 1105
router.Path ... 1108
router.Route ... 1113
router.Segment ... 1119
router.ViaPoint ... 1122
router.WayPoint ... 1127

shape ... 1133
shape.Circle ... 1133
shape.LineString ... 1134
shape.MultiGeometry ... 1135
shape.MultiPolygon ... 1136
shape.Polygon ... 1137
shape.Rectangle ... 1139
shape.storage ... 1140

suggest ... 1141
SuggestView ... 1141

template ... 1146
template.filtersStorage ... 1146

Template ... 1148
templateLayoutFactory ... 1150

traffic ... 1153

traffic.provider ... 1153
traffic.provider.Actual ... 1153
traffic.provider.Archive ... 1155
traffic.provider.Forecast ... 1160
traffic.provider.storage ... 1163

util ... 1164
util.AsyncStorage ... 1164
util.augment ... 1167
util.bind ... 1168
util.bounds ... 1169

util.cursor ... 1174
util.cursor.Accessor ... 1174
util.cursor.Manager ... 1175

util.defineClass ... 1178
util.Dragger ... 1180
util.extend ... 1183
util.hd ... 1184

util.math ... 1185
util.math.areEqual ... 1185
util.math.cycleRestrict ... 1186
util.math.restrict ... 1186

util.pixelBounds ... 1188
util.requireCenterAndZoom ... 1191
util.Storage ... 1192

vow ... 1194
vow.Deferred ... 1196
vow.Promise ... 1197

Yandex.Maps API Reference

This reference guide describes the JavaScript API version 2.1.63.

The release date is April 17, 2018.

Changed:

- New design for the "Open in Yandex.Maps" block in the panorama player.
- API key validation for the free API host when it is specified through the API loading parameters. Statistics are now collected for API keys that are valid for commercial use.

Fixed:

- The 'autofocus: false' option for the routes panel didn't work when either the start point or end point of the route was already added.
- Excessive 'actionbegin' events occurred during map scaling in Safari on mobile devices.
- The 'viaPointsChange' attribute of the 'requestsuccess' event had the wrong value after dragging route throughpoints.
- Layout fixes.

Balloon

Extends **IBalloon**, **Popup**.

A balloon is a popup window that can display any HTML content. There is usually just one balloon instance on the map and it is managed via special managers (for example, maps, geo objects, hotspot layers and so on). Don't create them yourself, unless truly necessary.

Constructor | Fields | Events | Methods

Constructor

```
Balloon(map[, options])
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Reference to a map object.
options	—	Type: Object Options.
options.autoPan	true	Type: Boolean To move the map to show the opened balloon.
options.autoPanCheckZoomRange	false	Type: Boolean Enables autoscaling when it is impossible to display the map after dragging on the same scale.
options.autoPanDuration	500	Type: Number

Parameter	Default value	Description
		The duration of the movement to the point of the balloon (in milliseconds).
options.autoPanMargin	34	Type: Number|Number[] Offset or offsets from the edges of the visible map area when executing autoPan. The value can be set as a single number (equal margins on all sides), as two numbers (for vertical and horizontal margins), or as four numbers (in the order of upper, right, lower, and left margins). Keep in mind that this value will be added to the value calculated in the margins manager **map.margin.Manager**.
options.autoPanUseMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager** when executing autoPan.
options.closeButton	true	Type: Boolean Flag for the Close button.
options.closeTimeout	700	Type: Number Delay before closing (in ms).
options.contentLayout	—	Type: Function|String Layout for balloon content. (Type: constructor for an object with the **ILayout** interface or the layout key).
options.interactivityModel	—	Type: String Key for the interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.layout	islands#balloon	Type: Function|String External layout for the balloon. (Type: constructor for an object with the **ILayout** interface or the layout key).
options.maxHeight	—	Type: Number Maximum height, in pixels.
options.maxWidth	—	Type: Number Maximum width, in pixels.
options.minHeight	—	Type: Number Minimum height, in pixels.
options.minWidth	—	Type: Number Minimum width, in pixels.

Parameter	Default value	Description
options.offset	—	Type: Number[] Additional position offset relative to the anchor point.
options.openTimeout	150	Type: Number Delay before opening (in ms).
options.pane	'balloon'	Type: String Key of the pane that the balloon overlay is placed in.
options.panelContentLayout	null	Type: Function|String The layout of the balloon contents in the panel mode. If this option is omitted, the value of the contentLayout option is used. (Type: constructor for an object with the **ILayout** interface or the layout key).
options.panelMaxHeightRatio	—	Type: Number The maximum height of the balloon panel. Defined as the coefficient relative to the map height: a number from 0 to 1.
options.panelMaxMapArea	—	Type: Number The maximum area of the map at which the balloon will be displayed in the panel mode. You can disable panel mode by setting the value to *0*, and vice versa, you can always show the balloon in panel mode by setting the value to *Infinity*.
options.shadow	true	Type: Boolean Flag for whether there is a shadow.
options.shadowLayout	—	Type: Function|String Layout for the shadow. (Type: constructor for an object with the **ILayout** interface or the layout key).
options.shadowOffset	—	Type: Number[] Additional position offset of the shadow relative to the anchor point.
options.zIndex	—	Type: String The z-index of the balloon.

* Mandatory parameter/option.

Example:

```
// Creating an independent balloon instance and displaying it in the center of
the map.
var balloon = new ymaps.Balloon(myMap);
// Here map options are set to parent options,
// where they contain default values for mandatory options.
balloon.options.setParent(myMap.options);
// Opening a balloon at the center of the map:
balloon.open(myMap.getCenter());
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
autopanbegin	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the Event class.
autopanend	End of automatic shifting of the map center initiated by the autoPan method. Instance of the Event class.
beforeuserclose	The event which precedes Balloon.event:userclose. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class.
close	Closing the info object. Inherited from IPopup.
open	Opening the info object. Inherited from IPopup.
optionschange	Change to the object options. Inherited from ICustomizable.
userclose	Balloon closed by the user. Instance of the Event class.

Methods

Name	Returns	Description
autoPan()	vow.Promise	Moves the map so that the balloon is visible. Inherited from IBalloon.
close([force])	vow.Promise	Closes the info object. Inherited from IPopup.
getData()		Returns info object data. Inherited from IPopup.
getOverlay()	vow.Promise	Returns the promise object to return the overlay. Inherited from IPopup.

Name	Returns	Description
getOverlaySync()	**IOverlay**	Returns the overlay, if one exists. Inherited from **IPopup**.
getPosition()		Returns the coordinates of the info object. Inherited from **IPopup**.
isOpen()	Boolean	Returns the info object state: open/ closed. Inherited from **IPopup**.
open([position[, data]])	**vow.Promise**	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the **IProjection** that is in the options. Inherited from **IPopup**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopup**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopup**.

Events details

autopanbegin

Start of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class.

autopanend

End of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class.

beforeuserclose

The event which precedes **Balloon.event:userclose**. Allows you to cancel the user's action by calling the preventDefault method. Instance of the **Event** class.

userclose

Balloon closed by the user. Instance of the **Event** class.

behavior

behavior.DblClickZoom

Extends **IBehavior**.

The "zooming the map with double-click" behavior.

Constructor | Fields | Events | Methods

Constructor

```
behavior.DblClickZoom([options])
```

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options.
options.centering	true	Type: Boolean If true, the map is zoomed on double-click so that the point under the mouse cursor becomes the map center; if false, the point under the mouse cursor stays in the same position when zooming on double-click.
options.duration	200	Type: Number Duration of animation for zooming on double-click (0 — no animation).
options.useMapMargin	true	Type: Boolean When centering, whether to consider margins that were calculated in the margins manager **map.margin.Manager**.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
disable	Disabling behaviors. Inherited from **IBehavior**.
enable	Enabling behaviors.

Name	Description
	Inherited from **IBehavior**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Methods

Name	Returns	Description
disable()		Disables the behavior. Inherited from **IBehavior**.
enable()		Enables the behavior. Inherited from **IBehavior**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
isEnabled()	Boolean	Checks whether the behavior is enabled. Inherited from **IBehavior**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

behavior.Drag

Extends **IBehavior**.

The "dragging the map using the mouse or single touch" behavior.

Constructor | Fields | Events | Methods

Constructor

```
behavior.Drag([options])
```

Parameters:

Parameter	Default value	Description
`options`	—	Type: Object Options.
`options.actionCursor`	'grabbing'	Type: String

Parameter	Default value	Description
		Cursor for the behavior **behavior.Drag** when dragging the map.
options.cursor	'grab'	Type: String Cursor for the behavior **behavior.Drag** when pointing at the map.
options.inertia	true	Type: Boolean Enables kinetic inertia at the end of dragging.
options.inertiaDuration	400	Type: Number|String Duration of inertia, in ms. The "auto" string value sets the duration of inertia proportional to the distance.
options.tremor	2	Type: Integer Minimal cursor movement after pressing the mouse button, before the map begins to move.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
disable	Disabling behaviors. Inherited from **IBehavior**.
enable	Enabling behaviors. Inherited from **IBehavior**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Methods

Name	Returns	Description
disable()		Disables the behavior. Inherited from **IBehavior**.
enable()		Enables the behavior. Inherited from **IBehavior**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
isEnabled()	Boolean	Checks whether the behavior is enabled. Inherited from **IBehavior**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

behavior.LeftMouseButtonMagnifier

Extends **IBehavior**.

The "zooming the map when selecting an area with the left mouse button" behavior.

Constructor | Fields | Events | Methods

Constructor

```
behavior.LeftMouseButtonMagnifier([options])
```

Creates the "zooming the map when selecting an area with the left mouse button" behavior.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options.
options.actionCursor	'crosshair'	Type: String The cursor when selecting the area to magnify with the enabled behavior **behavior.LeftMouseButtonMagnifier**. The list of all available cursors is provided in the cursors manager.
options.cursor	'zoom'	Type: String The cursor for the enabled behavior **behavior.LeftMouseButtonMagnifier**. The list of all available cursors is provided in the cursors manager.
options.duration	300	Type: Number

Parameter	Default value	Description
		The duration of magnification animation when using the behavior **behavior.LeftMouseButtonMagnifier**, in ms.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
disable	Disabling behaviors. Inherited from **IBehavior**.
enable	Enabling behaviors. Inherited from **IBehavior**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Methods

Name	Returns	Description
disable()		Disables the behavior. Inherited from **IBehavior**.
enable()		Enables the behavior. Inherited from **IBehavior**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
isEnabled()	Boolean	Checks whether the behavior is enabled. Inherited from **IBehavior**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

Name	Returns	Description
		Inherited from **IChildOnMap**.

behavior.MultiTouch

Extends **IBehavior**.

The "zooming the map using multitouch" behavior.

Constructor | Fields | Events | Methods

Constructor

```
behavior.MultiTouch([options])
```

Parameters:

Parameter	Default value	Description
`options`	—	Type: Object Options.
`options.tremor`	2	Type: Number The minimal movement on the device screen (in pixels) to trigger the behavior **behavior.MultiTouch**.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
disable	Disabling behaviors. Inherited from **IBehavior**.
enable	Enabling behaviors. Inherited from **IBehavior**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Methods

Name	Returns	Description
disable()		Disables the behavior. Inherited from **IBehavior**.
enable()		Enables the behavior. Inherited from **IBehavior**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
isEnabled()	Boolean	Checks whether the behavior is enabled. Inherited from **IBehavior**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

behavior.RightMouseButtonMagnifier

Extends **IBehavior**.

The "zooming the map when selecting an area with the right mouse button" behavior.

Constructor | Fields | Events | Methods

Constructor

```
behavior.RightMouseButtonMagnifier([options])
```

Creates the "zooming the map when selecting an area with the right mouse button" behavior.

Parameters:

Parameter	Default value	Description
`options`	—	Type: Object Options.
`options.actionCursor`	'crosshair'	Type: String The cursor when selecting the area to magnify with the enabled behavior **behavior.RightMouseButtonMagnifier**. The list of all available cursors is provided in the cursors manager.
`options.duration`	300	Type: Number The duration of animation when selecting an area using the behavior **behavior.RightMouseButtonMagnifier**, in ms.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
disable	Disabling behaviors. Inherited from IBehavior.
enable	Enabling behaviors. Inherited from IBehavior.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Methods

Name	Returns	Description
disable()		Disables the behavior. Inherited from IBehavior.
enable()		Enables the behavior. Inherited from IBehavior.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
isEnabled()	Boolean	Checks whether the behavior is enabled. Inherited from IBehavior.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

behavior.RouteEditor

Extends IBehavior.

"Route editor" behavior.

Constructor | Fields | Events | Methods

Constructor

```
behavior.RouteEditor()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
disable	Disabling behaviors. Inherited from IBehavior.
enable	Enabling behaviors. Inherited from IBehavior.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
routechange	A changed route event resulting from calling the setState method or enabling/disabling behaviors. You can get the old and new routes from the properties of the "oldRoute" and "newRoute" events, respectively.

Methods

Name	Returns	Description
disable()		Disables the behavior. Inherited from IBehavior.
enable()		Enables the behavior. Inherited from IBehavior.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getRoute()	router.Route	Returns route.
getState()	String	Returns the current state of the route editor in encoded format.

Name	Returns	Description
isEnabled()	Boolean	Checks whether the behavior is enabled. Inherited from **IBehavior**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.
setState(state)		Restores the state of the route editor from the encoded string. When setting a new route, the "routechange" event is called.

Events details

routechange

A changed route event resulting from calling the setState method or enabling/disabling behaviors. You can get the old and new routes from the properties of the "oldRoute" and "newRoute" events, respectively.

Methods details

getRoute

```
{router.Route} getRoute()
```

Returns route.

getState

```
{String} getState()
```

Returns the current state of the route editor in encoded format.

setState

```
{} setState(state)
```

Restores the state of the route editor from the encoded string. When setting a new route, the "routechange" event is called.

Parameters:

Parameter	Default value	Description
state*	—	Type: String|null Encoded state of the route editor. The state is set in the following format: rt=point1~point2~point3~point4...&via=via-point-indexes For example, if the encoded string looks like this: "rt=50,30~42,35~45,32~40,30&via=1,2", it means the waypoints on the route should be "50,30" and "40,30", and the midpoints are "42,35" and "45,32". If the "via" parameter is omitted, the route will consist solely of waypoints.

* Mandatory parameter/option.

Example:

```
var behavior = map.behaviors.get('routeEditor');
behavior.events.add('routechange', function (e) {
   var newRoute = e.get('newRoute');
   alert(newRoute.getLength()); // length of the new route
   setTimeout(function () {
       // deleting the route
       behavior.setState(null);
   })
});
behavior.setState('rt=55.874872,37.562677~55.92517867214157,37.62725433916199~55
.920011490602526,37.6629269628905&via=1');
```

behavior.Ruler

Extends **IBehavior**.

The "Ruler" behavior. For marking points on the map and displaying the distance between them.

Constructor | Fields | Events | Methods

Constructor

```
behavior.Ruler([options])
```

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options.
options.balloonAutoPan	true	Type: Boolean Whether to auto-position the map when opening the ruler balloon.
options.balloonAutoPanUseMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager** when executing autoPan for the ruler balloon.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**geometry.LineString**	"Line" behavior geometry.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
disable	Disabling behaviors. Inherited from **IBehavior**.
enable	Enabling behaviors. Inherited from **IBehavior**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Methods

Name	Returns	Description
close()	Boolean	Deletes all the points on the ruler. If the current number of points is more than two, confirmation of this action will be requested.
disable()		Disables the behavior. Inherited from **IBehavior**.
enable()		Enables the behavior. Inherited from **IBehavior**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getState()	String	The ruler state is described by a string consisting of sequences separated by the "~" symbol. Each sequence is a substring in the format "longitude,latitude" that describes the increment in coordinates relative to the previous ruler point.
isEnabled()	Boolean	Checks whether the behavior is enabled. Inherited from **IBehavior**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.
setState(state)		Retrieves the ruler state from the encoded string. See behavior.Ruler.getState.

Fields details

geometry

```
{geometry.LineString} geometry
```

"Line" behavior geometry.

Example:

```
// Setting coordinates of the first point on the ruler.
myMap.behaviors.get('ruler').geometry.set(0, [0, 0]);
```

Methods details

close

```
{Boolean} close()
```

Deletes all the points on the ruler. If the current number of points is more than two, confirmation of this action will be requested.

Returns true, if the action was completed successfully.

getState

```
{String} getState()
```

The ruler state is described by a string consisting of sequences separated by the "~" symbol. Each sequence is a substring in the format "longitude,latitude" that describes the increment in coordinates relative to the previous ruler point.

Returns the current state of the ruler, in encoded format.

setState

```
{} setState(state)
```

Retrieves the ruler state from the encoded string. See behavior.Ruler.getState.

Parameters:

Parameter	Default value	Description
state*	—	Type: String Encoded state of the ruler.

* Mandatory parameter/option.

behavior.ScrollZoom

Extends **IBehavior**.

The "zooming the map with the mouse wheel" behavior.

Constructor | Fields | Events | Methods

Constructor

```
behavior.ScrollZoom([options])
```

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options.
options.maximumDelta	5	Type: Number The maximum change in the map zoom for a single scroll event (uninterrupted turn of the mouse wheel). When this limit is reached, the map is re-drawn (tiles are shown for the zoom level that was reached).
options.speed	5	Type: Number The speed of zooming the map with the enabled behavior **behavior.ScrollZoom**, in zoom levels per second.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
disable	Disabling behaviors. Inherited from **IBehavior**.
enable	Enabling behaviors. Inherited from **IBehavior**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Methods

Name	Returns	Description
disable()		Disables the behavior. Inherited from **IBehavior**.

Name	Returns	Description
enable()		Enables the behavior. Inherited from **IBehavior**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
isEnabled()	Boolean	Checks whether the behavior is enabled. Inherited from **IBehavior**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

behavior.storage

Static object.

Instance of **util.Storage**

Storage for map behavior classes. A new behavior is added to the map via this storage.

By default, the following behaviors are added to the storage:

- "drag" — Dragging the map when the left mouse button is held down, or by a single touch **behavior.Drag**.
- "scrollZoom" — Changing the zoom with the mouse wheel **behavior.ScrollZoom** (only for desktop browsers).
- "dblClickZoom" — Zooming the map on a double click **behavior.DblClickZoom**.
- "multiTouch" — Zooming the map with a multi touch (i.e. on a touch screen) **behavior.MultiTouch** (only for mobile browsers).
- "rightMouseButtonMagnifier" — Magnifying the area that is selected using the right mouse button (for desktop browsers only), **behavior.RightMouseButtonMagnifier** (for desktop browsers only).
- "leftMouseButtonMagnifier" — Magnifying the area selected by the left mouse button or a single touch, **behavior.LeftMouseButtonMagnifier**.
- "ruler" — Measuring distance **behavior.Ruler**.
- "routeEditor" — Route editor **behavior.RouteEditor**.

Methods

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

borders

borders.load

Static function.

Provides access to the geometry of various regions and countries. If you using content security policy then you need to add 'connect-src' rule for following hosts: https://api-maps.yandex.ru https://suggest-maps.yandex.ru https://*.maps.yandex.net https://yandex.ru

Returns Promise object.

```
{ vow.Promise } borders.load(region[, options])
```

Parameters:

Parameter	Default value	Description
region*	—	Type: String The ISO_3166-1 country code (RU, UA, BY, KZ, TR) for loading the regional area, '001' for loading the geometry of country borders, or AQ for loading the geometry of Antarctic.
options	—	Type: Object Display options.
options.disputedBorders	—	Type: String Two-letter code of the country to use as the official reference for determining the administrative subordination of disputed territories. Accepted values: 'RU', 'UA', 'UN'. By default, it coincides with the country code that is specified when loading the API. Unsupported country codes are reset to RU. For the region '001' (borders of countries), the code 'UN' is supported — world borders according to the United Nations.
options.lang	—	Type: String Language (ru, en, uk, be, kk, tr).
options.quality	1	Type: Number Quality level. Available values: • 0 — minimal quality • 1 — standard quality • 2 — improved quality • 3 — high quality The quality level affects how accurately curves are represented, as well as the volume of the data file.

* Mandatory parameter/option.

Examples:

1.

```
// Show objects on the map using ObjectManager
ymaps.borders.load('RU', {
    lang: 'en'
}).then(function (geojson) {
    var features = geojson.features.map(function (feature) {
        feature.id = feature.properties.iso3166;
        return feature;
    });
    var objectManager = new ymaps.ObjectManager();
    objectManager.add(features);
    myMap.geoObjects.add(objectManager);
});
```

2.

```
// Show objects on the map using GeoObject
ymaps.borders.load('RU', {
    lang: 'en'
}).then(function (geojson) {
    for (var i = 0; i < geojson.features.length; i++) {
        var geoObject = new ymaps.GeoObject(geojson.features[i]);
        myMap.geoObjects.add(geoObject);
    }
});
```

Circle

Extends **GeoObject**.

Circle. A geo object with the geometry **geometry.Circle**.

See GeoObject geometry.Circle

Constructor | Fields | Events | Methods

Constructor

```
Circle(geometry[, properties[, options]])
```

Creates an instance of a circle.

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **ICircleGeometry**|Number[][]|Object Reference to a point geometry object or an array, in which the first item is the coordinates of the center of the circle, and the second is the radius in meters, or an object with the parameters of the geometry.
properties	—	Type: Object|**IDataManager** Circle data. Corresponds to the data for the **GeoObject** object. Can be set as an instance of a class that implements the **IDataManager** interface, or as a hash. When options are set to default values, the following data fields are interpreted by a circle: • hintContent — Content of the circle's popup hint. • balloonContent — Content of the circle's balloon. • balloonContentHeader — Content of the circle balloon title. • balloonContentBody — Content of the main part of the circle's balloon. • balloonContentFooter — Content of the lower part of the circle's balloon. The balloonContent field is a shortcut for the balloonContentBody field, but if they are both set simultaneously, balloonContentBody takes priority. You can also add your own custom fields to the circle data and use them wherever possible. For example, in the circle layout or balloon layout.
options	—	Type: Object

Parameter	Default value	Description
		Circle options. Using this parameter, you can set options for the circle itself, as well as for its parts: • Options for the circle's balloon with the `balloon` prefix. • Options for the circle's popup hint with the `hint` prefix. • Options for the circle's geometry editor with the `editor` prefix. See the description of the **geometryEditor.Circle** class. • Geometry options can be set without a prefix. See the description of the **IGeometry** class for the **geometry.Circle** geometry.
`options.circleOverlay`	"default#circle"	Type: String|Function Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelCircleGeometry** — The pixel geometry itself. • data: Object — The overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
`options.cursor`	"pointer"	Type: String Type of cursor over a circle.
`options.draggable`	false	Type: Boolean Checks whether the circle can be dragged.
`options.fill`	true	Type: Boolean Whether the shape is filled.
`options.fillColor`	"0066ff99"	Type: String Fill color.
`options.fillImageHref`	—	Type: String Background image. When this option is enabled in **stretch** mode, the "fillColor" value is ignored.
`options.fillMethod`	'stretch'	Type: String

Parameter	Default value	Description
		Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	1	Type: Number Fill transparency.
options.hasBalloon	true	Type: Boolean Checks whether the circle has the "balloon" field.
options.hasHint	true	Type: Boolean Checks whether the circle has the "hint" field.
options.hideIconOnBalloonOpen	true	Type: Boolean Hide the icon when opening the balloon.
options.interactiveZIndex	false	Type: Boolean Enables automatically modifying the z-index of the circle depending on its state.
options.interactivityModel	"default#geoObject"	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.opacity	1	Type: Number Transparency.
options.openBalloonOnClick	true	Type: Boolean Checks whether to show the balloon when the circle is clicked on.
options.openEmptyBalloon	false	Type: Boolean Checks whether to show an empty balloon when the circle is clicked on.
options.openEmptyHint	false	Type: Boolean Checks whether to show an empty hint when the mouse pointer hovers over the circle.
options.openHintOnHover	true	Type: Boolean

Parameter	Default value	Description
		Checks whether to show the hint when the mouse pointer hovers over the circle.
options.outline	true	Type: Boolean Whether the circle has an outline.
options.pane	"areas"	Type: String The key of the pane where the circle overlay is placed.
options.strokeColor	"0066ffff"	Type: String|String[] Color of the line or outline. You can set multiple values for a multistroke outline.
options.strokeOpacity	1	Type: Number|Number[] Transparency of the line or outline. You can set multiple values for a multistroke outline.
options.strokeStyle	—	Type: String|Object|String[]|Object[] Style of the line or outline. You can set multiple values for a multistroke outline.
options.strokeWidth	1	Type: Number|Number[] Thickness of the line or outline. You can set multiple values for a multistroke outline.
options.syncOverlayInit	false	Type: Boolean Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a circle to the map.
options.useMapMarginInDragging	true	Type: Boolean When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with **map.margin.Manager**.
options.visible	true	Type: Boolean Checks circle visibility.
options.zIndex	—	Type: Number

Parameter	Default value	Description
		The z-index of a circle in its normal state. Lowest priority.
options.zIndexActive	—	Type: Number The z-index of a circle with an open balloon. Highest priority.
options.zIndexDrag	—	Type: Number The z-index of a circle that is being dragged.
options.zIndexHover	—	Type: Number The z-index of a circle when the mouse pointer is hovering over it.

* Mandatory parameter/option.

Example:

```
// Creating a geodesic circle with a radius of 1000 kilometers.
var circle = new ymaps.Circle([[50, 75], 1000000], {}, {
    geodesic: true
});
// Adding the circle to the map.
myMap.geoObjects.add(circle);
```

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object. Inherited from **GeoObject**.
editor	**geometryEditor.Circle**	The "Circle" geometry editor.
events	**event.Manager**	Event manager. Inherited from **GeoObject**.
geometry	**geometry.Circle**	The "Circle" type of geometry.
hint	**geoObject.Hint**	Geo object hint. Inherited from **GeoObject**.
indices	ArrayBuffer	
options	**option.Manager**	Geo object options manager. Inherited from **GeoObject**.
properties	**data.Manager**	Geo object data manager. Inherited from **GeoObject**.
state	**data.Manager**	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged

Name	Type	Description
		Inherited from **GeoObject**.
vertices	ArrayBuffer	

Events

Name	Description
balloonclose	Closing the balloon. Instance of the **Event** class. Inherited from **GeoObject**.
balloonopen	Opening a balloon on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
beforedrag	Event preceding the "drag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent drag event will be canceled. Inherited from **GeoObject**.
beforedragstart	Event preceding the "dragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from **GeoObject**.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Name	Description
drag	Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragend	End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragstart	Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent — Original event of the geometry editor. Inherited from **GeoObject**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
hintclose	Closing the hint. Instance of the **Event** class. Inherited from **GeoObject**.
hintopen	Opening a hint on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object.

Name	Description
	Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Fields details

editor

```
{geometryEditor.Circle} editor
```

The "Circle" geometry editor.

geometry

```
{geometry.Circle} geometry
```

The "Circle" type of geometry.

indices

```
{ArrayBuffer} indices
```

vertices

```
{ArrayBuffer} vertices
```

clusterer

clusterer.addon

clusterer.addon.balloon

Note:
The constructor of the clusterer.addon.balloon class is hidden, as this class is not intended for autonomous initialization.

Static object.

Methods

Methods

Name	Returns	Description
get(clusterer)	IPopupManager	Returns the clusterer's balloon manager.

Methods details

get

```
{IPopupManager} get(clusterer)
```

Returns the clusterer's balloon manager.

Parameters:

Parameter	Default value	Description
clusterer*	—	Type: Clusterer Clusterer

* Mandatory parameter/option.

Example:

```
ymaps.clusterer.addon.balloon.get(clusterer)
```

clusterer.addon.hint

Note:
The constructor of the clusterer.addon.hint class is hidden, as this class is not intended for autonomous initialization.

Static object.

Methods

Methods

Name	Returns	Description
get(clusterer)	IPopupManager	Returns the clusterer's hint manager.

Methods details

get

```
{IPopupManager} get(clusterer)
```

Returns the clusterer's hint manager.

Parameters:

Parameter	Default value	Description
clusterer*	—	Type: Clusterer Clusterer

* Mandatory parameter/option.

Example:

```
ymaps.clusterer.addon.hint.get(clusterer)
```

clusterer.Balloon

Extends **IBalloonManager**.

The clusterer's balloon manager. Provides management of the cluster's balloon — opening it and hiding it. It uses the map balloon manager **map.Balloon**. Clusterers contain an instance of this class, which is available as myGeoObject.balloon. Don't create new instances of this class unless necessary.

See Balloon

Constructor | Fields | Events | Methods

Constructor

```
clusterer.Balloon(clusterer)
```

Parameters:

Parameter	Default value	Description
clusterer *	—	Type: **Clusterer** Clusterer.

* Mandatory parameter/option.

Fields

Name	Type	Description
close		Closing the balloon. • target — Link to the clusterer balloon manager. • cluster — Reference to the cluster object. Instance of the Event class.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
open		Opening a balloon on a cluster. Names of fields that are available via the **Event.get** method: • target — Link to the clusterer balloon manager. • cluster — Reference to the cluster object. Instance of the Event class.

Events

Name	Description
autopanbegin	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.

Name	Description
autopanend	End of automatic shifting of the map center initiated by the autoPan method. Instance of the Event class. Names of fields that are available via the Event.get method: • target — Reference to the IBalloonOwner object. Inherited from IBalloonManager.
beforeuserclose	The event which precedes Balloon.event:userclose. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the Event.get method: • target — Reference to the IBalloonOwner object. Inherited from IBalloonManager.
close	Closing the info object. Names of fields available via Event.get: • target — Reference to the object where the closing occurred. Inherited from IPopupManager.
open	Opening the info object. Names of fields available via Event.get: • target — Reference to the object where the opening occurred. Inherited from IPopupManager.
userclose	Balloon closed by the user. Instance of the Event class. Names of fields that are available via the Event.get method: • target — Reference to the IBalloonOwner object. Inherited from IBalloonManager.

Methods

Name	Returns	Description
autoPan()	vow.Promise	Moves the map so that the balloon is visible. Inherited from IBalloonManager.
destroy()		Disables the info object manager. Inherited from IPopupManager.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from IPopupManager.
getOptions()	IOptionManager|null	Returns the options manager or 'null'. Inherited from IPopupManager.
getOverlay()	vow.Promise	Returns the promise object to return the overlay. Inherited from IPopupManager.
getOverlaySync()	IOverlay|null	Returns the overlay, if one exists. Inherited from IPopupManager.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from IPopupManager.

Name	Returns	Description
isOpen()	Boolean	Returns the info object state: open/ closed. Inherited from **IPopupManager**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopupManager**.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

Fields details

close

```
close
```

Closing the balloon.

- target — Link to the clusterer balloon manager.
- cluster — Reference to the cluster object.

Instance of the Event class.

open

```
open
```

Opening a balloon on a cluster. Names of fields that are available via the **Event.get** method:

- target — Link to the clusterer balloon manager.
- cluster — Reference to the cluster object.

Instance of the Event class.

Example:

```
clusterer.balloon.events.add('open', function (e) {
    var clusterPlacemark = e.get('cluster');
});
```

clusterer.Hint

Extends **IHintManager**.

The clusterer's hint manager. Provides management of the cluster's hint, opening it and hiding it. It uses the map hint manager **map.Hint** inside itself. Clusterers contain an instance of this class, which is available as myClusterer.hint. Don't create new instances of this class unless necessary.

See Hint

Constructor | Fields | Events | Methods

Constructor

```
clusterer.Hint(clusterer)
```

Parameters:

Parameter	Default value	Description
clusterer *	—	Type: **Clusterer** Clusterer.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
close	Closing the info object. Names of fields available via **Event.get**: • target — Reference to the object where the closing occurred. Inherited from **IPopupManager**.
open	Opening the info object. Names of fields available via **Event.get**: • target — Reference to the object where the opening occurred. Inherited from **IPopupManager**.

Methods

Name	Returns	Description
close([force])	**vow.Promise**	Closes the info object. Inherited from **IPopupManager**.
destroy()		Disables the info object manager. Inherited from **IPopupManager**.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from **IPopupManager**.
getOptions()	**IOptionManager**|null	Returns the options manager or 'null'. Inherited from **IPopupManager**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopupManager**.
getOverlaySync()	**IOverlay**|null	Returns the overlay, if one exists. Inherited from **IPopupManager**.

Name	Returns	Description
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from **IPopupManager**.
isOpen()	Boolean	Returns the info object state: open/ closed. Inherited from **IPopupManager**.
open([position[, data[, options]]])	**vow.Promise**	Opens the info object at the specified position. Inherited from **IPopupManager**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopupManager**.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

Clusterer

Extends **IChildOnMap**, **ICustomizable**, **IEventEmitter**, **IParentOnMap**.

Geo object clusterer. Clusterizes objects in the visible area of the map. If the object does not fall within the visible area of the map, it will not be added to the map. Note, that the clusterer does not react to changing the coordinates of objects (either programmatically, or as the result of dragging). If you want to change the coordinates of some object in the clusterer, you should first delete the object from the clusterer and then add it back.

See ClusterPlacemark

Constructor | Fields | Events | Methods

Constructor

```
Clusterer([options])
```

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options. Options for child cluster objects are set with the "cluster" prefix. See **ClusterPlacemark**.
options.gridSize	64	Type: Number The size of cluster cells, in pixels. The value must be equal 2^n (an area of 256 by 256 pixels should fit an even number of cells).
options.groupByCoordinates	false	Type: Boolean Special operation mode of the clusterer allowing clusters to be formed only from geo objects with the same coordinates.
options.hasBalloon	true	Type: Boolean Flag whether the clusterer has the .balloon field. If a balloon doesn't need to be opened when clicking the cluster, we recommend setting this option to the "false" value to avoid unnecessary initializations.
options.hasHint	true	Type: Boolean Flag whether the clusterer has the .hint field. If a popup hint doesn't need to be displayed when the cluster is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations.
options.margin	10	Type: Number|Number[]

Parameter	Default value	Description
		Number or array of numbers that set the offset for the cluster center relative to the clusterization cells. If a single number is set, it is applied to each side. If two numbers are set, they are the vertical and horizontal offsets, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins.
options.maxZoom	Infinity	Type: Number[] The maximum map zoom that object clusterization occurs at. Even if clusterization is disabled, only objects in the visible map area will be shown.
options.minClusterSize	2	Type: Number Minimum number of objects to make up a cluster.
options.preset	—	Type: String Key for the clusterer's preset options. A list of keys available in the package.clusters package can be found in the description of **option.presetStorage**.
options.showInAlphabeticalOrder	false	Type: Boolean Show placemarks in the balloon in alphabetical order when the cluster is clicked on. The cluster's geo objects are sorted by special fields in the data of these geo objects — clusterCaption (or balloonContentHeader, if the previous field is not defined). By default, geo objects are shown in the order in which they were added to the clusterer.
options.useMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager** when zooming the map in after a click on a cluster.
options.viewportMargin	128	Type: Number|Number[] The offset of the area in which you are clustering. Clustering is performed for the visible area of the map. Use this option to expand the area of clustering relative to the visible area of the map.
options.zoomMargin	0	Type: Number|Number[]

Parameter	Default value	Description
		Offset from the map viewport borders to observe when zooming in the map after a cluster is clicked. Recommended to set this option's value to match the size of the cluster icons and placemarks. For example, if only the bottom half of a placemark falls on the visible area of the map, a nonzero "top" offset should be set, so the placemark remains completely visible after the cluster is broken up. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins.

Example:

```
// Creating a clusterer

// Creating a map that requires clusterization of geo objects
var map = new ymaps.Map('mapsID', { center: [56.034, 36.992], zoom: 8 }),
// Creating an array of geo objects
myGeoObjects = [];
myGeoObjects[0] = new ymaps.GeoObject({
    geometry: { type: "Point", coordinates: [56.034, 36.992] },
    properties: {
        clusterCaption: 'Geo object №1',
        balloonContentBody: 'Balloon content for geo object №1'
    }
});
myGeoObjects[1] = new ymaps.GeoObject({
    geometry: { type: "Point", coordinates: [56.021, 36.983] },
    properties: {
        clusterCaption: 'Geo object №2',
        balloonContentBody: 'Balloon content for geo object №2'
   }
});

// Creating a cluster and prohibiting zooming the map in when the cluster is
clicked
var clusterer = new ymaps.Clusterer({ clusterDisableClickZoom: true });
clusterer.add(myGeoObjects);
map.geoObjects.add(clusterer);
```

Fields

Name	Type	Description
balloon	clusterer.Balloon	Clusterer's balloon.
balloonclose		Closing the balloon. • target — Link to the clusterer balloon manager. • cluster — Reference to the cluster object. Instance of the Event class.

Name	Type	Description
balloonopen		Opening a balloon on a cluster. Names of fields that are available via the **Event.get** method: • target — Link to the clusterer balloon manager. • cluster — Reference to the cluster object. Instance of the Event class.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
hint	clusterer.hint	Clusterer's hint.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
hintclose	Closing the hint. Instance of the **Event** class.
hintopen	Opening a hint on a cluster. Instance of the **Event** class.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Methods

Name	Returns	Description
add(objects)	**Clusterer**	Adds a geo object or array of geo objects to the clusterer.
createCluster(center, geoObjects)	**IGeoObject**	Function for creating a cluster using the clusterer. Called directly by the clusterer during the clusterization process. Accepts as input the cluster center and an array of geo objects that go in this cluster. Returns the cluster, which will later be added to the map. If you need the clusterer to create user cluster objects, you should redefine this method for the clusterer.
getBounds()	Number[][]|null	Returns the geographical coordinates of the rectangular area that includes all the clusterer elements.

Name	Returns	Description
getClusters()	**IGeoObject**[]	Method for getting the current array of cluster objects. Note that the cluster objects change when you change the map zoom or shift its center. If you want to perform operations on all clusters, it is better to redefine the createCluster method and add the required operations to its call.
getGeoObjects()	**IGeoObject**[]	Returns an array of geo objects added to the clusterer.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getObjectState(geoObject)	Object	Function for getting information about the current state of an object that was added to the clusterer.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
remove(objects)	**Clusterer**	Removes geo objects from the clusterer.
removeAll()	**Clusterer**	Removes all geo objects from the clusterer.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Fields details

balloon

```
{clusterer.Balloon} balloon
```

Clusterer's balloon.

balloonclose

```
balloonclose
```

Closing the balloon.

- target — Link to the clusterer balloon manager.
- cluster — Reference to the cluster object.

Instance of the Event class.

balloonopen

```
balloonopen
```

Opening a balloon on a cluster. Names of fields that are available via the **Event.get** method:

- target — Link to the clusterer balloon manager.
- cluster — Reference to the cluster object.

Instance of the Event class.

Example:

```
clusterer.events.add('balloonopen', function (e) {
    var clusterPlacemark = e.get('cluster');
});
```

hint

```
{clusterer.hint} hint
```

Clusterer's hint.

Events details

hintclose

Closing the hint. Instance of the **Event** class.

hintopen

Opening a hint on a cluster. Instance of the **Event** class.

Methods details

add

```
{Clusterer} add(objects)
```

Adds a geo object or array of geo objects to the clusterer.

Returns self-reference.

Parameters:

Parameter	Default value	Description
objects*	—	Type: IGeoObject|IGeoObject[] Array of geo objects, or a single geo object.

* Mandatory parameter/option.

Example:

```
var myPlacemark = new ymaps.Placemark([35, 21]);
clusterer.add(myPlacemark);
var placemarks = [
    new ymaps.Placemark([44, 55]),
    new ymaps.Placemark([34, 45])
];
clusterer.add(placemarks);
```

createCluster

```
{IGeoObject} createCluster(center, geoObjects)
```

Function for creating a cluster using the clusterer. Called directly by the clusterer during the clusterization process. Accepts as input the cluster center and an array of geo objects that go in this cluster. Returns the cluster, which will later be added to the map. If you need the clusterer to create user cluster objects, you should redefine this method for the clusterer.

Returns the cluster object. By default, creates instances of the **ClusterPlacemark**.

Parameters:

Parameter	Default value	Description
center*	—	Type: Number[] The cluster center in geocoordinates.
geoObjects*	—	Type: IGeoObject[] Array of placemarks in the cluster.

* Mandatory parameter/option.

Example:

```
// Setting a hint for clusters depending on their contents.
var clusterer = new ymaps.Clusterer();
clusterer.createCluster = function (center, geoObjects) {
    // Creating a cluster placemark using a standard implementation of the
method.
    var clusterPlacemark = ymaps.Clusterer.prototype.createCluster.call(this,
center, geoObjects),
        geoObjectsLength = clusterPlacemark.getGeoObjects().length,
        hintContent;
    if (geoObjectsLength &lt; 10) {
        hintContent = 'Very few placemarks';
    } else if (geoObjectsLength &lt; 100) {
        hintContent = 'That's OK with placemarks';
    } else {
        hintContent = 'A lot of placemarks';
    }
    clusterPlacemark.properties.set('hintContent', hintContent);
    return clusterPlacemark;
};
```

getBounds

```
{Number[][]|null} getBounds()
```

Returns the geographical coordinates of the rectangular area that includes all the clusterer elements.

Example:

```
var clusterer = new ymaps.Clusterer();
clusterer.add(myPlacemarks);
map.setBounds(clusterer.getBounds());
map.geoObjects.add(clusterer);
```

getClusters

```
{IGeoObject[]} getClusters()
```

Method for getting the current array of cluster objects. Note that the cluster objects change when you change the map zoom or shift its center. If you want to perform operations on all clusters, it is better to redefine the createCluster method and add the required operations to its call.

Returns an array of clusters added to the map at this moment.

Example:

```
map.events.add('boundschange', function () {
    var clusters = clusterer.getClusters();
    alert('The map has moved and ' + clusters.length + ' clusters are displayed
now.');
});
```

getGeoObjects

```
{IGeoObject[]} getGeoObjects()
```

Returns an array of geo objects added to the clusterer.

Example:

```
// Counting objects in the visible area of the map.
var geoObjects = clusterer.getGeoObjects(),
    shownObjectsCounter = 0;
for (var i = 0, l = geoObjects.length; i &lt; l; i++) {
    if (clusterer.getObjectState(geoObjects[i]).isShown) {
        shownObjectsCounter++;
    }
}
alert('shownObjectsCounter + ' out of ' + geoObjects.length + ' placemarks are
displayed on the map.');
```

getObjectState

```
{Object} getObjectState(geoObject)
```

Function for getting information about the current state of an object that was added to the clusterer.

Returns object with following fields:

- isShown — Indicates whether an object is in the visible area of the map.
- cluster — A reference to the cluster the object was added to.
- isClustered — Indicates whether the object was put in the cluster.

.

Parameters:

Parameter	Default value	Description
geoObject*	—	Type: **IGeoObject** The geo object to get the state for.

* Mandatory parameter/option.

Example:

```
// Opening the cluster balloon with the selected object.
// Getting the info about the object's state within the cluster.
var geoObjectState = clusterer.getObjectState(myGeoObjects[1]);
// Checking if the object is located inside the visible area of the map.
if (geoObjectState.isShown) {

    // If the object is put in the cluster, the cluster balloon with the
appropriate object is opened.
    if (geoObjectState.isClustered) {
        geoObjectState.cluster.state.set('activeObject', myGeoObjects[1]);
        clusterer.balloon.open(geoObjectState.cluster);

    } else {
        // If the object was not in the cluster, its own balloon is  opened.
        myGeoObjects[1].balloon.open();
    }
}
```

remove

```
{Clusterer} remove(objects)
```

Removes geo objects from the clusterer.

Returns self-reference.

Parameters:

Parameter	Default value	Description
objects*	—	Type: IGeoObject|IGeoObject[] Array of geo objects.

* Mandatory parameter/option.

Example:

```
var myPlacemark = new ymaps.Placemark([35, 21]),
    placemarks = [
        new ymaps.Placemark([44, 55]),
        new ymaps.Placemark([34, 45]),
        ...
    ];
clusterer.add(myPlacemark).add(placemarks);
...
// Deleting the first 10 objects of the array from the clusterer.
clusterer.remove(placemarks.slice(0, 10));
```

removeAll

```
{Clusterer} removeAll()
```

Removes all geo objects from the clusterer.

Returns self-reference.

Example:

```
clusterer.add(placemark1);
clusterer.add(placemark2);
clusterer.removeAll();
```

ClusterPlacemark

Extends **IGeoObject**, **collection.Item**.

Cluster of geo objects. Used by default in **Clusterer**.

Constructor | Fields | Events | Methods

Constructor

```
ClusterPlacemark(geometry, properties[, options])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: Number[]|Object| IPointGeometry

Parameter	Default value	Description
		Coordinates of the placemark, or a hash describing the geometry, or a reference to the point geometry object.
properties *	—	Type: **IDataManager** Cluster data.
properties.geoObjects *	—	Type: **IGeoObject**[] Array of geo objects that are located in the given cluster.
options	—	Type: Object Cluster options. In addition to special options, the cluster balloon supports the same options as **Balloon**. The cluster balloon options are indicated with the "balloon" prefix.
options.balloonContentLayout	'cluster#balloonTwoColumns'	Type: Function|String

Parameter	Default value	Description

		Layout of the cluster balloon in normal mode. You can pass the object's constructor with the **ILayout** interface or the key for one of the standard layouts. Each standard layout has several customized options. In standard layouts, geo object data fields are used by default: clustercaption, balloonContentHeader, balloonContent, and balloonContentFooter (see **Placemark**).
		• 'cluster#balloonTwoColumns' — Two-column layout. The left column lists the names of placemarks (the clusterCaption field of the placemark or balloonContentHeader, if the first field is not defined). The right column contains information about the geo object. Layout options:
		  • balloonLeftColumnWidth — The width of the column with the list of objects in the cluster balloon, in pixels. Default value: 125
		• 'cluster#balloonCarousel' — Information about the geo object is positioned in the center. The buttons to navigate to the previous and next geo object are positioned on each side. The lower part of the balloon has a quick navigation menu. Layout options:
		  • balloonCycling — Cycle the list when navigating with the side arrows. Default value: true
		  • balloonPagerSize — Number of navigational items in the lower panel. The default value depends on the type of device. For mobile phones it is 4, and for tablets and PCs — 9.
		  • balloonPagerType — Type of lower panel for navigation. Takes the values 'numeric' or 'marker'.
		  • numeric — Display the number of a geo object in the list.
		  • marker — Display markers without numbers. Recommended for use when the number of items in the cluster is less than or equal to the value

Parameter	Default value	Description
		of options.balloonPagerSize. Default value: numeric • balloonPagerVisible — Whether to display the navigation panel. Default value: true • 'cluster#balloonAccordion' — Information about geo objects in list format. Each list item is a small icon and the name of the placemark (the clusterCaption field of the placemark or balloonContentHeader, if the first field is not defined). The color of the icon corresponds to the color of the placemark. The color of the placemark is defined by the "iconColor" **Placemark** option. The icons might not always appear in Internet Explorer 8 and below. When a list item is clicked, information about the geo object is expanded beneath it. Layout options: • balloonAccordionShowIcons — Option that determines whether to display the geo object's icon in the 'cluster#balloonAccordionItemTitle' layout. This option is not used in Internet Explorer 8 and below. Default value: true
options.balloonContentLayoutHeight	—	Type: Number The height of the layout of the content of the balloon cluster. The balloon cluster option balloonMaxHeight is not set by default, because all standard layouts have fixed dimensions. The standard value depends on the layout. • 'cluster#balloonTwoColumns' — 210 pixels • 'cluster#balloonCarousel' — 177 pixels • 'cluster#balloonAccordion' — 283 pixels
options.balloonContentLayoutWidth	—	Type: Number

Parameter	Default value	Description
		The width of the layout of the content of the balloon cluster. The balloon cluster option balloonMaxWidth is not set by default, because all standard layouts have fixed dimensions. The standard value depends on the layout. • 'cluster#balloonTwoColumns' — 475 pixels • 'cluster#balloonCarousel' — 308 pixels • 'cluster#balloonAccordion' — 305 pixels The option is not used in the panel mode.
options.balloonItemContentLayout	—	Type: **ILayout**|String The layout containing the information about geo object. The standard value depends on the layout. • Displayed to the right of the list in the 'cluster#balloonTwoColumns' layout. The standard value of 'cluster#balloonTwoColumnsItemContent' • Displayed in the center in the 'cluster#balloonCarousel' layout. The standard value of 'cluster#balloonCarouselItemContent'. • Displayed after the list item is clicked in the 'cluster#balloonAccordion' layout. The standard value of 'cluster#balloonAccordionItemContent'. **The set of fields received by this layout differs from the parent and matches the fields that received by the standard layout of a geo object balloon. Additionally, ownerProperties, ownerOptions and ownerState fields have been added to access the cluster data.**
options.balloonPanelContentLayout	null	Type: Function|String Layout of the cluster balloon in the "panel" mode. You can pass the object's constructor with the **ILayout** interface. The available values are the same as in the 'balloonContentLayout' options. If null, then the value of the 'balloonContentLayout' option is used.
options.cursor	'pointer'	Type: String Cursor over the cluster marker.
options.disableClickZoom	false	Type: Boolean

Parameter	Default value	Description
		Flag that prohibits zooming in on the map when the cluster is clicked.
options.hideIconOnBalloonOpen	true	Type: Boolean Hide the icon when opening the balloon.
options.iconColor	—	Type: String The color of the cluster icon. This option is used for standard icons in browsers that support SVG.
options.iconContentLayout	'cluster#iconContent'	Type: Function|String Layout for cluster marker contents. (Type: constructor for an object with the **ILayout** interface or the layout key). If you do not want to display the content of the placemark, set the option value to null.
options.iconLayout	'cluster#icon'	Type: Function|String Layout of the cluster placemark (Type: constructor for an object with the **ILayout** interface or the layout key).
options.icons	—	Type: Object[] Array describing the icon for standard implementation of the cluster. An icon description is made up of an object with the following fields: • href — Link to the image. • size — Array of two numbers representing the icon size in pixels. • offset — Offset of the icon relative to the anchor point of the object. • shape — Optional field. An object that implements the **IShape** interface or a JSON description of the geometry. Allows to specify the description of the icon geometry. If the parameter is omitted, a rectangular area around the icon will be considered as an active area for events (mouse over, mouse click).
options.iconShape	—	Type: **IGeometryJson**|null The hotspot shape of the cluster. Specified as a JSON description of the pixel geometry of the icon. Use this option when creating your HTML layouts. The coordinates of the figure geometry are counted from the anchor point.
options.interactivityModel	'default#geoObject'	Type: String

Parameter	Default value	Description
		Cluster interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.numbers	[10, 100]	Type: Number[] Array describing the boundary values for cluster dimensions. The number of icons described in the "icons" options must be 1 more than the number in this array.
options.openBalloonOnClick	true	Type: Boolean Option that prevents opening the balloon when clicking on a cluster. By default, opening the balloon is allowed.
options.openEmptyHint	false	Type: Boolean Enables displaying empty popup hints. Empty cluster hints are not shown by default.
options.openHintOnHover	true	Type: Boolean Option that allows you to forbid displaying the popup hint when the cluster is pointed at. By default, showing hints is allowed.
options.zIndexHover	—	Type: Number The zIndex value, which is set to the cluster placemark on mouse hovering.

* Mandatory parameter/option.

Examples:

1.

```
    // Variable with a description of two types of cluster icons.
      var clusterIcons = [
      {
        href: 'small.png',
        size: [40, 40],
      // Offset so the image center matches the cluster center.
      offset: [-20, -20]
      },
      {
        href: 'big.png',
        size: [60, 60],
        offset: [-30, -30],
        // You can define the shape of the hotspot area
        // of the placemark.
        // The rectangular area is used by default.
        shape: {
           type: 'Circle',
           coordinates: [0, 0],
         radius: 30
      }
    }],
    // For a cluster size up to 100, the image small.jpg will be used.
    // For a cluster size over 100, big.png will be used.
```

```
     clusterNumbers = [100],
     // Making a layout for cluster icon contents,
     // in which numbers will be white.
     MyIconContentLayout = ymaps.templateLayoutFactory.createClass(
       '&lt;div style="color: #FFFFFF; font-weight: bold;"&gt;
{{ properties.geoObjects.length }}&lt;/div&gt;'),
       var clusterer = new ymaps.Clusterer({
        // If options for the cluster are set via the clusterer,
        // they must be specified with the "cluster" prefix.
        clusterIcons: clusterIcons,
     clusterNumbers: clusterNumbers,
clusterIconContentLayout: MyIconContentLayout
});
```

2.

```
       // Creating a clusterer with a carousel layout
       var clusterer = new ymaps.Clusterer({
        clusterDisableClickZoom: true,
        // Using the "carousel" layout
        clusterBalloonContentLayout: "cluster#balloonCarousel",
        // Prohibiting cycling the list for paginated navigation
        clusterBalloonCycling: false,
        // Configuring the appearance of the navigation panel. Markers will be
the items in the navigation panel.
        clusterBalloonPagerType: "marker",
     // Number of items in the navigation panel
clusterBalloonPagerSize: 6
});
```

3.

```
// Creating a clusterer with an accordian layout
var clusterer = new ymaps.Clusterer({
        clusterDisableClickZoom: true,
        // Using the "accordian' layout
       clusterBalloonContentLayout: "cluster#balloonAccordion"
    });
```

4.

```
// Creating a clusterer with an arbitrary HTML layout of the cluster icon.
var clusterer = new ymaps.Clusterer({
        // Defining the cluster placemark layout.
        clusterIconLayout: ymaps.templateLayoutFactory.createClass('<div
class="clusterIcon">{{ properties.geoObjects.length }}</div>'),
        // Redefining the active area of the placemark to make it clickable.
        clusterIconShape: {
            type: 'Rectangle',
            coordinates: [[0, 0], [20, 20]]
        }
    });
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**IDataManager**	Geo object data.

Name	Type	Description
		Inherited from **IGeoObject**.
state	**data.Manager**	Cluster state. Defined by the following fields: • activeObject — Reference to the cluster's active object. The active object is the one that is currently selected in the cluster balloon.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Name	Description
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Name	Description
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Returns the geographical coordinates of the rectangular area that includes all the cluster elements.
getGeoObjects()	**IGeoObject**[]	The method is a simplified call of cluster.properties.get('geoObjects');
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
onAddToMap(map)		Function that is called when adding an element to the map. To perform additional actions when adding the object to the map, redefine this function. Inherited from **collection.Item**.
onRemoveFromMap(oldMap)		Function that is called when deleting an element from the map. To perform additional actions when deleting the object from the map, redefine this function. Inherited from **collection.Item**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Fields details

state

```
{data.Manager} state
```

Cluster state. Defined by the following fields:

- activeObject — Reference to the cluster's active object. The active object is the one that is currently selected in the cluster balloon.

Example:

```
var geoObjects = cluster.properties.get('geoObjects');
// When opening the cluster's balloon, the third object in the list will be
selected.
cluster.state.set('activeObject', geoObjects[2]);
```

Methods details

getBounds

```
{Number[][]|null} getBounds()
```

Returns the geographical coordinates of the rectangular area that includes all the cluster elements.

getGeoObjects

```
{IGeoObject[]} getGeoObjects()
```

The method is a simplified call of cluster.properties.get('geoObjects');

Returns an array of geo objects that make up the cluster.

collection

collection.Item

Extends **IChildOnMap**, **ICustomizable**, **IEventEmitter**, **IParentOnMap**.

Base class for an item in a collection of map objects.

Constructor | Fields | Events | Methods

Constructor

```
collection.Item([options])
```

Parameters:

Parameter	Default value	Description
options	—	Type: Object Object options.

Example:

```
// Example of implementing a custom control based on inheritance from
collection.Item.
// The control displays the name of the object that is located in the center of
the map.
var map = new ymaps.Map('map', {
        center: [55.819543, 37.611619],
        zoom: 6
    }),
// Creating a custom class.
    CustomControl = function (options) {
        CustomControl.superclass.constructor.call(this, options);
    };
// And inheriting from collection.Item.
ymaps.util.defineClass(CustomControl, ymaps.collection.Item, {
    onAddToMap: function (map) {
        CustomControl.superclass.onAddToMap.call(this, map);
        // Creating an HTML element with text.
        this.getParent().getChildElement(this).then(this._onChildElementGet,
this);
    },

    onRemoveFromMap: function (oldMap) {
        CustomControl.superclass.onRemoveFromMap.call(this, oldMap);
    },

    _onChildElementGet: function(parentElementContainer) {
        // You can create a DOM representation for the control here
        // and add it as a child element in parentElementContaner.
        // ...
    }
});

var customControl = new CustomControl();
map.controls.add(customControl, {top: 10, left: 10});
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Methods

Name	Returns	Description
getMap()	Map	Returns the map that the collection item belongs to.
getParent()	IParentOnMap	Returns parent object.
onAddToMap(map)		Function that is called when adding an element to the map. To perform additional actions when adding the object to the map, redefine this function.
onRemoveFromMap(oldMap)		Function that is called when deleting an element from the map. To perform additional actions when deleting the object from the map, redefine this function.
setParent(parent)	collection.Item	Sets the parent for the selected item in a collection.

Methods details

getMap

```
{Map} getMap()
```

Returns the map that the collection item belongs to.

getParent

```
{IParentOnMap} getParent()
```

Returns parent object.

onAddToMap

```
{} onAddToMap(map)
```

Function that is called when adding an element to the map. To perform additional actions when adding the object to the map, redefine this function.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** The map that the object has been added to.

* Mandatory parameter/option.

onRemoveFromMap

```
{} onRemoveFromMap(oldMap)
```

Function that is called when deleting an element from the map. To perform additional actions when deleting the object from the map, redefine this function.

Parameters:

Parameter	Default value	Description
oldMap*	—	Type: **Map** The map that the object has been deleted from.

* Mandatory parameter/option.

setParent

```
{collection.Item} setParent(parent)
```

Sets the parent for the selected item in a collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
parent*	—	Type: **IParentOnMap** Parent object.

* Mandatory parameter/option.

Collection

Extends **ICollection**, **collection.Item**.

Basic implementation of an object collection on the map.

Constructor | Fields | Events | Methods

Constructor

```
Collection([options])
```

Parameters:

Parameter	Default value	Description
options	—	Type: Object Collection options.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
add	A child object was added. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • child — Child element that was added.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
remove	A child object was deleted. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • child — Child element that was deleted.

Methods

Name	Returns	Description
add(child)	**Collection**	Adds an item to the collection.
each(callback[, context])	**Collection**	Calls a handler function for all the items in the collection.
filter(filterFunction)	Object[]	Calls a filter function for all the items in the collection. When the filter returns a non-zero value, the collection item goes in the final array.
get(index)	Object	Returns a collection item, or null if the number is outside of the range for the collection's numbers.
getAll()	Object[]	Returns an array with all the collection items.
getIterator()	**IIterator**	Returns an iterator for selecting collection items.
getLength()	Number	Returns the number of items in the collection.
getMap()	**Map**	Returns the map that the collection item belongs to. Inherited from **collection.Item**.
getParent()	**IParentOnMap**	Returns parent object. Inherited from **collection.Item**.
indexOf(childToFind)	Number	Returns the sequential number of the object in the collection, or -1 if the object was not found.
onAddToMap(map)		Function that is called when adding an element to the map. To perform additional actions when adding the object to the map, redefine this function. Inherited from **collection.Item**.
onRemoveFromMap(oldMap)		Function that is called when deleting an element from the map. To perform additional actions when deleting the object from the map, redefine this function. Inherited from **collection.Item**.
remove(child)	**Collection**	Deletes an item from a collection.
removeAll()	**Collection**	Deletes all the items from a collection.
setParent(parent)	**collection.Item**	Sets the parent for the selected item in a collection. Inherited from **collection.Item**.

Events details

add

A child object was added. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- child — Child element that was added.

remove

A child object was deleted. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- child — Child element that was deleted.

Methods details

add

```
{Collection} add(child)
```

Adds an item to the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
child*	—	Type: **collection.Item** Element to add.

* Mandatory parameter/option.

each

```
{Collection} each(callback[, context])
```

Calls a handler function for all the items in the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function. Receives a collection item as input. If the function returns the value "false," processing stops.
context	—	Type: Object Context of the invoked function.

* Mandatory parameter/option.

filter

```
{Object[]} filter(filterFunction)
```

Calls a filter function for all the items in the collection. When the filter returns a non-zero value, the collection item goes in the final array.

Returns an array of items that were selected.

Parameters:

Parameter	Default value	Description
filterFunction*	—	Type: Function Function that works as a filter for the collection's objects. It takes an item from the collection as the first parameter. It should return a boolean value.

* Mandatory parameter/option.

get

```
{Object} get(index)
```

Returns a collection item, or null if the number is outside of the range for the collection's numbers.

Parameters:

Parameter	Default value	Description
index*	—	Type: Number The sequential number of the item in the collection.

* Mandatory parameter/option.

getAll

```
{Object[]} getAll()
```

Returns an array with all the collection items.

getIterator

```
{IIterator} getIterator()
```

Returns an iterator for selecting collection items.

getLength

```
{Number} getLength()
```

Returns the number of items in the collection.

indexOf

```
{Number} indexOf(childToFind)
```

Returns the sequential number of the object in the collection, or -1 if the object was not found.

Parameters:

Parameter	Default value	Description
childToFind*	—	Type: Object

Parameter	Default value	Description
		The required object.

* Mandatory parameter/option.

remove

```
{Collection} remove(child)
```

Deletes an item from a collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
child*	—	Type: **collection.Item** Item to delete.

* Mandatory parameter/option.

removeAll

```
{Collection} removeAll()
```

Deletes all the items from a collection.

Returns self-reference.

control

control.Button

Extends **ICustomizable**, **ISelectableControl**.

The "Button" control. The standard button layout changes its appearance depending on the size of the map. If the map is wide enough, the "image + text" version of the button is shown. If the map is medium in size, the "text" version of the button is shown. If the map is small, only the icon is shown in the button layout. If a button has no icon, then only text will be displayed in all states, and vice versa.

Constructor | Fields | Events | Methods

Constructor

```
control.Button([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object|String Parameters of a button or string — button contents in HTML format.
parameters.data	—	Type: Object Button data.
parameters.data.content	—	Type: String Button contents in HTML format.
parameters.data.image	—	Type: String URL of the button icon. The standard layout of a button is designed for the icon size 16x16 pixels.
parameters.data.title	—	Type: String Text of the popup hint that appears when the mouse cursor hovers over the button.
parameters.options	—	Type: Object Button options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the button registers its size in the map margins manager **map.margin.Manager**.
parameters.options.float	"right"	Type: String

Parameter	Default value	Description
		The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
parameters.options.floatIndex	0	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
parameters.options.layout	—	Type: **ISelectableControlLayout**|String Constructor of the control layout which implements the **ISelectableControlLayout** interface or the layout key in the **layout.storage**. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager control.Button.options. • data — Control data manager **control.Button.data**. • state — Control state manager **control.Button.state**. The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for **control.Button.state** depending on the commands received.
parameters.options.maxWidth	90	Type: Number|Number[] The maximum width of the button in different states. If a number is specified, it is assumed that the button has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width of the button in different states, from the lesser to the greater. The number of states is set in the instance of the class **control.Manager**, which is usually a field of **Map.controls**, via the "states" option. By default, the controls have three states ['small', 'medium', 'large'].
parameters.options.position	—	Type: Object

Parameter	Default value	Description
		Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.selectOnClick	true	Type: Boolean Options describing button behavior. • If true, the button becomes "pressed" after the click and changes its appearance and the value of the **control.Button.state** field is changed to "selected". • false — The button's appearance and state does not change after clicking it.
parameters.options.size	'auto'	Type: String Defines the appearance of the standard button layout. Takes the following values: • 'auto' — The default button layout is changed automatically depending on the dimensions of the map and the number of added controls. • 'small' — The button layout displays the icon, regardless of the map size. • 'medium' — The button layout displays only text, regardless of the map size. • 'large' — The button layout always displays both the icon and text, regardless of the map size.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object

Parameter	Default value	Description
		Object describing the state of the button.
parameters.state.enabled	true	Type: Boolean Flags if the button is active.
parameters.state.selected	false	Type: Boolean Flags if the button is pressed.

Examples:

1.

```
// Example 1.
// Creating a button and adding it to the map.
var button = new ymaps.control.Button({
         data: {
             // Setting an icon for the button.
             image: 'images/button.jpg',
             // Text on the icon.
             content: 'Save',
             // Text for the popup hint.
             title: 'Click to save the route'
         },
         options: {
             // Setting up the button options.
             selectOnClick: false,
             // The button will have three states - icon, text, and icon + text.
             // So we'll define three values for the button width for all
states.
             maxWidth: [30, 100, 150]
         }});
map.controls.add(button, { float: 'right', floatIndex: 100 });
```

2.

```
// Example 2.
// Creating buttons with a custom layout
var button = new ymaps.control.Button({
        data: {
            content: 'Red button',
            title: 'Press the button'
        },
        options: {
            layout: ymaps.templateLayoutFactory.createClass(
               // If the button is not pressed, the css style 'myButton' will be
applied
                // If the button is pressed, the css style 'myButton' and
'myButtonSelected' will be applied.
                "<div class='myButton {% if state.selected %}myButtonSelected{%
endif %}' title='{{ data.title }}'>" +
                "{{ data.content }}" +
                "</div>"
            ),
            // In order to correctly position all other controls horizontally,
            // you should specify the maximum width of the layout.
            maxWidth: 150
        }});
map.controls.add(button, { float: 'left', floatIndex: 0 });

// You can define the positioning relative to the edges of the map. In this
case,
// the value of the maxWidth option does not affect
//  the positioning of controls.
map.controls.add(button, { float: 'none', position: {left: '5px', top: '5px'} });
```

Fields

Name	Type	Description
data	data.Manager	Button data. Names of fields that are available via the data.Manager.get method: • image — Button icon, if available. • content — Button content in HTML format. • title — Text of the popup hint that appears when the mouse cursor hovers over the button.
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from IControl.
press		The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the Event class.
state	data.Manager	Button state. Names of fields that are available via the data.Manager.get method: • selected — Flag for whether the button is selected. • enabled — Flag for whether the button is active. • size — The size that is currently set for the button.

Events

Name	Description
click	Clicking the button. Instance of the Event class.
deselect	The control is not selected. Inherited from ISelectableControl.
disable	The control is unavailable. Inherited from ISelectableControl.
enable	The control is available. Inherited from ISelectableControl.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
select	The control is selected. Inherited from ISelectableControl.

Methods

Name	Returns	Description
deselect()		Cancels selection of the control (turns it off). Inherited from ISelectableControl.
disable()		Makes the control unavailable (user actions are not allowed). Inherited from ISelectableControl.
enable()		Makes the control available (user actions are allowed). Inherited from ISelectableControl.
getMap()	Map	Returns reference to the map.
getParent()	IControlParent|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IControl.
isEnabled()	Boolean	Returns true if the control is available, or false if it is unavailable. Inherited from ISelectableControl.
isSelected()	Boolean	Returns true if the control is selected, or false if it is not selected. Inherited from ISelectableControl.
select()		Selects (turns on) the control. Inherited from ISelectableControl.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IControl.

Fields details

data

```
{data.Manager} data
```

Button data. Names of fields that are available via the data.Manager.get method:

- image — Button icon, if available.
- content — Button content in HTML format.
- title — Text of the popup hint that appears when the mouse cursor hovers over the button.

press

```
press
```

The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the Event class.

state

```
{data.Manager} state
```

Button state. Names of fields that are available via the data.Manager.get method:

- selected — Flag for whether the button is selected.
- enabled — Flag for whether the button is active.
- size — The size that is currently set for the button.

Example:

```
var button = new ymaps.control.Button('Edit');
// Setting the button state to "selected" -
// similar to calling button.select();
button.state.set('selected', true);
```

Events details

click

Clicking the button. Instance of the **Event** class.

Methods details

getMap

```
{Map} getMap()
```

Returns reference to the map.

control.FullscreenControl

Extends **control.Button**.

The "Fullscreen mode" control. You can set the z-index property of the map container for full-screen mode using the Map.options.fullscreenZIndex option. Key for the control in **control.storage** — "fullscreenControl".

Constructor | Fields | Events | Methods

Constructor

```
control.FullscreenControl([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.
parameters.data	—	Type: Object Object describing the data of a control.
parameters.data.title	—	Type: String Text of the popup hint that appears when the mouse cursor hovers over the button.

Parameter	Default value	Description
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.float	"right"	Type: String The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map.
parameters.options.floatIndex	300	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none". See also the description of the position option.
parameters.options.layout	—	Type: **ISelectableControlLayout**|String Constructor of the control layout which implements the **ISelectableControlLayout** interface or the layout key in the **layout.storage**. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Options manager for the control.FullscreenControl.options. • data — Data manager for the control.FullscreenControl.data control. • state — Control state manager **control.FullscreenControl.state**. The layout's outward appearance changes based on the control's data, state and options. The manager element, in turn, reacts to the layout's interface events.
parameters.options.maxWidth	28	Type: Number|Number[]

Parameter	Default value	Description
		The maximum width of the control in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class **control.Manager**, which is usually a field of **Map.controls**, via the "states" option. By default, the controls have three states ['small', 'medium', 'large']. By default, the control does not change its size and always looks like a button with an icon.
parameters.options.position	—	Type: Object Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object Object describing the state of a control.
parameters.state.enabled	true	Type: Boolean Flags if the button is active.
parameters.state.selected	false	Type: Boolean Flags if the button is pressed.

Example:

```
// Adding the control to the map and immediately switching it
// to the full-screen mode.
var fullscreenControl = new ymaps.control.FullscreenControl();
myMap.controls.add(fullscreenControl);
fullscreenControl.enterFullscreen();
```

Fields

Name	Type	Description
data	**data.Manager**	Button data. Names of fields that are available via the data.Manager.get method: • image — Button icon, if available. • content — Button content in HTML format. • title — Text of the popup hint that appears when the mouse cursor hovers over the button. Inherited from **control.Button**.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **IControl**.
press		The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the Event class. Inherited from **control.Button**.
state	**data.Manager**	State of the control. Names of fields that are available via the data.Manager.get method: • fullscreen — Flag for whether the map is in full-screen mode.

Events

Name	Description
click	Clicking the button. Instance of the **Event** class. Inherited from **control.Button**.
deselect	The control is not selected. Inherited from **ISelectableControl**.
disable	The control is unavailable. Inherited from **ISelectableControl**.
enable	The control is available. Inherited from **ISelectableControl**.
fullscreenenter	The map switched to fullscreen mode. Instance of the **Event** class.
fullscreenexit	The map exited full-screen mode. Instance of the **Event** class.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed.

Name	Description
	Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
select	The control is selected. Inherited from **ISelectableControl**.

Methods

Name	Returns	Description
deselect()		Cancels selection of the control (turns it off). Inherited from **ISelectableControl**.
disable()		Makes the control unavailable (user actions are not allowed). Inherited from **ISelectableControl**.
enable()		Makes the control available (user actions are allowed). Inherited from **ISelectableControl**.
enterFullscreen()		Allows you to switch the map to full-screen mode.
exitFullscreen()		Allows you to take the map out of full-screen mode.
getMap()	**Map**	Returns reference to the map. Inherited from **control.Button**.
getParent()	**IControlParent**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IControl**.
isEnabled()	Boolean	Returns true if the control is available, or false if it is unavailable. Inherited from **ISelectableControl**.
isSelected()	Boolean	Returns true if the control is selected, or false if it is not selected. Inherited from **ISelectableControl**.
select()		Selects (turns on) the control. Inherited from **ISelectableControl**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IControl**.

Fields details

state

```
{data.Manager} state
```

State of the control. Names of fields that are available via the data.Manager.get method:

- fullscreen — Flag for whether the map is in full-screen mode.

Events details

fullscreenenter

The map switched to fullscreen mode. Instance of the **Event** class.

fullscreenexit

The map exited full-screen mode. Instance of the **Event** class.

Methods details

enterFullscreen

```
{} enterFullscreen()
```

Allows you to switch the map to full-screen mode.

exitFullscreen

```
{} exitFullscreen()
```

Allows you to take the map out of full-screen mode.

control.GeolocationControl

Extends **control.Button**.

The "geolocation" control. Allows you to render the user's position on the map. The key of the control in the storage. **control.storage** — "geolocationControl".

See geolocation

Constructor | Fields | Events | Methods

Constructor

```
control.GeolocationControl([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.
parameters.data	—	Type: Object Object describing the data of a control.

Parameter	Default value	Description
parameters.data.image	'geolocation'	Type: String URL of the button icon.
parameters.data.title	—	Type: String Text of the popup hint that appears when the mouse cursor hovers over the button.
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.float	"right"	Type: String The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
parameters.options.floatIndex	300	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
parameters.options.maxWidth	28	Type: Number|Number[] The maximum width of the control in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class **control.Manager**, which is usually a field of **Map.controls**, via the "states" option. By default, the control does not change its size and always looks like a button with an icon.
parameters.options.noPlacemark	false	Type: Boolean When set to true, the location marker is not shown on the map, and automatic shifting of the map center and scaling does not occur.

Parameter	Default value	Description
parameters.options.position	—	Type: Object Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object Object describing the state of a control.
options.useMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager** when centering the map.

Example:

```
// Adding the control with a custom geolocation placemark on the map.
var geolocationControl = new ymaps.control.GeolocationControl({
    options: {noPlacemark: true}
});
geolocationControl.events.add('locationchange', function (event) {
    var position = event.get('position'),
    // When creating a placemark, you can set any appearance for it.
        locationPlacemark = new ymaps.Placemark(position);

    myMap.geoObjects.add(locationPlacemark);
   // Setting the new map center to the user's current location.
    myMap.panTo(position);
});
myMap.controls.add(geolocationControl);
```

Fields

Name	Type	Description
data	data.Manager	Button data. Names of fields that are available via the data.Manager.get method: • image — Button icon, if available. • content — Button content in HTML format. • title — Text of the popup hint that appears when the mouse cursor hovers over the button. Inherited from control.Button.
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from IControl.
press		The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the Event class. Inherited from control.Button.
state	data.Manager	Button state. Names of fields that are available via the data.Manager.get method: • selected — Flag for whether the button is selected. • enabled — Flag for whether the button is active. • size — The size that is currently set for the button. Inherited from control.Button.

Events

Name	Description
click	Clicking the button. Instance of the Event class. Inherited from control.Button.
deselect	The control is not selected. Inherited from ISelectableControl.
disable	The control is unavailable. Inherited from ISelectableControl.
enable	The control is available. Inherited from ISelectableControl.
locationchange	The event of determining the user's position. The list of event fields that are available via the Event.get: • position — The user's position, in geo coordinates. • geoObjects — An instance of the GeoObjectCollection class with the object that represents the user's current location.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed.

Name	Description
	Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
select	The control is selected. Inherited from **ISelectableControl**.

Methods

Name	Returns	Description
deselect()		Cancels selection of the control (turns it off). Inherited from **ISelectableControl**.
disable()		Makes the control unavailable (user actions are not allowed). Inherited from **ISelectableControl**.
enable()		Makes the control available (user actions are allowed). Inherited from **ISelectableControl**.
getMap()	**Map**	Returns reference to the map. Inherited from **control.Button**.
getParent()	**IControlParent**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IControl**.
isEnabled()	Boolean	Returns true if the control is available, or false if it is unavailable. Inherited from **ISelectableControl**.
isSelected()	Boolean	Returns true if the control is selected, or false if it is not selected. Inherited from **ISelectableControl**.
select()		Selects (turns on) the control. Inherited from **ISelectableControl**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IControl**.

Events details

locationchange

The event of determining the user's position. The list of event fields that are available via the **Event.get**:

- position — The user's position, in geo coordinates.
- geoObjects — An instance of the **GeoObjectCollection** class with the object that represents the user's current location.

control.ListBox

Extends **ICollection**, **IControl**, **ICustomizable**.

Class for creating a control in the form of a drop-down list. The standard drop-down list layout changes its appearance depending on the size of the map. If the map is wide enough, text can be displayed in the header of a drop-down list. If the map is small in size, only an icon is displayed. If a button has no icon, then only text will be displayed in all states, and vice versa.

Constructor | Fields | Events | Methods

Constructor

```
control.ListBox([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Drop-down list parameters.
parameters.data	—	Type: Object Data.
parameters.data.content	—	Type: String List title.
parameters.data.image	—	Type: String URL of the button icon. The standard layout of a button is designed for the icon size 16x16 pixels.
parameters.data.title	—	Type: String Text of the popup hint that appears when the mouse cursor hovers over the list.
parameters.items	—	Type: **IControl**[] Array of child elements of the list.
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean

Parameter	Default value	Description
		Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.collapseOnBlur	true	Type: Boolean This flag enables to collapse the list when the button loses focus, such as when a user clicks on the document.
parameters.options.expandOnClick	true	Type: Boolean Flag that allows automatically expanding/collapsing the list when clicked.
parameters.options.float	"right"	Type: String The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
parameters.options.floatIndex	0	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
parameters.options.layout	—	Type: Function|String Constructor of the control layout which implements the **ISelectableControlLayout** and **IGroupControlLayout** interfaces or the layout key in the **layout.storage**. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager control.ListBox.options. • data — Control data manager **control.ListBox.data**. • state — Control state manager **control.ListBox.state**. The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for **control.ListBox.state** depending on the commands received.
parameters.options.maxWidth	90	Type: Number|Number[]

Parameter	Default value	Description
		The maximum width of the listbox in different states. If a number is specified, it is assumed that the button has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width of the button in different states, from the lesser to the greater. The number of states is set in the instance of the class **control.Manager**, which is usually a field of **Map.controls**, via the "states" option. By default, the controls have three states ['small', 'medium', 'large'].
parameters.options.popupFloat	—	Type: String The side to align the manager element's popup to. Takes two values: "left" and "right". By default, it is defined by the "float" option.
parameters.options.position	—	Type: Object Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object State of the drop-down list.
parameters.state.expanded	false	Type: Boolean Flags if the list is expanded.

Examples:

1.

```
// Example 1.
// Handling a click on list items.
var cityList = new ymaps.control.ListBox({
        data: {
            content: 'Select a city'
        },
        items: [
            new ymaps.control.ListBoxItem('Moscow'),
            new ymaps.control.ListBoxItem('Novosibirsk'),
            new ymaps.control.ListBoxItem({options: {type: 'separator'}}),
            new ymaps.control.ListBoxItem('New York'),
        ]
    });
cityList.get(0).events.add('click', function () {
    map.setCenter([55.752736, 37.606815]);
});
cityList.get(1).events.add('click', function () {
    map.setCenter([55.026366, 82.907803]);
});
cityList.get(3).events.add('click', function () {
    map.setCenter([40.695537, -73.97552]);
});
map.controls.add(cityList, { floatIndex: 0 });
```

2.

```
// Example 2.
// Creating a custom list.
// This example uses jQuery, downloaded from http://yandex.st/jquery/1.6.4/
jquery.min.js

// By default, the drop-down list reacts to the "click" event and automatically
// changes its state to expanded or collapsed.
var MyListBoxLayout = ymaps.templateLayoutFactory.createClass(
        '<div id="my-listbox-header" >{{ data.title }}</div >' +
        // This item will serve as a container for the child list items.
       '<div id="my-list-box" style="display: {% if state.expanded %}block{% else
%}none{% endif %};" >' +
        '</div >', {

        build: function() {
            MyListBoxLayout.superclass.build.call(this);
            this.childContainerElement = $('#my-list-box').get(0);
            // Each time we rebuild, we will generate an event
            // that signals that the container for child elements has changed.
            // The event format is described in the IGroupControlLayout
interface.
            this.events.fire('childcontainerchange', {
                newChildContainerElement: this.childContainerElement,
                oldChildContainerElement: null
            });
        },

        // Redefining the method that requires the IGroupControlLayout
interface.
        getChildContainerElement: function () {
            return this.childContainerElement;
        }
    }),
    // Creating a list and displaying the created layout via the options.
    listBox = new ymaps.control.ListBox({options: {layout: MyListBoxLayout}});
```

3.

```
// Example 3.
// In this example, the ListBox control filters objects to show
// on the map (multi-select is supported).
// ObjectManager is used here for adding objects to the map.

// Creating a drop-down list with 5 items.
var listBoxItems = ['School', 'Pharmacy', 'Store', 'Hospital', 'Bar']
    .map(function(title) {
        return new ymaps.control.ListBoxItem({
            data: {
                content: title
            },
            state: {
                selected: true
            }
        });
    }),

// Now creating the drop-down list with 5 items.
    listBoxControl = new ymaps.control.ListBox({
        data: {
            content: 'Filter',
            title: 'Filter'
        },
        items: listBoxItems,
        state: {
            // Indicates that the list is expanded.
            expanded: true,
            filters: listBoxItems.reduce(function(filters, filter) {
                filters[filter.data.get('content')] = filter.isSelected();
                return filters;
            }, {})
        }
    });

map.controls.add(listBoxControl);

// Adding tracking to the indicator to check if a list item is selected.
listBoxControl.events.add(['select', 'deselect'], function(e) {
    var listBoxItem = e.get('target');
    var filters = ymaps.util.extend({}, listBoxControl.state.get('filters'));
    filters[listBoxItem.data.get('content')] = listBoxItem.isSelected();
    listBoxControl.state.set('filters', filters);
});

// Tracking changes to the control.ListBox.state field.
var filterMonitor = new ymaps.Monitor(listBoxControl.state);

filterMonitor.add('filters', function(filters) {
    // Applying the filter to ObjectManager.
    objectManager.setFilter(getFilterFunction(filters));
});

function getFilterFunction(categories){
    return function(obj){
        var content = obj.properties.balloonContent;
        return categories[content]
    }
}
```

Fields

Name	Type	Description
data	data.Manager	Drop-down list data. Names of fields that are available via the data.Manager.get method: • content — Title of the drop-down list. • title — Text of the popup hint that appears when the mouse cursor hovers over the list.
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from IControl.
state	data.Manager	State of the drop-down list. Names of fields that are available via the data.Manager.get method: • expanded — Flag for whether the list is expanded. • size — The size that is currently set for the list.

Events

Name	Description
add	A child object was added. Inherited from ICollection.
click	Clicking the list title. Instance of the Event class.
collapse	The list is closed. Instance of the Event class.
expand	The list is open. Instance of the Event class.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
press	The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the Event class.
remove	A child object was deleted. Inherited from ICollection.

Methods

Name	Returns	Description
add(object)	ICollection	Adds a child object to the collection. Inherited from ICollection.
collapse()	control.ListBox	Collapses the list.

Name	Returns	Description
expand()	control.ListBox	Expands the list.
getIterator()	IIterator	Returns iterator for the collection. Inherited from ICollection.
getMap()	Map	Returns reference to the map.
getParent()	IControlParent|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IControl.
isExpanded()	Boolean	Returns a flag for whether the control is in the expanded state.
remove(object)	ICollection	Removes a child object from the collection. Inherited from ICollection.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IControl.

Fields details

data

```
{data.Manager} data
```

Drop-down list data. Names of fields that are available via the data.Manager.get method:

- content — Title of the drop-down list.
- title — Text of the popup hint that appears when the mouse cursor hovers over the list.

Example:

```
// Adding a drop-down list to the map and changing its hint
// depending on whether the list is collapsed or expanded.

// Creating a group of event listeners.
var listBoxListener = listBox.events.group()
    .add('expand', function () {
        listBox.data.set('title', 'List is expanded');
    })
    .add('collapse', function () {
        listBox.data.set('title', 'List is collapsed');
    });
map.controls.add(listBox, {float: 'none', top: 10, left: 10});
// ...
map.controls.remove(listBox);
// After deleting the item from the map, we delete the listeners.
listBoxListener.removeAll();
```

state

```
{data.Manager} state
```

State of the drop-down list. Names of fields that are available via the data.Manager.get method:

- expanded — Flag for whether the list is expanded.
- size — The size that is currently set for the list.

Example:

```
// Creating and adding a list that is initially open.
var listBox = new ymaps.control.ListBox();
listBox.state.set('expanded', true);
map.controls.add(listBox);
```

Events details

click

Clicking the list title. Instance of the **Event** class.

collapse

The list is closed. Instance of the **Event** class.

expand

The list is open. Instance of the **Event** class.

press

The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the **Event** class.

Methods details

collapse

```
{control.ListBox} collapse()
```

Collapses the list.

Returns self-reference.

expand

```
{control.ListBox} expand()
```

Expands the list.

Returns self-reference.

getMap

```
{Map} getMap()
```

Returns reference to the map.

isExpanded

```
{Boolean} isExpanded()
```

Returns a flag for whether the control is in the expanded state.

control.ListBoxItem

Extends **ICustomizable**, **ISelectableControl**.

Drop-down list item.

Constructor | Fields | Events | Methods

Constructor

```
control.ListBoxItem([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object|String The parameters of the item or string — the HTML content of the item.
parameters.data	—	Type: Object Item data.
parameters.data.content	—	Type: String Item contents.
parameters.options	—	Type: Object Control options.
parameters.options.layout	'islands#listBoxItemLayout'	Type: Function|String Constructor of the control layout which implements the **ISelectableControlLayout** interface or the layout key in the **layout.storage**. This is the base layout, which selects which of the sub-layouts to display, depending on the value of the "type" option — options.selectableLayout or options.separatorLayout. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Options manager for the control.ListBoxItem.options. • data — Control data manager **control.ListBoxItem.data**. • state — Control state manager **control.ListBoxItem.state**. The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for **control.ListBoxItem.state** depending on the commands received.
parameters.options.selectableLayout	'islands#listBoxItemSelectableLayout'	Type: Function|String

Parameter	Default value	Description
		Constructor of the list item layout which implements the ISelectableControlLayout interface or the layout key in the layout.storage. Applies to items with the option type='item'. Option for standard implementation of the list item layout.
parameters.options.selectOnClick	true	Type: Boolean Flag that allows automatically selecting a list item when clicked. • If true, the list item is selected after the click and changes the value of the control.ListBoxItem.state field to 'selected'. • If false, the list item's appearance and state do not change after a click.
parameters.options.separatorLayout	'islands#listBoxItemSeparatorLayout'	Type: Function|String Constructor of the list item separator layout, which implements the IControlLayout interface or the layout key in layout.storage. Applies to items with the option type='separator'. Option for standard implementation of the list item layout.
parameters.options.type	'selectable'	Type: String Type of menu item. Depending on the value of this option, the list item layout instantiates one of the sub-layouts — options.selectableLayout or options.separatorLayout. Possible values: • 'selectable' — The list item is selected by a checkmark to the right of the content. • 'separator' — Separator.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object Object describing the state of the menu item.
parameters.state.selected	false	Type: Boolean Indicates whether the item is selected.

Fields

Name	Type	Description
data	data.Manager	Data for a list item. Names of fields that are available via the data.Manager.get method: • content — List item content in HTML format. • title — Pop-up hint text.
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.
state	data.Manager	State of a list item. Names of fields that are available via the data.Manager.get method: • selected — Indicates whether the list item is selected. • enabled — Indicates whether the list item is active.

Events

Name	Description
click	Selecting a list item.
deselect	The control is not selected. Inherited from ISelectableControl.
disable	The control is unavailable. Inherited from ISelectableControl.
enable	The control is available. Inherited from ISelectableControl.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
select	The control is selected. Inherited from ISelectableControl.

Methods

Name	Returns	Description
deselect()		Cancels selection of the control (turns it off). Inherited from ISelectableControl.

Name	Returns	Description
disable()		Makes the control unavailable (user actions are not allowed). Inherited from ISelectableControl.
enable()		Makes the control available (user actions are allowed). Inherited from ISelectableControl.
getMap()	Map	Returns reference to the map.
getParent()	IControlParent|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IControl.
isEnabled()	Boolean	Returns true if the control is available, or false if it is unavailable. Inherited from ISelectableControl.
isSelected()	Boolean	Returns true if the control is selected, or false if it is not selected. Inherited from ISelectableControl.
select()		Selects (turns on) the control. Inherited from ISelectableControl.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IControl.

Fields details

data

```
{data.Manager} data
```

Data for a list item. Names of fields that are available via the data.Manager.get method:

- content — List item content in HTML format.
- title — Pop-up hint text.

state

```
{data.Manager} state
```

State of a list item. Names of fields that are available via the data.Manager.get method:

- selected — Indicates whether the list item is selected.
- enabled — Indicates whether the list item is active.

Events details

click

Selecting a list item.

Methods details

getMap

```
{Map} getMap()
```

Returns reference to the map.

control.Manager

Manager for map controls.

Constructor | Fields | Methods

Constructor

```
control.Manager(map[, controls[, options]])
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Instance of the map.
controls	—	Type: String[]|**IControl**[] The controls which should be initially placed on the map.
options	—	Type: Object Manager options.
options.margin	10	Type: Number Distance between adjacent controls and the indent from the edges of the map. Specified in pixels.
options.pane	—	Type: **IPane** Container for controls.
options.states	['small', 'medium', 'large']	Type: String[] The array of sizes, from smallest to largest.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**event.Manager**	Event manager.
options	**option.Manager**	Manager options.

Name	Type	Description
state	data.Manager	Manager state. Names of fields that are available via the data.Manager.get method: • size — The state of the controls.

Methods

Name	Returns	Description
add(control[, options])	control.Manager	Adds a control to the manager.
each(callback, context)	control.Manager	Calls a handler function for all the controls.
get(index)	IControl|null	Returns the control, or null if no item has been found.
getChildElement(control)	vow.Promise	Returns the Promise object, which is confirmed by the HTML element that should hold the child element.
getContainer()	HTMLElement	Returns container where elements of the control will be added.
getMap()	Map	Returns reference to the map.
indexOf(childToFind)	Integer	Returns -1 if the control has not been found, or the index of the item in the manager.
remove(control)	control.Manager	Removing a control from the manager.

Fields details

events

```
{event.Manager} events
```

Event manager.

options

```
{option.Manager} options
```

Manager options.

state

```
{data.Manager} state
```

Manager state. Names of fields that are available via the data.Manager.get method:

- size — The state of the controls.

Methods details

add

```
{control.Manager} add(control[, options])
```

Adds a control to the manager.

Returns self-reference.

Parameters:

Parameter	Default value	Description
control*	—	Type: **IControl**|String Controls that are set by instances of classes that implement the **IControl** interface, or by keys. Acceptable key values: • "fullscreenControl" — Button for opening the map in full-screen mode, **control.FullscreenControl**. • "geolocationControl" — Button for defining the user location, **control.GeolocationControl**. • "routeEditor" — Button for enabling or disabling the behavior "route editor" **control.RouteEditor**. • "rulerControl" — Button for enabling or disabling the behavior "ruler" **control.RulerControl**. • "searchControl" — Search box for processing the user's search query **control.SearchControl**. • "trafficControl" — Traffic panel **control.TrafficControl**. • "typeSelector" — Panel for selecting the map type **control.TypeSelector**. • "zoomControl" — Zoom slider **control.ZoomControl**.

Parameter	Default value	Description
		Also, you can specify one of the predefined sets of controls using special keys: • "smallMapDefaultSet" — Basic set of controls, optimized for small maps and mobile phone screens. It includes the following controls: "zoomControl", "searchControl", "typeSelector", "geolocationControl" and "fullscreenControl". All controls in this set are minimized to buttons with icons. • "mediumMapDefaultSet" — Basic set of controls, optimized for medium-sized maps and tablet screens. In addition to the basic set of controls (see above), this set includes "rulerControl" and "trafficControl". • "largeMapDefaultSet" — Basic set of controls, optimized for large maps and desktops. The "routeEditor" control is added to the set and the "fullScreen" control is removed from the set, compared to the "mediumMapDefaultSet". • "default" — Default set of controls, an alias for "mediumMapDefaultSet".
options	—	Type: Object Control options.
options.float	"right"	Type: String The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map.
options.floatIndex	0	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
options.position	—	Type: Object Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".

Parameter	Default value	Description
options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map. Only works with float = none.
options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map. Only works with float = none.
options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map. Only works with float = none.
options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map. Only works with float = none.

* Mandatory parameter/option.

Example:

```
map.controls
    .add('zoomControl')
    .add('typeSelector');
```

each

```
{control.Manager} each(callback, context)
```

Calls a handler function for all the controls.

Returns self-reference.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function. Receives a collection item as input. If the function returns the value "false," processing stops.
context*	—	Type: Object Context of the invoked function.

* Mandatory parameter/option.

get

```
{IControl|null} get(index)
```

Returns the control, or null if no item has been found.

Parameters:

Parameter	Default value	Description
index*	—	Type: Number|String

Parameter	Default value	Description
		The index of the desired element or key.

* Mandatory parameter/option.

getChildElement

```
{vow.Promise} getChildElement(control)
```

Returns the Promise object, which is confirmed by the HTML element that should hold the child element.

Parameters:

Parameter	Default value	Description
control*	—	Type: **IControl** Control.

* Mandatory parameter/option.

getContainer

```
{HTMLElement} getContainer()
```

Returns container where elements of the control will be added.

getMap

```
{Map} getMap()
```

Returns reference to the map.

indexOf

```
{Integer} indexOf(childToFind)
```

Returns -1 if the control has not been found, or the index of the item in the manager.

Parameters:

Parameter	Default value	Description
childToFind*	—	Type: String|**IControl** Control or its key.

* Mandatory parameter/option.

remove

```
{control.Manager} remove(control)
```

Removing a control from the manager.

Returns self-reference.

Parameters:

Parameter	Default value	Description
control*	—	Type: **IControl**|String

Parameter	Default value	Description
		The deleted control or its key.

* Mandatory parameter/option.

control.RouteButton

Extends **IControl**, **ICustomizable**.

The Route Button control: button with route panel in the popup. The key of the control in the storage **control.storage** — "routeButtonControl".

Constructor | Fields | Events | Methods

Constructor

```
control.RouteButton([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.
parameters.lazy	true	Type: Boolean If set to true, all modules needed to work with routes will be loaded lazily when the user opens the panel the first time.
parameters.options	—	Type: Object Control options. Use the 'routePanel' prefix to configure underlying **IRoutePanel** options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.autofocus	true	Type: Boolean Specifies whether route panel must automatically gain focus when popup opens.
parameters.options.collapseOnBlur	true	Type: Boolean This flag enables to collapse the panel when the control loses focus.
parameters.options.float	"right"	Type: String

Parameter	Default value	Description
		The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged in a line, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
parameters.options.floatIndex	0	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
parameters.options.popupAnimate	true	Type: Boolean This flag indicates whether popup expansion and collapse should be animated.
parameters.options.popupFloat	'auto'	Type: Boolean Specifies which button edge the popup should stick to. Takes the following values: • 'auto' — The side depends on the size of the map and the size of the popup. • 'right' — The right side of the popup sticks to the right side of the control. • 'left' — The left side of the popup sticks to the left side of the control.
parameters.options.popupWidth	'210px'	Type: String Specifies CSS width of popup. Width is restricted by 176px below and 400px above.
parameters.options.position	—	Type: Object Object describing the position of a control.
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	10	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String

Parameter	Default value	Description
		Position relative to the right edge of the map.
parameters.options.position.top	108	Type: Number|String Position relative to the top edge of the map.
parameters.options.size	'auto'	Type: String Defines the appearance of the control. Takes the following values: • 'auto' — The layout is changed automatically depending on the dimensions of the map and the number of added controls. • 'small' — The layout displays the icon, regardless of the map size. • 'medium' — The layout displays only text, regardless of the map size. • 'large' — The layout always displays both the icon and text, regardless of the map size.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object Object describing the state of a control.
parameters.state.expanded	false	Type: Boolean Indicates whether the popup with the panel is expanded.

Examples:

1.

```
// Example 1.
// Creating a small zoom control and adding it to the map.
var routeButton = new ymaps.control.RouteButton({
    options: {
        size: "small"
    }
});
myMap.controls.add(routeButton);
```

2.

```
// Example 2.
// Add control to the left corner of the map and set default points of arrival
and departure.
myMap.controls.add('routeButtonControl', {
    size: "large",
    float: "left",
    floatIndex: 1000,
});
myMap.controls.get('routeButtonControl').routePanel.state.set({
    fromEnabled: false,
    from: "moscow",
    to: "saint petersburg",
    type: "auto"
});
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from IControl.
routePanel	IRoutePanel	

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Methods

Name	Returns	Description
getParent()	IControlParent|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IControl.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IControl.

Fields details

routePanel

```
{IRoutePanel} routePanel
```

control.RouteEditor

Extends **control.Button**.

The "Route editor" control. The key of the control in the storage. **control.storage** — "routeEditor".

Constructor | Fields | Events | Methods

Constructor

```
control.RouteEditor([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.
parameters.data	—	Type: Object Object describing the data of a control.
parameters.data.image	'routes'	Type: String URL of the button icon.
parameters.data.title	—	Type: String Text of the popup hint that appears when the mouse cursor hovers over the button.
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.float	"right"	Type: String The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.

Parameter	Default value	Description
parameters.options.floatIndex	100	Type: Number The priority of the control positioning. Element with the highest possible priority.
parameters.options.layout	—	Type: ISelectableControlLayout|String Constructor of the control layout which implements the ISelectableControlLayout interface or the layout key in the layout.storage. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Options manager for the control.FullscreenControl.options. • data — Data manager for the control.FullscreenControl.data control. • state — Control state manager control.FullscreenControl.state. The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to the interface events of the layout
parameters.options.maxWidth	28	Type: Number|Number[] The maximum width of the control in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class control.Manager, which is usually a field of Map.controls, via the "states" option. By default, the controls have three states ['small', 'medium', 'large']. By default, the control does not change its size and always looks like a button with an icon.
parameters.options.position	—	Type: Object Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	'auto'	Type: Number|String

Parameter	Default value	Description
		Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object Object describing the state of a control.

Example:

```
// Adding the control to the map.
map.controls.add('routeEditor');
```

Fields

Name	Type	Description
data	**data.Manager**	Button data. Names of fields that are available via the data.Manager.get method: • image — Button icon, if available. • content — Button content in HTML format. • title — Text of the popup hint that appears when the mouse cursor hovers over the button. Inherited from **control.Button**.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **IControl**.
press		The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the Event class. Inherited from **control.Button**.
state	**data.Manager**	Button state. Names of fields that are available via the data.Manager.get method: • selected — Flag for whether the button is selected. • enabled — Flag for whether the button is active. • size — The size that is currently set for the button. Inherited from **control.Button**.

Events

Name	Description
click	Clicking the button. Instance of the **Event** class. Inherited from **control.Button**.
deselect	The control is not selected. Inherited from **ISelectableControl**.
disable	The control is unavailable. Inherited from **ISelectableControl**.
enable	The control is available. Inherited from **ISelectableControl**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
select	The control is selected. Inherited from **ISelectableControl**.

Methods

Name	Returns	Description
deselect()		Cancels selection of the control (turns it off). Inherited from **ISelectableControl**.
disable()		Makes the control unavailable (user actions are not allowed). Inherited from **ISelectableControl**.
enable()		Makes the control available (user actions are allowed). Inherited from **ISelectableControl**.
getMap()	**Map**	Returns reference to the map. Inherited from **control.Button**.
getParent()	**IControlParent**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IControl**.
getRoute()	**router.Route**	Returns route.
isEnabled()	Boolean	Returns true if the control is available, or false if it is unavailable. Inherited from **ISelectableControl**.

Name	Returns	Description
isSelected()	Boolean	Returns true if the control is selected, or false if it is not selected. Inherited from ISelectableControl.
select()		Selects (turns on) the control. Inherited from ISelectableControl.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IControl.

Methods details

getRoute

```
{router.Route} getRoute()
```

Returns route.

control.RoutePanel

Extends IControl, ICustomizable.

The Route Panel control. The key of the control in the storage control.storage — "routePanelControl".

Constructor | Fields | Events | Methods

Constructor

```
control.RoutePanel([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.
parameters.options	—	Type: Object Control options. Use prefix 'routePanel' to configure underlying IRoutePanel options. 'routePanelAdjustMapMargin' option is set to true by default.
parameters.options.autofocus	true	Type: Boolean Whether panel must automatically gain focus after it was added to map.
parameters.options.float	"left"	Type: String

Parameter	Default value	Description
		The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
parameters.options.floatIndex	0	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
parameters.options.maxWidth	'210px'	Type: String Specifies CSS width of popup. Width is restricted by 176px below and 400px above.
parameters.options.position	—	Type: Object Object describing the position of a control.
parameters.options.position.bottom	—	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	—	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	—	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	—	Type: Number|String Position relative to the top edge of the map.
parameters.options.showHeader	false	Type: Boolean Whether to show header.
parameters.options.title	'Routes'	Type: String Title to show at the top of the panel. Visible only if showHeader option is true.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.

Parameter	Default value	Description
parameters.state	—	Type: Object Object describing the state of a control.

Examples:

1.

```
// Example 1.
// Adding a route panel control to the map.
myMap.controls.add('routePanelControl')
```

2.

```
// Example 2.
// Creating 300px-wide route panel with header with filled starting point.
myMap.controls.add('routePanelControl', {
    maxWidth: 300,
});
var routePanel = myMap.controls.get('routePanelControl').routePanel;
routePanel.options.set('adjustMapMargin', true);
routePanel.state.set({
    fromEnabled: false,
    from: "moscow",
    to: "saint petersburg",
    type: "auto"
});
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from IControl.
routePanel	IRoutePanel	Route panel.

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Methods

Name	Returns	Description
getParent()	IControlParent|null	Returns link to the parent object, or null if the parent element was not set.

Name	Returns	Description
		Inherited from **IControl**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IControl**.

Fields details

routePanel

```
{IRoutePanel} routePanel
```

Route panel.

control.RulerControl

Extends **control.Button**.

The "Ruler" control. The key of the control in the storage. **control.storage** — "rulerControl".

Constructor | Fields | Events | Methods

Constructor

```
control.RulerControl([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.
parameters.data	—	Type: Object Object describing the data of a control.
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.position	—	Type: Object Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	30	Type: Number|String Position relative to the bottom edge of the map.

Parameter	Default value	Description
parameters.options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	10	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.scaleLine	true	Type: Boolean Flags if the ruler control should be displayed to the right of the ruler button.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object Object describing the state of a control.

Fields

Name	Type	Description
data	**data.Manager**	Button data. Names of fields that are available via the data.Manager.get method: • image — Button icon, if available. • content — Button content in HTML format. • title — Text of the popup hint that appears when the mouse cursor hovers over the button. Inherited from **control.Button**.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **IControl**.
press		The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the Event class. Inherited from **control.Button**.
state	**data.Manager**	Button state. Names of fields that are available via the data.Manager.get method: • selected — Flag for whether the button is selected. • enabled — Flag for whether the button is active. • size — The size that is currently set for the button. Inherited from **control.Button**.

Events

Name	Description
click	Clicking the button. Instance of the **Event** class. Inherited from **control.Button**.
deselect	The control is not selected. Inherited from **ISelectableControl**.
disable	The control is unavailable. Inherited from **ISelectableControl**.
enable	The control is available. Inherited from **ISelectableControl**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
select	The control is selected. Inherited from **ISelectableControl**.

Methods

Name	Returns	Description
deselect()		Cancels selection of the control (turns it off). Inherited from **ISelectableControl**.
disable()		Makes the control unavailable (user actions are not allowed). Inherited from **ISelectableControl**.
enable()		Makes the control available (user actions are allowed). Inherited from **ISelectableControl**.
getMap()	**Map**	Returns reference to the map. Inherited from **control.Button**.
getParent()	**IControlParent**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IControl**.
isEnabled()	Boolean	Returns true if the control is available, or false if it is unavailable. Inherited from **ISelectableControl**.
isSelected()	Boolean	Returns true if the control is selected, or false if it is not selected.

Name	Returns	Description
		Inherited from **ISelectableControl**.
select()		Selects (turns on) the control. Inherited from **ISelectableControl**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IControl**.

control.SearchControl

Extends **IControl**, **ICustomizable**.

The "Search on map" control. Allows you to process a user's search query and display the result in the panel and on the map.

Each search result represents a two-line block on the panel of a control. To generate the block, the "name" and "description" fields from the geocoding result object are used.

Key for the control in **control.storage** — "searchControl".
Constructor | Fields | Events | Methods

Constructor

```
control.SearchControl([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.
parameters.data	—	Type: Object Object describing the data of a control.
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. For ranking, the objects located inside the specified area will receive higher priority.
parameters.options.fitMaxWidth	false	Type: Boolean

Parameter	Default value	Description
		Whether to stretch control by the maximum width.
parameters.options.float	"right"	Type: String The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
parameters.options.floatIndex	200	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
parameters.options.formLayout	'islands#searchControlFormLayout'	Type: **ILayout**|String Constructor of the form layout for searching in the default control layout or the key in storage **layout.storage**. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager control.SearchControl.options. • data — Control data manager control.SearchControl.data. • state — Control state manager **control.SearchControl.state**.
parameters.options.kind	'house'	Type: String Type of toponym (only for reverse geocoding). List of acceptable values: • house — House or building. • street — Street. • metro — Subway station. • district — City district. • locality — City, town, village, etc.
parameters.options.layout	'islands#searchControlLayout'	Type: **ISearchControlLayout**|String Control layout.

Parameter	Default value	Description
		The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager control.SearchControl.options. • data — Control data manager control.SearchControl.data. • state — Control state manager **control.SearchControl.state**. The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for **control.SearchControl.state** depending on the commands received.
parameters.options.maxWidth	[30, 72, 315]	Type: Number|Number[] The maximum width of the control in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class **control.Manager**, which is usually a field of **Map.controls**, via the "states" option. Default search control layout width for large state can be set in range from 280 to 660 pixels. By default, the controls have three states ['small', 'medium', 'large'].
parameters.options.noCentering	false	Type: Boolean If false, the map center is automatically placed so that the object is entirely visible; if true, the map center is not changed when showing the found object. If noCentering = true and noPlacemark = true are set, no visible changes will occur on the map when search results are clicked.
parameters.options.noPlacemark	false	Type: Boolean If false, a placemark with an open balloon is automatically added to the center of the found object; if true, it is not. If noCentering = true and noPlacemark = true are set, no visible changes will occur on the map when search results are clicked. This option doesn't work with the yandex#search provider.
parameters.options.noPopup	false	Type: Boolean

Parameter	Default value	Description
		If true, the drop-down list of results is not shown; if false, it is shown.
parameters.options.noSelect	false	Type: Boolean If false, the search result will be automatically displayed in case it is the only object found; if true, the result will not be selected.
parameters.options.noSuggestPanel	false	Type: Boolean If true, the panel with the search suggestions is not shown; if false, it is shown.
parameters.options.placeholderContent	—	Type: String Text of the hint, displayed in the input field of the control.
parameters.options.popupItemLayout	'islands#searchControlPopupItemLayout'	Type: **ILayout**|String Layout constructor for a search result in the drop-down list in the default control layout or the key in storage **layout.storage**. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager control.SearchControl.options. • data — Control data manager control.SearchControl.data. • state — Control state manager **control.SearchControl.state**.
parameters.options.popupLayout	'islands#searchControlPopupLayout'	Type: **ILayout**|String Layout constructor for the drop-down list with search results in the default control layout or the key in storage **layout.storage**. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager control.SearchControl.options. • data — Control data manager control.SearchControl.data. • state — Control state manager **control.SearchControl.state**.
parameters.options.position	—	Type: Object

Parameter	Default value	Description
		Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.provider	'yandex#map'	Type: **IGeocodeProvider**|String Geocoding provider. One of the standard providers can be used: • 'yandex#map' — Search for toponyms on the map. • 'yandex#search' — Search for toponyms and businesses. When searching for businesses, the following options don't work: "noPlacemark", "noCentering", "noSelect", "strictBounds", "kind". Moreover, if the "results" option is not specified for the control, the map, by default, cannot show more than 20 objects simultaneously.
parameters.options.searchCoordOrder	'latlong'	Type: String Determines how to interpret the coordinates in the request. By default, coordinates will be processed as latitude-longitude. This option doesn't work with the yandex#search provider.
parameters.options.size	'auto'	Type: String

Parameter	Default value	Description
		Defines the appearance of the control. Takes the following values: • 'small' — Always show the button with the icon. • 'medium' — Always show the button with text. • 'large' — Always show a full search form. • 'auto' — Automatically select the control size, depending on the amount of free space in the toolbar.
parameters.options.strictBounds	—	Type: Boolean Search only inside the area defined by the "boundedBy" option. Objects that are outside of the specified area will not be in the output.
parameters.options.suppressYandexSearch	false	Type: Boolean Whether to hide the message offering to search on the Yandex portal if results were not found.
parameters.options.useMapBounds	—	Type: Boolean Flag for taking into account the boundaries of the visible map area when searching. If true, the visible area that is calculated has higher priority than the area defined by boundedBy.
parameters.options.zoomMargin	0	Type: Number Represents the margins from the boundaries of the visible area of the map when displaying search results. The option works only if the value of noCentering is false. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" option is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager **map.margin.Manager**.
parameters.state	—	Type: Object Object describing the state of a control.
options.useMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager** when showing search results on the map.

Examples:

1.

```
// Example 1.
// Creating a control and adding it to the map.

var searchControl = new ymaps.control.SearchControl({
     options: {
         float: 'right',
        floatIndex: 100,
         noPlacemark: true
     }
});
myMap.controls.add(searchControl);
```

2.

```
// Example 2.
// If the control was already added to the map using a key from control.storage,
// you can get its instance using the control.Manager.get method
// for control.Manager.

var searchControl = myMap.controls.get('searchControl');
searchControl.events.add('submit', function () {
    console.log('request: ' + searchControl.getRequestString());
}, this);
```

3.

```
// Example 3.
// Enabling the business search in a control that is already added to the map.
// If the control was already added to the map using a key from control.storage,
// you can get its instance using the control.Manager.get method.

var searchControl = myMap.controls.get('searchControl');
searchControl.options.set('provider', 'yandex#search');
```

4.

```
// Example 4.
// Creating the "map search" control with business search enabled.

var searchControl = new ymaps.control.SearchControl({
     options: {
         float: 'left',
        provider: 'yandex#search'
     }
});
```

5.

```
// Example 5.
// Enlarging search control in "large" state.
map.options.set({
    // Default option value is [30, 72, 315],
    // we need to correct only value for "large" state.
    searchControlMaxWidth: [30, 72, 500],
    // Expand large search control to maxWidth.
    fitMaxWidth: true
});
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager.

Name	Type	Description
		Inherited from **IControl**.
state	**data.Manager**	State of the control. Names of fields that are available via the data.Manager.get method: • size — Current size of the control. • results — Array containing search results. • currentIndex — Index of the currently selected element. • found — Total number of results found. • request — Currently active query. • correction — Corrected query. • noSuggestPanel — Indicates whether the suggestion panel should be hidden.

Events

Name	Description
clear	Event of clearing the search results. Instance of the **Event** class.
error	Error in getting search results from the server event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • error — Information about the error.
load	Getting search results from the server event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • skip — How many elements were omitted. • count — The number of downloaded items.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
resultselect	"Search result selection" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index — Index of the selected result.
resultshow	Event for displaying search results. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index — Index of the selected result.
submit	The event of sending a search query to the server. Instance of the **Event** class.

Methods

Name	Returns	Description
clear()		Clears the search results and the contents of the search bar control.

Name	Returns	Description
getMap()	**Map**	Returns reference to the map.
getParent()	**IControlParent**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IControl**.
getRequestString()	String	Returns search query.
getResponseMetaData()	Object	Returns the geo search metadata.
getResult(index)	**vow.Promise**	Getting search results.
getResultsArray()	Object[]	Returns array containing current search results.
getResultsCount()	Integer	Returns total number of results found.
getSelectedIndex()	Integer	Returns index of the selected element.
hideResult()		Hides the result shown on the map.
search(request, options)	**vow.Promise**	Performs the search.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IControl**.
showResult(index)	**control.SearchControl**	Displays the result with the specified index.

Fields details

state

```
{data.Manager} state
```

State of the control. Names of fields that are available via the data.Manager.get method:

- size — Current size of the control.
- results — Array containing search results.
- currentIndex — Index of the currently selected element.
- found — Total number of results found.
- request — Currently active query.
- correction — Corrected query.
- noSuggestPanel — Indicates whether the suggestion panel should be hidden.

Events details

clear

Event of clearing the search results. Instance of the **Event** class.

error

Error in getting search results from the server event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- error — Information about the error.

load

Getting search results from the server event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- skip — How many elements were omitted.
- count — The number of downloaded items.

resultselect

"Search result selection" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- index — Index of the selected result.

resultshow

Event for displaying search results. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- index — Index of the selected result.

submit

The event of sending a search query to the server. Instance of the **Event** class.

Methods details

clear

```
{} clear()
```

Clears the search results and the contents of the search bar control.

getMap

```
{Map} getMap()
```

Returns reference to the map.

getRequestString

```
{String} getRequestString()
```

Returns search query.

getResponseMetaData

```
{Object} getResponseMetaData()
```

Returns the geo search metadata.

getResult

```
{vow.Promise} getResult(index)
```

Getting search results.

Returns object of type **vow.Promise**.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of the result, starting from 0.

* Mandatory parameter/option.

getResultsArray

```
{Object[]} getResultsArray()
```

Returns array containing current search results.

getResultsCount

```
{Integer} getResultsCount()
```

Returns total number of results found.

getSelectedIndex

```
{Integer} getSelectedIndex()
```

Returns index of the selected element.

hideResult

```
{} hideResult()
```

Hides the result shown on the map.

Example:

```
// If we showed a result on the map using control.SearchControl.showResult,
// or it was displayed automatically during search, we can hide it, for example,
// when the button is clicked.
var myButton = new ymaps.control.Button("Hide results");
myButton.events.add('click', function () {
    searchControl.hideResult();
}, this);
myMap.controls.add(myButton, { selectOnClick: false });
```

search

```
{vow.Promise} search(request, options)
```

Performs the search.

Returns object of type **vow.Promise**.

Parameters:

Parameter	Default value	Description
request*	—	Type: String Request.
options*	—	Type: Object Additional provider options.

* Mandatory parameter/option.

Example:

```
// Finding Moscow and outputting the "name" field
// from the first result to the console.
searchControl.search('Moscow').then(function () {
    var geoObjectsArray = searchControl.getResultsArray();
    if (geoObjectsArray.length) {
        // Outputs the "name" property of the first geo object in the search
results.
        console.log(geoObjectsArray[0].properties.get('name'));
    }
});
```

showResult

```
{control.SearchControl} showResult(index)
```

Displays the result with the specified index.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of the result, starting from 0.

* Mandatory parameter/option.

Example:

```
// We want to always show the first result,
// regardless of the number of objects
// found on the map (1 or more).
var searchControl = new ymaps.control.SearchControl({
    // This option disables automatically selecting search results.
    options: { noSelect: true }
});
searchControl.events.add('load', function (event) {
    // Verifying that this event is not completing results loading,
    // but that at least one result was found for the query.
    if (!event.get('skip') && searchControl.getResultsCount()) {
        searchControl.showResult(0);
    }
});
```

control.storage

Static object.

Instance of **util.Storage**

Storage for map controls. Defines control keys with their constructors.

By default, the following controls are added to the storage:

- "rulerControl" — Ruler and scale line **control.RulerControl**.
- "searchControl" — Search box **control.SearchControl**.
- "trafficControl" — Traffic panel **control.TrafficControl**.
- "typeSelector" — Panel for selecting the map type **control.TypeSelector**.
- "zoomControl" — Zoom slider **control.ZoomControl**.
- "geolocationControl" — Geolocation control **control.GeolocationControl**.
- "routeEditor" — Route editor **control.RouteEditor**.
- "fullscreenControl" — Control for full-screen mode **control.FullscreenControl**.
- "routeButtonControl" — Button with route panel popup **control.RouteButton**.
- "routePanelControl" — The Route Panel control **control.RoutePanel**.

Methods

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

control.TrafficControl

Extends **IControl**, **ICustomizable**.

The traffic control panel on the map.

Constructor | Fields | Events | Methods

Constructor

```
control.TrafficControl([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean

Parameter	Default value	Description
		Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.collapseOnBlur	true	Type: Boolean This flag enables to collapse the panel when the control loses focus. For example, when a user clicks on the document.
parameters.options.float	"right"	Type: String The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
parameters.options.floatIndex	100	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
parameters.options.layout	—	Type: Function|String Control layout. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager for the control.TrafficControl.options control. • data — Control data manager for the **control.TrafficControl.data** control. • state — Control state manager for the **control.TrafficControl.state** control. The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for **control.TrafficControl.state** depending on the commands received. (Type: constructor for an object with the **ITrafficControlLayout** interface).
parameters.options.maxWidth	[26, 195, 195]	Type: Number|Number[]

Parameter	Default value	Description
		The maximum width of the button in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class **control.Manager**, which is usually a field of **Map.controls**, via the "states" option. By default, the controls have three states ['small', 'medium', 'large'].
parameters.options.position	—	Type: Object Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.size	'auto'	Type: String Defines the appearance of the standard traffic control layout. Takes the following values: • 'auto' — The layout is changed automatically depending on the dimensions of the map and the number of added controls. • 'small' — The button layout displays the traffic light icon, regardless of the map size. • 'large' — The button layout always displays both the traffic light icon and text, regardless of the map size.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object State of the control.

Parameter	Default value	Description
parameters.state.providerKey	'traffic#actual'	Type: String Key for the provider of traffic info shown on the map. • 'traffic#actual' — Traffic "right now". • 'traffic#archive' — Traffic "normally".
parameters.state.trafficShown	false	Type: Boolean Whether traffic data is shown on the map.

Example:

```
// Adding the traffic control to the map
// with "current" traffic enabled.
var trafficControl = new ymaps.control.TrafficControl({state: {trafficShown:
true}});
map.controls.add(trafficControl, {top: 10, left: 10});
```

Fields

Name	Type	Description
data	data.Manager	Panel data.
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from IControl.
state	data.Manager	State of the panel. Names of fields that are available via the data.Manager.get method: • trafficShown — Flag for whether the traffic provider is shown on the map. • providerKey — Key of the provider that the panel shows. Accepts the values 'traffic#actual' and 'traffic#archive'. • expanded — Flag for whether the panel is expanded.

Events

Name	Description
collapse	The traffic panel is collapsed. Instance of the Event class.
expand	The traffic panel is expanded. Instance of the Event class.
hidetraffic	Traffic is hidden. Instance of the Event class.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed.

Name	Description
	Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
providerkeychange	The provider key changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • newProviderKey — New value for the provider key. • oldProviderKey — Old key value.
showtraffic	Traffic is shown. Instance of the **Event** class.

Methods

Name	Returns	Description
collapse()		Collapse the traffic panel.
expand()		Expand the traffic panel.
getMap()	**Map**	Returns reference to the map.
getParent()	**IControlParent**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IControl**.
getProvider([key])	**ITrafficProvider**	Returns instance of the traffic provider.
hideTraffic()		Hide the traffic provider from the map.
isExpanded()	Boolean	Returns flag for whether the panel is expanded.
isTrafficShown()	Boolean	Returns the flag for whether the panel is shown.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IControl**.
showTraffic()		Show the traffic provider on the map.

Fields details

data

```
{data.Manager} data
```

Panel data.

state

```
{data.Manager} state
```

State of the panel. Names of fields that are available via the data.Manager.get method:

- trafficShown — Flag for whether the traffic provider is shown on the map.
- providerKey — Key of the provider that the panel shows. Accepts the values 'traffic#actual' and 'traffic#archive'.
- expanded — Flag for whether the panel is expanded.

Events details

collapse

The traffic panel is collapsed. Instance of the **Event** class.

expand

The traffic panel is expanded. Instance of the **Event** class.

hidetraffic

Traffic is hidden. Instance of the **Event** class.

providerkeychange

The provider key changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- newProviderKey — New value for the provider key.
- oldProviderKey — Old key value.

showtraffic

Traffic is shown. Instance of the **Event** class.

Methods details

collapse

```
{} collapse()
```

Collapse the traffic panel.

expand

```
{} expand()
```

Expand the traffic panel.

getMap

```
{Map} getMap()
```

Returns reference to the map.

getProvider

```
{ITrafficProvider} getProvider([key])
```

Returns instance of the traffic provider.

Parameters:

Parameter	Default value	Description
key	—	Type: String Key of the provider of traffic information. List of available keys: • 'traffic#actual' — Provider for traffic "right now". • 'traffic#archive' — Provider for "normal" traffic. If the parameter is omitted, the current provider is returned.

Example:

```
// Adding the traffic control to the map.
map.controls.add('trafficControl');
// When opened, the provider for traffic "now" will show the layer of traffic
events.
map.controls.get('trafficControl').getProvider('traffic#actual').state.set('info
LayerShown', true);
```

hideTraffic

```
{} hideTraffic()
```

Hide the traffic provider from the map.

isExpanded

```
{Boolean} isExpanded()
```

Returns flag for whether the panel is expanded.

isTrafficShown

```
{Boolean} isTrafficShown()
```

Returns the flag for whether the panel is shown.

showTraffic

```
{} showTraffic()
```

Show the traffic provider on the map.

control.TypeSelector

Extends **control.ListBox**.

The "Map types" control. For this control, you can add list items that describe map types, as well as additional elements. The key of the control in the storage. **control.storage** — "typeSelector".

Constructor | Fields | Events | Methods

Constructor

```
control.TypeSelector([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: String[]|**MapType**[]|Object Object with descriptions of control parameters. If an array is passed, it is interpreted as an array of map types.
parameters.mapTypes	—	Type: String[]|**MapType**[] Array of constructors for map types or keys. If the parameter is omitted, the item is added to the standard set of map types. List of available map types: • 'yandex#map' — "Roadmap" map type. • 'yandex#satellite' — "Satellite" map type. • 'yandex#hybrid' — "Hybrid" map type.
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.collapseOnBlur	true	Type: Boolean This flag enables to collapse the list when the button loses focus. For example, when a user clicks on the document.
parameters.options.collapseTimeout	3000	Type: Number Time delay, after which the open list closes automatically.
parameters.options.expandOnClick	true	Type: Boolean Flag that allows automatically expanding/collapsing the list when clicked.
parameters.options.float	"right"	Type: String The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.

Parameter	Default value	Description
parameters.options.floatIndex	200	Type: Number The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
parameters.options.layout	—	Type: Function|String Constructor of the control layout which implements the **ISelectableControlLayout** and **IGroupControlLayout** interfaces or the layout key in the **layout.storage**. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager control.ListBox.options. • data — Control data manager **control.ListBox.data**. • state — Control state manager **control.ListBox.state**. The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for **control.ListBox.state** depending on the commands received.
parameters.options.maxWidth	[30, 65, 85]	Type: Number|Number[] The maximum width of the listbox in different states. If a number is specified, it is assumed that the button has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width of the button in different states, from the lesser to the greater.
parameters.options.panoramasItemMode	'ifMercator'	Type: String Shows or hides the "Panoramas" element. Possible values: • 'on' — The "Panorama" element is always shown. • 'ifMercator' — the 'Panorama' element is only shown if the map projection is the Mercator projection. • 'off' — The "Panorama" element is never shown. The "Panoramas" element is available only when using the standard layout.
parameters.options.position	—	Type: Object

Parameter	Default value	Description
		Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	'auto'	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	'auto'	Type: Number|String Position relative to the top edge of the map.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.state	—	Type: Object State of the control.
parameters.state.expanded	false	Type: Boolean Flags if the list is expanded.

Example:

```
map.controls.add(new ymaps.control.TypeSelector(['yandex#map',
'yandex#hybrid']));
```

Fields

Name	Type	Description
data	**data.Manager**	Control data.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **IControl**.
state	**data.Manager**	State of the drop-down list. Names of fields that are available via the data.Manager.get method: • expanded — Flag for whether the list is expanded. • size — The size that is currently set for the list. Inherited from **control.ListBox**.

Events

Name	Description
add	A child object was added. Inherited from **ICollection**.
click	Clicking the list title. Instance of the **Event** class. Inherited from **control.ListBox**.
collapse	The list is closed. Instance of the **Event** class. Inherited from **control.ListBox**.
expand	The list is open. Instance of the **Event** class. Inherited from **control.ListBox**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
press	The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state isEnabled == true. Instance of the **Event** class. Inherited from **control.ListBox**.
remove	A child object was deleted. Inherited from **ICollection**.

Methods

Name	Returns	Description
add(object)	**ICollection**	Adds a child object to the collection. Inherited from **ICollection**.
addMapType(mapType[, positionIndex])	**control.TypeSelector**	Adds a map type to the list.
collapse()	**control.ListBox**	Collapses the list. Inherited from **control.ListBox**.
expand()	**control.ListBox**	Expands the list. Inherited from **control.ListBox**.
getIterator()	**IIterator**	Returns iterator for the collection. Inherited from **ICollection**.
getMap()	**Map**	Returns reference to the map.
getParent()	**IControlParent**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IControl**.

Name	Returns	Description
isExpanded()	Boolean	Returns a flag for whether the control is in the expanded state. Inherited from **control.ListBox**.
remove(object)	**ICollection**	Removes a child object from the collection. Inherited from **ICollection**.
removeAllMapTypes()	**control.TypeSelector**	Deletes all map types from the control.
removeMapType(mapType)	**control.TypeSelector**	Deletes the map type.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IControl**.

Fields details

data

```
{data.Manager} data
```

Control data.

Methods details

addMapType

```
{control.TypeSelector} addMapType(mapType[, positionIndex])
```

Adds a map type to the list.

Returns self-reference.

Parameters:

Parameter	Default value	Description
mapType*	—	Type: String|**MapType** Map type.
positionIndex	—	Type: Integer Position in the list (if omitted, the new map type is added to the end of the list). The list of default positionIndex values for standard map types: • 'yandex#map' — 5; • 'yandex#satellite' — 10; • 'yandex#hybrid' — 15.

* Mandatory parameter/option.

Examples:

1.

```
var typeSelector = new ymaps.control.TypeSelector([]);
typeSelector.addMapType('yandex#map', 1);
typeSelector.addMapType('yandex#hybrid', 0);
```

2.

```
// If a standard set of map types is being used,
// and we want to add our own set from mapType.storage // and insert it between
"satellite" and "map".
var typeSelector = myMap.controls.get('typeSelector');
typeSelector.addMapType('my#mapType', 6);
```

getMap

```
{Map} getMap()
```

Returns reference to the map.

removeAllMapTypes

```
{control.TypeSelector} removeAllMapTypes()
```

Deletes all map types from the control.

Returns self-reference.

removeMapType

```
{control.TypeSelector} removeMapType(mapType)
```

Deletes the map type.

Returns self-reference.

Parameters:

Parameter	Default value	Description
mapType*	—	Type: String|**MapType** Map type.

* Mandatory parameter/option.

control.ZoomControl

Extends **IControl**, **ICustomizable**.

The "Zoom" control. The key of the control in the storage. **control.storage** — "zoomControl".

Constructor | Fields | Events | Methods

Constructor

```
control.ZoomControl([parameters])
```

Parameters:

Parameter	Default value	Description
parameters	—	Type: Object Control parameters.

Parameter	Default value	Description
parameters.data	—	Type: Object Control data.
parameters.options	—	Type: Object Control options.
parameters.options.adjustMapMargin	false	Type: Boolean Whether the control registers its size in the map margins manager **map.margin.Manager**.
parameters.options.layout	—	Type: **IZoomControlLayout**|String Constructor of the control layout or the layout key in the **layout.storage**. The layout constructor is passed an object containing the fields: • control — Reference to the control. • options — Control options manager control.ZoomControl.options. • data — Control data manager control.ZoomControl.data. • state — Control state manager control.ZoomControl.state.
parameters.options.position	—	Type: Object Object describing the position of a control.
parameters.options.position.bottom	'auto'	Type: Number|String Position relative to the bottom edge of the map.
parameters.options.position.left	10	Type: Number|String Position relative to the left edge of the map.
parameters.options.position.right	'auto'	Type: Number|String Position relative to the right edge of the map.
parameters.options.position.top	108	Type: Number|String Position relative to the top edge of the map.
parameters.options.size	'auto'	Type: String

Parameter	Default value	Description
		Defines the appearance of the control. Takes the following values: • 'small' — Always show a small zoom control. • 'large' — Always show a large zoom control. • 'auto' — Automatically select the size of the control depending on the height of the map container.
parameters.options.visible	true	Type: Boolean Indicates if the control is displayed.
parameters.options.zoomDuration	200	Type: Number The length of animation playback when changing the zoom level.
parameters.options.zoomStep	1	Type: Number Step of the zoom level change.
parameters.state	—	Type: Object Object describing the state of a control.

Examples:

1.

```
// Example 1.
// Creating a small zoom control and adding it to the map.
var zoomControl = new ymaps.control.ZoomControl({
    options: {
        size: "small"
    }
});
myMap.controls.add(zoomControl);
```

2.

```
// Example 2.
// If the control was already added to the map using a key from control.storage.
myMap.controls.add('zoomControl', {
    size: "large"
});
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from IControl.

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map.
getParent()	IControlParent|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IControl.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IControl.

Methods details

getMap

```
{Map} getMap()
```

Returns reference to the map.

coordSystem

coordSystem.cartesian

Static object.

Instance of **ICoordSystem**

Class that describes the geometry of a Cartesian plane. Used as the default coordinate system when constructing geodetic lines in non-standard projections.

Methods

Methods

Name	Returns	Description
getDistance(point1, point2)	Number	Returns the shortest distance (along a geodetic line) between the two set points (in meters).
solveDirectProblem(startPoint, direction, distance)	Object	Solves the first (direct) geodesic problem: where we will end up, if we start from a specified point and move in the specified direction for the specified distance, without turning. The following data is a solution for the direct geodetic problem: • The end point. • The final direction. • The path function. • A function that allows to specify, for any given moment in time, which point we will be at and which direction we will be moving in.
solveInverseProblem(startPoint, endPoint[, reverseDirection])	Object	Solves the second (inverse) geodetic problem: construct the shortest route between two points on the mapped surface and determine the distance and direction of movement. Note that on the map of the Earth's surface, the shortest routes are shown as crooked lines. For geo objects in the API, you can enable the mode for displaying shortest distances between points using the "geodesic" option.

coordSystem.geo

Static object.

Instance of **ICoordSystem**

Object that describes the geometry of the Earth's surface. Used for constructing shortest routes (geodetic lines) between points on the Earth's surface and finding distances. Used for displaying geo objects in "geodesic: true" mode.

Methods

Methods

Name	Returns	Description
getDistance(point1, point2)	Number	Returns the shortest distance (along a geodetic line) between the two set points (in meters).
solveDirectProblem(startPoint, direction, distance)	Object	Solves the first (direct) geodesic problem: where we will end up, if we start from a specified point and move in the specified direction for the specified distance, without turning. The following data is a solution for the direct geodetic problem: • The end point. • The final direction. • The path function. • A function that allows to specify, for any given moment in time, which point we will be at and which direction we will be moving in.
solveInverseProblem(startPoint, endPoint[, reverseDirection])	Object	Solves the second (inverse) geodetic problem: construct the shortest route between two points on the mapped surface and determine the distance and direction of movement. Note that on the map of the Earth's surface, the shortest routes are shown as crooked lines. For geo objects in the API, you can enable the mode for displaying shortest distances between points using the "geodesic" option.

data

data.Manager

Extends **IDataManager**, **IFreezable**.

Custom data manager.

Constructor | Fields | Events | Methods

Constructor

```
data.Manager(data)
```

Parameters:

Parameter	Default value	Description
data *	—	Type: Object Data.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager for the object. Inherited from **IFreezable**.

Events

Name	Description
change	Change to the internal state of the object. Inherited from **IFreezable**.

Methods

Name	Returns	Description
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
get(path[, defaultValue])	Object	Returns the value of the data field with the specified name.
getAll()	Object	Returns an object containing all the data fields.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.

Name	Returns	Description
set(path[, value])	**data.Manager**	Sets values for the specified fields. Two signatures are supported: • A single argument consisting of a {"name": "value"} object. • Two arguments; the first is the field name, and the second is the value. The name can reference nested fields, i.e. it can contain '.'.
setAll()	**data.Manager**	Completely overwrites all data fields. Equal to consecutive calls of the "unsetAll" and "set" methods, but with better performance.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.
unset(path)	**data.Manager**	Clears the specified data fields.
unsetAll()	**data.Manager**	Clears all data fields.

Methods details

get

```
{Object} get(path[, defaultValue])
```

Returns the value of the data field with the specified name.

Parameters:

Parameter	Default value	Description
path*	—	Type: String String with the name of a data field. The name can reference nested fields, i.e. it can contain '.'.
defaultValue	—	Type: Object Default value.

* Mandatory parameter/option.

getAll

```
{Object} getAll()
```

Returns an object containing all the data fields.

set

```
{data.Manager} set(path[, value])
```

Sets values for the specified fields. Two signatures are supported:

- A single argument consisting of a {"name": "value"} object.
- Two arguments; the first is the field name, and the second is the value.

The name can reference nested fields, i.e. it can contain '.'.

Returns self-reference.

Parameters:

Parameter	Default value	Description
path*	—	Type: Object|String A string containing the name of a data field, or an object of the type {"field name": "value"}.
value	—	Type: Object The value, if the string containing the field name is passed as the first argument.

* Mandatory parameter/option.

Example:

```
var balloonData = new ymaps.data.Manager({
     balloonContentHeader: 'Balloon title'
});
balloonData.set({
    balloonContentBody: 'Balloon content',
    balloonContentFooter: '&lt;a href=&quot;//ya.ru&quot;&gt;test&lt;/a&gt;'
});
```

setAll

```
{data.Manager} setAll()
```

Completely overwrites all data fields. Equal to consecutive calls of the "unsetAll" and "set" methods, but with better performance.

Returns self-reference.

Parameters:

Parameter	Default value	Description
data*	—	Type:

* Mandatory parameter/option.

unset

```
{data.Manager} unset(path)
```

Clears the specified data fields.

Returns self-reference.

Parameters:

Parameter	Default value	Description
path*	—	Type: String|String[] Field name, or an array of names.

* Mandatory parameter/option.

unsetAll

```
{data.Manager} unsetAll()
```

Clears all data fields.

Returns self-reference.

DomEvent

Extends **IDomEvent**.

DOM event in the Yandex.Maps API system. Provides proxy methods for accessing fields and methods of the source DOM event. The returned values are standardized to account for browser variations. The "position" property is also redefined; an array of the type [pageX, pageY] is returned.

Constructor | Methods

Constructor

```
DomEvent(originalEvent[, type])
```

Creates a DOM event in the Yandex.Maps API system.

Parameters:

Parameter	Default value	Description
originalEvent *	—	Type: **DomEvent** DOM event.
type	—	Type: Object Type of event. If omitted, it is taken from originalEvent.type.

* Mandatory parameter/option.

Methods

Name	Returns	Description
allowMapEvent()		Allows the propagation of the event to the map. Inherited from **IEvent**.
callMethod(name)		Calls the specified method from the source event. The second and following arguments are passed to the method with the call. Inherited from **IEvent**.
get(name)	Object	Returns the value of a property. First it checks whether the property was set via "set", then it checks whether the property exists in "domEvent.overrideStorage". If it's not found, it looks in "originalEvent". Property values are cached.
getSourceEvent()	**IDomEvent**	Returns source DOM event. Inherited from **IDomEvent**.
isDefaultPrevented()	Boolean	Returns true if the default reaction to the event has been canceled. Inherited from **IEvent**.
isImmediatePropagationStopped()	Boolean	Returns true if the event processing has been interrupted. Inherited from **IEvent**.

Name	Returns	Description
isMapEventAllowed()	Boolean	Returns true if the map event is enabled. Inherited from **IEvent**.
isPropagationStopped()	Boolean	Returns true if event propagation has been interrupted. Inherited from **IEvent**.
preventDefault()		Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect how the browser processes the default action for the source DOM event. Inherited from **IDomEvent**.
stopImmediatePropagation()		Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called. Calling this method does not affect the processing of the original DOM-event at the browser level. Inherited from **IDomEvent**.
stopPropagation()		Stops propagation of the DOM event in the Yandex.Maps API event system. Calling this method does not affect propagation of the source DOM event through the DOM tree. Inherited from **IDomEvent**.

Methods details

get

```
{Object} get(name)
```

Returns the value of a property. First it checks whether the property was set via "set", then it checks whether the property exists in "domEvent.overrideStorage". If it's not found, it looks in "originalEvent". Property values are cached.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Property name.

* Mandatory parameter/option.

domEvent

domEvent.manager

Static object.

Provides a singular interface for working with DOM element events in all browsers and on all devices. For devices that do not have mouse support, events will be mapped to mouse events.

- The touch start event (touchstart/pointerdown) with a single touch point is interpeted as a sequence of "mouseenter", "mousemove" and "mousedown" events.
- The moving touch event (touchmove/pointermove) with a single touch point is mapped to the "mousemove" event.
- Events for ending touch (touchend/pointerup) or canceling touch (touchcancel/pointercancel) are mapped to a sequence of "mouseup", "mousemove", and "mouseleave" events, if a touch start event with a single touch point already occurred earlier.
- A quick succession of start and end events with a single touch point without movement is mapped to a "click" event.
- A quick succession of two "click" events is mapped to a "dblclick" event.
- If there was a lengthy pause between the start and end events with a single touch point and without movement, this is mapped to the "contextmenu" event.

Special events for handling multiple simultaneous touches are also supported.

- "multitouchstart" is sent when touch start events are received with two or more touch points.
- "multitouchmove" is sent when moving touch events are received with two or more touch points.
- "multitouchend" is sent when touch end events are received, if the "multitouchstart" event was sent earlier.
- When adding/removing a touch point, the "multitouchend" event and the "multitouchstart" event will be sent if the remaining number of points is greater than or equal to two.

Manager for working with DOM element events.

Methods

Examples:

1.

```
// Listening for events of a single DOM element.
var block = document.getElementById('block');
ymaps.domEvent.manager
    .add(block, 'click', function (event) {
        console.log(event.get('type')); // click
    })
    .add(block, 'mouseleave', function (event) {
        console.log(event.get('type')); // mouseleave
    })
    // Setting a single listener for multiple events.
    .add(block, ['mousedown', 'mouseup'], function (event) {
        console.log(event.get('type')); // mousedown / mouseup
    });
```

2.

```
// Using the events container.
var block = document.getElementById('block'),
    domEventsGroup = ymaps.domEvent.manager.group(block);
domEventsGroup.add('click', function (event) {
    console.log(event.get('type')); // click
    // Deleting all event listeners.
    domEventsGroup.removeAll();
});
```

3.

```
// Executing the listener in the context of a specific object.
var block = document.getElementById('block'),
    // Defining the class.
    someClass = function () {
        this.property = 'value';
    },
    // Creating a class implementation.
    someObj = new someClass();
ymaps.domEvent.manager.add(block, 'click', function (event) {
    console.log(this.property + ' ' + event.get('type')); // value click
}, someObj);
```

4.

```
// On devices with touch support, we can listen to special multitouch* events
var block = document.getElementById('block');
ymaps.domEvent.manager
    .add(block, ['multitouchstart', 'multitouchmove', 'multitouchend'], function
(event) {
        console.log(event.get('type')); // multitouchstart / multitouchmove /
multitouchend
        // Not allowing users to move and scale the page using touch.
        event.callMethod('preventDefault');
    });
```

Methods

Name	Static	Returns	Description
domEvent.manager.**add**(htmlElement, types, callback[, context[, capture]])	✔	**domEvent.manager**	Adds a listener for the object's DOM events.
domEvent.manager.**group**(htmlElement[, capture])	✔	**event.Group**	Returns group of event listeners for the specified DOM element.
domEvent.manager.**remove**(htmlElement, types, callback[, context[, capture]])	✔	**domEvent.manager**	Deletes the listener for the objects's DOM events.

Methods details

add

```
{domEvent.manager}  <static> domEvent.manager.add(htmlElement, types, callback[,
context[, capture]])
```

Adds a listener for the object's DOM events.

Returns self-reference.

Parameters:

Parameter	Default value	Description
htmlElement*	—	Type: HTMLElement|Document The DOM element whose events need to be listened for.
types*	—	Type: String|String[] Event type or types.

Parameter	Default value	Description
callback*	—	Type: Function Handler function for the event.
context	—	Type: Object Context for the handler function.
capture	—	Type: Boolean Indicates if the event should be monitored at the capture phase.

* Mandatory parameter/option.

group

```
{event.Group}  <static> domEvent.manager.group(htmlElement[, capture])
```

Returns group of event listeners for the specified DOM element.

Parameters:

Parameter	Default value	Description
htmlElement*	—	Type: HTMLElement|Document DOM element.
capture	—	Type: Boolean Indicates if the event should be monitored at the capture phase.

* Mandatory parameter/option.

remove

```
{domEvent.manager}  <static>  domEvent.manager.remove(htmlElement,  types,
callback[, context[, capture]])
```

Deletes the listener for the objects's DOM events.

Returns self-reference.

Parameters:

Parameter	Default value	Description
htmlElement*	—	Type: HTMLElement|Document The DOM element whose events are listened for.
types*	—	Type: String|String[] Event type or types.
callback*	—	Type: Function|String Handler function for the event, or the unique id of the callback-context pair.
context	—	Type: Object

Parameter	Default value	Description
		Context for the handler function.
capture	—	Type: Boolean Indicates if the event should be monitored at the capture phase.

* Mandatory parameter/option.

domEvent.MultiPointer

Extends IMultiTouchEvent.

Object describing the multitouch event that was called by multiple PointerEvent events. Provides proxy methods for accessing fields and methods of DOM events.

Constructor | Methods

Constructor

```
domEvent.MultiPointer(originalEvent[, type])
```

Creates an object that describes a multitouch event.

Parameters:

Parameter	Default value	Description
originalEvent *	—	Type: Object pointer event.
type	—	Type: String Type of event. If omitted, it is assumed to be 'multi' + originalEvent.type.

* Mandatory parameter/option.

Methods

Name	Returns	Description
get(name)	Object	Returns the value of the original event property. Property values are cached.

Methods details

get

```
{Object} get(name)
```

Returns the value of the original event property. Property values are cached.

Parameters:

Parameter	Default value	Description
name*	—	Type: String

Parameter	Default value	Description
		Property name.

* Mandatory parameter/option.

domEvent.MultiTouch

Extends IMultiTouchEvent.

Event object. Provides proxy methods for accessing fields and methods of DOM events. The returned values are standardized to account for browser variations.

Constructor | Methods

Constructor

```
domEvent.MultiTouch(originalEvent[, type])
```

Creates an event object that describes a multitouch event.

Parameters:

Parameter	Default value	Description
originalEvent *	—	Type: Object Multitouch event.
type	—	Type: String Type of event. If omitted, it is assumed to be 'multi' + originalEvent.type.

* Mandatory parameter/option.

Methods

Name	Returns	Description
get(name)	Object	Returns the value of the original event property. Property values are cached.

Methods details

get

```
{Object} get(name)
```

Returns the value of the original event property. Property values are cached.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Property name.

* Mandatory parameter/option.

domEvent.Pointer

Extends IMultiTouchEvent.

Event object. Provides proxy methods for accessing the fields and methods of a DOM event (single touch on the screen).

Constructor | Methods

Constructor

```
domEvent.Pointer(originalEvent[, type])
```

Creates an object that describes a pointer event.

Parameters:

Parameter	Default value	Description
originalEvent *	—	Type: Object DOM event.
type	—	Type: String Type of event. If omitted, it is taken from originalEvent.type.

* Mandatory parameter/option.

Methods

Name	Returns	Description
get(name)	Object	Returns the value of the original event property. Property values are cached.

Methods details

get

```
{Object} get(name)
```

Returns the value of the original event property. Property values are cached.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Property name.

* Mandatory parameter/option.

domEvent.Touch

Extends IMultiTouchEvent.

Event object. Provides proxy methods for accessing fields and methods of DOM events. The returned values are standardized to account for browser variations.

Constructor | Methods

Constructor

```
domEvent.Touch(originalEvent[, type])
```

Creates an event object that describes a touch event (single touch on the screen).

Parameters:

Parameter	Default value	Description
originalEvent *	—	Type: Object DOM event.
type	—	Type: String Type of event. If omitted, it is taken from originalEvent.type.

* Mandatory parameter/option.

Methods

Name	Returns	Description
get(name)	Object	Returns the value of the original event property. Property values are cached.

Methods details

get

```
{Object} get(name)
```

Returns the value of the original event property. Property values are cached.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Property name.

* Mandatory parameter/option.

event

event.Group

Extends **IEventGroup**.

A group of event listeners.

Constructor | Fields | Methods

Constructor

```
event.Group(events)
```

Creates a group of event listeners.

Parameters:

Parameter	Default value	Description
events *	—	Type: **IEventManager** The event manager that the group was created for.

* Mandatory parameter/option.

Example:

```
// Creating a group of event listeners.
var listeners = events.group()
    .add('click', function () {
        alert('click');
    })
    .add('dblclick', function () {
        alert('dblclick');
    });
// ...
// When the event handlers stored in the container
// are no longer needed, we simply clear the group.
listeners.removeAll();
```

Fields

Name	Type	Description
events	**IEventManager**	The event manager that the group was created for.

Methods

Name	Returns	Description
add(types, callback[, context[, priority]])	**IEventGroup**	Adds an event listener. Inherited from **IEventGroup**.
getLength()	Integer	Returns the current number of subscribers in the group.
remove(types, callback[, context[, priority]])	**IEventGroup**	Deletes an event listener from the group. Inherited from **IEventGroup**.

Name	Returns	Description
removeAll()	IEventGroup	Deletes all event listeners from the group. Inherited from IEventGroup.

Fields details

events

```
{IEventManager} events
```

The event manager that the group was created for.

Methods details

getLength

```
{Integer} getLength()
```

Returns the current number of subscribers in the group.

event.Manager

Extends **IEventManager**.

Event manager. Using an event manager, you can subscribe to and unsubscribe from events, as well as initiate the events themselves. The manager implements the possibility of building a hierarchy of event propagation using the method event.Manager.setParent.

Event propagation has three phases:

- 1. Direct subscribers get events.
- 2. Objects that are higher up in the hierarchy get events when they are relayed on the parent event manager.
- 3. Default action handlers get events via a type + 'defaultaction' service event; the default action is performed only if the event's "target" field matches the context for the event manager.

The manager also allows you to specify the priority when adding event handlers. When throwing events, the handlers will be called in order of decreasing priority.

Subscriptions with the same callback and context parameters but different priority settings are considered to be different. When removing subscriptions, you must specify the same priority as was set when it was added.

Constructor | Methods

Constructor

```
event.Manager([params])
```

Parameters:

Parameter	Default value	Description
params	—	Type: Object Parameters of the event manager.
params.context	—	Type: Object

Parameter	Default value	Description
		The context object of the event manager.
params.controllers	—	Type: **IEventWorkflowController**[] Controller or controllers for the event manager.
params.parent	—	Type: **IEventManager** Parent event manager.

Methods

Name	Returns	Description
add(types, callback[, context[, priority]])	**IEventManager**	Adds a new subscription. Inherited from **IEventManager**.
createEventObject(type, event, target)	**Event**	Function that creates the event object. Called in the "fire" method when the passed object is not an instance of the Event class or its descendant.
fire(type[, event])	**event.Manager**	Fires an event.
getParent()	**IEventManager**|null	Returns reference to the parent event manager. Inherited from **IEventManager**.
group()	**IEventGroup**	Returns the group of event listeners associated with the given event manager. Inherited from **IEventManager**.
once(types, callback[, context[, priority]])	**IEventManager**	Adds a listener, which calls the handler function only one time. Inherited from **IEventManager**.
remove(types, callback[, context[, priority]])	**IEventManager**	Removes an existing subscription. Inherited from **IEventManager**.
setParent(parent)	**event.Manager**	Sets the parent event manager.

Methods details

createEventObject

```
{Event} createEventObject(type, event, target)
```

Function that creates the event object. Called in the "fire" method when the passed object is not an instance of the Event class or its descendant.

Returns Event object.

Parameters:

Parameter	Default value	Description
type*	—	Type: String

Parameter	Default value	Description
		Type of event.
event*	—	Type: Object Object that describes the event.
target*	—	Type: Object Object that the event occurred on.

* Mandatory parameter/option.

fire

```
{event.Manager} fire(type[, event])
```

Fires an event.

Returns self-reference.

Parameters:

Parameter	Default value	Description
type*	—	Type: String Type of event.
event	—	Type: Object|**Event** Event. If a hash with data is passed, the createEventObject method will be called for it and further actions will be performed with the newly created event.

* Mandatory parameter/option.

setParent

```
{event.Manager} setParent(parent)
```

Sets the parent event manager.

Returns self-reference.

Parameters:

Parameter	Default value	Description
parent*	—	Type: **IEventManager**|null Parent event manager.

* Mandatory parameter/option.

event.Mapper

Extends **IEventTrigger**.

Event mapper. Allows managing propagation of events along the hierarchy of event managers.

Constructor | Methods

Constructor

```
event.Mapper(targetEventManager, mappingTable)
```

Parameters:

Parameter	Default value	Description
targetEventManager *	—	Type: **IEventManager** The event manger that the mapper propagates initiated events to.
mappingTable *	—	Type: Object Table of mapping rules. A hash where the keys are types of events and the values are the corresponding mapping functions, or Boolean values. A mapping function for a specific type of event receives an event instance that was initiated on the mapper, and must return the event instance for propagating further along the hierarchy, or null if propagation must be prohibited. The boolean values are interpreted as follows: • true — Events of this type are propagated through the hierarchy unchanged. • false — Events of this type are not propagated through the hierarchy. The "*" key is also available in the table for the default processing rule.

* Mandatory parameter/option.

Example:

```
// Creating and setting up the event mapper that will transform the "click" event
in the root
// geo object collection into a "geoobjectclick" event on the map itself.
var mapper = new ymaps.event.Mapper(myMap.events, {
    "*": false,
    "click": function (event) {
        return new ymaps.Event({
             type: "geoobjectclick",
             target: map,
             originalTarget: event.get("target")
        }, event);
    }
});

myMap.geoObjects.events.setParent(mapper);
```

Methods

Name	Returns	Description
fire(type[, eventObject])	**IEventTrigger**	Triggers an event. Inherited from **IEventTrigger**.

Event

Extends **IEvent**.

Event. Provides methods for accessing the originalObject object's fields and methods, with the possibility for redefining them.

Constructor | Methods

Constructor

```
Event(originalEvent[, sourceEvent])
```

Creates an event.

Parameters:

Parameter	Default value	Description
originalEvent *	—	Type: Object Source data.
sourceEvent	—	Type: **IEvent** Source event.

* Mandatory parameter/option.

Methods

Name	Returns	Description
allowMapEvent()		Allows the propagation of the event to the map. Inherited from **IEvent**.
callMethod(name)	Object	Calls the specified method. The operation is equivalent to searching fields via "get" and making a call that passes originalEvent as context. All arguments after the first one are passed as parameters to the method being called.
get(name)	Object	Returns the field value from originalEvent. originalEvent always has the following fields: • type — String event type. • target — Reference to the object that generated the event. .
getSourceEvent()	**IEvent**|null	Returns source event. Inherited from **IEvent**.
isDefaultPrevented()	Boolean	Checks whether the default reaction to the event is canceled in the Yandex.Maps API event system.
isImmediatePropagationStopped()	Boolean	Checks whether event propagation is stopped in the Yandex.Maps API event system.

Name	Returns	Description
isMapEventAllowed()	Boolean	Returns true if the map event is enabled. Inherited from **IEvent**.
isPropagationStopped()	Boolean	Checks whether event propagation up the hierarchy of objects and collections is stopped in the Yandex.Maps API event system.
preventDefault()		Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.
stopImmediatePropagation()		Stops event propagation in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.
stopPropagation()		Stops event propagation up the hierarchy of objects and collections in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.

Methods details

callMethod

```
{Object} callMethod(name)
```

Calls the specified method. The operation is equivalent to searching fields via "get" and making a call that passes originalEvent as context. All arguments after the first one are passed as parameters to the method being called.

Returns value.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Method name.

* Mandatory parameter/option.

get

```
{Object} get(name)
```

Returns the field value from originalEvent. originalEvent always has the following fields:

- type — String event type.
- target — Reference to the object that generated the event.

.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Property name.

* Mandatory parameter/option.

Example:

```
// Synchronizing two objects with each other.
object1.events.add(["add", "remove"], function (event) {
    object2[event.get("type")](event.get("child"));
});
```

isDefaultPrevented

```
{Boolean} isDefaultPrevented()
```

Checks whether the default reaction to the event is canceled in the Yandex.Maps API event system.

Returns true if the default reaction to the event is canceled, otherwise false.

isImmediatePropagationStopped

```
{Boolean} isImmediatePropagationStopped()
```

Checks whether event propagation is stopped in the Yandex.Maps API event system.

Returns true if propagation was stopped, or false if not.

isPropagationStopped

```
{Boolean} isPropagationStopped()
```

Checks whether event propagation up the hierarchy of objects and collections is stopped in the Yandex.Maps API event system.

Returns true if propagation up the hierarchy is canceled; otherwise, false.

preventDefault

```
{} preventDefault()
```

Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.

stopImmediatePropagation

```
{} stopImmediatePropagation()
```

Stops event propagation in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.

stopPropagation

```
{} stopPropagation()
```

Stops event propagation up the hierarchy of objects and collections in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.

formatter

Static object.

Static class containing methods for formatting measurement units depending on the current language.

Methods

Methods

Name	Returns	Description
distance(value[, significantDigits])	String	Returns string representation of distance formatted for the locale being used, and converted to the appropriate measurement system (for example, for en-US, distance will be in miles).
duration(value[, significantDigits])	String	Returns string representation of duration.

Methods details

distance

```
{String} distance(value[, significantDigits])
```

Returns string representation of distance formatted for the locale being used, and converted to the appropriate measurement system (for example, for en-US, distance will be in miles).

Parameters:

Parameter	Default value	Description
value*	—	Type: Number Length in meters.
significantDigits	2	Type: Integer Number of significant digits in the response.

* Mandatory parameter/option.

duration

```
{String} duration(value[, significantDigits])
```

Returns string representation of duration.

Parameters:

Parameter	Default value	Description
value*	—	Type: Number Duration in seconds.
significantDigits	2	Type: Integer Number of significant digits in the response.

* Mandatory parameter/option.

geocode

Static function.

Processes geocoding requests. The request result can be provided in JSON format or as a **GeoObjectCollection** object. The geocoder's response format is described in Geocoding.

See GeocodeResult

Returns Promise object.

```
{ vow.Promise } geocode(request[, options])
```

Parameters:

Parameter	Default value	Description
request*	—	Type: String|Number[] The address for which coordinates need to be obtained (forward geocoding), or the coordinates for which the address needs to be determined (reverse geocoding).
options	—	Type: Object Options.
options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located.
options.json	false	Type: Boolean If true, JSON is passed to the handler function. Otherwise, the handler function is passed an object containing the geoObjects field with the geocoding results as **GeoObjectCollection**. When geocoding using the 'yandex#map' geocoder, the collection contains **GeocodeResult** objects.
options.kind	'house'	Type: String Type of toponym (only for reverse geocoding). List of acceptable values: • house — House or building. • street — Street. • metro — Subway station. • district — City district. • locality — City, town, village, etc.
options.provider	'yandex#map'	Type: **IGeocodeProvider**|String

Parameter	Default value	Description
		Geocoding provider. One of the standard providers can be used: • 'yandex#map' — Search on the map.
options.results	10	Type: Integer Maximum number of results to be returned.
options.searchCoordOrder	—	Type: String Determines how to interpret the coordinates in the request.
options.skip	0	Type: Integer Number of results that must be skipped.
options.strictBounds	false	Type: Boolean Search only inside the area defined by the "boundedBy" option.

* Mandatory parameter/option.

Examples:

1.

```
// Performs a search for an object named "Moscow".
// The result is immediately displayed on the map.
var myGeocoder = ymaps.geocode("Moscow");
myGeocoder.then(
    function (res) {
        map.geoObjects.add(res.geoObjects);
        // Taking the data resulting from geocoding the object and outputting it
to the console.

console.log(result.geoObjects.get(0).properties.get('metaDataProperty').getAll()
);
    },
    function (err) {
        // error handling
    }
);
```

2.

```
// implements the IGeocodeProvider interface
var randomPointProvider = {
     geocode: function (request, options) {
         var deferred = ymaps.vow.defer(),
             geoObjects = new ymaps.GeoObjectCollection(),
             results = options.results || 10;

        for (var i = 0; i < results; i++) {
             geoObjects.add(new ymaps.GeoObject({
                 geometry: {
                     type: "Point",
                     coordinates: [(Math.random() - 0.5) * 180, (Math.random() -
0.5) * 360]
                 },
                 properties: {
                     name: request + ' ' + i,
                     description: request + '\'s description ' + i,
```

```
                    balloonContentBody: '<p>' + request + ' ' + i + '</p>'
                }
            }));
        }

        var response = {
            geoObjects: geoObjects, // search output geo objects
            // response metainformation
            metaData: {
                geocoder: {
                    request: request, // processed request string
                    found: results, // number of results found
                    results: results, // number of results returned
                    skip: options.skip || 0 // number of results skipped
                }
            }
        };

        // asynchronous processing
        setTimeout(function () {
            deferred.resolve(response);
        }, 0);

        return deferred.promise();
    }
},
    myGeocoder = ymaps.geocode("Moscow", { provider: randomPointProvider });

myGeocoder.then(
    function (res) {
        map.geoObjects.add(res.geoObjects);
    },
    function (err) {
        // handling errors
    }
);
```

GeocodeResult

Extends **IGeoObject**.

Geocoding result.

See Placemark geocode geolocation control.SearchControl

Constructor | Fields | Events | Methods

Constructor

```
GeocodeResult()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Name	Type	Description
properties	IDataManager	Geo object data. Inherited from IGeoObject.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Name	Description
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Name	Description
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getAddressLine()	String	Returns string with the address of the object.
getAdministrativeAreas()	String[]	Returns regional areas the object belongs to (federal district, region, or other district); no more than two.
getCountry()	String|null	Returns the country the toponym belongs to (if applicable).
getCountryCode()	String|null	Returns the code of the country the toponym belongs to (if applicable), as a two-letter ISO 3166 code.
getLocalities()	String[]	Returns the populated locality and, optionally, an area within the locality that the toponym belongs to.
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getPremise()	String|null	Returns the name of the building, if it has one (for example, the name of an airport terminal).
getPremiseNumber()	String|null	Returns the building number (including the unit, complex, or other additional characteristics).
getThoroughfare()	String|null	Returns the roadway (street, highway, road, and so on) that the toponym belongs to (if applicable).

Name	Returns	Description
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Methods details

getAddressLine

```
{String} getAddressLine()
```

Returns string with the address of the object.

getAdministrativeAreas

```
{String[]} getAdministrativeAreas()
```

Returns regional areas the object belongs to (federal district, region, or other district); no more than two.

getCountry

```
{String|null} getCountry()
```

Returns the country the toponym belongs to (if applicable).

getCountryCode

```
{String|null} getCountryCode()
```

Returns the code of the country the toponym belongs to (if applicable), as a two-letter ISO 3166 code.

getLocalities

```
{String[]} getLocalities()
```

Returns the populated locality and, optionally, an area within the locality that the toponym belongs to.

getPremise

```
{String|null} getPremise()
```

Returns the name of the building, if it has one (for example, the name of an airport terminal).

getPremiseNumber

```
{String|null} getPremiseNumber()
```

Returns the building number (including the unit, complex, or other additional characteristics).

getThoroughfare

```
{String|null} getThoroughfare()
```

Returns the roadway (street, highway, road, and so on) that the toponym belongs to (if applicable).

geolocation

Static object.

Provides information about the user's location.

Methods

Methods

Name	Returns	Description
get([options])	vow.Promise	Tries to determine the user's location. Returns the promise object, which will either be confirmed by the object with the field geoObjects or rejected with an error message. The geoObjects field is an instance of GeoObjectCollection. The object that indicates the user's current location will be added to the collection.

Methods details

get

```
{vow.Promise} get([options])
```

Tries to determine the user's location. Returns the promise object, which will either be confirmed by the object with the field geoObjects or rejected with an error message. The geoObjects field is an instance of **GeoObjectCollection**. The object that indicates the user's current location will be added to the collection.

Returns Promise object.

Parameters:

Parameter	Default value	Description
options	—	Type:
options.autoReverseGeocode	true	Type: If true, geocode the user position automatically; if false, return as it is. If automatic geocoding is used, the object marking the user's current position has the same structure as the result of executing **geocode**.
options.mapStateAutoApply	false	Type: If true, the map center and zoom level are adjusted automatically to show the current location of the user; if false, nothing happens.
options.provider	'auto'	Type:

Parameter	Default value	Description
		Geolocation provider. Accepted values: 'yandex' — geolocation according to the Yandex data, based on the user IP-address; 'browser' — built-in browser geolocation; 'auto' — try to locate the user by all means available and then choose the best value.
options.timeout	30000	Type: The response time, in milliseconds.
options.useMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager** when automatically centering and zooming the map.

Examples:

1.

```
ymaps.geolocation.get({
    // Setting the option for determining a user by IP
    provider: 'yandex',
    // The map is automatically centered by the user's position.
    mapStateAutoApply: true
}).then(function (result) {
    myMap.geoObjects.add(result.geoObjects);
});
```

2.

```
ymaps.geolocation.get({
    // Setting the option for detecting location by IP provider: 'yandex',
    // Automatically geocoding the result.
    autoReverseGeocode: true
}).then(function (result) {
    // Taking the data resulting from geocoding the object and outputting it to
the console.
    console.log(result.geoObjects.get(0).properties.get('metaDataProperty'));
});
```

geometry

geometry.base

geometry.base.Circle

Extends **IBaseCircleGeometry**.

The "Circle" base geometry.

Constructor | Fields | Events | Methods

Constructor

```
geometry.base.Circle([coordinates[, radius]])
```

Parameters:

Parameter	Default value	Description
coordinates	null	Type: Number[]|null Coordinates of the center of the circle.
radius	0	Type: Number Radius of the circle.

Example:

```
var myCircle = new ymaps.geometry.base.Circle([0, 0], 10);
myCircle.events.add('change', function () {
    alert('Geometry changed');
});
myCircle.freeze();
myCircle.setCoordinates([10, 10]);
myCircle.setRadius(20);
// At this moment, a single event will be generated, and a message will be
output.
myCircle.unfreeze();
```

Fields

Name	Type	Description
events	**event.Manager**	Geometry event manager.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates of the center. • newCoordinates — New coordinates of the center. • oldRadius — Old radius. • newRadius — New radius.

Name	Description
	Inherited from **ICircleGeometryAccess**.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the circle. Inherited from **ICircleGeometryAccess**.
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for a point on the circle closest to the anchorPosition. Inherited from **ICircleGeometryAccess**.
getCoordinates()	Number[]|null	Returns coordinates of the center of the circle. Inherited from **ICircleGeometryAccess**.
getRadius()	Number	Returns radius of the circle. Inherited from **ICircleGeometryAccess**.
getType()	String	Returns the "Circle" string. Inherited from **IBaseCircleGeometry**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
setCoordinates(coordinates)	**ICircleGeometryAccess**	Sets the coordinates of the center of the circle. Inherited from **ICircleGeometryAccess**.
setRadius(radius)	**ICircleGeometryAccess**	Sets the radius of the circle. Inherited from **ICircleGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Fields details

events

```
{event.Manager} events
```

Geometry event manager.

geometry.base.LinearRing

Extends **IBaseLinearRingGeometry**.

The "Closed contour" base geometry.

Constructor | Fields | Events | Methods

Constructor

```
geometry.base.LinearRing([coordinates[, fillRule]])
```

Parameters:

Parameter	Default value	Description
coordinates	[]	Type: Number[][] Geometry coordinates.
fillRule	"evenOdd"	Type: String String ID that defines the polygon fill rule. Accepts one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.

Example:

```
var linearRing = new ymaps.geometry.base.LinearRing([
    [0, 0], [0, 10], [10, 10], [10, 0], [0, 0]
]);
//...
linearRing.set(1, [5, 10]);
```

Fields

Name	Type	Description
events	event.Manager	Geometry event manager.

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. • oldFillRule — Old fill rule. • newFillRule — New fill rule. Inherited from ILinearRingGeometryAccess.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the contour. Inherited from ILinearRingGeometryAccess.
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(index)	Number[]	Returns coordinates of the point with the specified index. Inherited from ILinearRingGeometryAccess.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from IBaseGeometry.
getChildGeometry(index)	IPointGeometryAccess	Creates and returns an IPointGeometryAccess object for the specified contour on the polyline. Inherited from ILinearRingGeometryAccess.
getClosest(anchorPosition)	Object	Searches for a point on the contour closest to the anchorPosition.

Name	Returns	Description
		Inherited from **ILinearRingGeometryAccess**.
getCoordinates()	Number[][]	Returns an array of geometry coordinates. Inherited from **ILinearRingGeometryAccess**.
getFillRule()	String	Returns ID of the fill rule. Inherited from **ILinearRingGeometryAccess**.
getLength()	Integer	Returns the number of points in the geometry. Inherited from **ILinearRingGeometryAccess**.
getType()	String	Returns the "LinearRing" string. Inherited from **IBaseLinearRingGeometry**.
insert(index, coordinates)	**ILinearRingGeometryAccess**	Adds a new point with the specified index. Inherited from **ILinearRingGeometryAccess**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	Number[]	Removes the point with the specified index. Inherited from **ILinearRingGeometryAccess**.
set(index, coordinates)	**ILinearRingGeometryAccess**	Sets coordinates of the point with the specified index. Inherited from **ILinearRingGeometryAccess**.
setCoordinates(coordinates)	**ILinearRingGeometryAccess**	Sets an array of geometry coordinates. Inherited from **ILinearRingGeometryAccess**.
setFillRule(fillRule)	**ILinearRingGeometryAccess**	Sets the contour fill rule. Inherited from **ILinearRingGeometryAccess**.
splice(index, number)	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter. Inherited from **ILinearRingGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Fields details

events

```
{event.Manager} events
```

Geometry event manager.

geometry.base.LinearRing.fromEncodedCoordinates

Static function.

Creates a **geometry.base.LinearRing** geometry based on a string of Base64-encoded coordinates.

Returns a geometry.

```
{                    geometry.base.LinearRing                    }
geometry.base.LinearRing.fromEncodedCoordinates(encodedCoordinates)
```

Parameters:

Parameter	Default value	Description
encodedCoordinates*	—	Type: String The Base64-encoded coordinates of contour vertexes.

* Mandatory parameter/option.

geometry.base.LinearRing.toEncodedCoordinates

Static function.

Returns a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.base.LinearRing.toEncodedCoordinates(geometry)
```

Parameters:

Parameter	Default value	Description
geometry*	—	Type: **geometry.base.LinearRing** Geometry.

* Mandatory parameter/option.

geometry.base.LineString

Extends **IBaseLineStringGeometry**.

The "Polyline" base geometry.

Constructor | Fields | Events | Methods

Constructor

```
geometry.base.LineString([coordinates])
```

Parameters:

Parameter	Default value	Description
coordinates	[]	Type: Number[][] Geometry coordinates.

Example:

```
var lineString = new ymaps.geometry.base.LineString([
    [30, 50], [31, 51], [32, 52]
]);
//...
lineString.set(1, [20, 40]).remove(2);
```

Fields

Name	Type	Description
events	**event.Manager**	Geometry event manager.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. Inherited from **ILineStringGeometryAccess**.

Methods

Name	Returns	Description
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
get(index)	Number[]	Returns coordinates of the point with the specified index. Inherited from **ILineStringGeometryAccess**.

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getChildGeometry(index)	**IPointGeometryAccess**	Creates and returns an **IPointGeometryAccess** object for the specified vertex on the polyline. Inherited from **ILineStringGeometryAccess**.
getClosest(anchorPosition)	Object	Searches for a point on the polyline closest to the anchorPosition. Inherited from **ILineStringGeometryAccess**.
getCoordinates()	Number[][]	Returns an array of geometry coordinates. Inherited from **ILineStringGeometryAccess**.
getLength()	Integer	Returns the number of points in the geometry. Inherited from **ILineStringGeometryAccess**.
getType()	String	Returns the "LineString" string. Inherited from **IBaseLineStringGeometry**.
insert(index, coordinates)	**ILineStringGeometryAccess**	Adds a new point with the specified index. Inherited from **ILineStringGeometryAccess**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	Number[]	Removes the point with the specified index. Inherited from **ILineStringGeometryAccess**.
set(index, coordinates)	**ILineStringGeometryAccess**	Sets coordinates of the point with the specified index. Inherited from **ILineStringGeometryAccess**.
setCoordinates(coordinates)	**ILineStringGeometryAccess**	Sets an array of geometry coordinates. Inherited from **ILineStringGeometryAccess**.

Name	Returns	Description
splice(index, number)	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter. Inherited from **ILineStringGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Fields details

events

```
{event.Manager} events
```

Geometry event manager.

geometry.base.LineString.fromEncodedCoordinates

Static function.

Creates a **geometry.base.LineString** geometry based on a string of Base64-encoded coordinates.

Returns a geometry.

```
{                    geometry.base.LineString                    }
geometry.base.LineString.fromEncodedCoordinates(encodedCoordinates)
```

Parameters:

Parameter	Default value	Description
encodedCoordinates*	—	Type: String Base64-encoded coordinates of polyline points.

* Mandatory parameter/option.

geometry.base.LineString.toEncodedCoordinates

Static function.

Returns a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.base.LineString.toEncodedCoordinates(geometry)
```

Parameters:

Parameter	Default value	Description
geometry*	—	Type: **geometry.base.LineString** Geometry.

* Mandatory parameter/option.

geometry.base.Point

Extends **IBasePointGeometry**.

The "Point" base geometry.

Constructor | Fields | Events | Methods

Constructor

```
geometry.base.Point([coordinates])
```

Parameters:

Parameter	Default value	Description
coordinates	null	Type: Number[]|null Coordinates of a point.

Example:

```
var point = new ymaps.geometry.base.Point([30, 50]);

// This point will always correspond to the map center.
map.events.add('boundschange', function (e) {
    if (e.get('newCenter') != e.get('oldCenter')) {
        point.setCoordinates(e.get('newCenter'));
    }
});
```

Fields

Name	Type	Description
events	**event.Manager**	Geometry event manager.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. Inherited from **IPointGeometryAccess**.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.

Name	Returns	Description
getCoordinates()	Number[]|null	Returns coordinates of a point. Inherited from **IPointGeometryAccess**.
getType()	String	Returns the "Point" string. Inherited from **IBasePointGeometry**.
setCoordinates(coordinates)	**IPointGeometryAccess**	Sets coordinates of a point. Inherited from **IPointGeometryAccess**.

Fields details

events

```
{event.Manager} events
```

Geometry event manager.

geometry.base.Polygon

Extends **IBasePolygonGeometry**.

The "Polygon" base geometry.

Constructor | Fields | Events | Methods

Constructor

```
geometry.base.Polygon([coordinates[, fillRule]])
```

Parameters:

Parameter	Default value	Description
coordinates	[]	Type: Number[][][] Geometry coordinates.
fillRule	"evenOdd"	Type: String

Parameter	Default value	Description
		String ID that defines the polygon fill rule. Accepts one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.

Example:

```
var polygon = new ymaps.geometry.base.Polygon([
    // Outer contour.
    [
        [0, 0], [0, 5], [5, 5], [5, 0], [0, 0]
    ],
    // Inner contour.
    [
        [1, 1], [1, 2], [2, 2], [2, 1], [1, 1]
    ]
]);
```

Fields

Name	Type	Description
events	event.Manager	Geometry event manager.

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. • oldFillRule — Old fill rule. • newFillRule — New fill rule. Inherited from IPolygonGeometryAccess.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the polygon. Inherited from **IPolygonGeometryAccess**.
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
get(index)	Number[][]	Returns coordinates of the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getChildGeometry(index)	**ILinearRingGeometryAccess**	Creates and returns an **ILinearRingGeometryAccess** object for the specified contour. Inherited from **IPolygonGeometryAccess**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the polygon contour. Inherited from **IPolygonGeometryAccess**.
getCoordinates()	Number[][][]	Returns an array of geometry coordinates. Inherited from **IPolygonGeometryAccess**.
getFillRule()	String	Returns ID of the fill rule. Inherited from **IPolygonGeometryAccess**.
getLength()	Integer	Returns the number of contours in the geometry. Inherited from **IPolygonGeometryAccess**.
getType()	String	Returns the "Polygon" string. Inherited from **IBasePolygonGeometry**.
insert(index, path)	**IPolygonGeometryAccess**	Adds a new contour with the specified index. Inherited from **IPolygonGeometryAccess**.

Name	Returns	Description
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	**ILinearRingGeometryAccess**	Removes the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
set(index, path)	**IPolygonGeometryAccess**	Sets coordinates of the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
setCoordinates(coordinates)	**IPolygonGeometryAccess**	Sets an array of geometry coordinates. Inherited from **IPolygonGeometryAccess**.
setFillRule(fillRule)	**IPolygonGeometryAccess**	Sets the polygon's fill rule. Inherited from **IPolygonGeometryAccess**.
splice(index, number)	**ILinearRingGeometryAccess**[]	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter. Inherited from **IPolygonGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Fields details

events

```
{event.Manager} events
```

Geometry event manager.

geometry.base.Polygon.fromEncodedCoordinates

Static function.

Creates a **geometry.base.Polygon** geometry based on a string of Base64-encoded coordinates.

Returns a geometry.

```
{                    geometry.base.Polygon                    }
geometry.base.Polygon.fromEncodedCoordinates(encodedCoordinates)
```

Parameters:

Parameter	Default value	Description
encodedCoordinates*	—	Type: String

Parameter	Default value	Description
		Base64-encoded coordinates of polygon points.

* Mandatory parameter/option.

geometry.base.Polygon.toEncodedCoordinates

Static function.

Returns a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.base.Polygon.toEncodedCoordinates(geometry)
```

Parameters:

Parameter	Default value	Description
geometry*	—	Type: **geometry.base.Polygon** Geometry.

* Mandatory parameter/option.

geometry.base.Rectangle

Extends **IBaseRectangleGeometry**.

The "Rectangle" base geometry.

Constructor | Fields | Events | Methods

Constructor

```
geometry.base.Rectangle([coordinates])
```

Parameters:

Parameter	Default value	Description
coordinates	null	Type: Number[][]|null An array containing coordinates of two opposite corners of the rectangle.

Example:

```
var rectangle = new ymaps.geometry.base.Rectangle([
    [30, 50], [31, 51]
]);
```

Fields

Name	Type	Description
events	**event.Manager**	Geometry event manager.

Events

Name	Description
change	Change to corner coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates of the corners. • newCoordinates — New coordinates of the corners. Inherited from **IRectangleGeometryAccess**.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the rectangle. Inherited from **IRectangleGeometryAccess**.
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the rectangle contour. Inherited from **IRectangleGeometryAccess**.
getCoordinates()	Number[][]	Returns coordinates of two opposite corners of the rectangle. Inherited from **IRectangleGeometryAccess**.
getType()	String	Returns the "Rectangle" string. Inherited from **IBaseRectangleGeometry**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
setCoordinates(coordinates)	**IRectangleGeometryAccess**	Sets the coordinates of two opposite corners of the rectangle. Inherited from **IRectangleGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Fields details

events

```
{event.Manager} events
```

Geometry event manager.

geometry.Circle

Extends **ICircleGeometry**.

The "Circle" geometry.

See Circle

Constructor | Fields | Events | Methods

Constructor

```
geometry.Circle([coordinates[, radius[, options]]])
```

Parameters:

Parameter	Default value	Description
coordinates	null	Type: Number[]|null Coordinates of the center of the circle.
radius	0	Type: Number Radius of the circle in meters.
options	—	Type: Object Geometry options.
options.geodesic	false	Type: Boolean Enables display using geodesic lines.
options.pixelRendering	"jumpy"	Type: String Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: • jumpy — The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved. • static — The shape is always located in the initial world and does not move when the map is moved.
options.projection	—	Type: **IProjection** Projection.

Example:

```
// Creating an instance of the circle geometry class (specifying the coordinates
and radius in meters). var circleGeometry = new ymaps.geometry.Circle([30, 50],
10),
// Creating an instance of the geo object class and passing our geometry to the
constructor.
    circleGeoObject = new ymaps.GeoObject({ geometry: circleGeometry });

// Changing the geometry's radius via the geo object's "geometry" property.
circleGeoObject.geometry.setRadius(5)
// Or directly.
circleGeometry.setRadius(5);
// You can also access circleGeometry via circleGeoObject.geometry.
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates of the center. • newCoordinates — New coordinates of the center. • oldRadius — Old radius. • newRadius — New radius. Inherited from **ICircleGeometryAccess**.
mapchange	Map reference changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldMap — Old map. • newMap — New map. Inherited from **IGeometry**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
pixelgeometrychange	The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelGeometry — New **IPixelGeometry** pixel geometry. Inherited from **IGeometry**.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the circle. Inherited from **ICircleGeometryAccess**.

Name	Returns	Description
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getClosest(anchorPosition)	Object	Searches for a point on the circle closest to the anchorPosition. Inherited from **ICircleGeometryAccess**.
getCoordinates()	Number[]|null	Returns coordinates of the center of the circle. Inherited from **ICircleGeometryAccess**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getRadius()	Number	Returns radius of the circle. Inherited from **ICircleGeometryAccess**.
getType()	String	Returns the "Circle" string. Inherited from **ICircleGeometry**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
setCoordinates(coordinates)	**ICircleGeometryAccess**	Sets the coordinates of the center of the circle. Inherited from **ICircleGeometryAccess**.
setMap(map)		Sets the map. Inherited from **IGeometry**.
setRadius(radius)	**ICircleGeometryAccess**	Sets the radius of the circle. Inherited from **ICircleGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

geometry.json

geometry.json.circle

Extends **IGeometryJson**.

An object that defines the JSON representation of the "Circle" geometry.

Constructor | Fields

Constructor

```
geometry.json.circle()
```

Example:

```
var geometryJson = {
    type: "Circle",
    coordinates: [1, 2],
    radius: 100
};
```

Fields

Name	Type	Description
coordinates	Number[]|null	Coordinates of the center of the circle.
radius	Number	Radius of the circle.
type	String	ID of the "Circle" geometry type. It must always take the value "Circle".

Fields details

coordinates

```
{Number[]|null} coordinates
```

Coordinates of the center of the circle.

radius

```
{Number} radius
```

Radius of the circle.

type

```
{String} type
```

ID of the "Circle" geometry type. It must always take the value "Circle".

geometry.json.lineString

Extends **IGeometryJson**.

An object describing the JSON representation of the "Polyline" geometry.

Constructor | Fields

Constructor

```
geometry.json.lineString()
```

Example:

```
var geometryJson = {
    type: "LineString",
    coordinates: [[1, 2], [3, 5], [7, 11]]
};
```

Fields

Name	Type	Description
coordinates	Number[][]	Coordinates of a polyline.
type	String	Identifier of the "Polyline" geometry type. Must always take the value "LineString".

Fields details

coordinates

```
{Number[][]} coordinates
```

Coordinates of a polyline.

type

```
{String} type
```

Identifier of the "Polyline" geometry type. Must always take the value "LineString".

geometry.json.Point

Extends **IGeometryJson**.

An object that defines the JSON representation of the "Point" geometry.

Constructor | Fields

Constructor

```
geometry.json.Point()
```

Example:

```
var geometryJson = {
    type: "Point",
    coordinates: [1, 2]
};
```

Fields

Name	Type	Description
type	String	ID of the geometry type. Inherited from **IGeometryJson**.

geometry.json.polygon

Extends **IGeometryJson**.

An object that defines the JSON representation of the "Polygon" geometry.

Constructor | Fields

Constructor

```
geometry.json.polygon()
```

Example:

```
var geometryJson = {
        type: "Polygon",
        coordinates: [
            [[0, 0], [7, 11]],
            [[1, 2], [3, 5]]
        ]
    };
```

Fields

Name	Type	Description
coordinates	Number[][][]	Coordinates of the polygon.
fillRule	String	ID of the polygon fill rule. Accepts one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.
type	String	ID of the geometry type. Inherited from **IGeometryJson**.

Fields details

coordinates

```
{Number[][][]} coordinates
```

Coordinates of the polygon.

fillRule

```
{String} fillRule
```

ID of the polygon fill rule. Accepts one of two values:

- evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.
- nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.

geometry.json.rectangle

Extends **IGeometryJson**.

An object that defines the JSON representation of the "Rectangle" geometry.

Constructor | Fields

Constructor

```
geometry.json.rectangle()
```

Example:

```
var geometryJson = {
    type: "Rectangle",
    coordinates: [[1, 2], [3, 5]]
};
```

Fields

Name	Type	Description
coordinates	Number[][]|null	Coordinates of two opposite corners of the rectangle.
type	String	ID of the "Rectangle" geometry type. It must always take the value "Rectangle".

Fields details

coordinates

```
{Number[][]|null} coordinates
```

Coordinates of two opposite corners of the rectangle.

type

```
{String} type
```

ID of the "Rectangle" geometry type. It must always take the value "Rectangle".

geometry.LineString

Extends **ILineStringGeometry**.

"Polyline" geometry.

See Polyline

Constructor | Fields | Events | Methods

Constructor

```
geometry.LineString([coordinates[, options]])
```

Parameters:

Parameter	Default value	Description
coordinates	[]	Type: Number[][] Geometry coordinates.
options	—	Type: Object Geometry options.
options.coordRendering	—	Type: String String ID defining the algorithm for recalculating geometry coordinates as pixel coordinates. For the "Polyline" geometry, can accept one of two values: • shortestPath — Algorithm that considers projection cycling on the axes and generates pixel coordinates so that the distance between two neighboring points is minimal. • straightPath — Algorithm that does not consider projection cycling.
options.geodesic	false	Type: Boolean Enables display using geodesic lines.
options.pixelRendering	"jumpy"	Type: String Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: • jumpy — The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved. • static — The shape is always located in the initial world and does not move when the map is moved.
options.projection	—	Type: **IProjection**

Parameter	Default value	Description
		Projection.
options.simplification	true	Type: Boolean Enables simplification during rendering of a pixel geometry.

Example:

```
// Instantiates the point geometry (specifying coordinates).
var lineStringGeometry = new ymaps.geometry.LineString([
    [30, 50], [31, 51], [32, 52]
]),
// Instantiating the geo object and passing our geometry to the constructor.
    lineStringGeoObject = new ymaps.GeoObject({ geometry: lineStringGeometry });

lineStringGeometry.events.add('change', function (e) {
   alert([e.get('newCoordinates'), e.get('oldCoordinates')]);
});

// Changing vertexes via the geo object's "geometry" property (setting new
coordinates for the second point on the line).
lineStringGeoObject.geometry
    .set(1, [20, 40])
    .remove(2);
// Or directly.
lineStringGeometry
    .set(1, [20, 40])
    .remove(2);
// You can also access lineStringGeometry via lineStringGeoObject.geometry.
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. Inherited from ILineStringGeometryAccess.
mapchange	Map reference changed. Instance of the Event class. Names of fields that are available via the Event.get method: • oldMap — Old map. • newMap — New map. Inherited from IGeometry.
optionschange	Change to the object options. Inherited from ICustomizable.

Name	Description
pixelgeometrychange	The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelGeometry — New **IPixelGeometry** pixel geometry. Inherited from **IGeometry**.

Methods

Name	Returns	Description
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
get(index)	Number[]	Returns coordinates of the point with the specified index. Inherited from **ILineStringGeometryAccess**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getChildGeometry(index)	**IPointGeometryAccess**	Creates and returns an **IPointGeometryAccess** object for the specified vertex on the polyline. Inherited from **ILineStringGeometryAccess**.
getClosest(anchorPosition)	Object	Searches for a point on the polyline closest to the anchorPosition. Inherited from **ILineStringGeometryAccess**.
getCoordinates()	Number[][]	Returns an array of geometry coordinates. Inherited from **ILineStringGeometryAccess**.
getDistance([from[, to]])	Number	Returns the length of the specified line segment, or the whole line, if the delimiter is not set.
getLength()	Integer	Returns the number of points in the geometry. Inherited from **ILineStringGeometryAccess**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getType()	String	Returns the "LineString" string. Inherited from **ILineStringGeometry**.

Name	Returns	Description
insert(index, coordinates)	**ILineStringGeometryAccess**	Adds a new point with the specified index. Inherited from **ILineStringGeometryAccess**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	Number[]	Removes the point with the specified index. Inherited from **ILineStringGeometryAccess**.
set(index, coordinates)	**ILineStringGeometryAccess**	Sets coordinates of the point with the specified index. Inherited from **ILineStringGeometryAccess**.
setCoordinates(coordinates)	**ILineStringGeometryAccess**	Sets an array of geometry coordinates. Inherited from **ILineStringGeometryAccess**.
setMap(map)		Sets the map. Inherited from **IGeometry**.
splice(index, number)	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter. Inherited from **ILineStringGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Methods details

getDistance

```
{Number} getDistance([from[, to]])
```

Returns the length of the specified line segment, or the whole line, if the delimiter is not set.

Parameters:

Parameter	Default value	Description
from	0	Type: Number Specifies the start point for calculating the length.
to	—	Type: Number Specifies the end point for calculating the length. If not specified, the last point is used.

Example:

```
var lineStringGeometry = new ymaps.geometry.LineString([[30, 50], [31, 51], [32,
52]]),
    geoObject = new ymaps.GeoObject({ geometry: lineStringGeometry });
myMap.geoObjects.add(geoObject);
// The total length of the line.
console.log(geoObject.geometry.getDistance());
// The length of the segment from the first to the second point.
console.log(geoObject.geometry.getDistance(0, 1));
```

geometry.LineString.fromEncodedCoordinates

Static function.

Creates a **geometry.LineString** geometry based on a string of Base64-encoded coordinates.

Returns a geometry.

```
{                    geometry.LineString                    }
geometry.LineString.fromEncodedCoordinates(encodedCoordinates)
```

Parameters:

Parameter	Default value	Description
encodedCoordinates*	—	Type: String Base64-encoded coordinates of polyline points.

* Mandatory parameter/option.

Example:

```
var base64Coords = "gMPJAYDw-gJAQg8AQEIPAEBCDwBAQg8A",
    geometry = ymaps.geometry.LineString.fromEncodedCoordinates(base64Coords);
    polyline = new ymaps.Polyline(geometry);
```

geometry.LineString.toEncodedCoordinates

Static function.

Returns a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.LineString.toEncodedCoordinates(geometry)
```

Parameters:

Parameter	Default value	Description
geometry*	—	Type: **geometry.LineString** Geometry.

* Mandatory parameter/option.

geometry.pixel

geometry.pixel.Circle

Extends **IPixelCircleGeometry**.

The "Circle" pixel geometry.

Constructor | Fields | Methods

Constructor

```
geometry.pixel.Circle(coordinates, radius[, metaData])
```

Parameters:

Parameter	Default value	Description
coordinates *	—	Type: Number[]|null Coordinates of the center of the circle.
radius *	—	Type: Number|null Radius of the circle.
metaData	—	Type: Object Metadata.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getCoordinates()	Number[]	Returns coordinates of the center of the circle. Inherited from **IPixelCircleGeometry**.

Name	Returns	Description
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getRadius()	Number	Returns radius of the circle. Inherited from **IPixelCircleGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

geometry.pixel.LineString

Extends **IPixelLineStringGeometry**.

The "Polyline" pixel geometry.

Constructor | Fields | Methods

Constructor

```
geometry.pixel.LineString(coordinates[, metaData])
```

Parameters:

Parameter	Default value	Description
`coordinates` *	—	Type: Number[][] Coordinates of a line.
`metaData`	—	Type: Object Metadata.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for a point on the polyline closest to the anchorPosition. Inherited from **IPixelLineStringGeometry**.
getCoordinates()	Number[][]	Returns coordinates of a line. Inherited from **IPixelLineStringGeometry**.
getLength()	Integer	Returns the number of points in the geometry. Inherited from **IPixelLineStringGeometry**.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

geometry.pixel.MultiLineString

Extends **IPixelMultiLineGeometry**.

The "Multiline" pixel geometry.

Constructor | Fields | Methods

Constructor

```
geometry.pixel.MultiLineString(coordinates[, metaData])
```

Parameters:

Parameter	Default value	Description
`coordinates` *	—	Type: Number[][][] Line coordinates.
`metaData`	—	Type: Object

Parameter	Default value	Description
		Metadata.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for a point on the contour closest to the anchorPosition. Inherited from **IPixelMultiLineGeometry**.
getCoordinates()	Number[][][]	Returns coordinates of a multiline. Inherited from **IPixelMultiLineGeometry**.
getLength()	Integer	Returns the number of lines in the multiline. Inherited from **IPixelMultiLineGeometry**.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

geometry.pixel.MultiPolygon

Extends **IPixelMultiPolygonGeometry**.

The "Polygon from multiple shapes" pixel geometry.

Constructor | Fields | Methods

Constructor

```
geometry.pixel.MultiPolygon(coordinates, fillRule[, metaData])
```

Parameters:

Parameter	Default value	Description
coordinates *	—	Type: Number[][][][] Coordinates of the polygons.
fillRule *	—	Type: String String ID that defines the fill rule for the polygons. Accepts one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.
metaData	—	Type: Object Metadata.
metaData.convex	false	Type: Boolean Convex indicator for a polygon. If true, it is convex; if false, it is not. For convex polygons, it is faster to calculate whether points fall in the polygon.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the multipolygon. Inherited from IPixelMultiPolygonGeometry.
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from IPixelGeometry.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from IBaseGeometry.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the multipolygon contour. Inherited from IPixelMultiPolygonGeometry.
getCoordinates()	Number[][][][]	Returns coordinates of the multipolygon. Inherited from IPixelMultiPolygonGeometry.

Name	Returns	Description
getFillRule()	String	Returns the string ID that defines the multipolygon fill rule. The ID can have one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it. . Inherited from **IPixelMultiPolygonGeometry**.
getLength()	Integer	Returns the number of polygons in the multipolygon. Inherited from **IPixelMultiPolygonGeometry**.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

geometry.pixel.Point

Extends **IPixelPointGeometry**.

The "Point" pixel geometry.

Constructor | Fields | Methods

Constructor

```
geometry.pixel.Point(position[, metaData])
```

Parameters:

Parameter	Default value	Description
position *	—	Type: Number[]|null Coordinates of a point.
metaData	—	Type: Object Metadata.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getCoordinates()	Number[]	Returns coordinates of a point. Inherited from **IPixelPointGeometry**.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

geometry.pixel.Polygon

Extends **IPixelPolygonGeometry**.

The "Polygon" pixel geometry.

Constructor | Fields | Methods

Constructor

```
geometry.pixel.Polygon(coordinates, fillRule[, metaData])
```

Parameters:

Parameter	Default value	Description
coordinates *	—	Type: Number[][][] Coordinates of the polygon.
fillRule *	—	Type: String String ID that defines the polygon fill rule. Accepts one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.
metaData	—	Type: Object Metadata.
metaData.convex	false	Type: Boolean Convex indicator for a polygon. If true, it is convex; if false, it is not. For convex polygons, it is faster to calculate whether points fall in the polygon.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the polygon. Inherited from IPixelPolygonGeometry.
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from IPixelGeometry.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from IBaseGeometry.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the polygon contour. Inherited from IPixelPolygonGeometry.
getCoordinates()	Number[][][]	Returns coordinates of the polygon. Inherited from IPixelPolygonGeometry.

Name	Returns	Description
getFillRule()	String	Returns string ID that defines the polygon fill rule. The ID accepts one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it. . Inherited from **IPixelPolygonGeometry**.
getLength()	Integer	Returns the number of contours in the polygon. Inherited from **IPixelPolygonGeometry**.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

geometry.pixel.Rectangle

Extends **IPixelRectangleGeometry**.

The "Rectangle" pixel geometry.

Constructor | Fields | Methods

Constructor

```
geometry.pixel.Rectangle([coordinates[, metaData]])
```

Parameters:

Parameter	Default value	Description
coordinates	null	Type: Number[][]|null Coordinates of two opposite corners of the rectangle.
metaData	—	Type: Object Metadata.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the rectangle. Inherited from **IPixelRectangleGeometry**.
getCoordinates()	Number[][]	Returns coordinates of two opposite corners of the rectangle. Inherited from **IPixelRectangleGeometry**.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.

Name	Returns	Description
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

geometry.Point

Extends **IPointGeometry**.

The "Point" geometry.

See Placemark

Constructor | Fields | Events | Methods

Constructor

```
geometry.Point([position[, options]])
```

Parameters:

Parameter	Default value	Description
position	null	Type: Number[] Coordinates of a point.
options	—	Type: Object Geometry options.
options.pixelRendering	"jumpy"	Type: String Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: • jumpy — The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved. • static — The shape is always located in the initial world and does not move when the map is moved.
options.projection	—	Type: **IProjection** Projection.

Example:

```
// Creating an instance of the point geometry (specifying coordinates).
var pointGeometry = new ymaps.geometry.Point([30, 50]),
// Instantiating the geo object and passing our geometry to the constructor.
    placemark = new ymaps.GeoObject({ geometry: pointGeometry });

// Changing the vertexes via the geo object's "geometry" property.
placemark.geometry.setCoordinates([20, 40]);
// Or directly.
pointGeometry.setCoordinates([20, 40]);
// You can also access pointGeometry via placemark.geometry
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager.

Name	Type	Description
		Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. Inherited from **IPointGeometryAccess**.
mapchange	Map reference changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldMap — Old map. • newMap — New map. Inherited from **IGeometry**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
pixelgeometrychange	The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelGeometry — New **IPixelGeometry** pixel geometry. Inherited from **IGeometry**.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getCoordinates()	Number[]|null	Returns coordinates of a point. Inherited from **IPointGeometryAccess**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getType()	String	Returns the "Point" string. Inherited from **IPointGeometry**.

Name	Returns	Description
setCoordinates(coordinates)	**IPointGeometryAccess**	Sets coordinates of a point. Inherited from **IPointGeometryAccess**.
setMap(map)		Sets the map. Inherited from **IGeometry**.

geometry.Polygon

Extends **IPolygonGeometry**.

The "Polygon" geometry.

See Polygon

Constructor | Fields | Events | Methods

Constructor

```
geometry.Polygon([coordinates[, fillRule[, options]]])
```

Parameters:

Parameter	Default value	Description
coordinates	[]	Type: Number[][][] Geometry coordinates. A three-dimensional array of elements that are each two-dimensional coordinates of the polygon contours. The first element describes the outer contour, and the rest describe the inner contours.
fillRule	"evenOdd"	Type: String

Parameter	Default value	Description
		String ID that defines the polygon fill rule. Accepts one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.
options	—	Type: Object Geometry options.
options.coordRendering	"shortestPath"	Type: String String ID defining the algorithm for recalculating geometry coordinates as pixel coordinates. Accepts one of two values: • shortestPath — Algorithm that considers projection cycling on the axes and generates pixel coordinates so that the distance between two neighboring points is minimal. • straightPath — Algorithm that does not consider projection cycling.
options.geodesic	false	Type: Boolean Enables display using geodesic lines.
options.pixelRendering	"jumpy"	Type: String

Parameter	Default value	Description
		Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: • jumpy — The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved. • static — The shape is always located in the initial world and does not move when the map is moved.
options.projection	—	Type: **IProjection** Projection.
options.simplification	true	Type: Boolean Enables simplification during rendering of a pixel geometry.

Example:

```
// Creating an instance of the polygon geometry (specifying coordinates of
vertexes on contours).
var polygonGeometry = new ymaps.geometry.Polygon([
    // Outer contour.
    [
        [0, 0], [0, 5], [5, 5], [5, 0], [0, 0]
    ],
    // Inner contour.
    [
        [1, 1], [1, 2], [2, 2], [2, 1], [1, 1]
    ]
]),
// Creating a geo object instance and passing our geometry to the constructor.
    polygonGeoObject = new ymaps.GeoObject({ geometry: polygonGeometry });

// You can also access polygonGeometry via polygonGeoObject.geometry.
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. • oldFillRule — Old fill rule. • newFillRule — New fill rule. Inherited from **IPolygonGeometryAccess**.
mapchange	Map reference changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldMap — Old map. • newMap — New map. Inherited from **IGeometry**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
pixelgeometrychange	The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelGeometry — New **IPixelGeometry** pixel geometry. Inherited from **IGeometry**.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the polygon. Inherited from **IPolygonGeometryAccess**.
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
get(index)	Number[][]	Returns coordinates of the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getChildGeometry(index)	**ILinearRingGeometryAccess**	Creates and returns an **ILinearRingGeometryAccess** object for the specified contour. Inherited from **IPolygonGeometryAccess**.

Name	Returns	Description
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the polygon contour. Inherited from **IPolygonGeometryAccess**.
getCoordinates()	Number[][][]	Returns an array of geometry coordinates. Inherited from **IPolygonGeometryAccess**.
getFillRule()	String	Returns ID of the fill rule. Inherited from **IPolygonGeometryAccess**.
getLength()	Integer	Returns the number of contours in the geometry. Inherited from **IPolygonGeometryAccess**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getType()	String	Returns the "Polygon" string. Inherited from **IPolygonGeometry**.
insert(index, path)	**IPolygonGeometryAccess**	Adds a new contour with the specified index. Inherited from **IPolygonGeometryAccess**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	**ILinearRingGeometryAccess**	Removes the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
set(index, path)	**IPolygonGeometryAccess**	Sets coordinates of the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
setCoordinates(coordinates)	**IPolygonGeometryAccess**	Sets an array of geometry coordinates. Inherited from **IPolygonGeometryAccess**.
setFillRule(fillRule)	**IPolygonGeometryAccess**	Sets the polygon's fill rule. Inherited from **IPolygonGeometryAccess**.
setMap(map)		Sets the map.

Name	Returns	Description
		Inherited from **IGeometry**.
splice(index, number)	**ILinearRingGeometryAccess**[]	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter. Inherited from **IPolygonGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

geometry.Polygon.fromEncodedCoordinates

Static function.

Creates a **geometry.Polygon** geometry based on a string of Base64-encoded coordinates.

Returns a geometry.

```
{ geometry.Polygon } geometry.Polygon.fromEncodedCoordinates(encodedCoordinates)
```

Parameters:

Parameter	Default value	Description
encodedCoordinates*	—	Type: String Base64-encoded coordinates of polyline points.

* Mandatory parameter/option.

Example:

```
var base64Coords =
"AAAAAAAAAAAAAAAAQEtMAEBLTAAAAAAAAAAAAMC0s__AtLP_AAAAAA==;QEIPAEBCDwAAAAAQEIPAE
BCDwAAAAAAAAAAAMC98P_AvfD_AAAAAA==",
    geometry = ymaps.geometry.Polygon.fromEncodedCoordinates(base64Coords);
    polygon = new ymaps.Polygon(geometry);
```

geometry.Polygon.toEncodedCoordinates

Static function.

Returns a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.Polygon.toEncodedCoordinates(geometry)
```

Parameters:

Parameter	Default value	Description
geometry*	—	Type: **geometry.Polygon** Geometry.

* Mandatory parameter/option.

geometry.Rectangle

Extends **IRectangleGeometry**.

The "Rectangle" geometry.

See Rectangle

Constructor | Fields | Events | Methods

Constructor

```
geometry.Rectangle([coordinates[, options]])
```

Parameters:

Parameter	Default value	Description
coordinates	null	Type: Number[][]|null An array containing coordinates of two opposite corners of the rectangle.
options	—	Type: Object Geometry options.
options.coordRendering	—	Type: String String ID defining the algorithm for recalculating geometry coordinates as pixel coordinates. For the "Rectangle" geometry, can accept one of three values: • shortestPath — Algorithm that considers projection cycling on the axes and generates pixel coordinates so that the distance between opposite corners is minimal. • straightPath — Algorithm that does not consider projection cycling. • boundsPath — Algorithm that interprets the coordinates of rectangle corners as coordinates corresponding to the lower and upper corners of a bounding area. When calculating diagonals for projections with cycled axes, the counter-clockwise direction is always used.
options.geodesic	false	Type: Boolean Enables display using geodesic lines.
options.pixelRendering	"jumpy"	Type: String

Parameter	Default value	Description
		Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: • jumpy — The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved. • static — The shape is always located in the initial world and does not move when the map is moved.
options.projection	—	Type: **IProjection** Projection.

Example:

```
// Creating an instance of the point geometry (specifying coordinates).
var rectangleGeometry = new ymaps.geometry.Rectangle([[30, 50], [31, 51]]),
// Instantiating the geo object and passing our geometry to the constructor.
    rectangleGeoObject = new ymaps.GeoObject({ geometry: rectangleGeometry });

// Changing the coordinates via the geo object's "geometry" property.
rectangleGeoObject.geometry.setCoordinates([[10, 20], [51, 71]]);
// Or directly.
rectangleGeometry.setCoordinates([[10, 20], [51, 71]]);
// You can also access rectangleGeometry via rectangleGeoObject.geometry.
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
change	Change to corner coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates of the corners. • newCoordinates — New coordinates of the corners. Inherited from **IRectangleGeometryAccess**.
mapchange	Map reference changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldMap — Old map. • newMap — New map. Inherited from **IGeometry**.
optionschange	Change to the object options.

Name	Description
	Inherited from **ICustomizable**.
pixelgeometrychange	The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelGeometry — New **IPixelGeometry** pixel geometry. Inherited from **IGeometry**.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the rectangle. Inherited from **IRectangleGeometryAccess**.
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the rectangle contour. Inherited from **IRectangleGeometryAccess**.
getCoordinates()	Number[][]	Returns coordinates of two opposite corners of the rectangle. Inherited from **IRectangleGeometryAccess**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getType()	String	Returns the "Rectangle" string. Inherited from **IRectangleGeometry**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
setCoordinates(coordinates)	**IRectangleGeometryAccess**	Sets the coordinates of two opposite corners of the rectangle. Inherited from **IRectangleGeometryAccess**.
setMap(map)		Sets the map.

Name	Returns	Description
		Inherited from **IGeometry**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

geometryEditor

geometryEditor.Circle

Extends **IGeometryEditor**.

The "Circle" geometry editor.

Constructor | Fields | Events | Methods

Constructor

```
geometryEditor.Circle(geometry[, options])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **ICircleGeometry** The "Circle" geometry.
options	—	Type: Object Options for the geometry editor.
options.drawingCursor	"arrow"	Type: Boolean The mouse cursor in drawing mode.
options.drawOver	true	Type: Boolean Allows to put points on top of map objects in drawing mode.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**IGeometry**	The geometry being edited. Inherited from **IGeometryEditor**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**IDataManager**	Manager for the state of the geometry editor. Data fields that are available via the "get" and "set" methods: • editing — Checks whether editing mode is enabled. Type — Boolean. Default value — false. • drawing — Checks whether drawing mode is enabled. Type — Boolean. Default value — false.

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.
statechange	Change to the geometry editor state. Instance of the Event class. Inherited from IGeometryEditor.

Methods

Name	Returns	Description
startDrawing()	vow.Promise	Enables the mode for drawing a circle.
startEditing()		Enables editing mode. Inherited from IGeometryEditor.
stopDrawing()	vow.Promise	Disables the mode for drawing a circle.
stopEditing()		Disables editing mode. Inherited from IGeometryEditor.

Fields details

state

```
{IDataManager} state
```

Manager for the state of the geometry editor.

Data fields that are available via the "get" and "set" methods:

- editing — Checks whether editing mode is enabled. Type — Boolean. Default value — false.
- drawing — Checks whether drawing mode is enabled. Type — Boolean. Default value — false.

Methods details

startDrawing

```
{vow.Promise} startDrawing()
```

Enables the mode for drawing a circle.

Returns Promise object.

stopDrawing

```
{vow.Promise} stopDrawing()
```

Disables the mode for drawing a circle.

Returns Promise object.

geometryEditor.LineString

Extends IGeometryEditor.

The "Polyline" geometry editor.

Constructor | Fields | Events | Methods

Constructor

```
geometryEditor.LineString(geometry[, options])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: ILineStringGeometry "Polyline" geometry.
options	—	Type: Object Options for the geometry editor.
options.dblClickHandler	—	Type: Function Handler for double-clicking a vertex. Accepts a reference to a model of the vertex being edited. By default, the handler is defined by the function that removes the corresponding vertex.
options.drawingCursor	"arrow"	Type: Boolean The mouse cursor in the mode for adding new vertexes.
options.drawOver	true	Type: Boolean Allows to put points on top of map objects in the mode for adding new vertexes.
options.edgeInteractiveOptions	true	Type: Boolean Allows the intermediate vertices placemarks to use options with postfixes linked to the current state of the placemark. The following postfixes are available: • Hover — Options with this given postfix are used when the user hovers over a placemark with the mouse pointer. • Drag — Options with this given postfix are used when the user drags a placemark. Examples of such options: edgeLayoutHover, edgeIconImageSizeActive, edgeIconImageShapeHover, etc. If you do not want to change the options of intermediate vertices placemarks depending on their state, then you will need to disable this option.
options.edgeLayout	—	Type: Function

Parameter	Default value	Description
		Class of the layout for interim placemarks.
options.maxPoints	Infinity	Type: Number The maximum allowable number of vertexes on a polyline.
options.menuManager	—	Type: Function Context menu dispatcher for the menu that opens when clicking on a vertex. Accepts two arguments: • An array of objects describing the context menu items for this vertex. • A reference to a model of the vertex being edited. The dispatcher can change data in the passed array. Must return an array of objects that describe context menu items.
options.minPoints	0	Type: Number The minimum allowable number of vertexes on a polyline.
options.useAutoPanInDrawing	true	Type: Boolean Enables autopan of the map when dragging a vertex on the boundary.
options.useMapMarginInDrawing	true	Type: Boolean Whether to use map margins in drawing mode.
options.vertexInteractiveOptions	true	Type: Boolean Allows the vertices placemarks to use options with postfixes linked to the current state of the vertex. The following postfixes are available: • Hover — Options with this given postfix are used when the user hovers over a vertex with the mouse pointer. • Drag — Options with this given postfix are used when the user drags a vertex. • Active — Options with this given postfix are used when the context menu is opened for the vertex.

Parameter	Default value	Description
		Examples of such options: vertexLayoutHover, vertexIconImageSizeActive, vertexIconImageShapeHover, etc. If you do not want to change the options of vertices placemarks depending on their state, then you will need to disable this option.
`options.vertexLayout`	—	Type: Function Class of the layout for placemarks on polyline vertexes.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**IGeometry**	The geometry being edited. Inherited from **IGeometryEditor**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**IDataManager**	Manager for the state of the geometry editor. Data fields that are available via the "get" and "set" methods: • editing — Checks whether editing mode is enabled. Type — Boolean. Default value — false. • drawing — Checks whether vertex drawing mode is enabled. Type — Boolean. Default value — false. • drawingFrom — Checks how new points are added in drawing mode. Accepts one of two string values: "begin" — points are added at the beginning of the polyline; "end" — points are added at the end. Default value — "end".

Events

Name	Description
beforeedgedrag	Event preceding the "edgedrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset — Array of two numbers that describe the pixel offset at this step. • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

Name	Description
	If the **Event.preventDefault** method is called for this event, a subsequent "edgedrag" event will be canceled.
beforeedgedragstart	Event preceding the "edgedragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent — Source DOM event (as a **DomEvent** object), if there is one. • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "edgedragstart" event, will be canceled.
beforevertexadd	Event preceding the "vertexadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • parentModel — Reference to the parent data model for the vertex being added. • vertexIndex — Index of the vertex being added. • globalPixels — Coordinates of the vertex being added, in global pixel coordinates. Names of methods that are accessible via **Event.callMethod**: • setGlobalPixels — Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "vertexadd" event will be canceled.
beforevertexdrag	Event preceding the "vertexdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset — Array of two numbers that describe the pixel offset at this step. • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "vertexdrag" event will be canceled.
beforevertexdragstart	Event preceding the "vertexdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent — Source DOM event (as a **DomEvent** object), if there is one. • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the"vertexdragstart" event, will be canceled.

Name	Description
beforevertexdraw	Event preceding the "vertexdraw" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • parentModel — Reference to the parent data model for the vertex being added. • vertexIndex — Index of the vertex being added. • globalPixels — Coordinates of the vertex being added, in global pixel coordinates. Names of methods that are accessible via **Event.callMethod**: • setGlobalPixels — Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "vertexdraw" event will be canceled.
drawingstart	Enabling the mode for adding new vertexes. Instance of the **Event** class.
drawingstop	Disabling the mode for adding new vertexes. Instance of the **Event** class.
edgedrag	Dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset — Array of two numbers that describe the pixel offset at this step. • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.
edgedragend	End of dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.
edgedragstart	Start of dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent — Source DOM event (as a **DomEvent** object), if there is one. • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.
editingstart	Enabling the mode for editing vertexes. Instance of the **Event** class.
editingstop	Disabling the mode for editing vertexes. Instance of the **Event** class.
framingstart	Enabling the zoom mode. Instance of the **Event** class.
framingstop	Disabling the zoom mode. Instance of the **Event** class.
optionschange	Change to the object options. Inherited from **ICustomizable**.
statechange	Change to the geometry editor state. Instance of the **Event** class. Inherited from **IGeometryEditor**.

Name	Description
vertexadd	Adding a new vertex. Instance of the Event class. Names of fields that are available via the Event.get method: • parentModel — Reference to the parent data model for the vertex that was added. • vertexIndex — Index of the vertex that was added. • globalPixels — Coordinates of the vertex that was added, in global pixel coordinates.
vertexdrag	Dragging a vertex. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelOffset — Array of two numbers that describe the pixel offset at this step. • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.
vertexdragend	End of vertex dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.
vertexdragstart	Start of vertex dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • domEvent — Source DOM event (as a DomEvent object), if there is one. • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.
vertexdraw	Drawing a new vertex. This event usually precedes the add vertex event, and occurs when the mouse moves and the mode for adding new vertexes is enabled. Based on data passed in this event, guide lines are displayed in vertex drawing mode. Instance of the Event class. Names of fields that are available via the Event.get method: • parentModel — Reference to the parent data model for the vertex being added. • vertexIndex — Index of the vertex being added. • globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Methods

Name	Returns	Description
getModel()	vow.Promise	Returns the promise object, which is confirmed by the model object at the time it is actually created, or is rejected with one of the following error messages: • Canceled — Editing mode is disabled until the model is actually created. • Editor wasn't started — Editing mode is not enabled. .

Name	Returns	Description
getModelSync()	geometryEditor.model.RootLineString|null	Returns the editor's data model, or null if it is missing at the time of the call.
getView()	vow.Promise	Returns the promise object, which is confirmed by the representation object at the time it is actually created, or is rejected with one of the following error messages: • Canceled — Editing mode is disabled until the representation is actually created. • Editor wasn't started — Editing mode is not enabled. .
getViewSync()	geometryEditor.view.Path|null	Returns the editor's representation, or null if it is missing at the time of the call.
startDrawing()	vow.Promise	Enables vertex drawing mode for a polyline. Enabling occurs asynchronously.
startEditing()	vow.Promise	Enables editing mode. Enabling occurs asynchronously.
startFraming()	vow.Promise	Enables the zoom mode for a polyline. Enabling occurs asynchronously.
stopDrawing()		Disables vertex drawing mode for a polyline.
stopEditing()		Disables editing mode.
stopFraming()		Disables the zoom mode.

Fields details

state

```
{IDataManager} state
```

Manager for the state of the geometry editor.

Data fields that are available via the "get" and "set" methods:

- editing — Checks whether editing mode is enabled. Type — Boolean. Default value — false.
- drawing — Checks whether vertex drawing mode is enabled. Type — Boolean. Default value — false.
- drawingFrom — Checks how new points are added in drawing mode. Accepts one of two string values: "begin" — points are added at the beginning of the polyline; "end" — points are added at the end. Default value — "end".

Events details

beforeedgedrag

Event preceding the "edgedrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "edgedrag" event will be canceled.

beforeedgedragstart

Event preceding the "edgedragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent — Source DOM event (as a **DomEvent** object), if there is one.
- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "edgedragstart" event, will be canceled.

beforevertexadd

Event preceding the "vertexadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- parentModel — Reference to the parent data model for the vertex being added.
- vertexIndex — Index of the vertex being added.
- globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Names of methods that are accessible via **Event.callMethod**:

- setGlobalPixels — Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "vertexadd" event will be canceled.

Example:

```
// Correcting the coordinates of the vertex add event so that they stay within a
square with sides that are
// 100 pixels that is centered on the map center.
polyline.editor.events.add(["beforevertexdraw", "beforevertexadd"], function
(event) {
    var mapGlobalPixelCenter = geoMap.getGlobalPixelCenter(),
        globalPixels = event.get("globalPixels"),
        pixelBounds = [
            [mapGlobalPixelCenter[0] - 100, mapGlobalPixelCenter[1] - 100],
            [mapGlobalPixelCenter[0] + 100, mapGlobalPixelCenter[1] + 100]
        ];
    event.callMethod("setGlobalPixels", [
        Math.max(Math.min(globalPixels[0], pixelBounds[1][0]), pixelBounds[0]
[0]),
        Math.max(Math.min(globalPixels[1], pixelBounds[1][1]), pixelBounds[0]
[1])
    ]);
});
```

beforevertexdrag

Event preceding the "vertexdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "vertexdrag" event will be canceled.

Example:

```
// Inverting the offset when dragging the vertex.
polyline.editor.events.add("beforevertexdrag", function (event) {
    var pixelOffset = event.get("pixelOffset");
    event.callMethod("setPixelOffset", [-pixelOffset[0], -pixelOffset[1]]);
});
```

beforevertexdragstart

Event preceding the "vertexdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent — Source DOM event (as a **DomEvent** object), if there is one.
- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the"vertexdragstart" event, will be canceled.

beforevertexdraw

Event preceding the "vertexdraw" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- parentModel — Reference to the parent data model for the vertex being added.
- vertexIndex — Index of the vertex being added.
- globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Names of methods that are accessible via **Event.callMethod**:

- setGlobalPixels — Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "vertexdraw" event will be canceled.

drawingstart

Enabling the mode for adding new vertexes. Instance of the **Event** class.

drawingstop

Disabling the mode for adding new vertexes. Instance of the **Event** class.

edgedrag

Dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

edgedragend

End of dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

edgedragstart

Start of dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent — Source DOM event (as a **DomEvent** object), if there is one.
- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

editingstart

Enabling the mode for editing vertexes. Instance of the **Event** class.

editingstop

Disabling the mode for editing vertexes. Instance of the **Event** class.

framingstart

Enabling the zoom mode. Instance of the **Event** class.

framingstop

Disabling the zoom mode. Instance of the **Event** class.

vertexadd

Adding a new vertex. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- parentModel — Reference to the parent data model for the vertex that was added.
- vertexIndex — Index of the vertex that was added.
- globalPixels — Coordinates of the vertex that was added, in global pixel coordinates.

vertexdrag

Dragging a vertex. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

vertexdragend

End of vertex dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

vertexdragstart

Start of vertex dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent — Source DOM event (as a **DomEvent** object), if there is one.
- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

vertexdraw

Drawing a new vertex. This event usually precedes the add vertex event, and occurs when the mouse moves and the mode for adding new vertexes is enabled. Based on data passed in this event, guide lines are displayed in vertex drawing mode. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- parentModel — Reference to the parent data model for the vertex being added.
- vertexIndex — Index of the vertex being added.
- globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Methods details

getModel

```
{vow.Promise} getModel()
```

Returns the promise object, which is confirmed by the model object at the time it is actually created, or is rejected with one of the following error messages:

- Canceled — Editing mode is disabled until the model is actually created.
- Editor wasn't started — Editing mode is not enabled.

.

getModelSync

```
{geometryEditor.model.RootLineString|null} getModelSync()
```

Returns the editor's data model, or null if it is missing at the time of the call.

getView

```
{vow.Promise} getView()
```

Returns the promise object, which is confirmed by the representation object at the time it is actually created, or is rejected with one of the following error messages:

- Canceled — Editing mode is disabled until the representation is actually created.
- Editor wasn't started — Editing mode is not enabled.

.

getViewSync

```
{geometryEditor.view.Path|null} getViewSync()
```

Returns the editor's representation, or null if it is missing at the time of the call.

startDrawing

```
{vow.Promise} startDrawing()
```

Enables vertex drawing mode for a polyline. Enabling occurs asynchronously.

Returns the promise object, which is confirmed when drawing mode has actually started, or is rejected with one of the following error messages:

- Canceled — Drawing mode is disabled until it is actually started.

.

startEditing

```
{vow.Promise} startEditing()
```

Enables editing mode. Enabling occurs asynchronously.

Returns the promise object, which is confirmed when editing mode has actually started, or is rejected with one of the following error messages:

- Canceled — Editing mode is disabled until it is actually started.

.

startFraming

```
{vow.Promise} startFraming()
```

Enables the zoom mode for a polyline. Enabling occurs asynchronously.

Returns the promise object that is confirmed when zoom mode actually starts.

stopDrawing

```
{} stopDrawing()
```

Disables vertex drawing mode for a polyline.

stopEditing

```
{} stopEditing()
```

Disables editing mode.

stopFraming

```
{} stopFraming()
```

Disables the zoom mode.

geometryEditor.model

geometryEditor.model.ChildLinearRing

Note:
The constructor of the geometryEditor.model.ChildLinearRing class is hidden, as this class is not intended for autonomous initialization.

Extends **geometryEditor.model.ChildLineString**.

Model for a child closed contour. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Methods

Fields

Name	Type	Description
editor	**IGeometryEditor**	Geometry editor. Inherited from **IGeometryEditorChildModel**.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**IBaseGeometry**	Geometry of the model. Inherited from **IGeometryEditorChildModel**.

Methods

Name	Returns	Description
destroy()		Destructor. Inherited from **IGeometryEditorModel**.

Name	Returns	Description
getAllVerticesNumber()	Integer	Returns the total number of vertexes in the geometry being edited. Inherited from **geometryEditor.model.ChildLineString**.
getEdgeModels()	**geometryEditor.model.Edge**[]	Returns an array of models for interim placemarks. Inherited from **geometryEditor.model.ChildLineString**.
getIndex()	Integer	Returns the index of the child polyline in the parent model. Inherited from **geometryEditor.model.ChildLineString**.
getParent()	**IGeometryEditorModel**	Returns the parent data model. Inherited from **IGeometryEditorChildModel**.
getPixels()	Number[]	Returns the model's pixel data. Inherited from **IGeometryEditorModel**.
getVertexModels()	**geometryEditor.model.ChildVertex**[]	Returns an array of models for child vertexes. Inherited from **geometryEditor.model.ChildLineString**.
setIndex(index)		Sets the index of the child polyline in the parent model. Inherited from **geometryEditor.model.ChildLineString**.
setPixels(pixels)		Sets the model's pixel data. Inherited from **IGeometryEditorChildModel**.
spliceVertices(start, deleteCount)	Number[][]	Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter. Inherited from **geometryEditor.model.ChildLineString**.

geometryEditor.model.ChildLineString

Note:
The constructor of the geometryEditor.model.ChildLineString class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeometryEditorChildModel**.

Model of the child polyline. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Methods

Fields

Name	Type	Description
editor	**IGeometryEditor**	Geometry editor. Inherited from **IGeometryEditorChildModel**.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**IBaseGeometry**	Geometry of the model. Inherited from **IGeometryEditorChildModel**.

Methods

Name	Returns	Description
destroy()		Destructor. Inherited from **IGeometryEditorModel**.
getAllVerticesNumber()	Integer	Returns the total number of vertexes in the geometry being edited.
getEdgeModels()	**geometryEditor.model.Edge**[]	Returns an array of models for interim placemarks.
getIndex()	Integer	Returns the index of the child polyline in the parent model.
getParent()	**IGeometryEditorModel**	Returns the parent data model. Inherited from **IGeometryEditorChildModel**.
getPixels()	Number[]	Returns the model's pixel data. Inherited from **IGeometryEditorModel**.
getVertexModels()	**geometryEditor.model.ChildVertex**[]	Returns an array of models for child vertexes.
setIndex(index)		Sets the index of the child polyline in the parent model.
setPixels(pixels)		Sets the model's pixel data.

Name	Returns	Description
		Inherited from IGeometryEditorChildModel.
spliceVertices(start, deleteCount)	Number[][]	Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.

Methods details

getAllVerticesNumber

```
{Integer} getAllVerticesNumber()
```

Returns the total number of vertexes in the geometry being edited.

getEdgeModels

```
{geometryEditor.model.Edge[]} getEdgeModels()
```

Returns an array of models for interim placemarks.

getIndex

```
{Integer} getIndex()
```

Returns the index of the child polyline in the parent model.

getVertexModels

```
{geometryEditor.model.ChildVertex[]} getVertexModels()
```

Returns an array of models for child vertexes.

setIndex

```
{} setIndex(index)
```

Sets the index of the child polyline in the parent model.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a child vertex.

* Mandatory parameter/option.

spliceVertices

```
{Number[][]} spliceVertices(start, deleteCount)
```

Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.

Returns an array of coordinates of deleted vertexes.

Parameters:

Parameter	Default value	Description
start*	—	Type: Integer The index to start from for removing and adding vertexes.
deleteCount*	—	Type: Integer The number of deleted vertexes.

* Mandatory parameter/option.

geometryEditor.model.ChildVertex

Note:
The constructor of the geometryEditor.model.ChildVertex class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeometryEditorChildModel**.

Model for a child index. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Methods

Fields

Name	Type	Description
editor	**IGeometryEditor**	Geometry editor. Inherited from **IGeometryEditorChildModel**.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**IBaseGeometry**	Geometry of the model. Inherited from **IGeometryEditorChildModel**.

Methods

Name	Returns	Description
destroy()		Destructor. Inherited from **IGeometryEditorModel**.
getAllVerticesNumber()	Integer	Returns the total number of vertexes in the geometry being edited.
getIndex()	Integer	Returns the index of the child vertex in the parent model.
getNextVertex()	**geometryEditor.model.ChildVertex**|null	Returns a reference to the model for the next vertex.

Name	Returns	Description
getParent()	IGeometryEditorModel	Returns the parent data model. Inherited from IGeometryEditorChildModel.
getPixels()	Number[]	Returns the model's pixel data. Inherited from IGeometryEditorModel.
getPrevVertex()	geometryEditor.model.ChildVertex|null	Returns a reference to the model for the previous vertex.
setGlobalPixels(pixels)		Sets global pixel coordinates of the vertex.
setIndex(index)		Sets the index of the child vertex in the parent model.
setNextVertex(nextVertex)		Sets the reference to the model for the next vertex.
setPixels(pixels)		Sets the model's pixel data. Inherited from IGeometryEditorChildModel.
setPrevVertex(prevVertex)		Sets the reference to the model for the previous vertex.

Methods details

getAllVerticesNumber

```
{Integer} getAllVerticesNumber()
```

Returns the total number of vertexes in the geometry being edited.

getIndex

```
{Integer} getIndex()
```

Returns the index of the child vertex in the parent model.

getNextVertex

```
{geometryEditor.model.ChildVertex|null} getNextVertex()
```

Returns a reference to the model for the next vertex.

getPrevVertex

```
{geometryEditor.model.ChildVertex|null} getPrevVertex()
```

Returns a reference to the model for the previous vertex.

setGlobalPixels

```
{} setGlobalPixels(pixels)
```

Sets global pixel coordinates of the vertex.

Parameters:

Parameter	Default value	Description
pixels*	—	Type: Number[] Global pixel coordinates of the vertex.

* Mandatory parameter/option.

setIndex

```
{} setIndex(index)
```

Sets the index of the child vertex in the parent model.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a child vertex.

* Mandatory parameter/option.

setNextVertex

```
{} setNextVertex(nextVertex)
```

Sets the reference to the model for the next vertex.

Parameters:

Parameter	Default value	Description
nextVertex*	—	Type: **geometryEditor.model.ChildVertex**|null Model for the next vertex.

* Mandatory parameter/option.

setPrevVertex

```
{} setPrevVertex(prevVertex)
```

Sets the reference to the model for the previous vertex.

Parameters:

Parameter	Default value	Description
prevVertex*	—	Type: **geometryEditor.model.ChildVertex**|null Model for the previous vertex.

* Mandatory parameter/option.

geometryEditor.model.Edge

Note:
The constructor of the geometryEditor.model.Edge class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeometryEditorRootModel**.

Interim placemark model. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Methods

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
destroy()		Destructor. Inherited from **IGeometryEditorModel**.
getNextVertex()	**geometryEditor.model.ChildVertex**|null	Returns a reference to the model for the next vertex.
getPixels()	Number[]	Returns the model's pixel data. Inherited from **IGeometryEditorModel**.
getPrevVertex()	**geometryEditor.model.ChildVertex**|null	Returns a reference to the model for the previous vertex.
setNextVertex(nextVertex)		Sets the reference to the model for the next vertex.
setPrevVertex(prevVertex)		Sets the reference to the model for the previous vertex.

Methods details

getNextVertex

```
{geometryEditor.model.ChildVertex|null} getNextVertex()
```

Returns a reference to the model for the next vertex.

getPrevVertex

```
{geometryEditor.model.ChildVertex|null} getPrevVertex()
```

Returns a reference to the model for the previous vertex.

setNextVertex

```
{} setNextVertex(nextVertex)
```

Sets the reference to the model for the next vertex.

Parameters:

Parameter	Default value	Description
nextVertex*	—	Type: geometryEditor.model.ChildVertex|null Model for the next vertex.

* Mandatory parameter/option.

setPrevVertex

```
{} setPrevVertex(prevVertex)
```

Sets the reference to the model for the previous vertex.

Parameters:

Parameter	Default value	Description
prevVertex*	—	Type: geometryEditor.model.ChildVertex|null Model for the previous vertex.

* Mandatory parameter/option.

geometryEditor.model.EdgeGeometry

Note:
The constructor of the geometryEditor.model.EdgeGeometry class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeometry**.

Interim placemark geometry. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Events | Methods

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager.

Name	Type	Description
		Inherited from **ICustomizable**.

Events

Name	Description
mapchange	Map reference changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldMap — Old map. • newMap — New map. Inherited from **IGeometry**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
pixelgeometrychange	The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelGeometry — New **IPixelGeometry** pixel geometry. Inherited from **IGeometry**.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
setMap(map)		Sets the map. Inherited from **IGeometry**.

geometryEditor.model.RootLineString

Note:
The constructor of the geometryEditor.model.RootLineString class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeometryEditorRootModel**.

Model of the root polyline. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Methods

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
destroy()		Destructor. Inherited from **IGeometryEditorModel**.
getAllVerticesNumber()	Integer	Returns the total number of vertexes in the geometry being edited.
getPixels()	Number[]	Returns the model's pixel data. Inherited from **IGeometryEditorModel**.
getVertexModels()	**geometryEditor.model.ChildVertex**[]	Returns an array of models for child vertexes.
spliceVertices(start, deleteCount)	Number[][]	Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.

Methods details

getAllVerticesNumber

```
{Integer} getAllVerticesNumber()
```

Returns the total number of vertexes in the geometry being edited.

getVertexModels

```
{geometryEditor.model.ChildVertex[]} getVertexModels()
```

Returns an array of models for child vertexes.

spliceVertices

```
{Number[][]} spliceVertices(start, deleteCount)
```

Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.

Returns an array of coordinates of deleted vertexes.

Parameters:

Parameter	Default value	Description
start*	—	Type: Integer The index to start from for removing and adding vertexes.
deleteCount*	—	Type: Integer The number of deleted vertexes.

* Mandatory parameter/option.

geometryEditor.model.RootPolygon

Note:
The constructor of the geometryEditor.model.RootPolygon class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeometryEditorRootModel**.

Model for the root polygon. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Methods

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
destroy()		Destructor. Inherited from **IGeometryEditorModel**.
getAllVerticesNumber()	Integer	Returns the total number of vertexes in the geometry being edited.
getPathModels()	**geometryEditor.model.ChildLinearRing**[]	Returns an array of models for child contours.
getPixels()	Number[]	Returns the model's pixel data. Inherited from **IGeometryEditorModel**.

Name	Returns	Description
splicePaths(start, deleteCount)	Number[][]	Deletes a defined number of polygon contours, starting from the specified index. New contours can be added in place of the deleted ones. Global pixel coordinates of the new contours can be passed as additional arguments after the "deleteCount" parameter.

Methods details

getAllVerticesNumber

```
{Integer} getAllVerticesNumber()
```

Returns the total number of vertexes in the geometry being edited.

getPathModels

```
{geometryEditor.model.ChildLinearRing[]} getPathModels()
```

Returns an array of models for child contours.

splicePaths

```
{Number[][]} splicePaths(start, deleteCount)
```

Deletes a defined number of polygon contours, starting from the specified index. New contours can be added in place of the deleted ones. Global pixel coordinates of the new contours can be passed as additional arguments after the "deleteCount" parameter.

Returns an array of coordinates of deleted contours.

Parameters:

Parameter	Default value	Description
start*	—	Type: Integer The index to start from for removing and adding contours.
deleteCount*	—	Type: Integer The number of contours to be deleted.

* Mandatory parameter/option.

geometryEditor.Point

Extends **IGeometryEditor**.

The "Point" geometry editor.

Constructor | Fields | Events | Methods

Constructor

```
geometryEditor.Point(geometry[, options])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPointGeometry** The "Point" geometry.
options	—	Type: Object Options for the geometry editor.
options.dblClickHandler	—	Type: Function Handler for double-clicking a vertex. Accepts a reference to a model of the vertex being edited. By default, the handler is defined by the function that removes the corresponding vertex.
options.drawingCursor	"arrow"	Type: Boolean The mouse cursor in drawing mode.
options.drawOver	true	Type: Boolean Allows to put points on top of map objects in drawing mode.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**IGeometry**	The geometry being edited. Inherited from **IGeometryEditor**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**IDataManager**	Manager for the state of the geometry editor.

Name	Type	Description
		Data fields that are available via the "get" and "set" methods: • editing — Checks whether editing mode is enabled. Type — Boolean. Default value — false. • drawing — Checks whether drawing mode is enabled. Type — Boolean. Default value — false.

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.
statechange	Change to the geometry editor state. Instance of the Event class. Inherited from IGeometryEditor.

Methods

Name	Returns	Description
startDrawing()	vow.Promise	Enables the mode for drawing points.
startEditing()		Enables editing mode. Inherited from IGeometryEditor.
stopDrawing()	vow.Promise	Disables the mode for drawing points.
stopEditing()		Disables editing mode. Inherited from IGeometryEditor.

Fields details

state

```
{IDataManager} state
```

Manager for the state of the geometry editor.

Data fields that are available via the "get" and "set" methods:

- editing — Checks whether editing mode is enabled. Type — Boolean. Default value — false.
- drawing — Checks whether drawing mode is enabled. Type — Boolean. Default value — false.

Methods details

startDrawing

```
{vow.Promise} startDrawing()
```

Enables the mode for drawing points.

Returns Promise object.

stopDrawing

```
{vow.Promise} stopDrawing()
```

Disables the mode for drawing points.

Returns Promise object.

geometryEditor.Polygon

Extends **IGeometryEditor**.

The "Polygon" geometry editor.

Constructor | Fields | Events | Methods

Constructor

```
geometryEditor.Polygon(geometry[, options])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPolygonGeometry** The "Polygon" geometry.
options	—	Type: Object Options for the geometry editor.
options.dblClickHandler	—	Type: Function Handler for double-clicking a vertex. Accepts a reference to a model of the vertex being edited. By default, the handler is defined by the function that removes the corresponding vertex.
options.drawingCursor	"arrow"	Type: Boolean The mouse cursor in the mode for adding new vertexes.
options.drawOver	true	Type: Boolean Allows to put points on top of map objects in the mode for adding new vertexes.
options.edgeInteractiveOptions	true	Type: Boolean Allows the intermediate vertices placemarks to use options with postfixes linked to the current state of the placemark. The following postfixes are available: • Drag — Options with this given postfix are used when the user drags a placemark. • Hover — Options with this given postfix are used when the user hovers over a placemark with the mouse pointer.

Parameter	Default value	Description
		Examples of such options: edgeLayoutHover, edgeIconImageSizeActive, edgeIconImageShapeHover, etc. If you do not want to change the options of intermediate vertices placemarks depending on their state, then you will need to disable this option.
options.edgeLayout	—	Type: Function Class of the layout for interim placemarks.
options.maxPoints	Infinity	Type: Number The maximum allowable number of vertexes on a polygon.
options.menuManager	—	Type: Function Context menu dispatcher for the menu that opens when clicking on a vertex. Accepts two arguments: • An array of objects describing the context menu items for this vertex. • A reference to a model of the vertex being edited. The dispatcher can change data in the passed array. Must return an array of objects that describe context menu items.
options.minPoints	0	Type: Number The minimum allowable number of vertexes on a polygon.
options.useAutoPanInDrawing	true	Type: Boolean Enables autopan of the map when dragging a vertex on the boundary.
options.useMapMarginInDrawing	true	Type: Boolean Whether to use map margins in drawing mode.
options.vertexInteractiveOptions	true	Type: Boolean

Parameter	Default value	Description
		Allows the vertices placemarks to use options with postfixes linked to the current state of the vertex. The following postfixes are available: • Hover — Options with this given postfix are used when the user hovers over a vertex with the mouse pointer. • Drag — Options with this given postfix are used when the user drags a vertex. • Active — Options with this given postfix are used when the context menu is opened for the vertex. Examples of such options: vertexLayoutHover, vertexIconImageSizeActive, vertexIconImageShapeHover, etc. If you do not want to change the options of vertices placemarks depending on their state, then you will need to disable this option.
`options.vertexLayout`	—	Type: Function Class of the layout for placemarks on polygon vertexes.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**IGeometry**	The geometry being edited. Inherited from **IGeometryEditor**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**IDataManager**	Manager for the state of the geometry editor.

Name	Type	Description
		Data fields that are available via the "get" and "set" methods: • editing — Checks whether editing mode is enabled. Type — Boolean. Default value — false. • drawing — Checks whether vertex drawing mode is enabled. Type — Boolean. Default value — false. • drawingFrom — Checks how new points are added in drawing mode. Accepts one of two string values: "begin" — points are added at the beginning of the polygon; "end" — points are added at the end. • drawingFromIndex — The index of the polygon vertex after which new points will be added in drawing mode. This field is available only during editing, because saving changes rearranges the order of vertexes in the polygon so that the point with the specified index becomes the last one. Type — integer. • drawingPath — The index of the polygon contour where new points are added in drawing mode. Type — integer. Default value — 0.

Events

Name	Description
beforeedgedrag	Event preceding the "edgedrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset — Array of two numbers that describe the pixel offset at this step. • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "edgedrag" event will be canceled.
beforeedgedragstart	Event preceding the "edgedragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent — Source DOM event (as a **DomEvent** object), if there is one. • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "edgedragstart" event, will be canceled.
beforevertexadd	Event preceding the "vertexadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • parentModel — Reference to the parent data model for the vertex being added. • vertexIndex — Index of the vertex being added. • globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Name	Description
	Names of methods that are accessible via **Event.callMethod**: • setGlobalPixels — Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "vertexadd" event will be canceled.
beforevertexdrag	Event preceding the "vertexdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset — Array of two numbers that describe the pixel offset at this step. • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "vertexdrag" event will be canceled.
beforevertexdragstart	Event preceding the "vertexdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent — Source DOM event (as a **DomEvent** object), if there is one. • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the"vertexdragstart" event, will be canceled.
beforevertexdraw	Event preceding the "vertexdraw" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • parentModel — Reference to the parent data model for the vertex being added. • vertexIndex — Index of the vertex being added. • globalPixels — Coordinates of the vertex being added, in global pixel coordinates. Names of methods that are accessible via **Event.callMethod**: • setGlobalPixels — Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "vertexdraw" event will be canceled.
drawingstart	Enabling the mode for adding new vertexes. Instance of the **Event** class.
drawingstop	Disabling the mode for adding new vertexes. Instance of the **Event** class.

Name	Description
edgedrag	Dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset — Array of two numbers that describe the pixel offset at this step. • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.
edgedragend	End of dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.
edgedragstart	Start of dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent — Source DOM event (as a **DomEvent** object), if there is one. • edgeModel — Reference to the model of the draggable placemark. • globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.
editingstart	Enabling the mode for editing vertexes. Instance of the **Event** class.
editingstop	Disabling the mode for editing vertexes. Instance of the **Event** class.
framingstart	Enabling the zoom mode. Instance of the **Event** class.
framingstop	Disabling the zoom mode. Instance of the **Event** class.
optionschange	Change to the object options. Inherited from **ICustomizable**.
statechange	Change to the geometry editor state. Instance of the **Event** class. Inherited from **IGeometryEditor**.
vertexadd	Adding a new vertex. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • parentModel — Reference to the parent data model for the vertex that was added. • vertexIndex — Index of the vertex that was added. • globalPixels — Coordinates of the vertex that was added, in global pixel coordinates.
vertexdrag	Dragging a vertex. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset — Array of two numbers that describe the pixel offset at this step. • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.
vertexdragend	End of vertex dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

Name	Description
vertexdragstart	Start of vertex dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • domEvent — Source DOM event (as a DomEvent object), if there is one. • vertexModel — Reference to the model of the draggable vertex. • globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.
vertexdraw	Drawing a new vertex. This event usually precedes the add vertex event, and occurs when the mouse moves and the mode for adding new vertexes is enabled. Based on data passed in this event, guide lines are displayed in vertex drawing mode. Instance of the Event class. Names of fields that are available via the Event.get method: • parentModel — Reference to the parent data model for the vertex being added. • vertexIndex — Index of the vertex being added. • globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Methods

Name	Returns	Description
getModel()	vow.Promise	Returns the promise object, which is confirmed by the model object at the time it is actually created, or is rejected with one of the following error messages: • Canceled — Editing mode is disabled until the model is actually created. • Editor wasn't started — Editing mode is not enabled. .
getModelSync()	geometryEditor.model.RootPolygon|null	Returns the editor's model, or null if it is missing at the time of the call.
getView()	vow.Promise	Returns the promise object, which is confirmed by the representation object at the time it is actually created, or is rejected with one of the following error messages: • Canceled — Editing mode is disabled until the representation is actually created. • Editor wasn't started — Editing mode is not enabled. .
getViewSync()	geometryEditor.view.MultiPath|null	Returns the editor's representation, or null if it is missing at the time of the call.
startDrawing()	vow.Promise	Enables drawing mode (for adding new vertexes to a polygon). Enabling occurs asynchronously.

Name	Returns	Description
startEditing()	vow.Promise	Enables edit mode (for adding new vertexes to a polygon). Enabling occurs asynchronously.
startFraming()	vow.Promise	Enables the zoom mode for a polygon. Enabling occurs asynchronously.
stopDrawing()		Disables drawing mode (for adding new vertexes to a polygon).
stopEditing()		Disables edit mode.
stopFraming()		Disables the zoom mode.

Fields details

state

```
{IDataManager} state
```

Manager for the state of the geometry editor.

Data fields that are available via the "get" and "set" methods:

- editing — Checks whether editing mode is enabled. Type — Boolean. Default value — false.
- drawing — Checks whether vertex drawing mode is enabled. Type — Boolean. Default value — false.
- drawingFrom — Checks how new points are added in drawing mode. Accepts one of two string values: "begin" — points are added at the beginning of the polygon; "end" — points are added at the end.
- drawingFromIndex — The index of the polygon vertex after which new points will be added in drawing mode. This field is available only during editing, because saving changes rearranges the order of vertexes in the polygon so that the point with the specified index becomes the last one. Type — integer.
- drawingPath — The index of the polygon contour where new points are added in drawing mode. Type — integer. Default value — 0.

Events details

beforeedgedrag

Event preceding the "edgedrag" event. Instance of the Event class. Names of fields that are available via the Event.get method:

- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

Names of methods that are accessible via Event.callMethod:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the Event.preventDefault method is called for this event, a subsequent "edgedrag" event will be canceled.

beforeedgedragstart

Event preceding the "edgedragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent — Source DOM event (as a **DomEvent** object), if there is one.
- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "edgedragstart" event, will be canceled.

beforevertexadd

Event preceding the "vertexadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- parentModel — Reference to the parent data model for the vertex being added.
- vertexIndex — Index of the vertex being added.
- globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Names of methods that are accessible via **Event.callMethod**:

- setGlobalPixels — Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "vertexadd" event will be canceled.

Examples:

1.

```
// Prohibiting adding new vertexes further than 100 pixels from the map center.
polygon.editor.events.add("beforevertexadd", function (event) {
    var mapGlobalPixelCenter = geoMap.getGlobalPixelCenter(),
        globalPixels = event.get("globalPixels"),
        vector = [mapGlobalPixelCenter[0] - globalPixels[0],
mapGlobalPixelCenter[1] - globalPixels[1]],
        vectorLength = Math.sqrt(vector[0] * vector[0] + vector[1] * vector[1]);
    if (dist &gt; 100) {
        event.preventDefault();
    }
});
```

2.

```
// Adjusting coordinates of the add vertex events so they fall inside a square
// with 100-pixel sides that is centered at the map center.
polygon.editor.events.add(["beforevertexdraw", "beforevertexadd"], function
(event) {
    var mapGlobalPixelCenter = geoMap.getGlobalPixelCenter(),
        globalPixels = event.get("globalPixels"),
        pixelBounds = [
            [mapGlobalPixelCenter[0] - 100, mapGlobalPixelCenter[1] - 100],
            [mapGlobalPixelCenter[0] + 100, mapGlobalPixelCenter[1] + 100]
        ];
    event.callMethod("setGlobalPixels", [
        Math.max(Math.min(globalPixels[0], pixelBounds[1][0]), pixelBounds[0]
[0]),
        Math.max(Math.min(globalPixels[1], pixelBounds[1][1]), pixelBounds[0]
[1])
    ]);
});
```

beforevertexdrag

Event preceding the "vertexdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "vertexdrag" event will be canceled.

beforevertexdragstart

Event preceding the "vertexdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent — Source DOM event (as a **DomEvent** object), if there is one.
- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the"vertexdragstart" event, will be canceled.

Example:

```
// Prohibiting dragging vertexes with the index 0 in a contour with the same
index.
polygon.editor.events.add("beforevertexdragstart", function (event) {
    var vertexModel = event.get("vertexModel"),
        pathModel = vertexModel.getParent();
    if (pathModel.getIndex() == 0 && vertexModel.getIndex() == 0) {
        event.preventDefault();
    }
});
```

beforevertexdraw

Event preceding the "vertexdraw" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- parentModel — Reference to the parent data model for the vertex being added.
- vertexIndex — Index of the vertex being added.
- globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Names of methods that are accessible via **Event.callMethod**:

- setGlobalPixels — Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "vertexdraw" event will be canceled.

drawingstart

Enabling the mode for adding new vertexes. Instance of the **Event** class.

drawingstop

Disabling the mode for adding new vertexes. Instance of the **Event** class.

edgedrag

Dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

edgedragend

End of dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

edgedragstart

Start of dragging an interim placemark. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent — Source DOM event (as a **DomEvent** object), if there is one.
- edgeModel — Reference to the model of the draggable placemark.
- globalPixels — Coordinates of the draggable placemark, in global pixel coordinates.

editingstart

Enabling the mode for editing vertexes. Instance of the **Event** class.

editingstop

Disabling the mode for editing vertexes. Instance of the **Event** class.

framingstart

Enabling the zoom mode. Instance of the **Event** class.

framingstop

Disabling the zoom mode. Instance of the **Event** class.

vertexadd

Adding a new vertex. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- parentModel — Reference to the parent data model for the vertex that was added.
- vertexIndex — Index of the vertex that was added.
- globalPixels — Coordinates of the vertex that was added, in global pixel coordinates.

vertexdrag

Dragging a vertex. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

vertexdragend

End of vertex dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

vertexdragstart

Start of vertex dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent — Source DOM event (as a **DomEvent** object), if there is one.
- vertexModel — Reference to the model of the draggable vertex.
- globalPixels — Coordinates of the draggable vertex, in global pixel coordinates.

vertexdraw

Drawing a new vertex. This event usually precedes the add vertex event, and occurs when the mouse moves and the mode for adding new vertexes is enabled. Based on data passed in this event, guide lines are displayed in vertex drawing mode. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- parentModel — Reference to the parent data model for the vertex being added.
- vertexIndex — Index of the vertex being added.
- globalPixels — Coordinates of the vertex being added, in global pixel coordinates.

Methods details

getModel

```
{vow.Promise} getModel()
```

Returns the promise object, which is confirmed by the model object at the time it is actually created, or is rejected with one of the following error messages:

- Canceled — Editing mode is disabled until the model is actually created.
- Editor wasn't started — Editing mode is not enabled.

.

getModelSync

```
{geometryEditor.model.RootPolygon|null} getModelSync()
```

Returns the editor's model, or null if it is missing at the time of the call.

getView

```
{vow.Promise} getView()
```

Returns the promise object, which is confirmed by the representation object at the time it is actually created, or is rejected with one of the following error messages:

- Canceled — Editing mode is disabled until the representation is actually created.
- Editor wasn't started — Editing mode is not enabled.

.

getViewSync

```
{geometryEditor.view.MultiPath|null} getViewSync()
```

Returns the editor's representation, or null if it is missing at the time of the call.

startDrawing

```
{vow.Promise} startDrawing()
```

Enables drawing mode (for adding new vertexes to a polygon). Enabling occurs asynchronously.

Returns the promise object, which is confirmed when drawing mode has actually started, or is rejected with one of the following error messages:

- Canceled — Drawing mode is disabled until it is actually started.

.

startEditing

```
{vow.Promise} startEditing()
```

Enables edit mode (for adding new vertexes to a polygon). Enabling occurs asynchronously.

Returns the promise object, which is confirmed when editing mode has actually started, or is rejected with one of the following error messages:

- Canceled — Editing mode is disabled until it is actually started.

.

startFraming

```
{vow.Promise} startFraming()
```

Enables the zoom mode for a polygon. Enabling occurs asynchronously.

Returns the promise object that is confirmed when zoom mode actually starts.

stopDrawing

```
{} stopDrawing()
```

Disables drawing mode (for adding new vertexes to a polygon).

stopEditing

```
{} stopEditing()
```

Disables edit mode.

stopFraming

```
{} stopFraming()
```

Disables the zoom mode.

geometryEditor.view

geometryEditor.view.Edge

Note:
The constructor of the geometryEditor.view.Edge class is hidden, as this class is not intended for autonomous initialization.

View of an interim placemark. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Methods

Methods

Name	Returns	Description
getPlacemark()	GeoObject	Returns a reference to the interim placemark geo object. The placemark data contains a reference to the data model for the interim placemark (called "model").

Methods details

getPlacemark

```
{GeoObject} getPlacemark()
```

Returns a reference to the interim placemark geo object. The placemark data contains a reference to the data model for the interim placemark (called "model").

geometryEditor.view.MultiPath

Note:
The constructor of the geometryEditor.view.MultiPath class is hidden, as this class is not intended for autonomous initialization.

View of a set of contours. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Methods

Methods

Name	Returns	Description
getEdgePlacemarks()	GeoObjectCollection	Returns a collection of interim placemarks of child contour views.

Name	Returns	Description
getPathViews()	geometryEditor.view.Path[]	Returns an array of views of child contours.
getVertexPlacemarks()	GeoObjectCollection	Returns a collection of vertex placemarks of child contour views.

Methods details

getEdgePlacemarks

```
{GeoObjectCollection} getEdgePlacemarks()
```

Returns a collection of interim placemarks of child contour views.

getPathViews

```
{geometryEditor.view.Path[]} getPathViews()
```

Returns an array of views of child contours.

getVertexPlacemarks

```
{GeoObjectCollection} getVertexPlacemarks()
```

Returns a collection of vertex placemarks of child contour views.

geometryEditor.view.Path

Note:
The constructor of the geometryEditor.view.Path class is hidden, as this class is not intended for autonomous initialization.

Contour view. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Methods

Methods

Name	Returns	Description
getEdgePlacemarks()	GeoObjectCollection	Returns a collection of interim placemarks that are on the contour.
getEdgeViews()	geometryEditor.view.Edge[]	Returns an array of views of interim placemarks.
getVertexPlacemarks()	GeoObjectCollection	Returns a collection of vertex placemarks that are on the contour.
getVertexViews()	geometryEditor.view.Vertex[]	Returns an array of views of child vertexes.

Methods details

getEdgePlacemarks

```
{GeoObjectCollection} getEdgePlacemarks()
```

Returns a collection of interim placemarks that are on the contour.

getEdgeViews

```
{geometryEditor.view.Edge[]} getEdgeViews()
```

Returns an array of views of interim placemarks.

getVertexPlacemarks

```
{GeoObjectCollection} getVertexPlacemarks()
```

Returns a collection of vertex placemarks that are on the contour.

getVertexViews

```
{geometryEditor.view.Vertex[]} getVertexViews()
```

Returns an array of views of child vertexes.

geometryEditor.view.Vertex

Note:
The constructor of the geometryEditor.view.Vertex class is hidden, as this class is not intended for autonomous initialization.

Vertex view. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Methods

Methods

Name	Returns	Description
getPlacemark()	GeoObject	Returns geo object of the placemark that marks the endpoint. The placemark data contains a reference to the data model for the vertex (called "model").

Methods details

getPlacemark

```
{GeoObject} getPlacemark()
```

Returns geo object of the placemark that marks the endpoint. The placemark data contains a reference to the data model for the vertex (called "model").

geoObject

geoObject.addon

geoObject.addon.balloon

Note:
The constructor of the geoObject.addon.balloon class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use a balloon for a geo object. Adds the **IBalloonOwner** interface to the geo object (**GeoObject**). When enabling package.full (the standard set of modules), it is available by default. If **GeoObject** is enabled separately, this module must be explicitly specified in the loader. If **geoObject.addon.balloon** is enabled separately after creating **GeoObject**, the **IBalloonOwner** interface won't be added. Then in order to initialize the balloon manager, you will need to use the geoObject.addon.balloon#get method.

Methods

Methods

Name	Returns	Description
get(geoObject)	IPopupManager	Returns geo object balloon manager.

Methods details

get

```
{IPopupManager} get(geoObject)
```

Returns geo object balloon manager.

Parameters:

Parameter	Default value	Description
geoObject*	—	Type: IGeoObject Geo object

* Mandatory parameter/option.

Example:

```
ymaps.geoObject.addon.balloon.get(geoObject)
```

geoObject.addon.editor

Note:
The constructor of the geoObject.addon.editor class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use the editor for a geo object. Adds the GeoObject#editor field to **GeoObject**. When enabling package.full (the standard set of modules), it is available by default. If **GeoObject** is enabled separately, this module must be explicitly specified in the loader. If **geoObject.addon.editor** is enabled separately after creating **GeoObject**, the GeoObject#editor field won't be added. Then you will need to use the geoObject.addon.editor#get method for the editor.

Methods

Methods

Name	Returns	Description
create(geoObject)		A function that creates the "editor" field for the passed geo object.
get(geoObject)	**IGeometryEditor**	Returns geo object editor.

Methods details

create

```
{} create(geoObject)
```

A function that creates the "editor" field for the passed geo object.

Parameters:

Parameter	Default value	Description
geoObject*	—	Type: **IGeoObject** Geo object.

* Mandatory parameter/option.

get

```
{IGeometryEditor} get(geoObject)
```

Returns geo object editor.

Parameters:

Parameter	Default value	Description
geoObject*	—	Type: **IGeoObject** Geo object

* Mandatory parameter/option.

Example:

```
ymaps.geoObject.addon.editor.get(geoObject)
```

geoObject.addon.hint

Note:
The constructor of the geoObject.addon.hint class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use a hint for a geo object. Adds the **IHintOwner** interface to the geo object (**GeoObject**). When enabling package.full (the standard set of modules), it is available by default. If **GeoObject** is enabled separately, this module must be explicitly specified in the loader. If **geoObject.addon.hint** is enabled separately after creating **GeoObject**, the **IHintOwner** interface won't be added. Then in order to initialize the balloon manager, you will need to use the geoObject.addon.hint#get method.

Methods

Methods

Name	Returns	Description
get(geoObject)	**IPopupManager**	Returns geo object hint manager.

Methods details

get

```
{IPopupManager} get(geoObject)
```

Returns geo object hint manager.

Parameters:

Parameter	Default value	Description
geoObject*	—	Type: **IGeoObject** Geo object

* Mandatory parameter/option.

Example:

```
ymaps.geoObject.addon.hint.get(geoObject)
```

geoObject.Balloon

Extends **IBalloonManager**.

Geo object balloon manager. Allows a geo object to manage a balloon by opening it and hiding it. Passes data to the balloon in the **IGeoObjectPopupData**. Uses the map balloon manager **map.Balloon** internally. Geo objects contain an instance of this class, which is available as myGeoObject.balloon. Don't create new instances of this class unless necessary.

See Balloon GeoObject.balloon

Constructor | Fields | Events | Methods

Constructor

```
geoObject.Balloon(geoObject)
```

Parameters:

Parameter	Default value	Description
geoObject *	—	Type: Object Geo object.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
autopanbegin	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
autopanend	End of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
beforeuserclose	The event which precedes **Balloon.event:userclose**. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
close	Closing the info object. Names of fields available via **Event.get**: • target — Reference to the object where the closing occurred.

Name	Description
	Inherited from **IPopupManager**.
open	Opening the info object. Names of fields available via **Event.get**: • target — Reference to the object where the opening occurred. Inherited from **IPopupManager**.
userclose	Balloon closed by the user. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.

Methods

Name	Returns	Description
autoPan()	**vow.Promise**	Moves the map so that the balloon is visible. Inherited from **IBalloonManager**.
close([force])	**vow.Promise**	Closes the info object. Inherited from **IPopupManager**.
destroy()		Disables the info object manager. Inherited from **IPopupManager**.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from **IPopupManager**.
getOptions()	**IOptionManager**|null	Returns the options manager or 'null'. Inherited from **IPopupManager**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopupManager**.
getOverlaySync()	**IOverlay**|null	Returns the overlay, if one exists. Inherited from **IPopupManager**.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from **IPopupManager**.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from **IPopupManager**.
open([position[, data[, options]]])	**vow.Promise**	Opens the balloon for the geo object.
setData([data])	**vow.Promise**	Sets new user data.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

Methods details

open

```
{vow.Promise} open([position[, data[, options]]])
```

Opens the balloon for the geo object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
position	—	Type: Number[] Coordinates of the point where the hint is opened. By default: the point on the geoobject that is closest to the current center of the map. The projection of the coordinates can be specified in the options, otherwise the geo object projection is used.
data	—	Type: Object Data to add to the *userData* field for the object of data passed to the balloon.
options	—	Type: Object Options.

setData

```
{vow.Promise} setData([data])
```

Sets new user data.

Returns Promise object.

Parameters:

Parameter	Default value	Description
data	—	Type: Object Data to add to the *userData* field for the object of data passed to the balloon.

geoObject.Hint

Extends **IHintManager**.

Geo object hint manager. Allows a geo object to manage a hint by opening it and hiding it. Passes data to the hint in the format **IGeoObjectPopupData**. Uses the map hint manager **map.Hint** internally. Geo objects contain an instance of this class, which is available as myGeoObject.hint. Don't create new instances of this class unless necessary.

See Hint GeoObject.hint

Constructor | Fields | Events | Methods

Constructor

```
geoObject.Hint(geoObject)
```

Parameters:

Parameter	Default value	Description
geoObject *	—	Type: Object Geo object.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
close	Closing the info object. Names of fields available via Event.get: • target — Reference to the object where the closing occurred. Inherited from IPopupManager.
open	Opening the info object. Names of fields available via Event.get: • target — Reference to the object where the opening occurred. Inherited from IPopupManager.

Methods

Name	Returns	Description
close([force])	vow.Promise	Closes the info object. Inherited from IPopupManager.
destroy()		Disables the info object manager. Inherited from IPopupManager.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from IPopupManager.
getOptions()	IOptionManager|null	Returns the options manager or 'null'. Inherited from IPopupManager.
getOverlay()	vow.Promise	Returns the promise object to return the overlay. Inherited from IPopupManager.
getOverlaySync()	IOverlay|null	Returns the overlay, if one exists. Inherited from IPopupManager.

Name	Returns	Description
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from **IPopupManager**.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from **IPopupManager**.
open([position[, data[, options]]])	**vow.Promise**	Opens the hint for the geo object.
setData([data])	**vow.Promise**	Sets new user data.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

Methods details

open

```
{vow.Promise} open([position[, data[, options]]])
```

Opens the hint for the geo object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
position	—	Type: Number[] Coordinates of the point where the hint is opened. By default: the geometric center of gravity of the geoobject. The projection of the coordinates can be specified in the options, otherwise the geo object projection is used.
data	—	Type: Object Data to add to the *userData* field for the object of data passed to the hint.
options	—	Type: Object Options.

setData

```
{vow.Promise} setData([data])
```

Sets new user data.

Returns Promise object.

Parameters:

Parameter	Default value	Description
data	—	Type: Object Data to add to the *userData* field for the object of data passed to the hint.

geoObject.Sequence

Extends **IGeoObject**, **IGeoObjectSequence**.

An unchanged collection of geo objects. Allows you to group geo objects for adding them to the map, setting options, etc. Collections are geo objects too.

Constructor | Fields | Events | Methods

Constructor

```
geoObject.Sequence([feature[, options]])
```

Creates an unchangeable collection of geo objects.

Parameters:

Parameter	Default value	Description
feature	—	Type: Object Collection description. For all purposes, corresponds to the geo object description. See **GeoObject** object.
feature.children	—	Type: **IGeoObject**[] Array of child geoobjects.
feature.geometry	—	Type: **IGeometry**|Object Geometry of a collection.
feature.properties	—	Type: **IDataManager**|Object Collection data.
options	—	Type: Object Collection options. You can set all the options described in the **GeoObject** object. Option values will be applied both to the collection itself and to its child objects, if these options are not set for them.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry.

Name	Type	Description
		Inherited from **IGeoObject**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**IDataManager**	Geo object data. Inherited from **IGeoObject**.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Events

Name	Description
boundschange	Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the **Event** class. Inherited from **IGeoObjectSequence**.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

Name	Description
	Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed.

Name	Description
	Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
pixelboundschange	Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the **Event** class. Inherited from **IGeoObjectSequence**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
each(callback[, context])		Calls a handler function for each child geo object. Inherited from **IGeoObjectSequence**.
get(index)	**IGeoObject**	Returns a child geo object with the specified index. Inherited from **IGeoObjectSequence**.
getBounds()	Number[][]|null	Returns geographical coordinates of the area that covers the collection and its child geo objects. Inherited from **IGeoObjectSequence**.
getIterator()	**IIterator**	Returns iterator for child geo objects in the collection. Inherited from **IGeoObjectSequence**.
getLength()	Integer	Returns length of the collection. Inherited from **IGeoObjectSequence**.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.

Name	Returns	Description
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getPixelBounds()	Number[][]|null	Returns global pixel coordinates of the area that spans the collection and its child geo objects. Inherited from **IGeoObjectSequence**.
indexOf(object)	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned. Inherited from **IGeoObjectSequence**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

GeoObject

Extends **IGeoObject**.

Geo object. Can be displayed as a placemark, polyline, polygon, etc., depending on the geometry type. You can also use auxiliary classes for simplified creation of geo objects with a specific geometry type.

See Placemark Polyline Polygon Circle Rectangle

Constructor | Fields | Events | Methods

Constructor

```
GeoObject([feature[, options]])
```

Parameters:

Parameter	Default value	Description
feature	—	Type: Object Geo object description.
feature.geometry	—	Type: **IGeometry**|**IGeometryJson** Geo object geometry. Can be either set directly by the object, or in the form of an object of the JSON representation of the geometry. Examples of classes and objects that can represent a geometry — **geometry.Point**, **geometry.json.Point**.
feature.properties	—	Type: **IDataManager**|Object Geo object data. Can be set as a class instance implementing the **IDataManager** interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object: • iconContent — Content of the geo object's icon. • iconCaption — Caption for the geo object's icon. • hintContent — Content of the geo object's popup hint. • balloonContent — Content of the geo object's balloon. • balloonContentHeader — Content of the geo object balloon title. • balloonContentBody — Content of the main part of the geo object's balloon. • balloonContentFooter — Content of the lower part of the geo object's balloon.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The balloonContent field is a shortcut for the balloonContentBody field, but if they are both set simultaneously, balloonContentBody takes priority. You can also add your own custom fields to the geo object data and use them wherever possible. For example, in the placemark layout or balloon layout.
options	—	Type: Object Geo object options. Using this parameter, you can set options for the geo object itself, as well as for its parts: • Options for the geo object balloon with the balloon prefix. • Options for the geo object's popup hint with the hint prefix. • Options for the geo object's geometry editor with the editor prefix. The type of editor and list of available options depends on the geo object's geometry type. See the descriptions of **geometryEditor.LineString**, **geometryEditor.Polygon**, and **geometryEditor.Point**. • Geometry options can be set without a prefix. See the descriptions of the **IGeometry** classes for geometries **geometry.Point**, **geometry.Polygon**, and others.
options.circleOverlay	"default#circle"	Type: String|Function Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelCircleGeometry** — The pixel geometry itself. • data: Object — The overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
options.cursor	"pointer"	Type: String Type of cursor over a geo object.
options.draggable	false	Type: Boolean Checks whether the geo object can be dragged.
options.fill	true	Type: Boolean Whether the shape is filled.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.fillColor	"0066ff99"	Type: String Fill color.
options.fillImageHref	—	Type: String Background image. When this option is enabled in **stretch** mode, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	1	Type: Number Fill transparency.
options.hasBalloon	true	Type: Boolean Checks whether the geo object has the "balloon" field.
options.hasHint	true	Type: Boolean Checks whether the geo object has the "hint" field.
options.hideIconOnBalloonOpen	true	Type: Boolean Hide the icon when opening the balloon.
options.iconCaptionMaxWidth	188	Type: Number Maximum width of the placemark caption.
options.iconColor	—	Type: String Icon color. This option is used for standard icons in browsers that support SVG.
options.iconContentLayout	—	Type: Function|String Layout for icon contents. (Type: constructor for an object with the ILayout interface, or its key in the storage).
options.iconContentOffset	—	Type: Number[]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Pixel offset for the icon contents relative to the parent element. Used in the default#imageWithContent layout.
options.iconContentPadding	—	Type: Number[] Padding for the icon contents.
options.iconContentSize	—	Type: Number[] Size of contents. Used in the default#imageWithContent layout.
options.iconImageClipRect	[[0, 0], [{imageWidth}, {imageHeight}]]	Type: Number[][] A rectangular area (the upper-left and lower-right corners are specified) that will be cut from the source image file and scaled to icon size (for example, for using sprites). By default, the entire source picture is used.
options.iconImageHref	—	Type: String URL of the icon's graphic file. Used only in combination with layouts (iconLayout) 'default ' #image' and 'default#imageWithContent'.
options.iconImageOffset	—	Type: Number[] Pixel offset for the icon image inside the parent element.
options.iconImageShape	—	Type: **IShape**|null The hotspot shape. If not set, the rectangular shape based on the size and offset of the icon will be calculated automatically. The coordinates of the figure geometry are counted from the anchor point. Used only in combination with layouts (iconLayout) 'default ' #image' and 'default#imageWithContent'.
options.iconImageSize	—	Type: Number[] Size of the icon, in pixels.
options.iconLayout	—	Type: Function|String Icon layout. (Type: constructor for an object with the ILayout interface, or its key in the storage).
options.iconMaxHeight	—	Type: Number Maximum height of an icon with contents.
options.iconMaxWidth	—	Type: Number Maximum width of an icon with contents.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.iconOffset	[0, 0]	Type: Number[] Pixel offset of the upper-left corner of the icon relative to the geographic anchor point of a placemark.
options.iconShadow	false	Type: Boolean Flag for whether the icon has a shadow.
options.iconShadowImageClipRect	[[0, 0], [{imageWidth}, {imageHeight}]]	Type: Number[][] A rectangular area (the upper-left and lower-right corners are specified) that will be cut from the source image file and scaled to shadow size (for example, for using sprites). By default, the entire source picture is used.
options.iconShadowImageHref	—	Type: String URL of the graphic file for the icon shadow. Used only in combination with layouts (iconShadowLayout) 'default#image' and 'default#imageWithContent'.
options.iconShadowImageOffset	—	Type: Number[] Pixel offset for the icon shadow image inside the parent element.
options.iconShadowImageSize	—	Type: Number[] Size of the icon shadow.
options.iconShadowLayout	—	Type: Function|String Layout for the icon shadow. (Type: constructor for an object with the ILayout interface, or its key in the storage).
options.iconShadowOffset	—	Type: Number[] The pixel offset of the icon shadow relative to its set position.
options.interactiveZIndex	—	Type: Boolean Enables to automatically modify z-index of the geo object depending on its state. By default, takes the true value for **geometry.Point** and false for other geometries.
options.interactivityModel	"default#geoObject"	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.lineStringOverlay	"default#polyline"	Type: String|Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelLineStringGeometry** — The pixel geometry itself. • data: **IDataManager** or Object — Overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
options.opacity	1	Type: Number Transparency.
options.openBalloonOnClick	true	Type: Boolean Checks whether to show the balloon when the geo object is clicked on.
options.openEmptyBalloon	false	Type: Boolean Checks whether to show an empty balloon when the geo object is clicked on.
options.openEmptyHint	false	Type: Boolean Checks whether to show an empty hint when the mouse pointer hovers over the geo object.
options.openHintOnHover	true	Type: Boolean Checks whether to show a hint when the mouse pointer hovers over the geo object.
options.outline	true	Type: Boolean Whether the shape has an outline.
options.pane	—	Type: String The key of the pane where the geo object overlay is placed. The default value is defined by the current overlay.
options.pointOverlay	"default#placemark"	Type: String|Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelPointGeometry** — The pixel geometry itself. • data: **IDataManager** or Object — Overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
options.polygonOverlay	"default#polygon"	Type: String|Function Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelPolygonGeometry** — The pixel geometry itself. • data: **IDataManager** or Object — Overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
options.preset	—	Type: String Key for the geo object's preset options. The list of keys is provided in the description of **option.presetStorage**.
options.rectangleOverlay	"default#rectangle"	Type: String|Function Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelCircleGeometry** — The pixel geometry itself. • data: **IDataManager** or Object — Overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
options.setMapCursorInDragging	false	Type: Boolean true – when dragging an object, set the cursor to "grabbing" for both the object and the map; false – only use the "grabbing" cursor on the object.
options.strokeColor	"0066ffff"	Type: String|String[] Color of the line or outline. You can set multiple values for a multistroke outline.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.strokeOpacity	1	Type: Number|Number[] Transparency of the line or outline. You can set multiple values for a multistroke outline.
options.strokeStyle	—	Type: String|Object|String[]|Object[] Style of the line or outline. You can set multiple values for a multistroke outline.
options.strokeWidth	1	Type: Number|Number[] Thickness of the line or outline. You can set multiple values for a multistroke outline.
options.syncOverlayInit	false	Type: Boolean Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a geo object to the map.
options.useMapMarginInDragging	true	Type: Boolean When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with **map.margin.Manager**.
options.visible	true	Type: Boolean Checks geo object visibility.
options.zIndex	—	Type: Number The z-index of a geo object in its normal state. Lowest priority.
options.zIndexActive	—	Type: Number The z-index of a geo object with an open balloon. Highest priority.
options.zIndexDrag	—	Type: Number The z-index of a geo object that is being dragged.
options.zIndexHover	—	Type: Number The z-index of a geo object when the mouse pointer is hovering over it.

Examples:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

1.

```
   var myGeoObject = new ymaps.GeoObject({
        // Defining a "Polyline" type of geometry.
        geometry: {
            type: "LineString",
            coordinates: [
                [55.75, 37.61], [52.51, 13.38]
            ]
        },
        // Defining the geo object data.
        properties: {
            hintContent: "Moscow-Berlin"
        }
    }, {
        // Enabling the display mode as geodesic curves.
        geodesic: true,
        // Setting the width to 5 pixels.
        strokeWidth: 5,
        // Setting the line color.
        strokeColor: "#F008"
  *     });
// Adding the geo object to the map.
geoMap.geoObjects.add(myGeoObject);
```

2.

```
   var myGeoObject = new ymaps.GeoObject({
        // Defining a "Point" geometry.
        geometry: {
            type: "Point",
            coordinates: [55.75, 37.61]
        },
        // Defining the geo object data.
        properties: {
            hintContent: "Moscow",
            balloonContentHeader: "Moscow",
            balloonContentBody: "Capital of Russia",
            population: 11848762
        }
    }, {
        // Setting the preset for a placemark with a dot and no content.
        preset: "islands#redDotIcon",
        // Enabling dragging.
        draggable: true,
        // Overriding the layout for the content in the lower part of the
balloon.
        balloonContentFooterLayout: ymaps.templateLayoutFactory.createClass(
            'population: {{ properties.population }}, coordinates:
{{ geometry.coordinates }}'
        ),
        // Disabling the delay for closing the popup hint.
        hintCloseTimeout: null
  *     });
// Adding the geo object to the map.

geoMap.geoObjects.add(myGeoObject);
```

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object.
editor	**IGeometryEditor**	Editor for the geo object geometry.
events	**event.Manager**	Event manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
geometry	IGeometry|null	Geo object geometry.
hint	geoObject.Hint	Geo object hint.
options	option.Manager	Geo object options manager.
properties	data.Manager	Geo object data manager.
state	data.Manager	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged

Events

Name	Description
balloonclose	Closing the balloon. Instance of the Event class.
balloonopen	Opening a balloon on a geo object. Instance of the Event class.
beforedrag	Event preceding the "drag" event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a DomEvent object), if there is one. Names of methods that are accessible via Event.callMethod: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the Event.preventDefault method is called for this event, a subsequent drag event will be canceled.
beforedragstart	Event preceding the "dragstart" event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a DomEvent object), if there is one. If the Event.preventDefault method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
drag	Dragging a geo object. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a DomEvent object), if there is one.
dragend	End of geo object dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a DomEvent object), if there is one.
dragstart	Start of geo object dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a DomEvent object), if there is one.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent — Original event of the geometry editor.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
hintclose	Closing the hint. Instance of the Event class.
hintopen	Opening a hint on a geo object. Instance of the Event class.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Fields details

balloon

```
{geoObject.Balloon} balloon
```

Balloon for a geo object.

editor

```
{IGeometryEditor} editor
```

Editor for the geo object geometry.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Start editing the geo object geometry.
geoObject.editor.startEditing();
// ...
// Finish editing.
geoObject.editor.stopEditing();
```

events

```
{event.Manager} events
```

Event manager.

geometry

```
{IGeometry|null} geometry
```

Geo object geometry.

Example:

```
// When changing the coordinates of a geo object's geometry, we set the map
borders
// so that the geo object is covered entirely.
myMap.geoObjects.add(myGeoObject);
myGeoObject.geometry.events.add("change", function () {
    myMap.setBounds(myGeoObject.geometry.getBounds());
});
```

hint

```
{geoObject.Hint} hint
```

Geo object hint.

options

```
{option.Manager} options
```

Geo object options manager.

properties

```
{data.Manager} properties
```

Geo object data manager.

Example:

```
// When changing data, output a custom ID
// if the object's "synchronized" field is set to false.
myGeoObject.properties.events.add("change", function () {
    if (!myGeoObject.properties.get("synchronized")) {
        console.log(myGeoObject.properties.get("myID"));
    }
});
```

state

```
{data.Manager} state
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

State of the geo object. Defined by the following fields:

- active: Boolean — Indicates that a balloon is open on the geo object.
- hover: Boolean — Indicates that the mouse is currently pointed at the geo object.
- drag: Boolean — Indicates that the geo object is being dragged

Events details

balloonclose

Closing the balloon. Instance of the **Event** class.

balloonopen

Opening a balloon on a geo object. Instance of the **Event** class.

beforedrag

Event preceding the "drag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- position — Coordinates relative to the document. Array in the format [pageX, pageY].
- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent drag event will be canceled.

Example:

```
// Allowing dragging the geo object only along the horizontal axis.
geoObject.events.add("beforedrag", function (event) {
    var originalOffset = event.get("pixelOffset");
    event.callMethod("setPixelOffset", [originalOffset[0], 0]);
});
```

beforedragstart

Event preceding the "dragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- position — Coordinates relative to the document. Array in the format [pageX, pageY].
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.

drag

Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- position — Coordinates relative to the document. Array in the format [pageX, pageY].
- pixelOffset — Array of two numbers that describe the pixel offset at this step.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

dragend

End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- position — Coordinates relative to the document. Array in the format [pageX, pageY].
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

dragstart

Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- position — Coordinates relative to the document. Array in the format [pageX, pageY].
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

editorstatechange

Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- originalEvent — Original event of the geometry editor.

hintclose

Closing the hint. Instance of the **Event** class.

hintopen

Opening a hint on a geo object. Instance of the **Event** class.

GeoObjectCollection

Extends **IGeoObject**, **IGeoObjectCollection**.

Collection of geo objects. Allows you to group geo objects for adding them to the map, setting options, etc. Collections are geo objects too.

Constructor | Fields | Events | Methods

Constructor

```
GeoObjectCollection([feature[, options]])
```

Creates a collection of geo objects.

Parameters:

Parameter	Default value	Description
feature	—	Type: Object Collection description. For all purposes, corresponds to the geo object description. See **GeoObject** object.
feature.children	—	Type: **IGeoObject**[] Array of child geoobjects.
feature.geometry	—	Type: **IGeometry**|Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Geometry of a collection.
feature.properties	—	Type: **IDataManager**|Object Collection data.
options	—	Type: Object Collection options. You can set all the options described in the **GeoObject** object. Option values will be applied both to the collection itself and to its child objects, if these options are not set for them.

Example:

```
// Creating a collection of geo objects and setting options.
var myGeoObjects = new ymaps.GeoObjectCollection({}, {
    preset: "islands#redCircleIcon",
    strokeWidth: 4,
    geodesic: true
});

// Adding placemarks and a polyline to the collection.
myGeoObjects.add(new ymaps.Placemark([13.38, 52.51]));
myGeoObjects.add(new ymaps.Placemark([30.30, 50.27]));
myGeoObjects.add(new ymaps.Polyline([[13.38, 52.51], [30.30, 50.27]]));

// Adding the collection to the map.
myMap.geoObjects.add(myGeoObjects);
// Setting the map center and scale so that the whole collection is visible.
myMap.setBounds(myGeoObjects.getBounds());
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**IDataManager**	Geo object data. Inherited from **IGeoObject**.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
add	A child geo object has been added (inserted). Instance of the Event class. Names of fields that are available via the Event.get method: • index: Integer — Index of the added geo object. • child: IGeoObject — Reference to the added geo object. Inherited from IGeoObjectCollection.
boundschange	Change to coordinates of the geographical area that includes the collection and all its child geo objects. Instance of the Event class.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
pixelboundschange	Change to pixel coordinates of the area that includes the collection and all its child geo objects. Instance of the **Event** class.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
remove	A child geo object has been removed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index: Integer — Index of the deleted geo object. • child: **IGeoObject** — Reference to the deleted geo object. Inherited from **IGeoObjectCollection**.
set	A new child geo object has been added to the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index: Integer — Index of the geo object. • child: **IGeoObject** — Reference to the new geo object. • prevChild: **IGeoObject** — Reference to the previous value for this index. Inherited from **IGeoObjectCollection**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
add(child)	**GeoObjectCollection**	Adds a geo object to the collection.
each(callback[, context])		Iterates through all the items in the collection and calls a handler function for each of them.
get(index)	**IGeoObject**	Returns a child geo object with the specified index. Inherited from **IGeoObjectCollection**.
getBounds()	Number[][]|null	Returns geographical coordinates of the area that covers the collection and all its child geo objects.
getIterator()	**IIterator**	Returns iterator for the collection.
getLength()	Integer	Returns the number of geo objects in the collection.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getPixelBounds()	Number[][]|null	Returns global pixel coordinates of the area that spans the collection and all its child geo objects.
indexOf(object)	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned. Inherited from **IGeoObjectCollection**.
remove(child)	**GeoObjectCollection**	Removes a geo object from the collection.
removeAll()	**GeoObjectCollection**	Removes all the geo objects from the collection.
set(index, child)	**GeoObjectCollection**	Adds a new child geo object to the collection.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.
splice(index, number)	**GeoObjectCollection**	Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).
toArray()	**IGeoObject**[]	Returns an array containing all the collection's geo objects at the time of the method call.

Events details

boundschange

Change to coordinates of the geographical area that includes the collection and all its child geo objects. Instance of the **Event** class.

pixelboundschange

Change to pixel coordinates of the area that includes the collection and all its child geo objects. Instance of the **Event** class.

Methods details

add

```
{GeoObjectCollection} add(child)
```

Adds a geo object to the collection.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns a reference to the collection.

Parameters:

Parameter	Default value	Description
child*	—	Type: IGeoObject Child object.

* Mandatory parameter/option.

each

```
{} each(callback[, context])
```

Iterates through all the items in the collection and calls a handler function for each of them.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function.
context	—	Type: Object Context for the function.

* Mandatory parameter/option.

getBounds

```
{Number[][]|null} getBounds()
```

Returns geographical coordinates of the area that covers the collection and all its child geo objects.

Example:

```
// Setting the map center and zoom so that the entire collection is displayed.
myMap.setBounds(myCollection.getBounds());
```

getIterator

```
{IIterator} getIterator()
```

Returns iterator for the collection.

Example:

```
// Searching the collection for a geo object with the "Polyline" geometry.
var iterator = myGroup.getIterator(),
    object;
while ((object = iterator.getNext()) != iterator.STOP_ITERATION) {
    if (object.geometry.getType() == "LineString") {
        break;
    }
}
```

getLength

```
{Integer} getLength()
```

Returns the number of geo objects in the collection.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Returns global pixel coordinates of the area that spans the collection and all its child geo objects.

remove

```
{GeoObjectCollection} remove(child)
```

Removes a geo object from the collection.

Returns a reference to the collection.

Parameters:

Parameter	Default value	Description
child*	—	Type: **IGeoObject** Child object.

* Mandatory parameter/option.

removeAll

```
{GeoObjectCollection} removeAll()
```

Removes all the geo objects from the collection.

Returns a reference to the collection.

set

```
{GeoObjectCollection} set(index, child)
```

Adds a new child geo object to the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index.
child*	—	Type: **IGeoObject** Child object.

* Mandatory parameter/option.

splice

```
{GeoObjectCollection} splice(index, number)
```

Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

Returns collection of deleted geo objects.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of the geo object to start deletion from.
number*	—	Type: Integer The number of geo objects to be deleted.

* Mandatory parameter/option.

Example:

```
// Removes the second object.
myGeoObjects.splice(1, 1);
// Puts a new "obj" object in the second position.
myGeoObjects.splice(1, 0, obj);
// Replaces the second object with the new "obj" object.
myGeoObjects.splice(1, 1, obj);
```

toArray

```
{IGeoObject[]} toArray()
```

Returns an array containing all the collection's geo objects at the time of the method call.

geoQuery

Static function.

Forms a data set from the specified source and creates an instance of GeoQueryResult based on it.

Returns result containing data from the source.

```
{ GeoQueryResult } geoQuery(source)
```

Parameters:

Parameter	Default value	Description
source*	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Source of geo objects: • IGeoObject — Object that implements the corresponding interface. • IGeoObject[] — Array of objects that implement the corresponding interface. • ICollection — Collection of objects that implement the IGeoObject interface. • ICollection[] — Array of object collections that implement the IGeoObject interface. • vow.Promise — A promise object that passes the data source for geoQuery to the handler function. The handler function can also be passed an object with the geoObjects field containing the data source for geoQuery. • GeoQueryResult — Object of the GeoQueryResult class. • String|Object — String or object with a JSON description of objects. JSON object descriptions are formed using the following approach (see the example below). An object may be an entity or a collection of entities. A collection of entities is made up of an object with the following fields: • type — Type of object. The value of the field must be "FeatureCollection". • features — Array of child entities in the collection. Child objects may be entities or nested collections of entities. An entity is made up of an object with the following fields: • type — Type of object. The value of the field must be "Feature". • geometry — Object geometry. Contains the "type" and "coordinates" fields. Corresponds to the parameter passed to the constructor for the ymaps.GeoObject object. • options — Options for the geo object. • properties — Geo object data.

* Mandatory parameter/option.

Examples:

1.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Creating GeoQueryResult from a single geo object.
var placemark = new ymaps.Placemark([34, 56]);
ymaps.geoQuery(placemark).addToMap(myMap);
```

2.

```
// Creating GeoQueryResult from an array of geo objects.
var objects = [
        new ymaps.Placemark([34, 56]),
        new ymaps.Rectangle([[34, 56], [36, 57]])
    ];
ymaps.geoQuery(objects).addToMap(myMap);
```

3.

```
// Creating GeoQueryResult from a collection of geo objects.
var result = ymaps.geoQuery(myMap.geoObjects).searchIntersect(myMap);
alert("Number of geo objects in the visible map area: " + result.getLength());
```

4.

```
// Creating GeoQueryResult from vow.Deferred.
var result = ymaps.geoQuery(ymaps.geocode('Siromyatnicheskiy
lane')).searchInside(myGeoBounds);
// Because the data source is asynchronous, we must wait for result processing.
result.then(function () {
    alert('Number of objects located inside the specified area: ' +
result.getLength());
});
```

5.

```
// Creating GeoQueryResult from JSON.
var result = ymaps.geoQuery({
        type: 'FeatureCollection',
            features: [
                {
                    type: 'Feature',
                    geometry: {
                        type: 'Circle',
                        coordinates: [15, 15],
                        radius: 100
                    }
                },
                {
                    type: 'Feature',
                    geometry: {
                        type: 'LineString',
                        coordinates: [[15, 16], [66, 23]]
                    }
                },
                {
                    type: 'FeatureCollection',
                    features: [
                        {
                            type: 'Feature',
                            geometry: {
                                type: 'Point',
                                coordinates: [12, 41]
                            },
                            properties: {
                              name: 'point'
                            },
                            options: {
                                preset: 'islands#yellowIcon'
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
                                }
                            }
                        ]
                    }
                ]
        });
// Adding non-point objects to the map as-is.
result.search('geometry.type != "Point"').addToMap(myMap);
// Adding point objects to the map via the clusterer.
myMap.geoObjects.add(result.search('geometry.type == "Point"').clusterize());
```

GeoQueryResult

Extends **IPromiseProvider**.

Geo query result.

Constructor | Methods

Constructor

```
GeoQueryResult(source)
```

Parameters:

Parameter	Default value	Description
source *	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Source of geo objects: • IGeoObject — Object that implements the corresponding interface. • IGeoObject[] — Array of objects that implement the corresponding interface. • ICollection — Collection of objects that implement the IGeoObject interface. • ICollection[] — Array of object collections that implement the IGeoObject interface. • IPromiseProvider — A promise object that passes the data source for geoQuery to the handler function. The handler function can also be passed an object with the geoObjects field containing the data source for geoQuery. • GeoQueryResult — Object of the GeoQueryResult class. • String|Object — String or object with a JSON description of objects. • Object[] — Array of JSON descriptions of geometries. An array item is an object with the "type" and "coordinates" fields. When describing a circle, the "radius" field is also mandatory. When describing a polygon, the optional "fillRule" field may also be specified. JSON object descriptions are formed using the following approach (see the example below). An object may be an entity or a collection of entities. A collection of entities is made up of an object with the following fields: • type — Type of object. The value of the field must be "FeatureCollection". • features — Array of child entities in the collection. Child objects may be entities or nested collections of entities.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		An entity is made up of an object with the following fields: • type — Type of object. The value of the field must be "Feature". • geometry — Object geometry. Contains the "type" and "coordinates" fields. Corresponds to the parameter passed to the constructor for the ymaps.GeoObject object. • options — Options for the geo object. • properties — Geo object data.

* Mandatory parameter/option.

Examples:

1.

```
// Creating GeoQueryResult from a single geo object.
var placemark = new ymaps.Placemark([34, 56]);
ymaps.geoQuery(placemark).addToMap(myMap);
```

2.

```
// Creating GeoQueryResult from an array of geo objects.
var objects = [
        new ymaps.Placemark([34, 56]),
        new ymaps.Rectangle([[34, 56], [36, 57]])
    ];
ymaps.geoQuery(objects).addToMap(myMap);
```

3.

```
// Creating GeoQueryResult from a collection of geo objects.
var result = ymaps.geoQuery(myMap.geoObjects).searchIntersect(myMap);
alert("Number of geo objects in the visible map area: " + result.getLength());
```

4.

```
// Creating GeoQueryResult from vow.Deferred.
var result = ymaps.geoQuery(ymaps.geocode('Siromyatnicheskiy
lane')).searchInside(myGeoBounds);
// Because the data source is asynchronous, we must wait for result processing.
result.then(function () {
    alert('Number of objects located inside the specified area: ' +
result.getLength());
});
```

5.

```
// Creating GeoQueryResult from JSON.
var result = ymaps.geoQuery({
        type: 'FeatureCollection',
            features: [
                {
                    type: 'Feature',
                    geometry: {
                        type: 'Circle',
                        coordinates: [15, 15],
                        radius: 100
                    }
                },
                {
                    type: 'Feature',
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
                    geometry: {
                        type: 'LineString',
                        coordinates: [[15, 16], [66, 23]]
                    }
                },
                // Object collections can be nested.
                {
                    type: 'FeatureCollection',
                    features: [
                        {
                            type: 'Feature',
                            geometry: {
                                type: 'Point',
                                coordinates: [12, 41]
                            },
                            properties: {
                              name: 'point'
                            },
                            options: {
                                preset: 'islands#yellowIcon'
                            }
                        }
                    ]
                }
            ]
        });
// Adding non-point objects to the map as-is.
result.search('geometry.type != "Point"').addToMap(myMap);
// Adding point objects to the map via the clusterer.
myMap.geoObjects.add(result.search('geometry.type == "Point"').clusterize());
```

Methods

Name	Returns	Description
add(source)	**GeoQueryResult**	Adds objects from the source to objects in the result. It does not change the source object; it creates a new object containing a combined set of geo objects.
addEvents(events, callback, context)	**GeoQueryResult**	Assigns event handlers to selection items.
addTo(collection)	**GeoQueryResult**	Method for adding objects to a collection of geo objects.
addToMap(map)	**GeoQueryResult**	Method for adding objects to the map.
applyBoundsToMap(map[, options])	**GeoQueryResult**	Method for setting the visible area of the map so that all the objects in the selection are visible.
clusterize([options])	**Clusterer**	This method creates a clusterer and adds objects from the selection to it. If the selection data are not ready yet, they will be added to the clusterer immediately after processing, and the returned clusterer will be empty at first. Only objects with the "Point" geometry will be added to the clusterer.
each(callback, context)	**GeoQueryResult**	Returns self-reference.
get(index)	**IGeoObject**	Returns a selection item by index.
getBounds()	Number[][]|null	Returns geographical coordinates of the area that includes all the result objects.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getCenter(map)	Number[]	Method that returns the center of the area that covers all the result objects, in geographical coordinates.
getCentralObject(map)	**IGeoObject**|null	Method for getting the object that is closest to the center of the visible area of the map.
getClosestTo(object)	**IGeoObject**|null	Method that returns the selection object that is closest to the one specified. If an object that is already in the selection is given as input, it returns a different object from the selection that is closest to the one specified. Note that many of the geo objects must be added to the map for correct calculations.
getExtreme(key)	Number	Returns the maximum and minimum coordinate values among the coordinates of objects in the selection.
getExtremeObject(key)	**IGeoObject**	Method that returns the object with the minimum or maximum coordinates among the coordinates of objects in the selection.
getGlobalPixelBounds(map)	Number[][]|null	Method that returns global pixel coordinates of the area for the current map zoom.
getGlobalPixelCenter(map)	Number[]	Returns coordinates of the center of the area, that covers all the result objects, in global pixel coordinates for the current map zoom.
getIterator()	**IIterator**	Returns iterator for objects in the result.
getLength()	Number	Returns number of items in the result.
getMaxZoom(map[, options])	Number	Method that calculates the maximum zoom level at which all the objects fall within the visible area of the map.
getParent()	**GeoQueryResult**|null	Returns reference to the parent selection, if the current selection was created as the result of changing another GeoQueryResult object.
indexOf(item)	Number	Returns index of the item in the selection. If the item was not found, it returns -1.
intersect(result)	**GeoQueryResult**	This method creates a new selection containing common items for two different selections.
isReady()	Boolean	Returns whether the selection results are ready or are still being processed.
map(callback[, context])	**GeoQueryResult**	Method that calls the "callback" method for all the selection items and forms a new selection based on the results received.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
remove(objects)	**GeoQueryResult**	Deletes objects from the result. It does not change the source object; it creates a new object containing the resulting set of geo objects.
removeEvents(events, callback, context)		Deletes the subscription to events from an object. Note that in order to unsubscribe correctly, the arguments passed must be exactly the same as the ones for subscribing via the "addEvents" method.
removeFrom(collection)	**GeoQueryResult**	Method for deleting objects from a collection.
removeFromMap(map)	**GeoQueryResult**	Method for deleting objects from the map.
reverse()	**GeoQueryResult**	Rearranges the selection items in reverse order and returns the new selection.
search(condition)	**GeoQueryResult**	Method for searching for selection objects that meet the conditions.
searchContaining(object)	**GeoQueryResult**	Method that creates a new selection from objects containing the specified object. Note that many of the geo objects must be added to the map for correct calculations.
searchInside(object)	**GeoQueryResult**	Method that creates a new selection from objects that are completely included in the specified object. Note that many of the geo objects must be added to the map for correct calculations.
searchIntersect(object[, options])	**GeoQueryResult**	Method that creates a new selection from objects intersecting the specified object. Note that many of the geo objects must be added to the map for correct calculations.
setOptions(key[, value])	**GeoQueryResult**	Method for setting option values for all the selection items.
setProperties(path, value)	**GeoQueryResult**	Method for setting the value of the "properties" field for all the selection items.
slice(begin[, end])	**GeoQueryResult**	Method that returns a slice of the selection.
sort(comparator)		Method for sorting selection objects. It does not change the original selection; it creates a new one containing sorted objects.
sortByDistance(object)	**GeoQueryResult**	Method for getting a selection containing objects sorted by their distance from the specified object. Note that many of the geo objects must be added to the map for correct calculations. It does not change the original selection.
then([onFulfill[, onReject, context)	**GeoQueryResult**	Subscription to the promise.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
unsetOptions(keys)	GeoQueryResult	Method for nullifying option values for all the selection items.
unsetProperties(path)	GeoQueryResult	Method for nullifying the value of the "properties" field for all the selection items.

Methods details

add

```
{GeoQueryResult} add(source)
```

Adds objects from the source to objects in the result. It does not change the source object; it creates a new object containing a combined set of geo objects.

Returns a new object with the combined set of geo objects.

Parameters:

Parameter	Default value	Description
source*	—	Type: Object Source of geo objects: • IGeoObject — Object that implements the corresponding interface. • IGeoObject[] — Array of objects that implement the corresponding interface. • ICollection — Collection of objects that implement the IGeoObject interface. • ICollection[] — Array of object collections that implement the IGeoObject interface. • vow.Promise — A promise object that passes the data source for geoQuery to the handler function. The handler function can also be passed an object with the geoObjects field containing the data source for geoQuery. • GeoQueryResult — Object of the GeoQueryResult class. • String|Object — String or object with a JSON description of objects. • Object[] — Array of JSON descriptions of geometries. An array item is an object with the "type" and "coordinates" fields. When describing a circle, the "radius" field is also mandatory. When describing a polygon, the optional "fillRule" field may also be specified.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		JSON object descriptions are formed using the following approach (see the example below). An object may be an entity or a collection of entities. A collection of entities is made up of an object with the following fields: • type — Type of object. The value of the field must be "FeatureCollection". • features — Array of child entities in the collection. Child objects may be entities or nested collections of entities. An entity is made up of an object with the following fields: • type — Type of object. The value of the field must be "Feature". • geometry — Object geometry. Contains the "type" and "coordinates" fields. Corresponds to the parameter passed to the constructor for the ymaps.GeoObject object. • options — Options for the geo object. • properties — Geo object data.

* Mandatory parameter/option.

Examples:

1.

```
// Adding a single geo object to GeoQueryResult.
var placemark = new ymaps.Placemark([34, 56]);
myGeoQueryResult.add(placemark);
```

2.

```
// Adding an array of geo objects to GeoQueryResult.
var objects = [
        new ymaps.Placemark([34, 56]),
        new ymaps.Rectangle([[34, 56], [36, 57]])
    ];
// Note that a different GeoQueryResult object will be obtained in the result,
// while the old one remains unchanged.
var newResult = myGeoQueryResult.add(objects);
```

3.

```
// Adding collections of geo objects to GeoQueryResult.
// Note that a different GeoQueryResult object will be obtained in the result,
// while the old one remains unchanged.
var newResult =
myGeoQueryResult.add(myMap.geoObjects).searchIntersect(myBounds);
alert("Number of geo objects in the specified area: " +
myGeoQueryResult.getLength());
```

4.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Adding geocoding results to GeoQueryResult.
// Note that a different GeoQueryResult will be obtained in the result,
// while the old one remains unchanged.
var newResult = myGeoQueryResult.add(ymaps.geocode('Siromyatnicheskiy lane'));
newResult.searchInside(myGeoBounds);
// Because the data source is asynchronous, we must wait for the result to be
processed.
result.then(function () {
    // The source object still exists.
    alert('Before adding objects, the number of objects was:' +
myGeoQueryResult.getLength());
    // The new object has both the old geo objects and the ones that were added.
    alert('After adding objects, the number of objects is: ' +
result.getLength());
});
```

5.

```
// Adding objects from a JSON string.
var result = ymaps.geoQuery(myMap.geoObjects),
    extendedResult = result.add('{"type": "Feature", "geometry": { "type":
"Point", "coordinates": [15, 64] }}');
extendedResult.addToMap(myMap);
```

addEvents

```
{GeoQueryResult} addEvents(events, callback, context)
```

Assigns event handlers to selection items.

Returns self-reference.

Parameters:

Parameter	Default value	Description
events*	—	Type: String|String[] Event type or array of event types that the subscription ends on.
callback*	—	Type: Function Handler function.
context*	—	Type: Object Context for the handler function.

* Mandatory parameter/option.

Example:

```
ymaps.geoQuery(map.geoObjects).search('geometry.type="Circle"').addEvents('click
', function () {
    alert('You clicked a circle!');
});
```

addTo

```
{GeoQueryResult} addTo(collection)
```

Method for adding objects to a collection of geo objects.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
collection*	—	Type: ICollection The collection that objects will be added to.

* Mandatory parameter/option.

Example:

```
// Showing objects in the northern hemisphere on the map.
var result1 = ymaps.geoQuery(placemarks).search('lat &gt;
0').addTo(myMap.geoObjects);
```

addToMap

```
{GeoQueryResult} addToMap(map)
```

Method for adding objects to the map.

Returns self-reference.

Parameters:

Parameter	Default value	Description
map*	—	Type: Map The map that objects will be added to.

* Mandatory parameter/option.

Example:

```
// Showing objects in the northern hemisphere on the map.
var result1 = ymaps.geoQuery(placemarks).search('lat &gt; 0').addToMap(myMap);
```

applyBoundsToMap

```
{GeoQueryResult} applyBoundsToMap(map[, options])
```

Method for setting the visible area of the map so that all the objects in the selection are visible.

Returns self-reference.

Parameters:

Parameter	Default value	Description
map*	—	Type: Map Map.
options	—	Type: Object Options.
options.checkZoomRange	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Checks whether the specified zoom level can be set. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
options.duration	0	Type: Number Animation duration, in milliseconds.
options.preciseZoom	false	Type: Boolean The ability to use fractional zoom levels.
options.timingFunction	'linear'	Type: String Timing function. The same as the value of the CSS property transition-timing-function. Full list of values: http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag
options.useMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager**.
options.zoomMargin	0	Type: Number|Number[] Offset from the borders of the visible area of the map. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" option is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager **map.margin.Manager**.

* Mandatory parameter/option.

Examples:

1.

```
// Making a geocoding request and setting the visible area of the map
// so that all the results will be visible on it immediately.
var result = ymaps.geoQuery(ymaps.geocode('Lenin
street')).applyBoundsToMap(myMap);
// Note that the request to the server is asynchronous and we need to wait for
results.
result.then(function () {
    alert("Got results and adjusted the visible area of the map.");
}, function () {
    alert("An error occurred.");
});
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// For synchronous requests, the new map area is set immediately.
var result = ymaps.geoQuery(objects).applyBoundsToMap(myMap);
alert('The visible area of the map changed.');
```

clusterize

```
{Clusterer} clusterize([options])
```

This method creates a clusterer and adds objects from the selection to it. If the selection data are not ready yet, they will be added to the clusterer immediately after processing, and the returned clusterer will be empty at first. Only objects with the "Point" geometry will be added to the clusterer.

Returns clusterer.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options passed to the object constructor **Clusterer**.

Example:

```
// Selecting only cafes and adding them to the clusterer.
var clusterer =
ymaps.geoQuery(objects).search('properties.type="Cafe"').clusterize();
myMap.geoObjects.add(clusterer);
```

each

```
{GeoQueryResult} each(callback, context)
```

Returns self-reference.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function. Called for each of the selection items and gets the item as input.
context*	—	Type: Object Context for the handler function.

* Mandatory parameter/option.

Example:

```
// Hiding red placemarks in the visible area of the map.
ymaps.geoQuery(placemarks).searchIntersect(myMap).each(function(pm) {
    if (pm.options.get('preset') == 'islands#redIcon') {
        myMap.geoObjects.remove(pm);
    }
});
```

get

```
{IGeoObject} get(index)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns a selection item by index.

Parameters:

Parameter	Default value	Description
index*	—	Type: Number Index of items.

* Mandatory parameter/option.

Examples:

1.

```
// Example with synchronous processing.
var result = ymaps.geoQuery(placemarks).sort('lat'),
    // The southernmost object.
    southObject = result.get(0),
    // The northernmost object.
    northObject = result.get(result.getLength() - 1);
```

2.

```
// Example with asynchronous processing.
var result = ymaps.geoQuery(ymaps.geocode('Moscow cafe')).sort('lat');
// We have to wait for the result to be ready before processing further.
result.then(function () {
    // Southernmost object.
    var southObject = result.get(0),
    // Northermost object.
        northObject = result.get(result.getLength() - 1);
});
```

getBounds

```
{Number[][]|null} getBounds()
```

Returns geographical coordinates of the area that includes all the result objects.

Examples:

1.

```
// Example of adding the objects synchronously.
// Setting the map center and zoom so that all the objects added to the map are
displayed.
var myResult = ymaps.geoQuery(myMap.geoObjects);
myMap.setBounds(myResult.getBounds(), { checkZoomRange: true });
```

2.

```
// Example of adding the objects asynchronously.
// Waiting for the data and only then call the getBounds method.
var myResult = ymaps.geoQuery(ymaps.geocode(' Zveronozhka river')).then(function
() {
     myMap.setBounds(myResult.getBounds(), { checkZoomRange: true });
});
```

getCenter

```
{Number[]} getCenter(map)
```

Method that returns the center of the area that covers all the result objects, in geographical coordinates.

Returns coordinates of the center of the area, in geographical coordinates.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** The map to make calculations for.

* Mandatory parameter/option.

Example:

```
// Moving the map center to the center of the area that covers the objects.
myMap.setCenter(ymaps.geoQuery(objects).getCenter());
```

getCentralObject

```
{IGeoObject|null} getCentralObject(map)
```

Method for getting the object that is closest to the center of the visible area of the map.

Returns reference to a geo object, or null if the selection is empty.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** Map.

* Mandatory parameter/option.

Examples:

1.

```
// Example of working with synchronous operations.
// Opening a balloon at the object closest to the center of the visible area of
the map.
ymaps.geoQuery(objects).getCentralObject(myMap).balloon.open();
```

2.

```
// Usage example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Ulitsa Stroiteley'));
// We just sent the geocoding request so we need to wait for
// the result.
result.then(function () {
    result.getCentralObject(myMap).balloon.open();
});
```

getClosestTo

```
{IGeoObject|null} getClosestTo(object)
```

Method that returns the selection object that is closest to the one specified. If an object that is already in the selection is given as input, it returns a different object from the selection that is closest to the one specified. Note that many of the geo objects must be added to the map for correct calculations.

Returns the selection object closest to the one specified, or null if the object cannot be found.

Parameters:

Parameter	Default value	Description
object*	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The object that the search will be relative to. Accepts the following values: • IGeoObject — Object that implements the **IGeoObject** interface. • IGeometry — Object that implements the **IGeometry** interface. • Map — The map. In this case, the reference object is the rectangular border of the map. • Number[] — Coordinates of a point. • Object — JSON description of a geometry. Contains the "type" and "coordinates" fields. When describing a circle, the "radius" field is also mandatory. When describing a polygon, the optional "fillRule" field may also be specified.

* Mandatory parameter/option.

Examples:

1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);

// 1. IGeoObject.
// A search can be made relative to an outside object.
var polyline = new ymaps.Polyline([[35, 65], [35, 66], [34, 62], [34, 63]]);
myMap.geoObjects.add(polyline);
var closestObject = result.getClosestTo(polyline);
// The search can be based on an object from the same selection.
var closestToFirst = result.getClosestTo(result.get(0));

// 2. IGeometry.
var closestToGeometry = result.getClosestTo(placemark.geometry);

// 3. Map.
// Finding the object closest to the border of the visible area of the map.
var edgeObject = result.getClosestTo(myMap);

// 4. Object nearest a point.
var closestObject = result.getClosestTo([34, 53]);
```

2.

```
// Example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Paris')).addToMap(myMap);
// Waiting for the server response and getting the object nearest the point.
result.then(function () {
    var closestObject = result.getClosestTo([34, 65]);
    // If the response is empty, the nearest object will not be found.
    if (closestObject) {
        closestObject.balloon.open();
    }
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getExtreme

```
{Number} getExtreme(key)
```

Returns the maximum and minimum coordinate values among the coordinates of objects in the selection.

Parameters:

Parameter	Default value	Description
key*	—	Type: String Key for receiving data. Accepts the following values: • top • right • bottom • left

* Mandatory parameter/option.

Examples:

1.

```
// Example with synchronous operations.
alert('Northernmost coordinate: ',
ymaps.geoQuery(myMap.geoObjects).getExtreme('top'));
```

2.

```
// Example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Novgorod'));
// Waiting for the server response and getting the northernmost coordinate.
result.then(function () {
    alert('Northernmost coordinate in the response: ' +
result.getExtreme('top'));
});
```

getExtremeObject

```
{IGeoObject} getExtremeObject(key)
```

Method that returns the object with the minimum or maximum coordinates among the coordinates of objects in the selection.

Returns object with the required coordinate.

Parameters:

Parameter	Default value	Description
key*	—	Type: String Key for searching for an object. Accepts the following values: • top • right • bottom • left

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Examples:

1.

```
// Example with synchronous operations.
// Opening a balloon on the northernmost object.
var topObject = ymaps.geoQuery(myMap.geoObjects).getExtremeObject('top');
topObject.balloon.open();
```

2.

```
// Example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Laptev Sea'));
// Waiting for the server response in order to work with data.
result.then(function () {
    var topObject = result.getExtremeObject('top');
    // If the response is empty, the object will not be found.
    if (topObject) {
        topObject.balloon.open();
    }
});
```

getGlobalPixelBounds

```
{Number[][]|null} getGlobalPixelBounds(map)
```

Method that returns global pixel coordinates of the area for the current map zoom.

Returns global pixel coordinates of the area that includes all objects in the result.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** The map that calculations are being made for.

* Mandatory parameter/option.

Example:

```
var result =
ymaps.geoQuery(placemarks).search('properties.type="shop"').getGlobalPixelBounds
(myMap);
if (Math.abs(result[0][0] - result[1][0]) &gt; myMap.container.getSize()[0]) {
   alert('Objects are too wide to fit on the map!');
}
```

getGlobalPixelCenter

```
{Number[]} getGlobalPixelCenter(map)
```

Returns coordinates of the center of the area, that covers all the result objects, in global pixel coordinates for the current map zoom.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** The map to make calculations for.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Calculating the tile number that contains the center of the area that spans
the result.
var globalPixelCenter = ymaps.geoQuery(objects).getGlobalPixelCenter(myMap),
    tileNumber = [
        Math.floor(globalPixelCenter[0] / 256),
        Math.floor(globalPixelCenter[1] / 256)
    ];
alert('Number of the center tile: ' + tileNumber[0] + ' ' + tileNumber[1]);
```

getIterator

```
{IIterator} getIterator()
```

Returns iterator for objects in the result.

Examples:

1.

```
// Using the iterator with synchronous operations.
// Searching for elements that match the click coordinates.
myMap.events.add('click', function (event) {
     var iterator = ymaps.geoQuery(myMap.geoObjects)
             .searchContaining(event.getCoordinates())
             .getIterator(),
         obj;
    while ((obj = iterator.getNext()) != iterator.STOP_ITERATION) {
        // Performing the necessary actions on a geo object.
    }
});
```

2.

```
// Using an iterator with asynchronous operations.
// Creating a result from the geocoder query.
var result = ymaps.geoQuery(ymaps.geocode("Starokolpakskiy street")).search("lat
> 20");
// Note that we only sent the request to the server, and the response will come
later.
// Because the request is asynchronous, we need to wait for it to be ready before
getting the result.
result.then(function () {
     var iterator = result.getIterator(),
         obj;
     while ((obj = iterator.getNext())  != iterator.STOP_ITERATION) {
         // Performing the necessary actions on the geo object.
     }

});

});
```

getLength

```
{Number} getLength()
```

Returns number of items in the result.

Examples:

1.

```
var result = ymaps.geoQuery(myMap.geoObject).searchIntersect(myPolygon);
alert('The number of geo objects that intersect with the polygon: ' +
result.getLength());
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
var result = ymaps.geoQuery(ymaps.geocode('Ivanovo')).searchInside(myMap);
// Because we sent the request to the server, we cannot immediately count
// the number of elements in the result. We have to wait for the response.
result.then(function () {
    alert('Number of objects in the visible area of the map: ' +
result.getLength());
});
```

getMaxZoom

```
{Number} getMaxZoom(map[, options])
```

Method that calculates the maximum zoom level at which all the objects fall within the visible area of the map.

Returns maximum map zoom level.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** The map that the calculation is being made for.
options	—	Type: Object Options.
options.useMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager**.

* Mandatory parameter/option.

Example:

```
// Calculating the maximum zoom level
// at which all the selection objects are visible
// and setting a restriction for the map.
var maxZoom = ymaps.geoQuery(objects).getMaxZoom();
myMap.options.set('maxZoom', maxZoom);
```

getParent

```
{GeoQueryResult|null} getParent()
```

Returns reference to the parent selection, if the current selection was created as the result of changing another GeoQueryResult object.

Example:

```
ymaps.geoQuery(objectsArray)
    // Selecting only objects of the type "cafe" and defining their styles.
    .search('properties.type == "cafe"')
    .setOptions('preset', 'islands#yellowDotIcon')
    // Then, without interrupting the chain of calls,
    // getting back to the original selection and defining styles for another
group of objects.
    .getParent()
    .search('properties.type == "shop"')
    .setOptions('preset', 'islands#greenDotIcon');
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

indexOf

```
{Number} indexOf(item)
```

Returns index of the item in the selection. If the item was not found, it returns -1.

Parameters:

Parameter	Default value	Description
item*	—	Type: **IGeoObject** The required object.

* Mandatory parameter/option.

Example:

```
// Sorting the selection by the "name" field.
var result = ymaps.geoQuery(polygons).sort('properties.name');
alert('New position of the first item: ' + result.indexOf(polygons[0]));
```

intersect

```
{GeoQueryResult} intersect(result)
```

This method creates a new selection containing common items for two different selections.

Returns the new selection that contains the intersection result.

Parameters:

Parameter	Default value	Description
result*	—	Type: **GeoQueryResult** The selection to intersect the first one with.

* Mandatory parameter/option.

Examples:

1.

```
// Example of intersection with synchronous operations.
var result = ymaps.geoQuery(placemarks),
    greenObjects = result.search('properties.color="green"'),
    roundObjects = result.search('properties.shape="round"'),
    greenRoundObjects = greenObjects.intersect(roundObjects);
alert('Number of round green objects: ' + greenRoundObjects.getLength());
```

2.

```
// Example of intersection with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Ivanovka')),
    filteredByLat = result.search('lat > 56'),
    filteredByLong = result.search('long > 36'),
    intersectedResult = filteredByLat.intersect(filteredByLong);
// Because the original request is asynchronous, we must wait for data to be
ready.
intersectedResult.then(function () {
    alert('Number of objects with the name "Ivanovka" +
        'with coordinates higher than [56, 36]: ' +
        intersectedResult.getLength());
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

isReady

```
{Boolean} isReady()
```

Returns whether the selection results are ready or are still being processed.

Example:

```
var result = ymaps.geoQuery(ymaps.geocode('Ivanovo'));
if (!result.isReady()) {
    result.then(function () {
        // Processing data.
    });
} else {
    // Processing data.
}
```

map

```
{GeoQueryResult} map(callback[, context])
```

Method that calls the "callback" method for all the selection items and forms a new selection based on the results received.

Returns the new selection.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function. Takes a selection item as input. Returns an instance of **IGeoObject**.
context	—	Type: Object Context for the handler function.

* Mandatory parameter/option.

Example:

```
// Adding only circle objects to the map.
var circlesResult =
ymaps.geoQuery(objects).search('geometry.type="Circle"').addToMap(myMap),
// We'll also add placemarks that indicate the circle centers.
    centers = circlesResult.map(function (object) {
        return new ymaps.Placemark(object.geometry.getCenter());
    }).addToMap(myMap);
```

remove

```
{GeoQueryResult} remove(objects)
```

Deletes objects from the result. It does not change the source object; it creates a new object containing the resulting set of geo objects.

Returns new object with the resulting set of geo objects.

Parameters:

Parameter	Default value	Description
objects*	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Objects can be presented in various forms: • IGeoObject — Object that implements the corresponding interface. • IGeoObject[] — Array of objects that implement the corresponding interface. • ICollection — Collection of objects that implement the IGeoObject interface. • ICollection[] — Array of object collections that implement the IGeoObject interface. • GeoQueryResult — Object of the GeoQueryResult class. Note that with asynchronous operations, the result has to be prepared before it can be deleted correctly. • vow.Promise — Object of the vow.Deferred class. Must be resolved by an array of geo objects or an object with the "geoObjects" field.

* Mandatory parameter/option.

Examples:

1.

```
var objects = [
        new ymaps.Placemark([34, 56]),
        new ymaps.Rectangle([[34, 56], [36, 57]])
    ],
    result = ymaps.geoQuery(objects);
// Note that a different GeoQueryResult object will be obtained in the result,
// while the old one remains unchanged.
var newResult = result.remove(objects[1]);
```

2.

```
// Deleting an array of geo objects from GeoQueryResult.
var objects = [
        new ymaps.Placemark([34, 56]),
        new ymaps.Rectangle([[34, 56], [36, 57]]),
        new ymaps.Placemark([35, 64])
    ],
    result = ymaps.geoQuery(objects);
// Note that a different GeoQueryResult object will be obtained in the result,
// while the old one remains unchanged.
var newResult = result.remove([objects[1], objects[2]]);
```

3.

```
// Deleting collections of geo objects from GeoQueryResult.
// Adding objects to the map that aren't on it yet.
myGeoQueryResult.remove(myMap.geoObjects).addToMap(myMap);
```

4.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Deleting data of a different GeoQueryResult from GeoQueryResult.
var result1 = ymaps.geoQuery(placemarks).search('properties.color="green"'),
    result2 = ymaps.geoQuery(placemarks).search('properties.shape="circle"'),
    result3 = result1.remove(result2);
alert('Number of green non-round shapes: ' + result3.getLength());
```

removeEvents

```
{} removeEvents(events, callback, context)
```

Deletes the subscription to events from an object. Note that in order to unsubscribe correctly, the arguments passed must be exactly the same as the ones for subscribing via the "addEvents" method.

Parameters:

Parameter	Default value	Description
events*	—	Type: String|String[] Event type or array of types that a subscription was made to.
callback*	—	Type: Function Handler function that was specified when subscribing.
context*	—	Type: Object Context that was specified when subscribing.

* Mandatory parameter/option.

Example:

```
var callback = function () {
        alert('You clicked a circle!');
    };
ymaps.geoQuery(map.geoObjects).search('geometry.type="Circle"').addEvents('click
', callback);
// ...
ymaps.geoQuery(map.geoObjects).search('geometry.type="Circle"').removeEvents('cl
ick', callback);
```

removeFrom

```
{GeoQueryResult} removeFrom(collection)
```

Method for deleting objects from a collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
collection*	—	Type: ICollection The collection to delete objects from.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Showing all objects on the map.
var result1 = ymaps.geoQuery(placemarks).addTo(myMap.geoObjects),
    // Then hiding objects in the northern hemisphere.
    result2 = result1.search('lat &gt; 0').removeFrom(myMap.geoObjects);
```

removeFromMap

```
{GeoQueryResult} removeFromMap(map)
```

Method for deleting objects from the map.

Returns self-reference.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** The map to delete objects from.

* Mandatory parameter/option.

Example:

```
// Showing all objects on the map.
var result1 = ymaps.geoQuery(placemarks).addToMap(myMap),
// Then hiding objects in the northern hemisphere.
    result2 = result1.search('lat &gt; 0').removeFromMap(myMap);
```

reverse

```
{GeoQueryResult} reverse()
```

Rearranges the selection items in reverse order and returns the new selection.

Returns the new selection with items in reverse order.

Examples:

1.

```
// Usage with synchronous requests.
var result = ymaps.geoQuery(myMap.geoObjects).sort('x'),
    invertedResult = result.reverse();
```

2.

```
// Usage with asynchronous requests.
var result = ymaps.geoQuery(ymaps.geocode('Friendly Bees village')).sort('x'),
    invertedResult = result.reverse();
invertedResult.then(function () {
    // Got the result. Ready to use.
});
```

search

```
{GeoQueryResult} search(condition)
```

Method for searching for selection objects that meet the conditions.

Returns a new selection containing search results.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
condition*	—	Type: String|Function String template for search or a filter function. A string template has the structure "<fieldname> <condition> <expression>". Acceptable values for <fieldname>: • 'lat' — Latitude. Supported only for point objects. • 'lng', 'long' — Longitude. Supported only for point objects. • 'x' — Global pixel coordinates on the X axis. Supported only for point objects. • 'y' — Global pixel coordinates on the Y axis. Supported only for point objects. • 'geometry.type' — Type of geometry. Result of executing the geometry.getType() method. • 'geometry.coordinates.<index>' — Coordinates. Result of executing the geometry.getCoordinates() method. To get access to a coordinate, specify its indexes after a dot — 'geometry.coordinates.0.1'. • 'properties.<path>' — Value of the data field. • 'options.<key>' — Value of options. Acceptable values for <condition>: • '>' • '>=' • '==', '=' • '!=' • '<=' • '<' • 'rlike', 'regexp' — Matches the <fieldname> value to a regular expression. Acceptable values for <expression>: • Number • String (must be put in quotation marks) • true • false • null • undefined

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The filter function takes a selection object as input and returns true/false. True means the object is in the resulting selection, and false means it is not.

* Mandatory parameter/option.

Examples:

1.

```
// Example of searching for objects with synchronous operations.
var result = ymaps.geoQuery(myMap.geoObjects);

         // Searching for objects with a specific type of geometry. Note that
// the field value shown in quotes is a string.
result.search('geometry.type = "Circle"')
// Searching by a coordinate.
    .search('geometry.coordinates.0 &gt; 100')
// Searching by latitude.
    .search('lat &lt; 0')
// Searching by pixel coordinates.
    .search('x &gt;= 100')
// Searching by the field value in "properties".
    .search('properties.name != null').search('properties.name rlike "(.) \\""')
    .search('properties.author.name = "Stepan"')
// Searching by option value.
    .search('options.visible = true');
```

2.

```
// Using the method with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Moose'));
// Waiting for the result to be ready before working with it.
result.then(function () {
    result.search('properties.description regExp "*village*"')
        .addToMap(myMap)
        .applyBoundsToMap(myMap);
});
```

searchContaining

```
{GeoQueryResult} searchContaining(object)
```

Method that creates a new selection from objects containing the specified object. Note that many of the geo objects must be added to the map for correct calculations.

Returns a new selection containing the desired objects.

Parameters:

Parameter	Default value	Description
object*	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The object that the search will be relative to. Accepts the following values: • IGeoObject — Object that implements the **IGeoObject** interface. • IGeometry — Object that implements the **IGeometry** interface. • Map — The map. In this case, the reference object is the rectangular border of the map. • Number[] — Coordinates of a point. If one parameter is passed, it is interpreted as a point. If two arguments are passed, it is interpreted as a circle. • Number[][] — Coordinates of a rectangular area. • Object — JSON descriptions of a geometry.

* Mandatory parameter/option.

Examples:

1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);

// 1. IGeoObject.
var polygon = new ymaps.Polygon([[[35, 65], [35, 66], [34, 62], [34, 63], [35,
65]]]);
myMap.geoObjects.add(polygon);
var objectsContainingPolygon = result.searchContaining(polygon);

// 2. IGeometry.
var objectsContainingGeometry = result.searchContaining(polygon.geometry);

// 3. Map.
var objectsContainingMapBounds = result.searchContaining(myMap);
```

2.

```
// Example of usage with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('cafe'))
        .addToMap(myMap);
// Waiting for the server response before processing results.
result.then(function () {
    var areas = result.map(function (object) {
                return new ymaps.Circle(object.geometry.getCoordinates(), 100);
            })
            .setOptions('visible', false)
            .addToMap(myMap);
    myMap.events.add('click', function (event) {
        if (areas.searchContaining(event.getCoordinates()).getLength()) {
            alert('You clicked near a cafe.');
        }
    });
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

searchInside

```
{GeoQueryResult} searchInside(object)
```

Method that creates a new selection from objects that are completely included in the specified object. Note that many of the geo objects must be added to the map for correct calculations.

Returns a new selection containing the desired objects.

Parameters:

Parameter	Default value	Description
object*	—	Type: Object The object that the search will be relative to. Accepts the following values: • IGeoObject — Object that implements the **IGeoObject** interface. • IGeometry — Object that implements the **IGeometry** interface. • Map — The map. In this case, the reference object is the rectangular border of the map.

* Mandatory parameter/option.

Examples:

1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);

// 1. IGeoObject.
var polygon = new ymaps.Polygon([[[35, 65], [35, 66], [34, 62], [34, 63], [35,
65]]]);
myMap.geoObjects.add(polygon);
var objectsInsidePolygon = result.searchInside(polygon);

// 2. IGeometry.
var objectsInsideGeometry = result.searchInside(polygon.geometry);

// 3. Map.
// We'll find objects that fit entirely inside the visible area of the map.
var visibleObject = result.searchInside(myMap);
```

2.

```
// Example of usage with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Ivanovo'))
        .setOptions('visible', false)
        .addToMap(myMap);
// Waiting for the server response and then showing objects that fit inside the
visible area of the map.
result.then(function () {
    result.searchInside(myMap).setOptions('visible', true);
});
```

searchIntersect

```
{GeoQueryResult} searchIntersect(object[, options])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Method that creates a new selection from objects intersecting the specified object. Note that many of the geo objects must be added to the map for correct calculations.

Returns a new selection containing the desired objects.

Parameters:

Parameter	Default value	Description
object*	—	Type: Object The object that the search will be relative to. Accepts the following values: • IGeoObject — Object that implements the **IGeoObject** interface. • IGeometry — Object that implements the **IGeometry** interface. • Map — The map. In this case, the reference object is the rectangular border of the map.
options	—	Type: Object Options.
options.considerOccurance	true	Type: Object Flag indicating whether we consider it an intersection when one figure is completely contained inside another.

* Mandatory parameter/option.

Examples:

1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);

// 1. IGeoObject.
var polygon = new ymaps.Polygon([[[35, 65], [35, 66], [34, 62], [34, 63], [35,
65]]]);
myMap.geoObjects.add(polygon);
var objectsIntersectPolygon = result.searchIntersect(polygon);

// 2. IGeometry.
var objectsIntersectGeometry = result.searchIntersect(polygon.geometry);

// 3. Map.
// We will only search for objects that directly intersect
// the map border. In other words, objects that are entirely inside the visible
area of the map
// will not be in the final selection.
var objectsIntersectMapBounds = result.searchIntersect(myMap,
{considerOccurance: false});
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Example of usage with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('cafe'))
        .setOptions('visible', false)
        .addToMap(myMap);

// Waiting for the server response and then showing objects that fall within the
map viewport.
result.then(function () {
    result.searchIntersect(myMap).setOptions('visible', true);
});
```

setOptions

```
{GeoQueryResult} setOptions(key[, value])
```

Method for setting option values for all the selection items.

Returns self-reference.

Parameters:

Parameter	Default value	Description
key*	—	Type: String|Object Option name or hash with options and their values.
value	—	Type: Object Option value.

* Mandatory parameter/option.

Example:

```
var result = ymaps.geoQuery(placemarks);
// Making elements visible if they fall within a rectangular area.
result.searchIntersect(myBounds).setOptions('visible', true);

// Setting options using a hash.
result.setOptions({zIndex: 10, fillColor: '#ff0005'});
```

setProperties

```
{GeoQueryResult} setProperties(path, value)
```

Method for setting the value of the "properties" field for all the selection items.

Returns self-reference.

Parameters:

Parameter	Default value	Description
path*	—	Type: String Name of the field that the value is assigned to. It may contain ".".
value*	—	Type: Object Field value.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
var result = ymaps.geoQuery(objects);
// Marking elements that fall inside the area.
result.searchIntersect(myBounds1).setProperties('intersectBounds', true);
result.searchIntersect(myBounds2).setProperties('intersectBounds', true);
// ...
result.search('properties.intersectBounds = true').addToMap(myMap);
```

slice

```
{GeoQueryResult} slice(begin[, end])
```

Method that returns a slice of the selection.

Returns the new selection containing items in the slice.

Parameters:

Parameter	Default value	Description
begin*	—	Type: Number Index of the first item in the slice.
end	—	Type: Number Index of the selection item that ends the slice. However, the last item in the new slice will be the item with the index end-1.

* Mandatory parameter/option.

Examples:

1.

```
// Making a slice with asynchronous processing.
var result = ymaps.geoQuery(map.geoObjects).slice(0, 10);
alert('Number of items in new selection:' + result.getLength());
```

2.

```
// Getting a slice with asynchronous processing.
var result = ymaps.geoQuery(ymaps.geocode('cafe Moscow')).slice(0, 10);
// The result is not ready immediately after the request.
alert('The selection is still empty. Number of items in selection: ' +
result.getLength());
// Waiting for the result to be ready and then we will see how many items will be
in the selection.
result.then(function () {
    alert('Response received. Amount of data in the selection: ' +
result.getLength());
});
```

sort

```
{} sort(comparator)
```

Method for sorting selection objects. It does not change the original selection; it creates a new one containing sorted objects.

Parameters:

Parameter	Default value	Description
comparator*	—	Type: String|Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		String with a sorting template or a compare function. The template string can be in the format '<fieldname> [<order>=asc]'. Acceptable values for <fieldname>: • 'lat' — Latitude. Supported only for point objects. • 'lng', 'long' — Longitude. Supported only for point objects. • 'x' — Global pixel coordinates on the X axis. Supported only for point objects. • 'y' — Global pixel coordinates on the Y axis. Supported only for point objects. • 'geometry.type' — Type of geometry. Result of executing the geometry.getType() method. • 'geometry.coordinates.<index>' — Coordinates. Result of executing the geometry.getCoordinates() method. To get access to a coordinate, specify its indexes after a dot — 'geometry.coordinates.0.1'. • 'properties.<path>' — Value of the data field. • 'options.<key>' — Value of options. Acceptable values for the optional <order>: parameter: • 'asc' — Sort in ascending order. • 'desc' — Sort in descending order. The compare function takes two selection items as input. Returned values: If the first object is larger than the second, the function returns the value > 0. If the first object is equal to the second, the function returns 0. If the first object is smaller than the second, the function returns < 0.

* Mandatory parameter/option.

Example:

```
// Example with synchronous operations.
var result = ymaps.geoQuery(myMap.geoObjects);
result.sort('lat').sort('x')
    .sort('properties.name desc')
    .sort('options.preset')
    .sort(function (a, b) {
        if (a.properties.get('name') == b.properties.get('name')) {
            return a.geometry.getCoordinates()[0] - b.geometry.getCoordinates()
[0];
        } else {
           return (a.properties.get('name') &gt; b.properties.get('name')) ? 1 :
-1;
        }
    });
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

sortByDistance

```
{GeoQueryResult} sortByDistance(object)
```

Method for getting a selection containing objects sorted by their distance from the specified object. Note that many of the geo objects must be added to the map for correct calculations. It does not change the original selection.

Returns the new ordered selection.

Parameters:

Parameter	Default value	Description
object*	—	Type: Object The object that the distance will be calculated to. Accepts the following values: • IGeoObject — Object that implements the **IGeoObject** interface. • IGeometry — Object that implements the **IGeometry** interface. • Map — The map. In this case, the reference object is the rectangular border of the map. • Number[] — Coordinates of a point. • Number[][] — Coordinates of a rectangular area. • Object — JSON description of a geometry. Contains the "type" and "coordinates" fields. When describing a circle, the "radius" field is also mandatory. When describing a polygon, the optional "fillRule" field may also be specified.

* Mandatory parameter/option.

Examples:

1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);

// 1. IGeoObject.
var polyline = new ymaps.Polyline([[35, 65], [35, 66], [34, 62], [34, 63]]);
myMap.geoObjects.add(polyline);
var sortedByPolyline = result.sortByDistance(polyline);

// 2. IGeometry.
var sortedByGeometry = result.sortByDistance(placemark.geometry);

// 3. Map.
var sortedByMapBounds = result.sortByDistance(myMap);

// 4. The selection sorted by a point.
var sortedByPoint = result.sortByDistance([34, 53]);
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Example with asynchronous operations.
var result =
ymaps.geoQuery(ymaps.geocode('Paris')).addToMap(myMap).sortByDistance([45, 64]);
// Waiting for the result from the server and getting the nearest and furthest
// object relative to the point.
result.then(function () {
    alert('The nearest object has the coordinates ' +
result.get(0).geometry.getCoordinates());
    alert('The furthest object has the coordinates ' +
result.get(result.getLength() - 1).geometry.getCoordinates());
});
```

then

```
{GeoQueryResult} then([onFulfill[, onReject, context)
```

Subscription to the promise.

Returns self-reference.

Parameters:

Parameter	Default value	Description
onFulfill	—	Type: Function Handler function that is called if the promise was fulfilled.
onReject	—	Type: Function Handler function that is called if the promise was not fulfilled (an error occurred).
context*	—	Type: Object Context for the handler function.

* Mandatory parameter/option.

Example:

```
var result = ymaps.geoQuery(ymaps.geocode('Lena river'));
result.then(function () {
    alert('Number of objects found: ' + result.getLength());
}, function () {
    alert('Error occurred.');
});
```

unsetOptions

```
{GeoQueryResult} unsetOptions(keys)
```

Method for nullifying option values for all the selection items.

Returns self-reference.

Parameters:

Parameter	Default value	Description
keys*	—	Type: String|String Name or array of names of options that should be canceled.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

Example:

```
result.unsetOptions('visible');
```

unsetProperties

```
{GeoQueryResult} unsetProperties(path)
```

Method for nullifying the value of the "properties" field for all the selection items.

Returns self-reference.

Parameters:

Parameter	Default value	Description
path*	—	Type: String Name of a field to cancel the value for. It may contain ".".

* Mandatory parameter/option.

Example:

```
var result = ymaps.geoQuery(objects);
// Marking items that fall inside the first area, but do not fall inside the
second one.
result.searchIntersect(myBounds1).setProperties('intersectBounds', true);
result.searchIntersect(myBounds2).unsetProperties('intersectBounds', true);
// ...
result.search('properties.intersectBounds = true').addToMap(myMap);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

geoXml

geoXml.load

Static function.

Loads an XML file with geographic data and converts it into a **GeoObjectCollection**. The generated collection can be passed to the specified function for subsequent processing. Supported XML data formats: YMapsML, KML, GPX. For the topmost collection of geo objects from a GPX file, the following presets are available:

- 'gpx#interactive' — Provides outputting information about a point on a route when clicked. Additionally, when this preset is used, the following geo object properties become available in the balloon layout: time, velocity, trackName, trackDescription, pointName, pointDescription, lon, lat, sym. Used by default.
- 'gpx#plain' — Items in GPX collections behave like normal geo objects.

Returns Promise object. If the XML file at the specified URL is downloaded successfully, the promise will be resolved and will get an object with the following fields (as parameters):

- geoObjects — Collection of geo objects **GeoObjectCollection**.
- mapState — Description of the map state **IMapState** (only for YMapsML).

.

```
{ vow.Promise } geoXml.load(url)
```

Parameters:

Parameter	Default value	Description
url*	—	Type: String The URL of a file with geographical data.

* Mandatory parameter/option.

Example:

```
// Creating and initializing map...

// Loading and displaying a ymapsml file from the My Maps service
ymaps.geoXml.load('http://maps.yandex.ru/export/usermaps/
HNQ5uTUgbjy6L0dW2uReUjSoXb1Ad7jw/')
     .then(function (res) {
         // Adding elements of the ymapsml file to the map
         map.geoObjects.add(res.geoObjects);
         // Setting the map boundaries and type.
         // res.mapState.getBounds() boundaries are applied to the map
asynchronously,
         // because we need to get information about available zoom levels for
these bounds
         // res.mapState.getType() is applied to the map synchronously.
         if (res.mapState) {
             res.mapState.applyToMap(map).then(function () {
                 alert('Boundaries applied to the map ' +
res.mapState.getBounds().toString());
             });
         }
         // If information about boundaries isn't provided in repr:View in the
YMapsML file,
         // we can apply gml:boundedBy for the top ymaps:GeoObjectCollection
element
         else if (res.geoObjects.properties.get('boundedBy')) {
             map.setBounds(res.geoObjects.properties.get('boundedBy'), {
                 checkZoomRange: true
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
            });
        }
    });

// Loading and displaying a KML file
ymaps.geoXml.load('http://api.yandex.ru/maps/doc/jsapi/1.x/examples/kml/
demonstration.xml')
     .then(function (res) {
         map.geoObjects.add(res.geoObjects);
    });

// Loading and displaying a GPX file
ymaps.geoXml.load('http://karmatsky.narod2.ru/MskChel2.xml')
     .then(function (res) {
         res.geoObjects.options.set({
             balloonContentBodyLayout: ymaps.templateLayoutFactory.createClass(
             // The balloon will only show the geo object's name property and the
speed
             '&lt;b&gt;{{ properties.name }}&lt;/b&gt;
{{ properties.velocity }}'
             )
        });

         map.geoObjects.add(res.geoObjects);
         // Metadata about boundaries from a GPX file is stored in properties of
the res.geoObjects collection.
         // Applying these boundaries to the map.
         if (res.geoObjects.properties.get('boundedBy')) {
             map.setBounds(res.geoObjects.properties.get('boundedBy'), {
                 checkZoomRange: true
             });
         }
     });
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getZoomRange

Static function.

Checks the available range of zoom levels at the specified point for the specified map type.

Returns a Promise, which will be resolved and will get an array of two numbers as a parameter — the maximum and minimum zoom at the given point.

```
{ vow.Promise } getZoomRange(mapType, coords, customizable)
```

Parameters:

Parameter	Default value	Description
mapType*	—	Type: String|**MapType** Map type. Key string from **mapType.storage**, or an instance of the **MapType** class.
coords*	—	Type: Number[] Coordinates of the point to find out the available range of zoom levels for.
customizable*	—	Type: **ICustomizable**=null Object that contains the options manager. The object options will be considered when getting the result.

* Mandatory parameter/option.

Examples:

1.

```
// Let's say we want to initialize the map at the maximum zoom.
var myMap;
ymaps.getZoomRange('yandex#map', [55.750516, 37.615924]).then(function (result)
{
     myMap = new ymaps.Map('mapContainer', {
         center: [55.750516, 37.615924],
         zoom: result[1]
     });
});
```

2.

```
// Initialize the map using the geocoder, centered at Lev Tolstoy street, number
16
// at the maximum zoom possible.
var myMap;
ymaps.geocode("Moscow, Lev Tolstoy, 16").then(function (geoData) {
    var coords = geoData.geoObjects.get(0).geometry.getCoordinates();
    ymaps.getZoomRange('yandex#map', coords).then(function (zoomRange) {
        myMap = new ymaps.Map('mapContainer', {
           center: coords,
           zoom: zoomRange[1]
        });
    });
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

graphics

graphics.style

graphics.style.color

Static object.

Sets the color of a graphic shape in the formats #RGB,#RGBA,#RRGGBB,#RRGGBBAA,rgb(r,b,a),rgba(r,g,b,a)

Example:

```
strokeColor: '#F00';
strokeColor: '#FF0000';
strokeColor: '#FF0000AA';
strokeColor: 'rgba(255,0,0,1)'.
```

graphics.style.stroke

Static object.

For changing line style. The value can be set by using the keys described below, or in array format. It is worth noting that in VML(IE<9) display mode, only keys may be used. Additionally, in the line style, the offset for beginning a dotted line can be set via the "offset" field.

Fields

Example:

```
strokeStyle: 'dot';
strokeStyle: [1,2];
strokeStyle: {
  style: 'dot',
  offset: 10
}
```

Fields

Name	Type	Description
dash		Dash
dashdot		Long dash — short dash
dot		Dots
longdash		Long dashes
longdashdot		Extra long dash — dot
longdashdotdot		Long dash — dot — dot
shortdash		Short dashes
shortdashdot		Dash — dot
shortdashdotdot		Dash — dot — dot
shortdot		Dots with double spacing

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
solid		Solid line

Fields details

dash

```
dash
```

Dash

dashdot

```
dashdot
```

Long dash — short dash

dot

```
dot
```

Dots

longdash

```
longdash
```

Long dashes

longdashdot

```
longdashdot
```

Extra long dash — dot

longdashdotdot

```
longdashdotdot
```

Long dash — dot — dot

shortdash

```
shortdash
```

Short dashes

shortdashdot

```
shortdashdot
```

Dash — dot

shortdashdotdot

```
shortdashdotdot
```

Dash — dot — dot

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

shortdot

```
shortdot
```

Dots with double spacing

solid

```
solid
```

Solid line

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Hint

Extends **IHint**, **Popup**.

A hint is a popup text that can display any HTML content. There is usually just one hint instance on the map and it is managed via special managers (maps, geo objects, hotspot layers and so on). Don't create them yourself, unless truly necessary.

Constructor | Fields | Events | Methods

Constructor

```
Hint(map[, options])
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Reference to a map object.
options	—	Type: Object Options.
options.closeTimeout	700	Type: Number Delay before closing (in ms).
options.contentLayout	—	Type: Function|String Layout for hint content. (Type: constructor for an object with the ILayout interface).
options.fitPane	true	Type: Boolean Flag that forces the info object to move its position so that it doesn't extend beyond the container boundaries.
options.holdByMouse	true	Type: Boolean Flag that cancels closing a hint that is located under the cursor.
options.interactivityModel	—	Type: String Key for the interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.layout	islands#hint	Type: Function|String External layout for the hint. (Type: constructor for an object with the ILayout interface).
options.offset	—	Type: Number[] Additional position offset relative to the anchor point.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.openTimeout	150	Type: Number Delay before opening (in ms).
options.pane	'outerHint'	Type: String Key of the pane that the hint overlay is placed in.
options.zIndex	—	Type: String The z-index of the hint.

* Mandatory parameter/option.

Example:

```
// Creating an independent hint instance and showing it with a custom layout in
the center of the map by setting the geo coordinates.
(new ymaps.Hint(myMap, {
    projection: ymaps.projection.wgs84Mercator,
    layout: ymaps.templateLayoutFactory.createClass('{{ content }}')
})).open(myMap.getCenter(), 'Hello');
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
close	Closing the info object. Inherited from IPopup.
open	Opening the info object. Inherited from IPopup.
optionschange	Change to the object options. Inherited from ICustomizable.

Methods

Name	Returns	Description
close([force])	vow.Promise	Closes the info object. Inherited from IPopup.
getData()		Returns info object data. Inherited from IPopup.
getOverlay()	vow.Promise	Returns the promise object to return the overlay.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IPopup**.
getOverlaySync()	**IOverlay**	Returns the overlay, if one exists. Inherited from **IPopup**.
getPosition()		Returns the coordinates of the info object. Inherited from **IPopup**.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from **IPopup**.
open([position[, data]])	**vow.Promise**	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the **IProjection** that is in the options. Inherited from **IPopup**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopup**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopup**.

Hotspot

Extends **IHotspot**.

Hotspot.

Constructor | Fields | Events | Methods

Constructor

```
Hotspot(shape[, zIndex])
```

Hotspot.

Parameters:

Parameter	Default value	Description
`shape` *	—	Type: **IShape** The hotspot shape.
`zIndex`	0	Type: Number zIndex of the hotspot.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getShape()	IShape	Returns the hotspot shape. Inherited from IHotspot.
getZIndex()	Number	Returns zIndex of the hotspot. Inherited from IHotspot.
setShape(shape)		Sets the hotspot shape. Inherited from IHotspot.
setZIndex(zIndex)		Sets the z-index of the hotspot. Inherited from IHotspot.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

hotspot

hotspot.layer

hotspot.layer.addon

hotspot.layer.addon.balloon

Note:
The constructor of the hotspot.layer.addon.balloon class is hidden, as this class is not intended for autonomous initialization.

Static object.

Methods

Methods

Name	Returns	Description
get(layer)	**IPopupManager**	Returns manager of the balloon of a hotspot layer.

Methods details

get

```
{IPopupManager} get(layer)
```

Returns manager of the balloon of a hotspot layer.

Parameters:

Parameter	Default value	Description
layer*	—	Type: **hotspot.Layer** Hotspot layer.

* Mandatory parameter/option.

Example:

```
ymaps.hotspot.layer.addon.balloon.get(layer)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

hotspot.layer.addon.hint

Note:
The constructor of the hotspot.layer.addon.hint class is hidden, as this class is not intended for autonomous initialization.

Static object.

Methods

Methods

Name	Returns	Description
get(layer)	IPopupManager	Returns manager of the hint on a hotspot layer.

Methods details

get

```
{IPopupManager} get(layer)
```

Returns manager of the hint on a hotspot layer.

Parameters:

Parameter	Default value	Description
layer*	—	Type: **hotspot.Layer** Hotspot layer.

* Mandatory parameter/option.

Example:

```
ymaps.hotspot.layer.addon.hint.get(layer)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

hotspot.layer.Balloon

Extends **IBalloonManager**.

Manager of the balloon of a hotspot layer. Allows a geo object to manage the hotspot layer by opening it and hiding it. It uses the map balloon manager **map.Balloon**. Hotspot layers contain an instance of this class available as myHotspotLayer.balloon. Don't create new instances of this class unless necessary.

See Balloon hotspot.Layer.balloon

Constructor | Fields | Events | Methods

Constructor

```
hotspot.layer.Balloon(hotspotLayer)
```

Parameters:

Parameter	Default value	Description
hotspotLayer *	—	Type: Object Hotspot layer.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
autopanbegin	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
autopanend	End of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
beforeuserclose	The event which precedes **Balloon.event:userclose**. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
close	Closing the info object. Names of fields available via **Event.get**: • target — Reference to the object where the closing occurred. Inherited from **IPopupManager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
open	Opening the info object. Names of fields available via Event.get: • target — Reference to the object where the opening occurred. Inherited from IPopupManager.
userclose	Balloon closed by the user. Instance of the Event class. Names of fields that are available via the Event.get method: • target — Reference to the IBalloonOwner object. Inherited from IBalloonManager.

Methods

Name	Returns	Description
autoPan()	vow.Promise	Moves the map so that the balloon is visible. Inherited from IBalloonManager.
close([force])	vow.Promise	Closes the info object. Inherited from IPopupManager.
destroy()		Disables the info object manager. Inherited from IPopupManager.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from IPopupManager.
getOptions()	IOptionManager|null	Returns the options manager or 'null'. Inherited from IPopupManager.
getOverlay()	vow.Promise	Returns the promise object to return the overlay. Inherited from IPopupManager.
getOverlaySync()	IOverlay|null	Returns the overlay, if one exists. Inherited from IPopupManager.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from IPopupManager.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from IPopupManager.
open([position[, data[, options]]])	vow.Promise	Opens the balloon at the specified position.
setData(data)	vow.Promise	Defines new data for the info object. Inherited from IPopupManager.
setOptions(options)	vow.Promise	Defines new options for the info object. Inherited from IPopupManager.
setPosition(position)	vow.Promise	Specifies a new position for the info object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IPopupManager**.

Methods details

open

```
{vow.Promise} open([position[, data[, options]]])
```

Opens the balloon at the specified position.

Returns Promise object.

Parameters:

Parameter	Default value	Description
position	—	Type: Number[] The coordinates of the balloon opening in the global pixel coordinates.
data	—	Type: Object Data.
options	—	Type: Object Options.

hotspot.layer.Hint

Extends **IHintManager**.

Manager of the hint on a hotspot layer. Allows to manage the hint on a hotspot layer by opening it and hiding it. It uses the map hint manager **map.Hint** inside itself. Hotspot layers contain an instance of this class available as myHotspotLayer.hint. Don't create new instances of this class unless necessary.

See Hint hotspot.Layer.hint

Constructor | Fields | Events | Methods

Constructor

```
hotspot.layer.Hint(hotspotLayer)
```

Parameters:

Parameter	Default value	Description
hotspotLayer *	—	Type: Object Hotspot layer.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
		Inherited from **IEventEmitter**.

Events

Name	Description
close	Closing the info object. Names of fields available via **Event.get**: • target — Reference to the object where the closing occurred. Inherited from **IPopupManager**.
open	Opening the info object. Names of fields available via **Event.get**: • target — Reference to the object where the opening occurred. Inherited from **IPopupManager**.

Methods

Name	Returns	Description
close([force])	**vow.Promise**	Closes the info object. Inherited from **IPopupManager**.
destroy()		Disables the info object manager. Inherited from **IPopupManager**.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from **IPopupManager**.
getOptions()	**IOptionManager**|null	Returns the options manager or 'null'. Inherited from **IPopupManager**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopupManager**.
getOverlaySync()	**IOverlay**|null	Returns the overlay, if one exists. Inherited from **IPopupManager**.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from **IPopupManager**.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from **IPopupManager**.
open([position[, data[, options]]])	**vow.Promise**	Opens a hint in the specified position.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopupManager**.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IPopupManager**.

Methods details

open

```
{vow.Promise} open([position[, data[, options]]])
```

Opens a hint in the specified position.

Returns Promise object.

Parameters:

Parameter	Default value	Description
position	—	Type: Number[] The coordinates of the balloon opening in the global pixel coordinates.
data	—	Type: Object Data.
options	—	Type: Object Options.

hotspot.layer.Object

Extends **IHotspotLayerObject**.

Object of the hotspot layer.

Constructor | Fields | Events | Methods

Constructor

```
hotspot.layer.Object(shape, feature, options)
```

Creates an object of the hotspot layer.

Parameters:

Parameter	Default value	Description
shape *	—	Type: **IShape** The hotspot shape.
feature *	—	Type: Object The description of the **GeoObject** object.
options *	—	Type: Object Object options.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
shapechange	Change to the shape that defines a hotspot. Instance of the Event class. Inherited from IHotspotLayerObject.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getGeometry()	Object	Returns the actual geometry of an object. Inherited from IHotspotLayerObject.
getHotspot()	IHotspot	Returns object describing a hotspot. Inherited from IHotspotLayerObject.
getId()	Number	Returns object ID. Inherited from IHotspotLayerObject.
getProperties()	Object	Returns object data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IHotspotLayerObject**.
setGeometry(geometry)		Defines the actual geometry of the object. Inherited from **IHotspotLayerObject**.
setId(id)		Sets the object ID. Inherited from **IHotspotLayerObject**.
setProperties(properties)		Sets the object's data. Inherited from **IHotspotLayerObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

hotspot.Layer

Extends **IChildOnMap**, **ICustomizable**.

Hotspot layer.

Constructor | Fields | Events | Methods

Constructor

```
hotspot.Layer(objectSource[, options])
```

Creates the hotspot layer. Each individual area on the layer consists of an object — **hotspot.layer.Object**.

Parameters:

Parameter	Default value	Description
objectSource *	—	Type: **IHotspotObjectSource** Source of objects for the layer.
options	—	Type: Object Layer options. Options for balloons **Balloon** and hints **Hint** on the hotspot layer must be indicated with the "balloon" and "hint" prefixes.
options.cursor	'pointer'	Type: String Type of cursor.
options.dontChangeCursor	false	Type: Boolean Option that prevents changing the cursor when pointing at an object on the layer. By default, cursors change.
options.hasBalloon	true	Type: Boolean Flag showing whether the layer has balloons. If the flag value is "false", the ".balloon" field will not be created for the layer.
options.hasHint	true	Type: Boolean Flag showing whether the layer has hints. If the flag value is "false", the ".hint" field will not be created for the layer.
options.interactivityModel	'default#layer'	Type: String Interactivity model for the layer. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.openBalloonOnClick	true	Type: Boolean Option that prevents opening the balloon when clicking on a hotspot object. By default, opening the balloon is allowed.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.openEmptyBalloon	false	Type: String Open the balloon with empty contents.
options.openEmptyHint	false	Type: String Show a popup hint with empty contents.
options.openHintOnHover	true	Type: Boolean Option that prevents showing the hint when pointing at a hotspot object. By default, showing hints is allowed.
options.pane	'events'	Type: **IEventPane** Container with map elements that events are listened on.
options.zIndex	—	Type: Number The zIndex value of the layer. When searching for the active object on a single layer and the cursor falls on two objects simultaneously, the object with the larger zIndex value is selected. From that point on — for example, when determining which layer's shape takes priority — the zIndex value of the layer is used. Please note that the zIndex of the layer will determine the relative position of the objects on the layer, as well as single objects added to particular map panes. For example, if you want to place hotspot objects over the area objects, you should set zIndex=201 for the hotspot layer. See **map.pane.Manager**.

* Mandatory parameter/option.

Example:

```
// Creating the source for hotspot data. We aren't setting the key value,
// so the name of the handler function (padding jsonp in the request) is generated
automatically.
var objectSource = new ymaps.hotspot.ObjectSource('tiles/?%c'),
    hotspotLayer = new ymaps.hotspot.Layer(objectSource, {
        zIndex: 100,
        // Allowing the balloon to be opened without content.
        showEmptyBalloon: true
    });
```

Fields

Name	Type	Description
balloon	**hotspot.layer.Balloon**	Hotspot layer balloon.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
hint	**hotspot.layer.Hint**	Hotspot layer hint.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
addtomap	Layer added to the map.
balloonclose	Closing the balloon. Instance of the Event class.
balloonopen	Opening the balloon on the hotspot layer. Instance of the Event class.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Instance of the Event class. Names of fields that are available via the Event.get method: • activeObject (hotspot.layer.Object) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Instance of the Event class. Names of fields that are available via the Event.get method: • activeObject (hotspot.layer.Object) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Instance of the Event class. Names of fields that are available via the Event.get method: • activeObject (hotspot.layer.Object) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.
hintclose	Closing the hint. Instance of the Event class.
hintopen	Opening the hint on the hotspot layer. Instance of the Event class.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Instance of the Event class. Names of fields that are available via the Event.get method: • activeObject (hotspot.layer.Object) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Instance of the Event class. Names of fields that are available via the Event.get method: • activeObject (hotspot.layer.Object) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Instance of the Event class. Names of fields that are available via the Event.get method: • activeObject (hotspot.layer.Object) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. Instance of the Event class. Names of fields that are available via the Event.get method: • activeObject (hotspot.layer.Object) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Instance of the Event class. Names of fields that are available via the Event.get method: • activeObject (hotspot.layer.Object) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method: • activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method: • activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent.get method: • activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IChild**.
removefrommap	Layer removed from the map.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method: • activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map.
getObjectInPosition(coords)	**vow.Promise**	Allows to get the layer object in the specifed geo point at the current zoom level. The method call automatically downloads the tile which covers the point, if it is not yet downloaded. After that, the downloaded data will be used for searching.
getObjectsInPosition(coords)	**vow.Promise**	Allows to get the layer object in the specifed geo point at the current zoom level. The method call automatically downloads the tile which covers the point, if it is not yet downloaded. After that, the downloaded data will be used for searching.
getObjectSource()	**IHotspotObjectSource**	Returns source of objects for the hotspot layer.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.
update()		Updates the hotspot layer. After this command has been executed, previously loaded objects are removed from the container and new data is requested.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

balloon

```
{hotspot.layer.Balloon} balloon
```

Hotspot layer balloon.

hint

```
{hotspot.layer.Hint} hint
```

Hotspot layer hint.

Events details

addtomap

Layer added to the map.

balloonclose

Closing the balloon. Instance of the **Event** class.

balloonopen

Opening the balloon on the hotspot layer. Instance of the **Event** class.

click

Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

contextmenu

Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

dblclick

Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

hintclose

Closing the hint. Instance of the **Event** class.

hintopen

Opening the hint on the hotspot layer. Instance of the **Event** class.

mousedown

Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

mouseenter

Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

mouseleave

Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

mousemove

Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

mouseup

Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

multitouchend

End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multitouchmove

Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

multitouchstart

Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent.get method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

removefrommap

Layer removed from the map.

wheel

Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- activeObject (**hotspot.layer.Object**) — The object of the layer where the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Methods details

getMap

```
{Map} getMap()
```

Returns reference to the map.

getObjectInPosition

```
{vow.Promise} getObjectInPosition(coords)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Allows to get the layer object in the specifed geo point at the current zoom level. The method call automatically downloads the tile which covers the point, if it is not yet downloaded. After that, the downloaded data will be used for searching.

Returns the Promise object that will be resolved by the layer object covering the point. If the point falls on several layer objects, the object with the maximum zIndex will be returned.

Parameters:

Parameter	Default value	Description
coords*	—	Type: Number[] The geo coordinates of the point.

* Mandatory parameter/option.

getObjectsInPosition

```
{vow.Promise} getObjectsInPosition(coords)
```

Allows to get the layer object in the specifed geo point at the current zoom level. The method call automatically downloads the tile which covers the point, if it is not yet downloaded. After that, the downloaded data will be used for searching.

Returns the Promise object that will be resolved by an array of layer objects covering the point.

Parameters:

Parameter	Default value	Description
coords*	—	Type: Number[] The geo coordinates of the point.

* Mandatory parameter/option.

getObjectSource

```
{IHotspotObjectSource} getObjectSource()
```

Returns source of objects for the hotspot layer.

Example:

```
// Updating the data source for the hotspot layer.
hotspotLayer.getObjectSource().setTileUrlTemplate('newSource/?%c');
hotspotLayer.update();
```

update

```
{} update()
```

Updates the hotspot layer. After this command has been executed, previously loaded objects are removed from the container and new data is requested.

hotspot.ObjectSource

Extends **IHotspotObjectSource**.

The standard implementation of the **IHotspotObjectSource** interface. Works with the standard format of the server response.

Constructor | Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
hotspot.ObjectSource(tileUrlTemplate[, keyTemplate[, options]])
```

Returns the data source for the hotspot layer.

Parameters:

Parameter	Default value	Description
tileUrlTemplate *	—	Type: String|Function Template for the URL of the data for the tile. Supports special constructions and the use of functions similar to **Layer**.
keyTemplate	—	Type: String|Function Template for the tile ID. Used for creating padding in the jsonp request for data. Set in the same way as the previous parameter. All characters other than letters, numbers, and the "_" symbol will be replaced with "_". If this parameter is omitted, the request's padding will be generated automatically. Conversion examples for tileNumber=[3, 1], zoom=9: • 'myCallback=%x' => 'myCallback_3' • '%c' => 'x_3_y_1_z_9' • 'callback2_%c' => 'callback2_x_3_y_1_z_9' • 'callback%test' => 'callback_test'
options	—	Type: Object Options for the data source.
options.bounds	null	Type: Bounds The area on the map that has data, or null if there is data everywhere. Option for the standard implementation of the "restrict" method.
options.maxZoom	0	Type: Bounds The maximum zoom level to make data requests for. Option for the standard implementation of the "restrict" method.
options.minZoom	0	Type: Bounds The minimum zoom level to make data requests for. Option for the standard implementation of the "restrict" method.
options.noCache	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		false — Use the built-in browser cache; true — don't use it (each URL will have a random GET parameter appended automatically in order to prevent the browser from caching tiles).

* Mandatory parameter/option.

Example:

```
// Example of the standard format of the server response
hotspot_callback({
    // the response can contain the "data" or "error" field.
    "data": {
        "type": "FeatureCollection",
        // Array of hotspot objects.
        "features": [{
            "type": "Feature",
            // Object data.
            // Passes the constructor hotspot.layer.Object as data
            "properties": {
                "hintContent": "Content of the text hint.",
                "balloonContentBody": "Balloon content.",
                "balloonContentHeader": "Balloon title.",
                "balloonContentFooter": "Lower part of the balloon.",
               // You can set the balloonContent property instead of Body/Header/
Footer

                // Required field that describes hotspot.layer.Object.
                "HotspotMetaData": {
                    // ID of the hotspot object.
                   // hotspot.layer.Object.getId returns the value of this field
by default.
                    "id": 10469893,
                    "zIndex": 10,
                    // Data used for creating the geometry of
hotspot.layer.Object.
                    // Required field.
                  // The standard implementation allows you to pass the following
types of geometries:
                    //    "Rectangle" - a rectangle.
                    //      Creates the geometry.pixel.Rectangle geometry.
                    //    "Polygon" - a multi-contour polygon.
                    //      Creates the geometry.pixel.Polygon geometry.
                    //    "MultiPolygon" - a complex shape made up of multiple
multi-contour polygons.
                    //      Creates the geometry.pixel.MultiPolygon geometry.
                    //    "ConvexPolygon" - a multi-contour polygon. All the
contours should be convex.
                    //       It works faster than "Polygon".
                    //       Creates the geometry.pixel.Polygon geometry.
                  //    "MultiConvexPolygon" - a complex shape made up of multiple
multi-contour polygons.
                   //      All the contour should be convex. It works faster than
"MultiPolygon".
                    //      Creates the geometry.pixel.MultiPolygon geometry.
                    "RenderedGeometry": {
                        "type": "Polygon",
                        // Coordinates are passed in pixel coordinates that are
calculated from the upper-left corner of the tile.
                        "coordinates": [
                            // The first contour of the polygon.
                            [
                                [-315, 280], [32, 442], [141, 208], [-206, 46],
[-315, 280]
                            ],
                            // The second contour of the polygon.
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
                            [
                                [-186, 155], [-238, 265], [-152, 306], [-100,
196], [-186, 155]
                            ]
                        ]
                    }
                }
            },
            // The object's actual geometry.
            // Optional field.
            // Passed to hotspot.layer.Object and available via the
hotspot.layer.Object.getGeometry method.
            "geometry": {
                "type": "Polygon",
                "coordinates": [
                    // The first contour of the polygon.
                    [
                        [29.176096525, 40.904183940],
                        [29.177027467, 40.903854324],
                        [29.177319900, 40.904329679],
                        [29.176389040, 40.904659406],
                        [29.176096525, 40.904183940]
                    ],
                    // The second contour of the polygon.
                    [
                        [29.176442530, 40.904437683],
                        [29.176303055, 40.904213830],
                        [29.176533525, 40.904131168],
                        [29.176673032, 40.904355010],
                        [29.176442530, 40.904437683]
                    ]
                ]
            }
        },
        {
            "type": "Feature",
            "properties": {
                // description of the next hotspot object...
            }
        }]
    }
});
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
cancelLastRequest()		Cancels the last request for data. Inherited from **IHotspotObjectSource**.
getKey(tileNumber, zoom)	String	Returns the name of the callback function (padding) in the jsonp request if it is static, or null if a dynamic jsonp callback can be used. Templates support the same substitutions as in the template for the tile URL. All characters other than letters, numbers, and the "_" symbol will be replaced with "_".
getKeyTemplate()	String	Returns template for the tile ID.
getTileUrl(tileNumber, zoom)	String	Returns the URL of data for a specific tile.
getTileUrlTemplate()	String	Returns template for the URL for tile data.
parseResponse(layer, res, callback, tileNumber, zoom)		Parses the server response. Sends the callback an array of hotspot objects generated using the json description.
requestObjects(layer, tileNumber, zoom, callback)		Builds an array of **IHotspotLayerObject** objects corresponding to a particular layer, tile, and map zoom level, and passes it to the callback function. Inherited from **IHotspotObjectSource**.
restrict(layer, tileNumber, zoom)	Boolean	Method that is called before sending a request for tile data. If the method returns true, the request will not be sent to the server, and an empty array of objects will be returned as the response. The standard implementation of the method checks whether the "zoom" parameter is within the limits of [options.minZoom, options.maxZoom]. It also checks whether the center of the requested tile falls within the specified options.bounds. If options.bounds=null, this parameter is not checked.
setKeyTemplate(template)		Sets a new tile ID template.
setTileUrlTemplate(template)		Sets a new template for the tile data URL.

Methods details

getKey

```
{String} getKey(tileNumber, zoom)
```

Returns the name of the callback function (padding) in the jsonp request if it is static, or null if a dynamic jsonp callback can be used. Templates support the same substitutions as in the template for the tile URL. All characters other than letters, numbers, and the "_" symbol will be replaced with "_".

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns tile ID. Used when creating padding in the jsonp request for data.

Parameters:

Parameter	Default value	Description
tileNumber*	—	Type: Number[] Tile number (tile coordinates).
zoom*	—	Type: Integer Zoom level.

* Mandatory parameter/option.

getKeyTemplate

```
{String} getKeyTemplate()
```

Returns template for the tile ID.

getTileUrl

```
{String} getTileUrl(tileNumber, zoom)
```

Returns the URL of data for a specific tile.

Parameters:

Parameter	Default value	Description
tileNumber*	—	Type: Number[] Tile number (tile coordinates).
zoom*	—	Type: Integer Zoom level.

* Mandatory parameter/option.

Example:

```
var hotspotObjectSource = new ymaps.hotspot.ObjectSource('dataSource/?%c');
hotspotObjectSource.getTileUrl = function (tileNumber, zoom) {
    if (zoom &gt; 10) {
        // For large scales, use the provided data path.
        return ymaps.hotspot.ObjectSource.prototype.call(this, tileNumber,
zoom);
    } else {
        // For small scales, use a different path.
       return 'otherSource/getHotspots.xml?z=' + zoom + '&lt;codeph&gt;&amp;&lt;/
codeph&gt;x=' + tileNumber[0] + '&lt;codeph&gt;&amp;&lt;/codeph&gt;y=' +
tileNumber[1]';
    }
};
```

getTileUrlTemplate

```
{String} getTileUrlTemplate()
```

Returns template for the URL for tile data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

parseResponse

```
{} parseResponse(layer, res, callback, tileNumber, zoom)
```

Parses the server response. Sends the callback an array of hotspot objects generated using the json description.

Parameters:

Parameter	Default value	Description
layer*	—	Type: **hotspot.Layer** The layer that the objects belong to.
res*	—	Type: Object Server response.
callback*	—	Type: Function Handler function.
tileNumber*	—	Type: Number[] Number of the tile that the response is for.
zoom*	—	Type: Number The zoom level that the response is for; array of objects.

* Mandatory parameter/option.

restrict

```
{Boolean} restrict(layer, tileNumber, zoom)
```

Method that is called before sending a request for tile data. If the method returns true, the request will not be sent to the server, and an empty array of objects will be returned as the response. The standard implementation of the method checks whether the "zoom" parameter is within the limits of [options.minZoom, options.maxZoom]. It also checks whether the center of the requested tile falls within the specified options.bounds. If options.bounds=null, this parameter is not checked.

Returns true if the tile extends beyond the boundaries of the data area (there is no data for this tile), or false if it does not (there is data).

Parameters:

Parameter	Default value	Description
layer*	—	Type: **hotspot.Layer** Hotspot layer.
tileNumber*	—	Type: Number[] Tile number.
zoom*	—	Type: Integer Zoom level.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Example:

```
// Example of redefining the "restrict" method
// Let's assume there is only data for Murmansk and Novosibirsk.
var myMap = new ymaps.Map('map', {center: [32.5, 68.9] , zoom: 9}),
    geoBounds = [
         [[31.729958, 69.369182], [34.203324, 68.666473]], // Murmansk
         [[82.179084, 55.341085], [83.725642, 54.670738]]  // Novosibirsk
    ],
    projection = myMap.options.get('projection');

var myHotspotSource = new ymaps.hotspot.ObjectSource('http://www.myDomain.ru/
tiles/?%c', '%c');

myHotspotSource.restrict = function(layer, tileNumber, zoom) {

   // Calculating the pixel boundaries of the cities for this zoom level.
   var boundsFromPoints = ymaps.util.bounds.fromPoints,
       toGlobalPixels = projection.toGlobalPixels,
       pixelBounds = [
           boundsFromPoints(
               toGlobalPixels(geoBounds[0][0]], zoom),
               toGlobalPixels(geoBounds[0][1]], zoom)
           ),
           boundsFromPoints(
               toGlobalPixels(geoBounds[1][0]], zoom),
               toGlobalPixels(geoBounds[1][1]], zoom)
           )
      ],

       // Calculating the pixel boundaries of the tile
       leftTop = [tileNumber[0] * 256, tileNumber[1] * 256],
       tileBounds = [leftTop, [leftTop[0] + 256, leftTop[1] + 256]],
       intersects = ymaps.util.bounds.intersects;
   // If the tile's pixel boundaries intersect with the pixel boundaries of the
specified areas,
   // we have to send a request for data
   if (intersects(pixelBounds[0], tileBounds) || (intersects(pixelBounds[1],
tileBounds)) {
         return false;
  }

   // For all the other tiles, this source doesn't have any data.
   return true;
}
```

setKeyTemplate

```
{} setKeyTemplate(template)
```

Sets a new tile ID template.

Parameters:

Parameter	Default value	Description
template*	—	Type: String Template for the ID.

* Mandatory parameter/option.

setTileUrlTemplate

```
{} setTileUrlTemplate(template)
```

Sets a new template for the tile data URL.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
template*	—	Type: String URL template

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

interactivityModel

interactivityModel.storage

Static object.

Instance of **util.Storage**

Storage for the interactivity model. Interactivity models allow objects to handle DOM events in different ways. Available interactivity model keys:

- 'default#opaque' — The object generates all DOM events and does not throw them to the map. Map behaviors will not work when hovering over or clicking on objects with this interactivity model.
- 'default#geoObject' — The object generates all DOM events. The events "wheel", "mousedown", "dblclick", "contextmenu", "multitouchstart", "multitouchmove" and "multitouchend" are thrown to the map. If the "scrollZoom", "dblClickZoom" or "magnifier" behaviors are enabled on the map, they will work via objects with this interactivity model, in contrast to the objects with the "default#opaque" model.
- 'default#layer' — The object generates all DOM events. The events "wheel", "mousedown", "contextment", "multitouchstart", "multitouchmove", and "multitouchend" are thrown to the map. If the "scrollZoom", "drag", or "magnifier" behaviors are enabled on the map, they will work via objects with this interactivity model.
- 'default#transparent' — The object generates all DOM events, then throws them to the map.
- 'default#silent' — The object does not generate all DOM events but throws them to the map.

Methods

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Interfaces

IBalloon

Extends **IPopup**.

Interface for a balloon.

Constructor | Fields | Events | Methods

Constructor

```
IBalloon()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
close	Closing the info object. Inherited from **IPopup**.
open	Opening the info object. Inherited from **IPopup**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Methods

Name	Returns	Description
autoPan()	**vow.Promise**	Moves the map so that the balloon is visible.
close([force])	**vow.Promise**	Closes the info object. Inherited from **IPopup**.
getData()		Returns info object data. Inherited from **IPopup**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopup**.
getOverlaySync()	**IOverlay**	Returns the overlay, if one exists. Inherited from **IPopup**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getPosition()		Returns the coordinates of the info object. Inherited from IPopup.
isOpen()	Boolean	Returns the info object state: open/ closed. Inherited from IPopup.
open([position[, data]])	vow.Promise	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the IProjection that is in the options. Inherited from IPopup.
setData(data)	vow.Promise	Defines new data for the info object. Inherited from IPopup.
setPosition(position)	vow.Promise	Specifies a new position for the info object. Inherited from IPopup.

Methods details

autoPan

```
{vow.Promise} autoPan()
```

Moves the map so that the balloon is visible.

Returns Promise object.

IBalloonLayout

Extends ILayout.

Interface for the balloon layout.

Constructor | Fields | Events | Methods

Constructor

```
IBalloonLayout()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty layout indicator. Instance of the **Event** class. Inherited from **ILayout**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
parentelementchange	Change to the parent element. Instance of the Event class. Inherited from ILayout.
shapechange	Change to the shape of the area spanning the layout. Instance of the Event class. Inherited from ILayout.
userclose	The Close button is selected. The user closes the balloon. Implements the IEvent interface.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getData()	Object	Returns layout data object. Inherited from ILayout.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from ILayout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getShape()	IShape|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from ILayout.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from ILayout.
setData(data)		Sets layout data. Inherited from ILayout.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from ILayout.

Events details

userclose

The Close button is selected. The user closes the balloon. Implements the IEvent interface.

IBalloonManager

Extends IPopupManager.

Interface for the balloon manager.

Constructor | Fields | Events | Methods

Constructor

```
IBalloonManager()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
autopanbegin	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the Event class. Names of fields that are available via the Event.get method: • target — Reference to the IBalloonOwner object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
autopanend	End of automatic shifting of the map center initiated by the autoPan method. Instance of the Event class. Names of fields that are available via the Event.get method: • target — Reference to the IBalloonOwner object.
beforeuserclose	The event which precedes Balloon.event:userclose. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the Event.get method: • target — Reference to the IBalloonOwner object.
close	Closing the info object. Names of fields available via Event.get: • target — Reference to the object where the closing occurred. Inherited from IPopupManager.
open	Opening the info object. Names of fields available via Event.get: • target — Reference to the object where the opening occurred. Inherited from IPopupManager.
userclose	Balloon closed by the user. Instance of the Event class. Names of fields that are available via the Event.get method: • target — Reference to the IBalloonOwner object.

Methods

Name	Returns	Description
autoPan()	vow.Promise	Moves the map so that the balloon is visible.
close([force])	vow.Promise	Closes the info object. Inherited from IPopupManager.
destroy()		Disables the info object manager. Inherited from IPopupManager.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from IPopupManager.
getOptions()	IOptionManager|null	Returns the options manager or 'null'. Inherited from IPopupManager.
getOverlay()	vow.Promise	Returns the promise object to return the overlay. Inherited from IPopupManager.
getOverlaySync()	IOverlay|null	Returns the overlay, if one exists. Inherited from IPopupManager.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from IPopupManager.
isOpen()	Boolean	Returns the info object state: open/ closed.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IPopupManager**.
open([position[, data[, options]]])	**vow.Promise**	Opens the info object at the specified position. Inherited from **IPopupManager**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopupManager**.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

Events details

autopanbegin

Start of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- target — Reference to the **IBalloonOwner** object.

autopanend

End of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- target — Reference to the **IBalloonOwner** object.

beforeuserclose

The event which precedes **Balloon.event:userclose**. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- target — Reference to the **IBalloonOwner** object.

userclose

Balloon closed by the user. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- target — Reference to the **IBalloonOwner** object.

Methods details

autoPan

```
{vow.Promise} autoPan()
```

Moves the map so that the balloon is visible.

Returns Promise object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

IBalloonOwner

An object with a balloon, which can be accessed through the "balloon" property.

Constructor | Fields | Events

Constructor

```
IBalloonOwner()
```

Fields

Name	Type	Description
balloon	IBalloonManager	Balloon for an object.

Events

Name	Description
balloonclose	Closing the balloon.
balloonopen	Opening the balloon.

Fields details

balloon

```
{IBalloonManager} balloon
```

Balloon for an object.

Events details

balloonclose

Closing the balloon.

balloonopen

Opening the balloon.

IBaseCircleGeometry

Extends **IBaseGeometry**, **ICircleGeometryAccess**.

Interface for the "Circle" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IBaseCircleGeometry()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates of the center. • newCoordinates — New coordinates of the center. • oldRadius — Old radius. • newRadius — New radius. Inherited from ICircleGeometryAccess.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the circle. Inherited from ICircleGeometryAccess.
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from IBaseGeometry.
getClosest(anchorPosition)	Object	Searches for a point on the circle closest to the anchorPosition. Inherited from ICircleGeometryAccess.
getCoordinates()	Number[]|null	Returns coordinates of the center of the circle. Inherited from ICircleGeometryAccess.
getRadius()	Number	Returns radius of the circle. Inherited from ICircleGeometryAccess.
getType()	String	Returns the "Circle" string.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from IFreezable.
setCoordinates(coordinates)	ICircleGeometryAccess	Sets the coordinates of the center of the circle. Inherited from ICircleGeometryAccess.
setRadius(radius)	ICircleGeometryAccess	Sets the radius of the circle. Inherited from ICircleGeometryAccess.
unfreeze()	IFreezable	Switches the object to active mode. Inherited from IFreezable.

Methods details

getType

```
{String} getType()
```

Returns the "Circle" string.

IBaseGeometry

Extends **IEventEmitter**.

Interface of a basic geometry.

Constructor | Fields | Methods

Constructor

```
IBaseGeometry()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.
getType()	String	Returns ID of the geometry type.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

getBounds

```
{Number[][]|null} getBounds()
```

Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.

getType

```
{String} getType()
```

Returns ID of the geometry type.

IBaseLinearRingGeometry

Extends **IBaseGeometry**, **ILinearRingGeometryAccess**.

Interface for the "Closed contour" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IBaseLinearRingGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. • oldFillRule — Old fill rule. • newFillRule — New fill rule. Inherited from **ILinearRingGeometryAccess**.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the contour. Inherited from **ILinearRingGeometryAccess**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
get(index)	Number[]	Returns coordinates of the point with the specified index. Inherited from **ILinearRingGeometryAccess**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getChildGeometry(index)	**IPointGeometryAccess**	Creates and returns an **IPointGeometryAccess** object for the specified contour on the polyline. Inherited from **ILinearRingGeometryAccess**.
getClosest(anchorPosition)	Object	Searches for a point on the contour closest to the anchorPosition. Inherited from **ILinearRingGeometryAccess**.
getCoordinates()	Number[][]	Returns an array of geometry coordinates. Inherited from **ILinearRingGeometryAccess**.
getFillRule()	String	Returns ID of the fill rule. Inherited from **ILinearRingGeometryAccess**.
getLength()	Integer	Returns the number of points in the geometry. Inherited from **ILinearRingGeometryAccess**.
getType()	String	Returns the "LinearRing" string.
insert(index, coordinates)	**ILinearRingGeometryAccess**	Adds a new point with the specified index. Inherited from **ILinearRingGeometryAccess**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	Number[]	Removes the point with the specified index. Inherited from **ILinearRingGeometryAccess**.
set(index, coordinates)	**ILinearRingGeometryAccess**	Sets coordinates of the point with the specified index.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **ILinearRingGeometryAccess**.
setCoordinates(coordinates)	**ILinearRingGeometryAccess**	Sets an array of geometry coordinates. Inherited from **ILinearRingGeometryAccess**.
setFillRule(fillRule)	**ILinearRingGeometryAccess**	Sets the contour fill rule. Inherited from **ILinearRingGeometryAccess**.
splice(index, number)	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter. Inherited from **ILinearRingGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Methods details

getType

```
{String} getType()
```

Returns the "LinearRing" string.

IBaseLineStringGeometry

Extends **IBaseGeometry**, **ILineStringGeometryAccess**.

Interface for the "Polyline" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IBaseLineStringGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. Inherited from ILineStringGeometryAccess.

Methods

Name	Returns	Description
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(index)	Number[]	Returns coordinates of the point with the specified index. Inherited from ILineStringGeometryAccess.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from IBaseGeometry.
getChildGeometry(index)	IPointGeometryAccess	Creates and returns an IPointGeometryAccess object for the specified vertex on the polyline. Inherited from ILineStringGeometryAccess.
getClosest(anchorPosition)	Object	Searches for a point on the polyline closest to the anchorPosition. Inherited from ILineStringGeometryAccess.
getCoordinates()	Number[][]	Returns an array of geometry coordinates. Inherited from ILineStringGeometryAccess.
getLength()	Integer	Returns the number of points in the geometry. Inherited from ILineStringGeometryAccess.
getType()	String	Returns the "LineString" string.
insert(index, coordinates)	ILineStringGeometryAccess	Adds a new point with the specified index. Inherited from ILineStringGeometryAccess.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	Number[]	Removes the point with the specified index. Inherited from **ILineStringGeometryAccess**.
set(index, coordinates)	**ILineStringGeometryAccess**	Sets coordinates of the point with the specified index. Inherited from **ILineStringGeometryAccess**.
setCoordinates(coordinates)	**ILineStringGeometryAccess**	Sets an array of geometry coordinates. Inherited from **ILineStringGeometryAccess**.
splice(index, number)	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter. Inherited from **ILineStringGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Methods details

getType

```
{String} getType()
```

Returns the "LineString" string.

IBasePointGeometry

Extends **IBaseGeometry**, **IPointGeometryAccess**.

Interface for the "Point" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IBasePointGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. Inherited from IPointGeometryAccess.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from IBaseGeometry.
getCoordinates()	Number[]|null	Returns coordinates of a point. Inherited from IPointGeometryAccess.
getType()	String	Returns the "Point" string.
setCoordinates(coordinates)	IPointGeometryAccess	Sets coordinates of a point. Inherited from IPointGeometryAccess.

Methods details

getType

```
{String} getType()
```

Returns the "Point" string.

IBasePolygonGeometry

Extends IBaseGeometry, IPolygonGeometryAccess.

Interface for the "Polygon" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IBasePolygonGeometry()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. • oldFillRule — Old fill rule. • newFillRule — New fill rule. Inherited from IPolygonGeometryAccess.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the polygon. Inherited from IPolygonGeometryAccess.
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(index)	Number[][]	Returns coordinates of the contour with the specified index. Inherited from IPolygonGeometryAccess.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from IBaseGeometry.
getChildGeometry(index)	ILinearRingGeometryAccess	Creates and returns an ILinearRingGeometryAccess object for the specified contour. Inherited from IPolygonGeometryAccess.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the polygon contour. Inherited from IPolygonGeometryAccess.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getCoordinates()	Number[][][]	Returns an array of geometry coordinates. Inherited from **IPolygonGeometryAccess**.
getFillRule()	String	Returns ID of the fill rule. Inherited from **IPolygonGeometryAccess**.
getLength()	Integer	Returns the number of contours in the geometry. Inherited from **IPolygonGeometryAccess**.
getType()	String	Returns the "Polygon" string.
insert(index, path)	**IPolygonGeometryAccess**	Adds a new contour with the specified index. Inherited from **IPolygonGeometryAccess**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	**ILinearRingGeometryAccess**	Removes the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
set(index, path)	**IPolygonGeometryAccess**	Sets coordinates of the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
setCoordinates(coordinates)	**IPolygonGeometryAccess**	Sets an array of geometry coordinates. Inherited from **IPolygonGeometryAccess**.
setFillRule(fillRule)	**IPolygonGeometryAccess**	Sets the polygon's fill rule. Inherited from **IPolygonGeometryAccess**.
splice(index, number)	**ILinearRingGeometryAccess**[]	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter. Inherited from **IPolygonGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

getType

```
{String} getType()
```

Returns the "Polygon" string.

IBaseRectangleGeometry

Extends **IBaseGeometry**, **IRectangleGeometryAccess**.

Interface for the "Rectangle" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IBaseRectangleGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
change	Change to corner coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates of the corners. • newCoordinates — New coordinates of the corners. Inherited from **IRectangleGeometryAccess**.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the rectangle. Inherited from **IRectangleGeometryAccess**.
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the rectangle contour. Inherited from **IRectangleGeometryAccess**.
getCoordinates()	Number[][]	Returns coordinates of two opposite corners of the rectangle. Inherited from **IRectangleGeometryAccess**.
getType()	String	Returns the "Rectangle" string.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
setCoordinates(coordinates)	**IRectangleGeometryAccess**	Sets the coordinates of two opposite corners of the rectangle. Inherited from **IRectangleGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Methods details

getType

```
{String} getType()
```

Returns the "Rectangle" string.

IBehavior

Extends **IChildOnMap**, **ICustomizable**.

Map behavior. Adds a map reaction to certain user actions (such as dragging, right-click zooming, or multitouch). The default behavior is disabled and can be enabled using the "enable" method.

Constructor | Fields | Events | Methods

Constructor

```
IBehavior([options])
```

Parameters:

Parameter	Default value	Description
options	—	Type: Object Behavior options.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
disable	Disabling behaviors.
enable	Enabling behaviors.
optionschange	Change to the object options. Inherited from ICustomizable.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Methods

Name	Returns	Description
disable()		Disables the behavior.
enable()		Enables the behavior.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
isEnabled()	Boolean	Checks whether the behavior is enabled.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Events details

disable

Disabling behaviors.

enable

Enabling behaviors.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

disable

```
{} disable()
```

Disables the behavior.

enable

```
{} enable()
```

Enables the behavior.

isEnabled

```
{Boolean} isEnabled()
```

Checks whether the behavior is enabled.

Returns true if the behavior is enabled, otherwise false.

ICanvasTile

Interface for tiles that can be displayed in the canvas object's 2d context.

Constructor | Fields | Events | Methods

Constructor

```
ICanvasTile(url)
```

Parameters:

Parameter	Default value	Description
url *	—	Type: String Tile URL.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Tile event manager.

Events

Name	Description
ready	Tile ready event.

Methods

Name	Returns	Description
destroy()		Destroys the tile.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
isReady()	Boolean	Checks tile readiness.
renderAt(context, canvasSize, bounds[, animate])		Draws an image tile in the canvas object's 2d context.

Fields details

events

```
{IEventManager} events
```

Tile event manager.

Events details

ready

Tile ready event.

Methods details

destroy

```
{} destroy()
```

Destroys the tile.

isReady

```
{Boolean} isReady()
```

Checks tile readiness.

Returns true if the tile is ready, or false if it is not ready.

renderAt

```
{} renderAt(context, canvasSize, bounds[, animate])
```

Draws an image tile in the canvas object's 2d context.

Parameters:

Parameter	Default value	Description
context*	—	Type: Object 2D context of the canvas object.
canvasSize*	—	Type: Number[] Dimensions of the canvas HTML element.
bounds*	—	Type: Number[][] Area in client coordinates where the tile should be drawn.
animate	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		If true, rendering is animated; if false, it is not.

* Mandatory parameter/option.

IChild

Extends **IEventEmitter**.

Interface for an object that has a parent.

Constructor | Fields | Events | Methods

Constructor

```
IChild()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent.

Methods

Name	Returns	Description
getParent()	Object	Returns reference to the parent object.
setParent(parent)	**IChild**	Sets the parent object.

Events details

parentchange

The parent object reference changed.

Data fields:

- oldParent — Old parent.
- newParent — New parent.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

getParent

```
{Object} getParent()
```

Returns reference to the parent object.

setParent

```
{IChild} setParent(parent)
```

Sets the parent object.

Returns self-reference.

Parameters:

Parameter	Default value	Description
parent*	—	Type: Object Parent object.

* Mandatory parameter/option.

IChildOnMap

Extends **IChild**.

Interface for an object whose parent belongs to a particular map.

Constructor | Fields | Events | Methods

Constructor

```
IChildOnMap()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

Methods details

getParent

```
{IParentOnMap|null} getParent()
```

Returns link to the parent object, or null if the parent element was not set.

setParent

```
{IChildOnMap} setParent(parent)
```

Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

Returns self-reference.

Parameters:

Parameter	Default value	Description
parent*	—	Type: IParentOnMap|null Parent object.

* Mandatory parameter/option.

ICircleGeometry

Extends ICircleGeometryAccess, IGeometry.

Interface for the "Circle" geometry.

Constructor | Fields | Events | Methods

Constructor

```
ICircleGeometry()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates of the center. • newCoordinates — New coordinates of the center. • oldRadius — Old radius. • newRadius — New radius. Inherited from ICircleGeometryAccess.
mapchange	Map reference changed. Instance of the Event class. Names of fields that are available via the Event.get method: • oldMap — Old map. • newMap — New map. Inherited from IGeometry.
optionschange	Change to the object options. Inherited from ICustomizable.
pixelgeometrychange	The pixel geometry changed. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelGeometry — New IPixelGeometry pixel geometry. Inherited from IGeometry.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the circle. Inherited from ICircleGeometryAccess.
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from IGeometry.
getClosest(anchorPosition)	Object	Searches for a point on the circle closest to the anchorPosition. Inherited from ICircleGeometryAccess.
getCoordinates()	Number[]|null	Returns coordinates of the center of the circle. Inherited from ICircleGeometryAccess.
getMap()	Map|null	Returns the current map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getRadius()	Number	Returns radius of the circle. Inherited from **ICircleGeometryAccess**.
getType()	String	Returns the "Circle" string.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
setCoordinates(coordinates)	**ICircleGeometryAccess**	Sets the coordinates of the center of the circle. Inherited from **ICircleGeometryAccess**.
setMap(map)		Sets the map. Inherited from **IGeometry**.
setRadius(radius)	**ICircleGeometryAccess**	Sets the radius of the circle. Inherited from **ICircleGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Methods details

getType

```
{String} getType()
```

Returns the "Circle" string.

ICircleGeometryAccess

Extends **IFreezable**.

Interface for access to the "Circle" geometry.

Constructor | Fields | Events | Methods

Constructor

```
ICircleGeometryAccess()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager for the object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
		Inherited from **IFreezable**.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates of the center. • newCoordinates — New coordinates of the center. • oldRadius — Old radius. • newRadius — New radius.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the circle.
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
getClosest(anchorPosition)	Object	Searches for a point on the circle closest to the anchorPosition.
getCoordinates()	Number[]|null	Returns coordinates of the center of the circle.
getRadius()	Number	Returns radius of the circle.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
setCoordinates(coordinates)	**ICircleGeometryAccess**	Sets the coordinates of the center of the circle.
setRadius(radius)	**ICircleGeometryAccess**	Sets the radius of the circle.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Events details

change

Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldCoordinates — Old coordinates of the center.
- newCoordinates — New coordinates of the center.
- oldRadius — Old radius.
- newRadius — New radius.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the circle.

Returns an indicator for whether the point belongs to the circle.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

Example:

```
var myCircle = new ymaps.geometry.base.Circle([0, 0], 10);
myCircle.contains([0, 10]); // =&gt; true
```

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the circle closest to the anchorPosition.

Returns an object with the following fields:

- position — Point on the circle closest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".

.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest point on the circle is calculated.

* Mandatory parameter/option.

Example:

```
var myCircle = new ymaps.geometry.base.Circle([0, 0], 10);
myCircle.getClosest([0, 15]).position; // =&gt; [0, 10]
```

getCoordinates

```
{Number[]|null} getCoordinates()
```

Returns coordinates of the center of the circle.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getRadius

```
{Number} getRadius()
```

Returns radius of the circle.

setCoordinates

```
{ICircleGeometryAccess} setCoordinates(coordinates)
```

Sets the coordinates of the center of the circle.

Returns self-reference.

Parameters:

Parameter	Default value	Description
coordinates*	—	Type: Number[]|null Coordinates of the center of the circle.

* Mandatory parameter/option.

setRadius

```
{ICircleGeometryAccess} setRadius(radius)
```

Sets the radius of the circle.

Returns self-reference.

Parameters:

Parameter	Default value	Description
radius*	—	Type: Number Radius of the circle.

* Mandatory parameter/option.

ICollection

Extends **IEventEmitter**.

Interface for a collection.

Constructor | Fields | Events | Methods

Constructor

```
ICollection()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
add	A child object was added.
remove	A child object was deleted.

Methods

Name	Returns	Description
add(object)	ICollection	Adds a child object to the collection.
getIterator()	IIterator	Returns iterator for the collection.
remove(object)	ICollection	Removes a child object from the collection.

Events details

add

A child object was added.

remove

A child object was deleted.

Methods details

add

```
{ICollection} add(object)
```

Adds a child object to the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
object*	—	Type: Object Object being added.

* Mandatory parameter/option.

getIterator

```
{IIterator} getIterator()
```

Returns iterator for the collection.

remove

```
{ICollection} remove(object)
```

Removes a child object from the collection.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
object*	—	Type: Object Object being removed.

* Mandatory parameter/option.

IContainerPane

Extends **IPane**, **IPositioningContext**.

Interface for the map pane intended for representations of the objects placed on the surface of the map. Allows you to transform global map pixels into a custom local coordinate system, thus enabling the object to be positioned within itself. Also informs about changes to the map status and thus allows the object to handle these changes.

Constructor | Fields | Events | Methods

Constructor

```
IContainerPane()
```

Fields

Name	Type	Description
clientpixelschange		Change to the pane's local coordinate system. This event means that objects that calculate their positions inside the pane from the map's global pixel coordinates must recalculate it and update their positions inside the pane's DOM element. Instance of IEvent.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
actionbegin	The start of pane movement. Instance of **IEvent**.
actionend	The end of pane movement. Instance of **IEvent**.
clientpixelschange	Change to the coordinate system of the client pixels.
overflowchange	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of **IEvent**. Inherited from **IPane**.
viewportchange	Change to a pane's viewport. Instance of **IEvent**.
zindexchange	Change to the zIndex value of a pane. Instance of **IEvent**. Inherited from **IPane**.
zoomchange	Change to the current zoom level of the pane. Instance of **IEvent**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
destroy()		Destroys the pane. Inherited from **IPane**.
fromClientPixels(clientPixelPoint)	Number[]	Converts client pixel coordinates to global coordinates. Inherited from **IPositioningContext**.
getElement()	HTMLElement	Returns a reference to the pane's DOM container. Inherited from **IPane**.
getMap()	**Map**	Returns the map that the pane belongs to. Inherited from **IPane**.
getOverflow()	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: • "visible" — When you go off the map container, the content of the pane remains visible. • "hidden" — The viewport for the pane content is limited by the map container. . Inherited from **IPane**.
getViewport()	Number[][]	Returns the viewport for the pane, in client coordinates.
getZIndex()	Number	Returns the zIndex of the pane. Inherited from **IPane**.
getZoom()	Number	Returns the current zoom level at which the positioning context works. Inherited from **IPositioningContext**.
toClientPixels(globalPixelPoint)	Number[]	Converts global pixel coordinates to client coordinates. Inherited from **IPositioningContext**.

Fields details

clientpixelschange

```
clientpixelschange
```

Change to the pane's local coordinate system. This event means that objects that calculate their positions inside the pane from the map's global pixel coordinates must recalculate it and update their positions inside the pane's DOM element. Instance of IEvent.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

actionbegin

The start of pane movement. Instance of **IEvent**.

actionend

The end of pane movement. Instance of **IEvent**.

clientpixelschange

Change to the coordinate system of the client pixels.

viewportchange

Change to a pane's viewport. Instance of **IEvent**.

zoomchange

Change to the current zoom level of the pane. Instance of **IEvent**.

Methods details

getViewport

```
{Number[][]} getViewport()
```

Returns the viewport for the pane, in client coordinates.

IControl

Extends **IChildOnMap**.

Control.

Constructor | Fields | Events | Methods

Constructor

```
IControl([options])
```

Interface for a control.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Control options.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
options	IOptionManager	Options manager.

Events

Name	Description
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Methods

Name	Returns	Description
getParent()	IControlParent|null	Returns link to the parent object, or null if the parent element was not set.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

Fields details

options

```
{IOptionManager} options
```

Options manager.

Methods details

getParent

```
{IControlParent|null} getParent()
```

Returns link to the parent object, or null if the parent element was not set.

setParent

```
{IChildOnMap} setParent(parent)
```

Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

Returns self-reference.

Parameters:

Parameter	Default value	Description
parent*	—	Type: IControlParent|null Parent object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

IControlParent

Extends **IParentOnMap**.

Interface of the parent object for the control.

Constructor | Fields | Events | Methods

Constructor

```
IControlParent()
```

Fields

Name	Type	Description
state	**IDataManager**	State manager.

Events

Name	Description
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.

Methods

Name	Returns	Description
getChildElement(child)	**vow.Promise**	Returns the promise object, which is confirmed by the HTML element that should hold the child element.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.

Fields details

state

```
{IDataManager} state
```

State manager.

Methods details

getChildElement

```
{vow.Promise} getChildElement(child)
```

Returns the promise object, which is confirmed by the HTML element that should hold the child element.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
child*	—	Type: **IControl** Child object.

* Mandatory parameter/option.

ICoordSystem

Interface for the map's coordinate system. This interface must be implemented if non-standard coordinates are used (such as cylindrical coordinates). To solve tasks with searching for the trajectory of movement on the earth's surface, use the **coordSystem.geo** object; on a Cartesian surface, use **coordSystem.cartesian**.

See coordSystem.geo coordSystem.cartesian

Constructor | Methods

Constructor

```
ICoordSystem()
```

Methods

Name	Returns	Description
getDistance(point1, point2)	Number	Returns the shortest distance (along a geodetic line) between the two set points (in meters).
solveDirectProblem(startPoint, direction, distance)	Object	Solves the first (direct) geodesic problem: where we will end up, if we start from a specified point and move in the specified direction for the specified distance, without turning. The following data is a solution for the direct geodetic problem: • The end point. • The final direction. • The path function. • A function that allows to specify, for any given moment in time, which point we will be at and which direction we will be moving in.
solveInverseProblem(startPoint, endPoint[, reverseDirection])	Object	Solves the second (inverse) geodetic problem: construct the shortest route between two points on the mapped surface and determine the distance and direction of movement. Note that on the map of the Earth's surface, the shortest routes are shown as crooked lines. For geo objects in the API, you can enable the mode for displaying shortest distances between points using the "geodesic" option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

getDistance

```
{Number} getDistance(point1, point2)
```

Returns the shortest distance (along a geodetic line) between the two set points (in meters).

Parameters:

Parameter	Default value	Description
point1*	—	Type: Number[] The first point.
point2*	—	Type: Number[] The second point.

* Mandatory parameter/option.

Example:

```
// Calculating the distance between Moscow and New York
// Coordinates of Moscow
ymaps.geocode('Moscow').then(function (res) {
    var moscowCoords = res.geoObjects.get(0).geometry.getCoordinates();
    // Coordinates of New York
    ymaps.geocode('New York').then(function (res) {
        var newYorkCoords = res.geoObjects.get(0).geometry.getCoordinates();
        // Distance
        alert(ymaps.formatter.distance(
            ymaps.coordSystem.geo.getDistance(moscowCoords, newYorkCoords)
        ));
    });
});
```

solveDirectProblem

```
{Object} solveDirectProblem(startPoint, direction, distance)
```

Solves the first (direct) geodesic problem: where we will end up, if we start from a specified point and move in the specified direction for the specified distance, without turning. The following data is a solution for the direct geodetic problem:

- The end point.
- The final direction.
- The path function.
- A function that allows to specify, for any given moment in time, which point we will be at and which direction we will be moving in.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns object with following fields:

- startPoint — The starting point in geocoordinates.
- startDirection — The starting direction of movement.
- endPoint — The end point in geocoordinates.
- endDirection — The final direction of movement.
- distance — The distance in meters.
- pathFunction — A function that accepts a number from 0 to 1 (the portion of the path completed) and returns a structure with the fields "point" and "direction".

.

Parameters:

Parameter	Default value	Description
startPoint*	—	Type: Number[] Point of departure.
direction*	—	Type: Number[] Direction. Set as a vector (an increment of coordinates) — either [dlat, dlon] or [dlon, dlat], depending on the "coordorder" parameter. In order to get the azimuth from a direction specified like this (the azimuth is the angle between the direction of movement and North), we need to calculate the arctangent of the dlon/dlat amount (in JavaScript — this is a standard function Math.atan2(dlon, dlat)). In order to calculate the direction of movement from the azimuth "a", put dlat = cos(a), dlon = sin(a).
distance*	—	Type: Number The distance walked, in meters.

* Mandatory parameter/option.

Example:

```
// Let's assume that we took off from Domodedovo airport (Moscow) going
// northeast and flew straight for 200 kilometers. We'll use placemarks to show
our path
// on the map.

// Finding the coordinates of the starting point, using geocoding.
ymaps.geocode('Domodedovo airport').then(function (res) {
    var startPoint = res.geoObjects.get(0).geometry.getCoordinates(),
    // Moving northeast, azimuth of 45 degrees
    // or radian pi/4.
        azimuth = Math.PI / 4,
    // Direction of movement.
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
        direction = [Math.cos(azimuth), Math.sin(azimuth)],
    // Path function
        path = ymaps.coordSystem.geo
            .solveDirectProblem(startPoint, direction, 2e5).pathFunction;

    // Showing the path on the map using placemarks set every 10 km
    for (var i = 0; i &lt;= 20; i++) {
        map.geoObjects.add(new ymaps.Placemark(path(i/20).point));
    }
});
```

solveInverseProblem

```
{Object} solveInverseProblem(startPoint, endPoint[, reverseDirection])
```

Solves the second (inverse) geodetic problem: construct the shortest route between two points on the mapped surface and determine the distance and direction of movement. Note that on the map of the Earth's surface, the shortest routes are shown as crooked lines. For geo objects in the API, you can enable the mode for displaying shortest distances between points using the "geodesic" option.

Returns object with following fields:

- startPoint — The starting point in geocoordinates.
- startDirection — The starting direction of movement.
- endPoint — The end point in geocoordinates.
- endDirection — The final direction of movement.
- distance — The distance in meters.
- pathFunction — A path function that accepts a number from 0 to 1 (the portion of the path completed) and returns a structure with the fields "point" and "direction".

.

Parameters:

Parameter	Default value	Description
startPoint*	—	Type: Number[] Point of departure.
endPoint*	—	Type: Number[] Point of arrival.
reverseDirection	false	Type: Boolean Direction of movement. "false" — select the shortest arc; "true" — select the opposite of the shortest arc.

* Mandatory parameter/option.

Example:

```
// Constructing the shortest route from Kaliningrad to Vladivostok
// Finding the coordinates of Kaliningrad
ymaps.geocode('Kaliningrad').then(function (res) {
    var startPoint = res.geoObjects.get(0).geometry.getCoordinates();
    // Finding the coordinates of Vladivostok
    ymaps.geocode('Vladivostok').then(function (res) {
        var endPoint = res.geoObjects.get(0).geometry.getCoordinates(),
        // Finding the function of the path between two points
        path = ymaps.coordSystem.geo
            .solveInverseProblem(startPoint, endPoint).pathFunction;
        // Showing the path with 20 points
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
        for (var i = 0; i &lt;= 20; i++) {
            // Finding an intermediate point
            var position = path(i/20).point;
            // Adding a placemark to an intermediate point
            map.geoObjects.add(new ymaps.Placemark(
                position, {
                    // Showing the distance covered in the placemark content
                    iconContent: ymaps.formatter.distance(
                        ymaps.coordSystem.geo.getDistance(startPoint, position)
                    )
                }, {
                    preset: 'isladns#redStretchyIcon'
                }
            ));
        }
    });
});
```

ICopyrightsAccessor

Extends **ICopyrightsProvider**.

Interface for an object that provides access to user copyright information that was added to the map using the method map.Copyrights.add.

See map.Copyrights.add

Constructor | Fields | Events | Methods

Constructor

```
ICopyrightsAccessor()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
copyrightschange	Event for changes to copyright information. Inherited from **ICopyrightsProvider**.

Methods

Name	Returns	Description
getCopyrights(coords, zoom)	**vow.Promise**	Requests information about copyrights at the specified point with the specified zoom. Inherited from **ICopyrightsProvider**.
remove()		Removes copyright information that was added via this object. Inherited from **ICopyrightsProvider**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setCopyrights(copyrights)		Sets a new value for the copyright information that was added via this object. Inherited from **ICopyrightsProvider**.

ICopyrightsProvider

Extends **IEventEmitter**.

Copyright information provider.

Constructor | Fields | Events | Methods

Constructor

```
ICopyrightsProvider()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
copyrightschange	Event for changes to copyright information.

Methods

Name	Returns	Description
getCopyrights(coords, zoom)	**vow.Promise**	Requests information about copyrights at the specified point with the specified zoom.
remove()		Removes copyright information that was added via this object.
setCopyrights(copyrights)		Sets a new value for the copyright information that was added via this object.

Events details

copyrightschange

Event for changes to copyright information.

Methods details

getCopyrights

```
{vow.Promise} getCopyrights(coords, zoom)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Requests information about copyrights at the specified point with the specified zoom.

Returns Promise that will be resolved and will pass as a result an array of strings or DOM elements with information about copyrights.

Parameters:

Parameter	Default value	Description
coords*	—	Type: Number[] The point on the map that copyright information is being requested for.
zoom*	—	Type: Number The zoom level that copyright information is being requested for.

* Mandatory parameter/option.

remove

```
{} remove()
```

Removes copyright information that was added via this object.

setCopyrights

```
{} setCopyrights(copyrights)
```

Sets a new value for the copyright information that was added via this object.

Parameters:

Parameter	Default value	Description
copyrights*	—	Type: String|HTMLElement|String[]|HTMLElement[] Copyright information.

* Mandatory parameter/option.

ICustomizable

Extends **IEventEmitter**.

Interface of an object that can be configured using the global options register.

Constructor | Fields | Events

Constructor

```
ICustomizable()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
options	IOptionManager	Options manager.

Events

Name	Description
optionschange	Change to the object options.

Fields details

options

```
{IOptionManager} options
```

Options manager.

Events details

optionschange

Change to the object options.

IDataManager

Extends IEventEmitter.

Interface for the custom data manager.

Constructor | Fields | Events | Methods

Constructor

```
IDataManager()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
change	Data change.

Methods

Name	Returns	Description
get(path, defaultValue)	Object	Returns value of data fields.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

change

Data change.

Methods details

get

```
{Object} get(path, defaultValue)
```

Returns value of data fields.

Parameters:

Parameter	Default value	Description
path*	—	Type: String String with the property name; can contain '.'.
defaultValue*	—	Type: Object Default value.

* Mandatory parameter/option.

Example:

```
dataManager.get("icon.shadow.size", [10, 10]);
```

IDomEvent

Extends **IEvent**.

The DOM event object in the Yandex.Maps API system. Wraps the browser's source DOM event in order to normalize the data field names. This means you can use the "get" and "callMethod" methods to access the fields and methods of the source DOM event. In addition, standardization occurs automatically for those fields and methods that are implemented differently in different browsers. So event.callMethod('stopPropagation') stops propagation of the DOM event in all browsers, including Internet Explorer.

Constructor | Methods

Constructor

```
IDomEvent(originalEvent)
```

Parameters:

Parameter	Default value	Description
originalEvent *	—	Type: Object Source DOM event.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
allowMapEvent()		Allows the propagation of the event to the map. Inherited from IEvent.
callMethod(name)		Calls the specified method from the source event. The second and following arguments are passed to the method with the call. Inherited from IEvent.
get(name)	Object	Returns the specified event property. Using this method, you can get access both to the properties of the source event and to additional properties provided by the Yandex.Maps API.
getSourceEvent()	IDomEvent	Returns source DOM event.
isDefaultPrevented()	Boolean	Returns true if the default reaction to the event has been canceled. Inherited from IEvent.
isImmediatePropagationStopped()	Boolean	Returns true if the event processing has been interrupted. Inherited from IEvent.
isMapEventAllowed()	Boolean	Returns true if the map event is enabled. Inherited from IEvent.
isPropagationStopped()	Boolean	Returns true if event propagation has been interrupted. Inherited from IEvent.
preventDefault()		Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect how the browser processes the default action for the source DOM event.
stopImmediatePropagation()		Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called. Calling this method does not affect the processing of the original DOM-event at the browser level.
stopPropagation()		Stops propagation of the DOM event in the Yandex.Maps API event system. Calling this method does not affect propagation of the source DOM event through the DOM tree.

Methods details

get

```
{Object} get(name)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns the specified event property. Using this method, you can get access both to the properties of the source event and to additional properties provided by the Yandex.Maps API.

Returns property value.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Property name. Additional properties are supported: • 'propagatedData' — Event data that is saved during event propagation through the DOM tree. • 'position' — An optional field that contains the coordinates of the event, relative to the document.

* Mandatory parameter/option.

getSourceEvent

```
{IDomEvent} getSourceEvent()
```

Returns source DOM event.

preventDefault

```
{} preventDefault()
```

Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect how the browser processes the default action for the source DOM event.

stopImmediatePropagation

```
{} stopImmediatePropagation()
```

Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called. Calling this method does not affect the processing of the original DOM-event at the browser level.

stopPropagation

```
{} stopPropagation()
```

Stops propagation of the DOM event in the Yandex.Maps API event system. Calling this method does not affect propagation of the source DOM event through the DOM tree.

IDomEventEmitter

Extends **IEventEmitter**.

Interface of an object that generates a "DOM event". This interface declares a list of events that are used for displaying interactive user manipulation of various objects in the Yandex.Maps API environment. Objects implementing this interface are not required to actually generate all the events declared by the interface.

Constructor | Fields | Events

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IDomEventEmitter()
```

Fields

Name	Type	Description
events	IEventManager	Event manager.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

Fields details

events

```
{IEventManager} events
```

Event manager.

Events details

click

Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

contextmenu

Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

dblclick

Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

mousedown

Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

mouseenter

Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

mouseleave

Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

mousemove

Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

mouseup

Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

multitouchend

End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface.

multitouchmove

Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields:

- clientX — X coordinate of the touch relative to the viewable area of the browser.
- clientY — Y coordinate of the touch relative to the viewable area of the browser.
- pageX — X coordinate of the touch relative to the beginning of the document.
- pageY — Y coordinate of the touch relative to the beginning of the document.

multitouchstart

Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields:

- clientX — X coordinate of the touch relative to the viewable area of the browser.
- clientY — Y coordinate of the touch relative to the viewable area of the browser.
- pageX — X coordinate of the touch relative to the beginning of the document.
- pageY — Y coordinate of the touch relative to the beginning of the document.

wheel

Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

IDomTile

Interface for tiles that make up a DOM object.

Constructor | Fields | Events | Methods

Constructor

```
IDomTile(url)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
`url` *	—	Type: String Tile URL.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Tile event manager.

Events

Name	Description
ready	Tile ready event.

Methods

Name	Returns	Description
destroy()		Destroys the tile.
isReady()	Boolean	Checks tile readiness.
renderAt(context, clientBounds, animate)		Adds a tile to the parent HTML element.

Fields details

events

```
{IEventManager} events
```

Tile event manager.

Events details

ready

Tile ready event.

Methods details

destroy

```
{} destroy()
```

Destroys the tile.

isReady

```
{Boolean} isReady()
```

Checks tile readiness.

Returns true if the tile is ready, or false if it is not ready.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

renderAt

```
{} renderAt(context, clientBounds, animate)
```

Adds a tile to the parent HTML element.

Parameters:

Parameter	Default value	Description
context*	—	Type: HTMLElement Parent HTML element
clientBounds*	—	Type: Number[][] The area in client coordinates that the tile should occupy.
animate*	—	Type: Boolean If true, rendering is animated; if false, it is not.

* Mandatory parameter/option.

IEvent

Event that is fired by the IEventManager event manager.

Constructor | Methods

Constructor

```
IEvent()
```

Methods

Name	Returns	Description
allowMapEvent()		Allows the propagation of the event to the map.
callMethod(name)		Calls the specified method from the source event. The second and following arguments are passed to the method with the call.
get(name)	Object	Returns an event property for a key. Using this method, you can get access both to the properties of the source event and to additional properties provided by the Yandex.Maps API.
getSourceEvent()	**IEvent**|null	Returns source event.
isDefaultPrevented()	Boolean	Returns true if the default reaction to the event has been canceled.
isImmediatePropagationStopped()	Boolean	Returns true if the event processing has been interrupted.
isMapEventAllowed()	Boolean	Returns true if the map event is enabled.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
isPropagationStopped()	Boolean	Returns true if event propagation has been interrupted.
preventDefault()		Cancels the default reaction to an event within the Yandex.Maps API event system.
stopImmediatePropagation()		Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called.
stopPropagation()		Stops event propagation in the Yandex.Maps API event system.

Methods details

allowMapEvent

```
{} allowMapEvent()
```

Allows the propagation of the event to the map.

callMethod

```
{} callMethod(name)
```

Calls the specified method from the source event. The second and following arguments are passed to the method with the call.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Method name.

* Mandatory parameter/option.

get

```
{Object} get(name)
```

Returns an event property for a key. Using this method, you can get access both to the properties of the source event and to additional properties provided by the Yandex.Maps API.

Returns property value.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Property name.

* Mandatory parameter/option.

getSourceEvent

```
{IEvent|null} getSourceEvent()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns source event.

isDefaultPrevented

```
{Boolean} isDefaultPrevented()
```

Returns true if the default reaction to the event has been canceled.

isImmediatePropagationStopped

```
{Boolean} isImmediatePropagationStopped()
```

Returns true if the event processing has been interrupted.

isMapEventAllowed

```
{Boolean} isMapEventAllowed()
```

Returns true if the map event is enabled.

isPropagationStopped

```
{Boolean} isPropagationStopped()
```

Returns true if event propagation has been interrupted.

preventDefault

```
{} preventDefault()
```

Cancels the default reaction to an event within the Yandex.Maps API event system.

stopImmediatePropagation

```
{} stopImmediatePropagation()
```

Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called.

stopPropagation

```
{} stopPropagation()
```

Stops event propagation in the Yandex.Maps API event system.

IEventController

Interface for an event controller. For controlling how events are subscribed to and unsubscribed from on a particular event manager.

Constructor | Methods

Constructor

```
IEventController()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Description
onStartListening(events, type)	Called on the first subscription to the specified event type using the specified event manager. This method is optional.
onStopListening(events, type)	Called when a particular event type stops listening on the specified event manager (the last subscription is deleted). This method is optional.

Methods details

onStartListening

```
{} onStartListening(events, type)
```

Called on the first subscription to the specified event type using the specified event manager. This method is optional.

Parameters:

Parameter	Default value	Description
events*	—	Type: **IEventManager** Event manager.
type*	—	Type: String Type of event.

* Mandatory parameter/option.

onStopListening

```
{} onStopListening(events, type)
```

Called when a particular event type stops listening on the specified event manager (the last subscription is deleted). This method is optional.

Parameters:

Parameter	Default value	Description
events*	—	Type: **IEventManager** Event manager.
type*	—	Type: String Type of event.

* Mandatory parameter/option.

IEventEmitter

Interface of the object that events can be listened on.

Constructor | Fields

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IEventEmitter()
```

Fields

Name	Type	Description
events	IEventManager	Event manager.

Fields details

events

```
{IEventManager} events
```

Event manager.

IEventGroup

A group of event listeners.

Constructor | Methods

Constructor

```
IEventGroup()
```

Methods

Name	Returns	Description
add(types, callback[, context[, priority]])	IEventGroup	Adds an event listener.
remove(types, callback[, context[, priority]])	IEventGroup	Deletes an event listener from the group.
removeAll()	IEventGroup	Deletes all event listeners from the group.

Methods details

add

```
{IEventGroup} add(types, callback[, context[, priority]])
```

Adds an event listener.

Returns self-reference.

Parameters:

Parameter	Default value	Description
types*	—	Type: String|String[] Type or array of types for the event.
callback*	—	Type: Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Handler function. The event object is passed to the function as a parameter.
context	—	Type: Object Context for the handler function.
priority	0	Type: Integer Subscription priority.

* Mandatory parameter/option.

remove

```
{IEventGroup} remove(types, callback[, context[, priority]])
```

Deletes an event listener from the group.

Returns self-reference.

Parameters:

Parameter	Default value	Description
types*	—	Type: String|String[] Type or array of types for the events.
callback*	—	Type: Function Handler function. The event object is passed to the function as a parameter.
context	—	Type: Object Context for the handler function.
priority	0	Type: Integer Subscription priority.

* Mandatory parameter/option.

removeAll

```
{IEventGroup} removeAll()
```

Deletes all event listeners from the group.

Returns self-reference.

IEventManager

Extends **IEventTrigger**.

Event manager. Using an event manager, you can subscribe to and unsubscribe from events, as well as initiate the events themselves.

Constructor | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IEventManager()
```

Methods

Name	Returns	Description
add(types, callback[, context[, priority]])	**IEventManager**	Adds a new subscription.
fire(type[, event])	**IEventManager**	Triggers an event.
getParent()	**IEventManager**|null	Returns reference to the parent event manager.
group()	**IEventGroup**	Returns the group of event listeners associated with the given event manager.
once(types, callback[, context[, priority]])	**IEventManager**	Adds a listener, which calls the handler function only one time.
remove(types, callback[, context[, priority]])	**IEventManager**	Removes an existing subscription.
setParent(parent)		Sets the parent event manager.

Methods details

add

```
{IEventManager} add(types, callback[, context[, priority]])
```

Adds a new subscription.

Returns self-reference.

Parameters:

Parameter	Default value	Description
types*	—	Type: String|String[] Type or array of types for the event.
callback*	—	Type: Function Handler function for the event. An object describing the event is passed to the function as a parameter. It can be either an arbitrary object, or implement the interface **IEvent**.
context	—	Type: Object Context for the handler.
priority	0	Type: Integer Subscription priority.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

fire

```
{IEventManager} fire(type[, event])
```

Triggers an event.

Returns self-reference.

Parameters:

Parameter	Default value	Description
type*	—	Type: String Type of event.
event	—	Type: Object|Event Event. If a hash with data is passed, the createEventObject method will be called for it and further actions will be performed with the newly created event.

* Mandatory parameter/option.

getParent

```
{IEventManager|null} getParent()
```

Returns reference to the parent event manager.

group

```
{IEventGroup} group()
```

Returns the group of event listeners associated with the given event manager.

once

```
{IEventManager} once(types, callback[, context[, priority]])
```

Adds a listener, which calls the handler function only one time.

Returns self-reference.

Parameters:

Parameter	Default value	Description
types*	—	Type: String|String[] Type or array of types for the event.
callback*	—	Type: Function Handler function for the event. The function is passed an object of the event IEvent.
context	—	Type: Object Context for the handler.
priority	0	Type: Integer

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Subscription priority.

* Mandatory parameter/option.

remove

```
{IEventManager} remove(types, callback[, context[, priority]])
```

Removes an existing subscription.

Returns self-reference.

Parameters:

Parameter	Default value	Description
types*	—	Type: String|String[] Type or array of types for the event.
callback*	—	Type: Function Handler function for the event. The function is passed an object of the event **IEvent**.
context	—	Type: Object Context for the handler.
priority	0	Type: Integer Subscription priority.

* Mandatory parameter/option.

setParent

```
{} setParent(parent)
```

Sets the parent event manager.

Parameters:

Parameter	Default value	Description
parent*	—	Type: **IEventManager**|null Parent event manager.

* Mandatory parameter/option.

IEventPane

Extends **IDomEventEmitter**, **IPane**.

Interface for a map events pane. Allows to trace the DOM events on the map but, unlike IContainerPane, does not allow to place representations of map objects inside its DOM element.

Constructor | Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IEventPane()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
overflowchange	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of IEvent. Inherited from IPane.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
zindexchange	Change to the zIndex value of a pane. Instance of IEvent. Inherited from IPane.

Methods

Name	Returns	Description
destroy()		Destroys the pane. Inherited from IPane.
getElement()	HTMLElement	Returns a reference to the pane's DOM container. Inherited from IPane.
getMap()	Map	Returns the map that the pane belongs to. Inherited from IPane.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getOverflow()	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: • "visible" — When you go off the map container, the content of the pane remains visible. • "hidden" — The viewport for the pane content is limited by the map container. . Inherited from **IPane**.
getZIndex()	Number	Returns the zIndex of the pane. Inherited from **IPane**.

IEventTrigger

The object which triggers the events.

Constructor | Methods

Constructor

```
IEventTrigger()
```

Methods

Name	Returns	Description
fire(type[, eventObject])	**IEventTrigger**	Triggers an event.

Methods details

fire

```
{IEventTrigger} fire(type[, eventObject])
```

Triggers an event.

Returns self-reference.

Parameters:

Parameter	Default value	Description
type*	—	Type: String Type of event.
eventObject	—	Type: Object|**IEvent**

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Object that describes the event. It can be either an arbitrary object, or implement the interface **IEvent**. In the latter case, after each handler is called, the value of the isImmediatePropagationStopped() method is checked, and if true, event propagation is stopped immediately.

* Mandatory parameter/option.

IEventWorkflowController

Extends **IEventController**.

Interface for an event controller that can affect event propagation through the tree.

Constructor | Methods

Constructor

```
IEventWorkflowController()
```

Methods

Name	Description
onAfterEventFiring(events, type[, event])	Function that is called after an event has been handled by the event manager. This method is optional.
onBeforeEventFiring(events, type[, event])	Function that is called before an event is handled by the event manager. This method is optional.
onStartListening(events, type)	Called on the first subscription to the specified event type using the specified event manager. This method is optional. Inherited from **IEventController**.
onStopListening(events, type)	Called when a particular event type stops listening on the specified event manager (the last subscription is deleted). This method is optional. Inherited from **IEventController**.

Methods details

onAfterEventFiring

```
{} onAfterEventFiring(events, type[, event])
```

Function that is called after an event has been handled by the event manager. This method is optional.

Parameters:

Parameter	Default value	Description
events*	—	Type: **IEventManager** Event manager.
type*	—	Type: String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Type of event.
event	—	Type: **IEvent** Event.

* Mandatory parameter/option.

onBeforeEventFiring

```
{} onBeforeEventFiring(events, type[, event])
```

Function that is called before an event is handled by the event manager. This method is optional.

Parameters:

Parameter	Default value	Description
events*	—	Type: **IEventManager** Event manager.
type*	—	Type: String Type of event.
event	—	Type: **IEvent** Event.

* Mandatory parameter/option.

IExpandableControlLayout

Extends **ILayout**.

Interface for a layout that can be in the collapsed or expanded state.

Constructor | Fields | Events | Methods

Constructor

```
IExpandableControlLayout()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
collapse	Event that initiates collapsing an object.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty layout indicator. Instance of the Event class. Inherited from ILayout.
expand	Event that initiates expanding an object.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
parentelementchange	Change to the parent element. Instance of the **Event** class. Inherited from **ILayout**.
shapechange	Change to the shape of the area spanning the layout. Instance of the **Event** class. Inherited from **ILayout**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from **ILayout**.
getData()	Object	Returns layout data object. Inherited from **ILayout**.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from **ILayout**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from **ILayout**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **ILayout**.
setData(data)		Sets layout data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **ILayout**.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from **ILayout**.

Events details

collapse

Event that initiates collapsing an object.

expand

Event that initiates expanding an object.

IFreezable

Interface for an object with a state change event that can be disabled. Object that implements **IFreezable** it can function in one of the following modes:

- 1. Active. In this mode, each change to the internal state of the object generates an **IFreezable.event:change** event.
- 2. Frozen. In this mode, changes to the object state do not cause an **IFreezable.event:change** event, but if changes occurred, the **IFreezable.event:change** event will be generated once when switching to active mode.

Constructor | Fields | Events | Methods

Constructor

```
IFreezable()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager for the object.

Events

Name	Description
change	Change to the internal state of the object.

Methods

Name	Returns	Description
freeze()	**IFreezable**	Switches the object to "frozen" mode.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false.
unfreeze()	**IFreezable**	Switches the object to active mode.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

events

```
{IEventManager} events
```

Event manager for the object.

Events details

change

Change to the internal state of the object.

Methods details

freeze

```
{IFreezable} freeze()
```

Switches the object to "frozen" mode.

Returns self-reference.

isFrozen

```
{Boolean} isFrozen()
```

Returns true if the object is in "frozen" mode, otherwise false.

unfreeze

```
{IFreezable} unfreeze()
```

Switches the object to active mode.

Returns self-reference.

IGeocodeProvider

Interface for a geocoder provider.

Constructor | Methods

Constructor

```
IGeocodeProvider()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
geocode(request[, options])	vow.Promise	Sends a request for geocoding. A handler function for processing geocoding results can be added via the returned promise object. The object input to the handler function can contain only the following types of fields: geoObjects, layers, mapState, and metaData.
suggest(request[, options])	vow.Promise	Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }. The displayName field represents the toponym in a user-friendly way, and the value field represents the value which should be inserted into the search field after the user selects the suggestion. This method is optional.

Methods details

geocode

```
{vow.Promise} geocode(request[, options])
```

Sends a request for geocoding. A handler function for processing geocoding results can be added via the returned promise object. The object input to the handler function can contain only the following types of fields: geoObjects, layers, mapState, and metaData.

Returns Promise object.

Parameters:

Parameter	Default value	Description
request*	—	Type: String Request string.
options	—	Type: Object Options.
options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
options.results	—	Type: Number Maximum number of results to be returned.
options.skip	—	Type: Number

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Number of results that must be skipped.
options.strictBounds	—	Type: Boolean Search only inside the area defined by the "boundedBy" option.

* Mandatory parameter/option.

suggest

```
{vow.Promise} suggest(request[, options])
```

Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }. The displayName field represents the toponym in a user-friendly way, and the value field represents the value which should be inserted into the search field after the user selects the suggestion.

This method is optional.

Returns a Promise object.

Parameters:

Parameter	Default value	Description
request*	—	Type: String Request string.
options	—	Type: Object Options.
options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
options.results	—	Type: Number Maximum number of results to be returned.
options.strictBounds	—	Type: Boolean Search only inside the area defined by the "boundedBy" option.

* Mandatory parameter/option.

IGeometry

Extends **IBaseGeometry**, **ICustomizable**.

Interface of a geometry.

Constructor | Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IGeometry()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
mapchange	Map reference changed. Instance of the Event class. Names of fields that are available via the Event.get method: • oldMap — Old map. • newMap — New map.
optionschange	Change to the object options. Inherited from ICustomizable.
pixelgeometrychange	The pixel geometry changed. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelGeometry — New IPixelGeometry pixel geometry.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.
getMap()	Map|null	Returns the current map.
getPixelGeometry([options])	IPixelGeometry	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.
getType()	String	Returns ID of the geometry type. Inherited from IBaseGeometry.
setMap(map)		Sets the map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

mapchange

Map reference changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldMap — Old map.
- newMap — New map.

pixelgeometrychange

The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelGeometry — New **IPixelGeometry** pixel geometry.

Methods details

getBounds

```
{Number[][]|null} getBounds()
```

Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.

Example:

```
// Setting the map center and zoom so that the geometry is displayed in its
entirety inside the visible area.
map.setBounds(myGeometry.getBounds());
```

getMap

```
{Map|null} getMap()
```

Returns the current map.

getPixelGeometry

```
{IPixelGeometry} getPixelGeometry([options])
```

Returns the pixel geometry corresponding to the given geometry, its options, and the map state.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Hash of options that allow to exclude part of the current geometry options for this calculation.

Example:

```
// Getting a pixel representation of the geometry with geodesics calculated, but
without optimizing deletion of invisible points.
myGeometry.getPixelGeometry({
    geodesic: true,
    simplification: false
}).getCoordinates();
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

setMap

```
{} setMap(map)
```

Sets the map.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map**|null Reference to the map.

* Mandatory parameter/option.

IGeometryEditor

Extends **ICustomizable**, **IEventEmitter**.

Interface for the geometry editor.

Constructor | Fields | Events | Methods

Constructor

```
IGeometryEditor()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**IGeometry**	The geometry being edited.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**IDataManager**	State of the geometry editor.

Events

Name	Description
optionschange	Change to the object options. Inherited from **ICustomizable**.
statechange	Change to the geometry editor state. Instance of the **Event** class.

Methods

Name	Description
startEditing()	Enables editing mode.
stopEditing()	Disables editing mode.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

geometry

```
{IGeometry} geometry
```

The geometry being edited.

state

```
{IDataManager} state
```

State of the geometry editor.

Events details

statechange

Change to the geometry editor state. Instance of the **Event** class.

Methods details

startEditing

```
{} startEditing()
```

Enables editing mode.

stopEditing

```
{} stopEditing()
```

Disables editing mode.

IGeometryEditorChildModel

Extends **IGeometryEditorModel**.

Interface for the child data model.

Constructor | Fields | Methods

Constructor

```
IGeometryEditorChildModel(geometry, editor, pixels, parent)
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IBaseGeometry** The child geometry being edited. The IBaseGeometry interface is not aware of the pixelgeometrychange event, therefore, the pixel data is retrieved from the parent data model.
editor *	—	Type: **IGeometryEditor**

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Link to the geometry editor.
pixels *	—	Type: Number[] The model's pixel data.
parent *	—	Type: IGeometryEditorModel The parent data model.

* Mandatory parameter/option.

Fields

Name	Type	Description
editor	IGeometryEditor	Geometry editor.
events	IEventManager	Event manager. Inherited from IEventEmitter.
geometry	IBaseGeometry	Geometry of the model.

Methods

Name	Returns	Description
destroy()		Destructor. Inherited from IGeometryEditorModel.
getParent()	IGeometryEditorModel	Returns the parent data model.
getPixels()	Number[]	Returns the model's pixel data. Inherited from IGeometryEditorModel.
setPixels(pixels)		Sets the model's pixel data.

Fields details

editor

```
{IGeometryEditor} editor
```

Geometry editor.

geometry

```
{IBaseGeometry} geometry
```

Geometry of the model.

Methods details

getParent

```
{IGeometryEditorModel} getParent()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns the parent data model.

setPixels

```
{} setPixels(pixels)
```

Sets the model's pixel data.

Parameters:

Parameter	Default value	Description
pixels*	—	Type: Number[] Pixel data.

* Mandatory parameter/option.

IGeometryEditorModel

Extends **IEventEmitter**.

Interface for the data model of the geometry editor.

Constructor | Fields | Methods

Constructor

```
IGeometryEditorModel(geometry, editor)
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IBaseGeometry** The geometry being edited.
editor *	—	Type: **IGeometryEditor** Link to the geometry editor.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
destroy()		Destructor.
getPixels()	Number[]	Returns the model's pixel data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

destroy

```
{} destroy()
```

Destructor.

getPixels

```
{Number[]} getPixels()
```

Returns the model's pixel data.

IGeometryEditorRootModel

Extends **IGeometryEditorModel**.

Interface for the root data model.

Constructor | Fields | Methods

Constructor

```
IGeometryEditorRootModel(geometry, editor)
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IGeometry** The geometry being edited.
editor *	—	Type: **IGeometryEditor** Link to the geometry editor.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
destroy()		Destructor. Inherited from **IGeometryEditorModel**.
getPixels()	Number[]	Returns the model's pixel data. Inherited from **IGeometryEditorModel**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

IGeometryJson

The interface of an object that describes a JSON representation of the geometry.

Constructor | Fields

Constructor

```
IGeometryJson()
```

Fields

Name	Type	Description
type	String	ID of the geometry type.

Fields details

type

```
{String} type
```

ID of the geometry type.

IGeoObject

Extends **IChildOnMap**, **ICustomizable**, **IDomEventEmitter**, **IParentOnMap**.

Interface for a geo object.

Constructor | Fields | Events | Methods

Constructor

```
IGeoObject()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from **IDomEventEmitter**.
geometry	IGeometry|null	Geo object geometry.
options	IOptionManager	Options manager. Inherited from **ICustomizable**.
properties	IDataManager	Geo object data.
state	IDataManager	State of the geo object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

geometry

```
{IGeometry|null} geometry
```

Geo object geometry.

properties

```
{IDataManager} properties
```

Geo object data.

state

```
{IDataManager} state
```

State of the geo object.

Events details

geometrychange

Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method:

- originalEvent: IEvent — Original event of the geometry.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

overlaychange

Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- overlay: **IOverlay**|null — Reference to the overlay.
- oldOverlay: **IOverlay**|null — Previous overlay of the geo object.

propertieschange

Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- originalEvent: **IEvent** — Original event of the data manager.

Methods details

getOverlay

```
{vow.Promise} getOverlay()
```

Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.

getOverlaySync

```
{IOverlay|null} getOverlaySync()
```

The method provides synchronous access to the overlay.

Returns a reference to the overlay, or null if the overlay is missing.

IGeoObjectCollection

Extends **ICustomizable**, **IEventEmitter**, **IParentOnMap**.

Interface of a collection of geo objects.

Constructor | Fields | Events | Methods

Constructor

```
IGeoObjectCollection()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
add	A child geo object has been added (inserted). Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index: Integer — Index of the added geo object. • child: **IGeoObject** — Reference to the added geo object.
boundschange	Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the **Event** class.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
pixelboundschange	Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the **Event** class.
remove	A child geo object has been removed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index: Integer — Index of the deleted geo object. • child: **IGeoObject** — Reference to the deleted geo object.
set	A new child geo object has been added to the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index: Integer — Index of the geo object. • child: **IGeoObject** — Reference to the new geo object. • prevChild: **IGeoObject** — Reference to the previous value for this index.

Methods

Name	Returns	Description
add(child[, index])	**IGeoObjectCollection**	Adds (inserts) a child geo object to the collection.
each(callback[, context])		Calls a handler function for each child geo object.
get(index)	**IGeoObject**	Returns a child geo object with the specified index.
getBounds()	Number[][]|null	Returns geographical coordinates of the area that covers the collection and its child geo objects.
getIterator()	**IIterator**	Returns iterator for the collection.
getLength()	Integer	Returns length of the collection.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getPixelBounds()	Number[][]|null	Returns global pixel coordinates of the area that spans the collection and its child geo objects.
indexOf(object)	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.
remove(child)	**IGeoObjectCollection**	Removes a child geo object from the collection.
removeAll()	**IGeoObjectCollection**	Clears the collection.
set(index, child)	**IGeoObjectCollection**	Adds a new child geo object to the collection.
splice(index, number)	**IGeoObjectCollection**	Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

Events details

add

A child geo object has been added (inserted). Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- index: Integer — Index of the added geo object.
- child: **IGeoObject** — Reference to the added geo object.

boundschange

Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the **Event** class.

pixelboundschange

Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the **Event** class.

remove

A child geo object has been removed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- index: Integer — Index of the deleted geo object.
- child: **IGeoObject** — Reference to the deleted geo object.

set

A new child geo object has been added to the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- index: Integer — Index of the geo object.
- child: **IGeoObject** — Reference to the new geo object.
- prevChild: **IGeoObject** — Reference to the previous value for this index.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

add

```
{IGeoObjectCollection} add(child[, index])
```

Adds (inserts) a child geo object to the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
child*	—	Type: **IGeoObject** Child object.
index	—	Type: Integer The index where the new object is added. By default, the object is added to the end of the collection.

* Mandatory parameter/option.

each

```
{} each(callback[, context])
```

Calls a handler function for each child geo object.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function.
context	—	Type: Object Context for the handler function.

* Mandatory parameter/option.

get

```
{IGeoObject} get(index)
```

Returns a child geo object with the specified index.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getBounds

```
{Number[][]|null} getBounds()
```

Returns geographical coordinates of the area that covers the collection and its child geo objects.

getIterator

```
{IIterator} getIterator()
```

Returns iterator for the collection.

getLength

```
{Integer} getLength()
```

Returns length of the collection.

getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Returns global pixel coordinates of the area that spans the collection and its child geo objects.

indexOf

```
{Integer} indexOf(object)
```

Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.

Parameters:

Parameter	Default value	Description
object*	—	Type: Object Child object.

* Mandatory parameter/option.

remove

```
{IGeoObjectCollection} remove(child)
```

Removes a child geo object from the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
child*	—	Type: IGeoObject Geo object being removed.

* Mandatory parameter/option.

removeAll

```
{IGeoObjectCollection} removeAll()
```

Clears the collection.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns self-reference.

set

```
{IGeoObjectCollection} set(index, child)
```

Adds a new child geo object to the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index.
child*	—	Type: **IGeoObject** Child object.

* Mandatory parameter/option.

splice

```
{IGeoObjectCollection} splice(index, number)
```

Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

Returns collection of deleted geo objects.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of the geo object to start deletion from.
number*	—	Type: Integer The number of geo objects to be deleted.

* Mandatory parameter/option.

IGeoObjectPopupData

Static object.

A data object that the geo object passes to its own info object (for example, to a balloon or hint).

Fields

Fields

Name	Type	Description
geometry	**IGeometry**	Reference to the geometry of the geo object.
geoObject	**IGeoObject**	Reference to the geo object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	IDataManager	Reference to the properties of the geo object.
userData	Object	User data.

Fields details

geometry

```
{IGeometry} geometry
```

Reference to the geometry of the geo object.

geoObject

```
{IGeoObject} geoObject
```

Reference to the geo object.

properties

```
{IDataManager} properties
```

Reference to the properties of the geo object.

userData

```
{Object} userData
```

User data.

IGeoObjectSequence

Extends ICustomizable, IEventEmitter, IParentOnMap.

Interface of an unchangeable collection of geo objects.

Constructor | Fields | Events | Methods

Constructor

```
IGeoObjectSequence()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
boundschange	Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the Event class.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
optionschange	Change to the object options. Inherited from ICustomizable.
pixelboundschange	Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the Event class.

Methods

Name	Returns	Description
each(callback[, context])		Calls a handler function for each child geo object.
get(index)	IGeoObject	Returns a child geo object with the specified index.
getBounds()	Number[][]|null	Returns geographical coordinates of the area that covers the collection and its child geo objects.
getIterator()	IIterator	Returns iterator for child geo objects in the collection.
getLength()	Integer	Returns length of the collection.
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getPixelBounds()	Number[][]|null	Returns global pixel coordinates of the area that spans the collection and its child geo objects.
indexOf(object)	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.

Events details

boundschange

Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the Event class.

pixelboundschange

Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the Event class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

each

```
{} each(callback[, context])
```

Calls a handler function for each child geo object.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function.
context	—	Type: Object Context for the handler function.

* Mandatory parameter/option.

get

```
{IGeoObject} get(index)
```

Returns a child geo object with the specified index.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index.

* Mandatory parameter/option.

getBounds

```
{Number[][]|null} getBounds()
```

Returns geographical coordinates of the area that covers the collection and its child geo objects.

getIterator

```
{IIterator} getIterator()
```

Returns iterator for child geo objects in the collection.

getLength

```
{Integer} getLength()
```

Returns length of the collection.

getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Returns global pixel coordinates of the area that spans the collection and its child geo objects.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

indexOf

```
{Integer} indexOf(object)
```

Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.

Parameters:

Parameter	Default value	Description
object*	—	Type: **IGeoObject** Child object.

* Mandatory parameter/option.

IGroupControlLayout

Extends **ILayout**.

Interface for the layout of a group control.

Constructor | Fields | Events | Methods

Constructor

```
IGroupControlLayout(data)
```

Parameters:

Parameter	Default value	Description
data *	—	Type: Object Layout data.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.

Events

Name	Description
childcontainerchange	Change to the container for child elements. Data fields: • newChildContainerElement — New element for child objects. • oldChildContainerElement — Old element for child objects.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty layout indicator. Instance of the Event class. Inherited from ILayout.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
parentelementchange	Change to the parent element. Instance of the Event class. Inherited from ILayout.
shapechange	Change to the shape of the area spanning the layout. Instance of the Event class. Inherited from ILayout.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getChildContainerElement()	Object	Returns a reference to the DOM element that the child elements should be added to.
getData()	Object	Returns layout data object. Inherited from ILayout.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from ILayout.
getShape()	IShape|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from ILayout.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from ILayout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setData(data)		Sets layout data. Inherited from **ILayout**.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from **ILayout**.

Events details

childcontainerchange

Change to the container for child elements.

Data fields:

- newChildContainerElement — New element for child objects.
- oldChildContainerElement — Old element for child objects.

Methods details

getChildContainerElement

```
{Object} getChildContainerElement()
```

Returns a reference to the DOM element that the child elements should be added to.

IHint

Extends **IPopup**.

Interface for a hint.

Constructor | Fields | Events | Methods

Constructor

```
IHint()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
close	Closing the info object. Inherited from **IPopup**.
open	Opening the info object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IPopup**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Methods

Name	Returns	Description
close([force])	**vow.Promise**	Closes the info object. Inherited from **IPopup**.
getData()		Returns info object data. Inherited from **IPopup**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopup**.
getOverlaySync()	**IOverlay**	Returns the overlay, if one exists. Inherited from **IPopup**.
getPosition()		Returns the coordinates of the info object. Inherited from **IPopup**.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from **IPopup**.
open([position[, data]])	**vow.Promise**	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the **IProjection** that is in the options. Inherited from **IPopup**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopup**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopup**.

IHintManager

Extends **IPopupManager**.

Interface for the hint manager.

Constructor | Fields | Events | Methods

Constructor

```
IHintManager()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
close	Closing the info object. Names of fields available via Event.get: • target — Reference to the object where the closing occurred. Inherited from IPopupManager.
open	Opening the info object. Names of fields available via Event.get: • target — Reference to the object where the opening occurred. Inherited from IPopupManager.

Methods

Name	Returns	Description
close([force])	vow.Promise	Closes the info object. Inherited from IPopupManager.
destroy()		Disables the info object manager. Inherited from IPopupManager.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from IPopupManager.
getOptions()	IOptionManager|null	Returns the options manager or 'null'. Inherited from IPopupManager.
getOverlay()	vow.Promise	Returns the promise object to return the overlay. Inherited from IPopupManager.
getOverlaySync()	IOverlay|null	Returns the overlay, if one exists. Inherited from IPopupManager.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from IPopupManager.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from IPopupManager.
open([position[, data[, options]]])	vow.Promise	Opens the info object at the specified position. Inherited from IPopupManager.
setData(data)	vow.Promise	Defines new data for the info object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IPopupManager**.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

IHintOwner

An object with a hint can can be accessed through the "hint" property.

Constructor | Fields | Events

Constructor

```
IHintOwner()
```

Fields

Name	Type	Description
hint	**IHintManager**	Object hint.

Events

Name	Description
hintclose	Hiding the hint.
hintopen	Opening the hint.

Fields details

hint

```
{IHintManager} hint
```

Object hint.

Events details

hintclose

Hiding the hint.

hintopen

Opening the hint.

IHotspot

Extends **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Interface of a shape that defines the hotspot geometry.

Note:
The shape should be set in global pixel coordinates.

Constructor | Fields | Events | Methods

Constructor

```
IHotspot()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getShape()	**IShape**	Returns the hotspot shape.
getZIndex()	Number	Returns zIndex of the hotspot.
setShape(shape)		Sets the hotspot shape.
setZIndex(zIndex)		Sets the z-index of the hotspot.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

getShape

```
{IShape} getShape()
```

Returns the hotspot shape.

getZIndex

```
{Number} getZIndex()
```

Returns zIndex of the hotspot.

setShape

```
{} setShape(shape)
```

Sets the hotspot shape.

Parameters:

Parameter	Default value	Description
shape*	—	Type: IShape Shape.

* Mandatory parameter/option.

setZIndex

```
{} setZIndex(zIndex)
```

Sets the z-index of the hotspot.

Parameters:

Parameter	Default value	Description
zIndex*	—	Type: Number The z-index that will be set for the hotspot.

* Mandatory parameter/option.

IHotspotLayerObject

Extends **ICustomizable**, **IDomEventEmitter**.

Interface for the hotspot layer object.

Note:
A figure describing the hotspot should be set in global pixel coordinates.

Constructor | Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IHotspotLayerObject()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape that defines a hotspot. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getGeometry()	Object	Returns the actual geometry of an object.
getHotspot()	**IHotspot**	Returns object describing a hotspot.
getId()	Number	Returns object ID.
getProperties()	Object	Returns object data.
setGeometry(geometry)		Defines the actual geometry of the object.
setId(id)		Sets the object ID.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setProperties(properties)		Sets the object's data.

Events details

shapechange

Change to the shape that defines a hotspot. Instance of the **Event** class.

Methods details

getGeometry

```
{Object} getGeometry()
```

Returns the actual geometry of an object.

getHotspot

```
{IHotspot} getHotspot()
```

Returns object describing a hotspot.

getId

```
{Number} getId()
```

Returns object ID.

getProperties

```
{Object} getProperties()
```

Returns object data.

setGeometry

```
{} setGeometry(geometry)
```

Defines the actual geometry of the object.

Parameters:

Parameter	Default value	Description
geometry*	—	Type: Object The actual geometry of an object.

* Mandatory parameter/option.

setId

```
{} setId(id)
```

Sets the object ID.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
id*	—	Type: Number Object ID.

* Mandatory parameter/option.

setProperties

```
{} setProperties(properties)
```

Sets the object's data.

Parameters:

Parameter	Default value	Description
properties*	—	Type: Object Object data.

* Mandatory parameter/option.

IHotspotObjectSource

Extends **ICustomizable**.

Source of objects for the hotspot layers.

Constructor | Fields | Events | Methods

Constructor

```
IHotspotObjectSource()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
optionschange	Change to the object options. Inherited from **ICustomizable**.

Methods

Name	Description
cancelLastRequest()	Cancels the last request for data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
requestObjects(layer, tileNumber, zoom, callback)	Builds an array of **IHotspotLayerObject** objects corresponding to a particular layer, tile, and map zoom level, and passes it to the callback function.

Methods details

cancelLastRequest

```
{} cancelLastRequest()
```

Cancels the last request for data.

requestObjects

```
{} requestObjects(layer, tileNumber, zoom, callback)
```

Builds an array of **IHotspotLayerObject** objects corresponding to a particular layer, tile, and map zoom level, and passes it to the callback function.

Parameters:

Parameter	Default value	Description
layer*	—	Type: **hotspot.Layer** Hotspot layer.
tileNumber*	—	Type: Number[] Tile coordinates.
zoom*	—	Type: Number Zoom level.
callback*	—	Type: Function Handler function.

* Mandatory parameter/option.

IHotspotShape

Extends **ICustomizable**, **IDomEventEmitter**.

Interface of a shape that defines the hotspot geometry.

Note:
Not supported, beginning from version 2.1.

Constructor | Fields | Events | Methods

Constructor

```
IHotspotShape()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
containerchange	Change to the container. Instance of the Event class.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape that defines a hotspot. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getContainer()	IHotspotContainer	Returns the hotspot container.
getGeometry()	Object	Returns the actual shape geometry.
getId()	Number	Returns object ID.
getProperties()	Object	Returns object data.
getShape()	**IShape**	Returns shape describing a hotspot.
setGeometry(geometry)		Sets the actual geometry of the shape.
setId(id)		Sets the object ID.
setProperties(properties)		Sets the object's data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setShape(shape)		

Events details

containerchange

Change to the container. Instance of the **Event** class.

shapechange

Change to the shape that defines a hotspot. Instance of the **Event** class.

Methods details

getContainer

```
{IHotspotContainer} getContainer()
```

Returns the hotspot container.

getGeometry

```
{Object} getGeometry()
```

Returns the actual shape geometry.

getId

```
{Number} getId()
```

Returns object ID.

getProperties

```
{Object} getProperties()
```

Returns object data.

getShape

```
{IShape} getShape()
```

Returns shape describing a hotspot.

setGeometry

```
{} setGeometry(geometry)
```

Sets the actual geometry of the shape.

Parameters:

Parameter	Default value	Description
geometry*	—	Type: Object Actual shape geometry.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

setId

```
{} setId(id)
```

Sets the object ID.

Parameters:

Parameter	Default value	Description
id*	—	Type: Number Object ID.

* Mandatory parameter/option.

setProperties

```
{} setProperties(properties)
```

Sets the object's data.

Parameters:

Parameter	Default value	Description
properties*	—	Type: Object Object data.

* Mandatory parameter/option.

setShape

```
{} setShape(shape)
```

Parameters:

Parameter	Default value	Description
shape*	—	Type: **IShape** Shape describing a hotspot.

* Mandatory parameter/option.

IIterator

Interface for an iterator. The iterator allows you to iterate through all the items in a collection.

Constructor | Fields | Methods

Constructor

```
IIterator()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
STOP_ITERATION	Object	The object returned by the method IIterator.getNext at the end of the iteration.

Methods

Name	Returns	Description
getNext()	Object|IIterator.STOP_ITERATION	Returns a reference to the next item in the collection or the IIterator.#STOP_ITERATION object if the iteration is completed.

Fields details

STOP_ITERATION

```
{Object} STOP_ITERATION
```

The object returned by the method IIterator.getNext at the end of the iteration.

Methods details

getNext

```
{Object|IIterator.STOP_ITERATION} getNext()
```

Returns a reference to the next item in the collection or the IIterator.#STOP_ITERATION object if the iteration is completed.

ILayer

Extends IChildOnMap, ICustomizable, IEventEmitter.

Interface for a map layer.

Constructor | Fields | Events | Methods

Constructor

```
ILayer()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
brightnesschange	Layer brightness change event.
copyrightschange	Event for changes to available copyright information.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
zoomrangechange	Event for changes to available information about the zoom level range.

Methods

Name	Returns	Description
getBrightness()	Number	Optional method.
getCopyrights(coords, zoom)	**vow.Promise**	Optional method. Requests information about copyrights at the specified point with the specified zoom.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getZoomRange(point)	**vow.Promise**	Optional method. Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers — the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Events details

brightnesschange

Layer brightness change event.

copyrightschange

Event for changes to available copyright information.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

zoomrangechange

Event for changes to available information about the zoom level range.

Methods details

getBrightness

```
{Number} getBrightness()
```

Optional method.

Returns the layer brightness value from 0 to 1 (0 is zero brightness, and 1 is maximum brightness). The total brightness of the layers added to the map determines which color is selected for the logo and copyrights on the map.

getCopyrights

```
{vow.Promise} getCopyrights(coords, zoom)
```

Optional method. Requests information about copyrights at the specified point with the specified zoom.

Returns Promise that will be resolved and will pass as a result an array of strings or DOM elements with information about copyrights.

Parameters:

Parameter	Default value	Description
coords*	—	Type: Number[] The point on the map that copyright information is being requested for.
zoom*	—	Type: Number The zoom level that copyright information is being requested for.

* Mandatory parameter/option.

getZoomRange

```
{vow.Promise} getZoomRange(point)
```

Optional method. Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers — the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error.

Returns Promise object.

Parameters:

Parameter	Default value	Description
point*	—	Type: Number[] Point

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

ILayout

Extends **IDomEventEmitter**.

Layout interface.

Constructor | Fields | Events | Methods

Constructor

```
ILayout(data)
```

Parameters:

Parameter	Default value	Description
data *	—	Type: Object Layout data.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty layout indicator. Instance of the **Event** class.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
parentelementchange	Change to the parent element. Instance of the **Event** class.
shapechange	Change to the shape of the area spanning the layout. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished.
getData()	Object	Returns layout data object.
getParentElement()	HTMLElement	Returns parent HTML element.
getShape()	IShape|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.
setData(data)		Sets layout data.
setParentElement(parent)		Adds the layout to the DOM tree.

Events details

emptinesschange

Change to the empty layout indicator. Instance of the **Event** class.

parentelementchange

Change to the parent element. Instance of the **Event** class.

shapechange

Change to the shape of the area spanning the layout. Instance of the **Event** class.

Methods details

destroy

```
{} destroy()
```

Destructor. Called when activity with the layout is finished.

getData

```
{Object} getData()
```

Returns layout data object.

getParentElement

```
{HTMLElement} getParentElement()
```

Returns parent HTML element.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getShape

```
{IShape|null} getShape()
```

Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.

isEmpty

```
{Boolean} isEmpty()
```

Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.

setData

```
{} setData(data)
```

Sets layout data.

Parameters:

Parameter	Default value	Description
data*	—	Type: Object Layout data.

* Mandatory parameter/option.

setParentElement

```
{} setParentElement(parent)
```

Adds the layout to the DOM tree.

Parameters:

Parameter	Default value	Description
parent*	—	Type: HTMLElement|null Parent HTML element. The parent element must be added to the DOM tree. If null is passed, the element is removed from the DOM tree.

* Mandatory parameter/option.

ILinearRingGeometryAccess

Extends **IFreezable**.

Interface for access to the "Closed contour" geometry.

Constructor | Fields | Events | Methods

Constructor

```
ILinearRingGeometryAccess()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager for the object. Inherited from IFreezable.

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. • oldFillRule — Old fill rule. • newFillRule — New fill rule.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the contour.
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(index)	Number[]	Returns coordinates of the point with the specified index.
getChildGeometry(index)	IPointGeometryAccess	Creates and returns an IPointGeometryAccess object for the specified contour on the polyline.
getClosest(anchorPosition)	Object	Searches for a point on the contour closest to the anchorPosition.
getCoordinates()	Number[][]	Returns an array of geometry coordinates.
getFillRule()	String	Returns ID of the fill rule.
getLength()	Integer	Returns the number of points in the geometry.
insert(index, coordinates)	ILinearRingGeometryAccess	Adds a new point with the specified index.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from IFreezable.
remove(index)	Number[]	Removes the point with the specified index.
set(index, coordinates)	ILinearRingGeometryAccess	Sets coordinates of the point with the specified index.
setCoordinates(coordinates)	ILinearRingGeometryAccess	Sets an array of geometry coordinates.
setFillRule(fillRule)	ILinearRingGeometryAccess	Sets the contour fill rule.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
splice(index, number)	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.
unfreeze()	IFreezable	Switches the object to active mode. Inherited from IFreezable.

Events details

change

Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method:

- oldCoordinates — Old coordinates
- newCoordinates — New coordinates.
- oldFillRule — Old fill rule.
- newFillRule — New fill rule.

Methods details

contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the contour.

Returns an indicator for whether the point belongs to the contour.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

get

```
{Number[]} get(index)
```

Returns coordinates of the point with the specified index.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a point.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getChildGeometry

```
{IPointGeometryAccess} getChildGeometry(index)
```

Creates and returns an **IPointGeometryAccess** object for the specified contour on the polyline.

Returns the "Point" geometry object that corresponds to the specified endpoint.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a contour vertex.

* Mandatory parameter/option.

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the contour closest to the anchorPosition.

Returns an object with the following fields:

- position — Point on the contour closest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".
- closestPointIndex — Index of the vertex closest to "position".
- nextPointIndex — Index of the vertex that follows "position".
- prevPointIndex — Index of the vertex that precedes "position".

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the contour vertexes.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest contour vertex is calculated.

* Mandatory parameter/option.

getCoordinates

```
{Number[][]} getCoordinates()
```

Returns an array of geometry coordinates.

getFillRule

```
{String} getFillRule()
```

Returns ID of the fill rule.

getLength

```
{Integer} getLength()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns the number of points in the geometry.

insert

```
{ILinearRingGeometryAccess} insert(index, coordinates)
```

Adds a new point with the specified index.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a point.
coordinates*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

remove

```
{Number[]} remove(index)
```

Removes the point with the specified index.

Returns the coordinates of a deleted point.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a point.

* Mandatory parameter/option.

set

```
{ILinearRingGeometryAccess} set(index, coordinates)
```

Sets coordinates of the point with the specified index.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a point.
coordinates*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

setCoordinates

```
{ILinearRingGeometryAccess} setCoordinates(coordinates)
```

Sets an array of geometry coordinates.

Returns self-reference.

Parameters:

Parameter	Default value	Description
coordinates*	—	Type: Number[][] Geometry coordinates.

* Mandatory parameter/option.

setFillRule

```
{ILinearRingGeometryAccess} setFillRule(fillRule)
```

Sets the contour fill rule.

Returns self-reference.

Parameters:

Parameter	Default value	Description
fillRule*	—	Type: String ID of the fill rule.

* Mandatory parameter/option.

splice

```
{Number[][]} splice(index, number)
```

Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.

Returns an array of coordinates of deleted points.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer The index to start from for removing and adding points.
number*	—	Type: Integer The number of deleted points.

* Mandatory parameter/option.

ILineStringGeometry

Extends **IGeometry**, **ILineStringGeometryAccess**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Interface for the "Polyline" geometry.

Constructor | Fields | Events | Methods

Constructor

```
ILineStringGeometry()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. Inherited from ILineStringGeometryAccess.
mapchange	Map reference changed. Instance of the Event class. Names of fields that are available via the Event.get method: • oldMap — Old map. • newMap — New map. Inherited from IGeometry.
optionschange	Change to the object options. Inherited from ICustomizable.
pixelgeometrychange	The pixel geometry changed. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelGeometry — New IPixelGeometry pixel geometry. Inherited from IGeometry.

Methods

Name	Returns	Description
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(index)	Number[]	Returns coordinates of the point with the specified index. Inherited from ILineStringGeometryAccess.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getChildGeometry(index)	**IPointGeometryAccess**	Creates and returns an **IPointGeometryAccess** object for the specified vertex on the polyline. Inherited from **ILineStringGeometryAccess**.
getClosest(anchorPosition)	Object	Searches for a point on the polyline closest to the anchorPosition. Inherited from **ILineStringGeometryAccess**.
getCoordinates()	Number[][]	Returns an array of geometry coordinates. Inherited from **ILineStringGeometryAccess**.
getLength()	Integer	Returns the number of points in the geometry. Inherited from **ILineStringGeometryAccess**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getType()	String	Returns the "LineString" string.
insert(index, coordinates)	**ILineStringGeometryAccess**	Adds a new point with the specified index. Inherited from **ILineStringGeometryAccess**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	Number[]	Removes the point with the specified index. Inherited from **ILineStringGeometryAccess**.
set(index, coordinates)	**ILineStringGeometryAccess**	Sets coordinates of the point with the specified index. Inherited from **ILineStringGeometryAccess**.
setCoordinates(coordinates)	**ILineStringGeometryAccess**	Sets an array of geometry coordinates. Inherited from **ILineStringGeometryAccess**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setMap(map)		Sets the map. Inherited from **IGeometry**.
splice(index, number)	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter. Inherited from **ILineStringGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Methods details

getType

```
{String} getType()
```

Returns the "LineString" string.

ILineStringGeometryAccess

Extends **IFreezable**.

Interface for access to the "Polyline" geometry.

Constructor | Fields | Events | Methods

Constructor

```
ILineStringGeometryAccess()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager for the object. Inherited from **IFreezable**.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(index)	Number[]	Returns coordinates of the point with the specified index.
getChildGeometry(index)	IPointGeometryAccess	Creates and returns an IPointGeometryAccess object for the specified vertex on the polyline.
getClosest(anchorPosition)	Object	Searches for a point on the polyline closest to the anchorPosition.
getCoordinates()	Number[][]	Returns an array of geometry coordinates.
getLength()	Integer	Returns the number of points in the geometry.
insert(index, coordinates)	ILineStringGeometryAccess	Adds a new point with the specified index.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from IFreezable.
remove(index)	Number[]	Removes the point with the specified index.
set(index, coordinates)	ILineStringGeometryAccess	Sets coordinates of the point with the specified index.
setCoordinates(coordinates)	ILineStringGeometryAccess	Sets an array of geometry coordinates.
splice(index, number)	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.
unfreeze()	IFreezable	Switches the object to active mode. Inherited from IFreezable.

Events details

change

Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method:

- oldCoordinates — Old coordinates
- newCoordinates — New coordinates.

Methods details

get

```
{Number[]} get(index)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns coordinates of the point with the specified index.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a point.

* Mandatory parameter/option.

Example:

```
// Marking the start of the line with a placemark:
map.geoObjects.add(
    new ymaps.Placemark(polyline.geometry.get(0), { iconContent: 'A' })
);
```

getChildGeometry

```
{IPointGeometryAccess} getChildGeometry(index)
```

Creates and returns an **IPointGeometryAccess** object for the specified vertex on the polyline.

Returns the "Point" geometry object that corresponds to the specified endpoint.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of an endpoint.

* Mandatory parameter/option.

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the polyline closest to the anchorPosition.

Returns an object with the following fields:

- position — Point on the polyline closest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".
- closestPointIndex — Index of the vertex closest to "position".
- nextPointIndex — Index of the vertex that follows "position".
- prevPointIndex — Index of the vertex that precedes "position".

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the polyline vertexes.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest polyline vertex is calculated.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Example:

```
// Deleting points from the line when clicked:
myPolyline.events.add('click', function (e) {
    myPolyline.geometry.remove(
        myPolyline.geometry.getClosest(e.get('coords')).closestPointIndex
    );
});
```

getCoordinates

```
{Number[][]} getCoordinates()
```

Returns an array of geometry coordinates.

getLength

```
{Integer} getLength()
```

Returns the number of points in the geometry.

insert

```
{ILineStringGeometryAccess} insert(index, coordinates)
```

Adds a new point with the specified index.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a point.
coordinates*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

Example:

```
// Adding a new point to the end of the line when the map is clicked.
myMap.events.add('click', function (e) {
    myLineString.insert(myLineString.getLength(), e.get('coords'))
});
```

remove

```
{Number[]} remove(index)
```

Removes the point with the specified index.

Returns the coordinates of a deleted point.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a point.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

set

```
{ILineStringGeometryAccess} set(index, coordinates)
```

Sets coordinates of the point with the specified index.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of a point.
coordinates*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

setCoordinates

```
{ILineStringGeometryAccess} setCoordinates(coordinates)
```

Sets an array of geometry coordinates.

Returns self-reference.

Parameters:

Parameter	Default value	Description
coordinates*	—	Type: Number[][] Geometry coordinates.

* Mandatory parameter/option.

splice

```
{Number[][]} splice(index, number)
```

Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.

Returns an array of coordinates of deleted points.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer The index to start from for removing and adding points.
number*	—	Type: Integer The number of deleted points.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Example:

```
// Adding a new point to the start of the line when the map is clicked.
myMap.events.add('click', function (e) {
    myLineString.splice(0, 0, myLineString.getLength(), e.get('coords'))
});
```

IMapAction

Extends **IEventEmitter**.

Interface of an object that manages map movement.

Constructor | Fields | Events | Methods

Constructor

```
IMapAction()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
end	Event that notifies the map that movement has finished.
tick	Event that notifies the map of the next step. Contains the fields: • globalPixelCenter — The new map center, in global pixels. • zoom — The new map zoom. • duration — The time that is allowed for performing the step. • timingFunction — Function describing the type of movement.

Methods

Name	Description
begin(mapActionManager)	Starts the movement to be performed by the map. This method is called automatically by the map movement manager. From the moment when **IMapAction.begin** is called, the movement manager listens for **IMapAction.event:tick** and **IMapAction.event:end** and executes them.
end()	Stops movement.

Events details

end

Event that notifies the map that movement has finished.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

tick

Event that notifies the map of the next step. Contains the fields:

- globalPixelCenter — The new map center, in global pixels.
- zoom — The new map zoom.
- duration — The time that is allowed for performing the step.
- timingFunction — Function describing the type of movement.

Methods details

begin

```
{} begin(mapActionManager)
```

Starts the movement to be performed by the map. This method is called automatically by the map movement manager. From the moment when **IMapAction.begin** is called, the movement manager listens for **IMapAction.event:tick** and **IMapAction.event:end** and executes them.

Parameters:

Parameter	Default value	Description
mapActionManager*	—	Type: **map.action.Manager** The movement manager for the map that movement is being performed on.

* Mandatory parameter/option.

end

```
{} end()
```

Stops movement.

IMapObjectCollection

Extends **ICollection**, **ICustomizable**, **IParentOnMap**.

Collection of objects on the map.

Constructor | Fields | Events | Methods

Constructor

```
IMapObjectCollection()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
add	A child object was added. Inherited from ICollection.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
optionschange	Change to the object options. Inherited from ICustomizable.
remove	A child object was deleted. Inherited from ICollection.

Methods

Name	Returns	Description
add(object)	ICollection	Adds a child object to the collection. Inherited from ICollection.
getIterator()	IIterator	Returns iterator for the collection. Inherited from ICollection.
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
remove(object)	ICollection	Removes a child object from the collection. Inherited from ICollection.

IMapState

Object descriptor of the map state. Objects with this interface are returned by the YMapsML loader service.

See geoXml.load

Constructor | Methods

Constructor

```
IMapState()
```

Methods

Name	Returns	Description
applyToMap(map)	vow.Promise	Applies the state to the map that was passed.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

applyToMap

```
{vow.Promise} applyToMap(map)
```

Applies the state to the map that was passed.

Returns Promise object.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** Map.

* Mandatory parameter/option.

IMultiRouteModelJson

Interface of the model description object of a multi-stop route.

Constructor | Fields

Constructor

```
IMultiRouteModelJson()
```

Fields

Name	Type	Description
params	**IMultiRoutePa rams**	Routing options.
referencePoints	**IMultiRouteRe ferencePoint**[]	The description of the reference points on the multi-stop route. Also see the description of the parameter **IMultiRouteParams.viaIndexes**.

Fields details

params

```
{IMultiRouteParams} params
```

Routing options.

referencePoints

```
{IMultiRouteReferencePoint[]} referencePoints
```

The description of the reference points on the multi-stop route. Also see the description of the parameter **IMultiRouteParams.viaIndexes**.

IMultiRouteParams

Interface of the object that describes the format of the parameters of a multi-stop route model.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor | Fields

Constructor

```
IMultiRouteParams()
```

Fields

Name	Type	Description
avoidTrafficJams	Boolean	Allows to construct a multi-stop route with consideration for the current traffic. Default value: false.
boundedBy	Number[][]|null	Defines the area on the map where the objects being searched for are presumably located. This is used if the route points are set using the mailing address, and not coordinates. Default value: null.
requestSendInterval	String|Number	Time interval between requests to the router service. Can be set in milliseconds, otherwise the optimal value will be calculated automatically. Default value: "auto".
results	Integer	The maximum number of routes returned by the multi-stop router. Default value is 3.
reverseGeocoding	Boolean	Whether to use reverse geocoding for points specified as coordinates.
routingMode	String	Routing type. Accepts one of the two string values: • "auto" — Driving route. • "masstransit" — Routing using public transport. • "pedestrian" — Walking route. Default value: "auto".
searchCoordOrder	String	Determines how to interpret descriptions of the reference points that can be defined either as arrays of coordinates or as geometries. Can take one of two values: "longlat" or "latlong". The current value of the API's coordorder parameter is used by default.
strictBounds	Boolean	To search for objects only within the area specified by the boundedBy parameter. This is used if the route points are set using the mailing address, and not coordinates. Default value: false.
viaIndexes	Integer[]	Contains indexes of throughpoints in the set of reference points of the multi-stop route. The array is empty by default.

Fields details

avoidTrafficJams

```
{Boolean} avoidTrafficJams
```

Allows to construct a multi-stop route with consideration for the current traffic. Default value: false.

boundedBy

```
{Number[][]|null} boundedBy
```

Defines the area on the map where the objects being searched for are presumably located. This is used if the route points are set using the mailing address, and not coordinates. Default value: null.

requestSendInterval

```
{String|Number} requestSendInterval
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Time interval between requests to the router service. Can be set in milliseconds, otherwise the optimal value will be calculated automatically. Default value: "auto".

results

```
{Integer} results
```

The maximum number of routes returned by the multi-stop router. Default value is 3.

reverseGeocoding

```
{Boolean} reverseGeocoding
```

Whether to use reverse geocoding for points specified as coordinates.

routingMode

```
{String} routingMode
```

Routing type. Accepts one of the two string values:

- "auto" — Driving route.
- "masstransit" — Routing using public transport.
- "pedestrian" — Walking route.

Default value: "auto".

searchCoordOrder

```
{String} searchCoordOrder
```

Determines how to interpret descriptions of the reference points that can be defined either as arrays of coordinates or as geometries. Can take one of two values: "longlat" or "latlong". The current value of the API's coordorder parameter is used by default.

strictBounds

```
{Boolean} strictBounds
```

To search for objects only within the area specified by the boundedBy parameter. This is used if the route points are set using the mailing address, and not coordinates. Default value: false.

viaIndexes

```
{Integer[]} viaIndexes
```

Contains indexes of throughpoints in the set of reference points of the multi-stop route. The array is empty by default.

IMultiRouteReferencePoint

Interface of the object that describes the format of a multi-stop route reference point.

Constructor

Constructor

```
IMultiRouteReferencePoint()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Each reference point of a multi-stop route can be set using one of the following ways:

- A string containing the postal address of the reference point.
- An array containing the latitude and longitude of the reference point.
- A geometry.Point geometry describing the reference point.

Reference points on a multi-stop route can be either waypoints or throughpoints. A waypoint signifies a stop, and waypoints divide the route into so-called paths. A throughpoint is not a stop. Thus, when passing via a throughpoint, the segment of the route path won't break.

IMultiRouterRouteBalloon

Note:
The constructor of the IMultiRouterRouteBalloon class is hidden, as this class is not intended for autonomous initialization.

Interface of the model description object of a multi-stop route.

Events | Methods

Events

Name	Description
close	Closing the balloon.
open	Opening the balloon.

Methods

Name	Returns	Description
close()	**vow.Promise**	Closes route's balloon.
isOpen()	Boolean	Returns the info object state: open/ closed.
open([position])	**vow.Promise**	Opens route's balloon at closest position on route to specified location. Also makes route active.

Events details

close

Closing the balloon.

open

Opening the balloon.

Methods details

close

```
{vow.Promise} close()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Closes route's balloon.

Returns Promise object.

isOpen

```
{Boolean} isOpen()
```

Returns the info object state: open/closed.

open

```
{vow.Promise} open([position])
```

Opens route's balloon at closest position on route to specified location. Also makes route active.

Returns Promise object.

Parameters:

Parameter	Default value	Description
position	—	Type: Number[] The point where you want to place the balloon. By default balloon will open in the middle of the route.

IOptionManager

Extends **IChild**, **IEventEmitter**, **IFreezable**.

Interface for the options manager. The options manager lets you set option values, build an options inheritance hierarchy, and resolve option values in the context of the existing inheritance hierarchy.

Constructor | Fields | Events | Methods

Constructor

```
IOptionManager()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager for the object. Inherited from **IFreezable**.

Events

Name	Description
change	Changes to the option.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(key[, defaultValue])		Returns the value of the specified option in the context of the existing options inheritance hierarchy. When this method is called, first values are searched for in the current options manager, then, if the value is not defined, the search continues in the hierarchy of parent managers.
getAll()	Object	Returns a reference to the internal hash that stores option values.
getName()	String	Returns name of the options manager.
getNative(key)	Object	Returns the value of the specified option that is defined for the given level of the options hierarchy, i.e. in this manager.
getParent()	IOptionManager|null	Returns parent options manager.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from IFreezable.
resolve(key[, name])	Object	Method intended to be called by child options managers.
setName(name)		Sets the name of the options manager.
setParent(parent)	IChild	Sets the parent options manager.
unfreeze()	IFreezable	Switches the object to active mode. Inherited from IFreezable.

Events details

change

Changes to the option.

Methods details

get

```
{} get(key[, defaultValue])
```

Returns the value of the specified option in the context of the existing options inheritance hierarchy. When this method is called, first values are searched for in the current options manager, then, if the value is not defined, the search continues in the hierarchy of parent managers.

Parameters:

Parameter	Default value	Description
key*	—	Type: String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Name of the option.
defaultValue	—	Type: Object Default value.

* Mandatory parameter/option.

getAll

```
{Object} getAll()
```

Returns a reference to the internal hash that stores option values.

getName

```
{String} getName()
```

Returns name of the options manager.

getNative

```
{Object} getNative(key)
```

Returns the value of the specified option that is defined for the given level of the options hierarchy, i.e. in this manager.

Parameters:

Parameter	Default value	Description
key*	—	Type: String Name of the option.

* Mandatory parameter/option.

getParent

```
{IOptionManager|null} getParent()
```

Returns parent options manager.

resolve

```
{Object} resolve(key[, name])
```

Method intended to be called by child options managers.

Returns the option's value in the parent context.

Parameters:

Parameter	Default value	Description
key*	—	Type: String Name of the option.
name	—	Type: String Name of the child options manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

setName

```
{} setName(name)
```

Sets the name of the options manager.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Name of the options manager.

* Mandatory parameter/option.

setParent

```
{IChild} setParent(parent)
```

Sets the parent options manager.

Returns self-reference.

Parameters:

Parameter	Default value	Description
parent*	—	Type: **IOptionManager**|null Parent options manager.

* Mandatory parameter/option.

IOverlay

Extends **ICustomizable**, **IDomEventEmitter**.

Interface for an overlay.

Constructor | Fields | Events | Methods

Constructor

```
IOverlay()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty overlay flag. Instance of the Event class.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
shapechange	Change to the shape of the area spanning the overlay. Instance of the Event class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object.
getGeometry()	IPixelGeometry	Returns the current pixel geometry.
getMap()	Map|null	Returns reference to the current map.
getShape()	IShape|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.
setData(data)		Sets the overlay data.
setGeometry(geometry)		Sets the overlay pixel geometry.
setMap(map)		Sets the map on which to display the overlay.

Events details

datachange

Data change. Data fields:

- oldData — Old data.
- newData — New data.

emptinesschange

Change to the empty overlay flag. Instance of the **Event** class.

geometrychange

Changed geometry. Data fields:

- oldGeometry — Old pixel geometry.
- newGeometry — New pixel geometry.

mapchange

Map reference changed. Data fields:

- oldMap — Old map.
- newMap — New map.

shapechange

Change to the shape of the area spanning the overlay. Instance of the **Event** class.

Methods details

getData

```
{Object} getData()
```

Returns the overlay data object.

getGeometry

```
{IPixelGeometry} getGeometry()
```

Returns the current pixel geometry.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getMap

```
{Map|null} getMap()
```

Returns reference to the current map.

getShape

```
{IShape|null} getShape()
```

Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.

isEmpty

```
{Boolean} isEmpty()
```

Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.

setData

```
{} setData(data)
```

Sets the overlay data.

Parameters:

Parameter	Default value	Description
data*	—	Type: Object Overlay data.

* Mandatory parameter/option.

setGeometry

```
{} setGeometry(geometry)
```

Sets the overlay pixel geometry.

Parameters:

Parameter	Default value	Description
geometry*	—	Type: **IPixelGeometry** The geometry in global pixel coordinates.

* Mandatory parameter/option.

setMap

```
{} setMap(map)
```

Sets the map on which to display the overlay.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map**|null Reference to the map.

* Mandatory parameter/option.

IPane

Extends **IEventEmitter**.

Interface for a map pane. The map pane is an object that hosts its DOM element with a certain zIndex inside the map container. A pane can serve as a container to hold representations of various map elements, such as tiles, overlays, static map controls, etc. A pane can also act as a non-transparent screen for DOM events where you can listen to mouse events.

Constructor | Fields | Events | Methods

Constructor

```
IPane()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
overflowchange	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of **IEvent**.
zindexchange	Change to the zIndex value of a pane. Instance of **IEvent**.

Methods

Name	Returns	Description
destroy()		Destroys the pane.
getElement()	HTMLElement	Returns a reference to the pane's DOM container.
getMap()	**Map**	Returns the map that the pane belongs to.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getOverflow()	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: • "visible" — When you go off the map container, the content of the pane remains visible. • "hidden" — The viewport for the pane content is limited by the map container. .
getZIndex()	Number	Returns the zIndex of the pane.

Events details

overflowchange

Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of **IEvent**.

zindexchange

Change to the zIndex value of a pane. Instance of **IEvent**.

Methods details

destroy

```
{} destroy()
```

Destroys the pane.

getElement

```
{HTMLElement} getElement()
```

Returns a reference to the pane's DOM container.

getMap

```
{Map} getMap()
```

Returns the map that the pane belongs to.

getOverflow

```
{String} getOverflow()
```

Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values:

- "visible" — When you go off the map container, the content of the pane remains visible.
- "hidden" — The viewport for the pane content is limited by the map container.

.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getZIndex

```
{Number} getZIndex()
```

Returns the zIndex of the pane.

IPanorama

Interface for describing a panorama.

Constructor | Methods

Constructor

```
IPanorama()
```

Methods

Name	Returns	Description
getAngularBBox()	Number[]	Returns spherical coordinates that define the area covered by the image on the panoramic sphere. Coordinates are specified in the format `[thetaTop, phiRight, thetaBottom, phiLeft]` (the same as CSS).
getConnectionArrows()	**IPanoramaConnectionArrow**[]	Returns array of connection arrows on the panorama.
getConnectionMarkers()	**IPanoramaConnectionMarker**[]	Returns array of connection markers on the panorama.
getCoordSystem()	**ICoordSystem**	Returns the coordinate system that is used for defining positions of the panorama and all the associated markers and connections.
getDefaultDirection()	Number[]	Returns default direction of view. It will be used by the player when opening the panorama.
getDefaultSpan()	Number[]	Returns default size of the viewing area. It will be used by the player when opening the panorama.
getGraph()	**IPanoramaGraph**|null	Returns the graph of panoramas connected to the current panorama for making quick transitions.
getMarkers()	**IPanoramaMarker**[]	Returns array of markers on the panorama.
getName()	String	Returns the name of the panorama displayed by the player in the interface.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getPosition()	Number[]	Returns the location of the panorama in the coordinate system set in the options. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` – the height of the panorama in meters, relative to some level (not necessarily sea level).
getTileLevels()	IPanoramaTileLevel[]	Returns array of zoom levels for the panorama image.
getTileSize()	Number[]	Returns the size of tiles that the panorama image is divided into.

Methods details

getAngularBBox

```
{Number[]} getAngularBBox()
```

Returns spherical coordinates that define the area covered by the image on the panoramic sphere. Coordinates are specified in the format `[thetaTop, phiRight, thetaBottom, phiLeft]` (the same as CSS).

getConnectionArrows

```
{IPanoramaConnectionArrow[]} getConnectionArrows()
```

Returns array of connection arrows on the panorama.

getConnectionMarkers

```
{IPanoramaConnectionMarker[]} getConnectionMarkers()
```

Returns array of connection markers on the panorama.

getCoordSystem

```
{ICoordSystem} getCoordSystem()
```

Returns the coordinate system that is used for defining positions of the panorama and all the associated markers and connections.

getDefaultDirection

```
{Number[]} getDefaultDirection()
```

Returns default direction of view. It will be used by the player when opening the panorama.

getDefaultSpan

```
{Number[]} getDefaultSpan()
```

Returns default size of the viewing area. It will be used by the player when opening the panorama.

getGraph

```
{IPanoramaGraph|null} getGraph()
```

Returns the graph of panoramas connected to the current panorama for making quick transitions.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getMarkers

```
{IPanoramaMarker[]} getMarkers()
```

Returns array of markers on the panorama.

getName

```
{String} getName()
```

Returns the name of the panorama displayed by the player in the interface.

getPosition

```
{Number[]} getPosition()
```

Returns the location of the panorama in the coordinate system set in the options. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` – the height of the panorama in meters, relative to some level (not necessarily sea level).

getTileLevels

```
{IPanoramaTileLevel[]} getTileLevels()
```

Returns array of zoom levels for the panorama image.

getTileSize

```
{Number[]} getTileSize()
```

Returns the size of tiles that the panorama image is divided into.

IPanoramaConnection

Interface describing an entity on the panorama that connects it to another panorama.

Constructor | Methods

Constructor

```
IPanoramaConnection()
```

Methods

Name	Returns	Description
getConnectedPanorama()	vow.Promise	Returns the promise object that will be resolved by the connected panorama object or rejected with an error.

Methods details

getConnectedPanorama

```
{vow.Promise} getConnectedPanorama()
```

Returns the promise object that will be resolved by the connected panorama object or rejected with an error.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

IPanoramaConnectionArrow

Extends **IPanoramaConnection**.

Interface describing a connection to another panorama that's rendered as a clickable arrow.

Constructor | Fields | Methods

Constructor

```
IPanoramaConnectionArrow()
```

Fields

Name	Type	Description
properties	**data.Manager**	Additional properties of the panorama connection.

Methods

Name	Returns	Description
getConnectedPanorama()	**vow.Promise**	Returns the promise object that will be resolved by the connected panorama object or rejected with an error. Inherited from **IPanoramaConnection**.
getDirection()	Number[]	Returns the direction to the panorama indicated by the connection from the panorama it belongs to.
getPanorama()	**IPanorama**	Returns the panorama the connection belongs to.

Fields details

properties

```
{data.Manager} properties
```

Additional properties of the panorama connection.

Methods details

getDirection

```
{Number[]} getDirection()
```

Returns the direction to the panorama indicated by the connection from the panorama it belongs to.

getPanorama

```
{IPanorama} getPanorama()
```

Returns the panorama the connection belongs to.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

IPanoramaConnectionMarker

Extends **IPanoramaConnection**, **IPanoramaMarker**.

Interface that describes panorama connections.

Constructor | Fields | Methods

Constructor

```
IPanoramaConnectionMarker()
```

Fields

Name	Type	Description
properties	**data.Manager**	Additional marker properties. Inherited from **IPanoramaMarker**.

Methods

Name	Returns	Description
getConnectedPanorama()	**vow.Promise**	Returns the promise object that will be resolved by the connected panorama object or rejected with an error. Inherited from **IPanoramaConnection**.
getIconSet()	**vow.Promise**	Returns a promise object that will be resolved by an object that implements the interface **IPanoramaMarkerIconSet** and contains images for all the marker states. Inherited from **IPanoramaMarker**.
getPanorama()	**IPanorama**	Returns the panorama the marker belongs to. Inherited from **IPanoramaMarker**.
getPosition()	Number[]	Returns the marker's position in the coordinate system of the panorama that the graph containing the node belongs to. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` – the height of the marker in meters, set relative to the same level as the height of the panorama. Inherited from **IPanoramaMarker**.

IPanoramaGraph

Interface that describes panorama graphs.

Constructor | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IPanoramaGraph()
```

Methods

Name	Returns	Description
getEdges()	IPanoramaGraphEdge[]	Returns an array of graph edges.
getNodes()	IPanoramaGraphNode[]	Returns an array of graph nodes.
getPanorama()	IPanorama	Returns the panorama the graph belongs to.

Methods details

getEdges

```
{IPanoramaGraphEdge[]} getEdges()
```

Returns an array of graph edges.

getNodes

```
{IPanoramaGraphNode[]} getNodes()
```

Returns an array of graph nodes.

getPanorama

```
{IPanorama} getPanorama()
```

Returns the panorama the graph belongs to.

IPanoramaGraphEdge

Interface describing an edge of the panorama graph.

Constructor | Methods

Constructor

```
IPanoramaGraphEdge()
```

Methods

Name	Returns	Description
getEndNodes()	IPanoramaGraphNode[]	Returns an array of two nodes of the graph connected by an edge.

Methods details

getEndNodes

```
{IPanoramaGraphNode[]} getEndNodes()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns an array of two nodes of the graph connected by an edge.

IPanoramaGraphNode

Extends **IPanoramaConnection**.

Interface describing a node of the panorama graph.

Constructor | Methods

Constructor

```
IPanoramaGraphNode()
```

Methods

Name	Returns	Description
getConnectedPanorama()	**vow.Promise**	Returns the promise object that will be resolved by the connected panorama object or rejected with an error. Inherited from **IPanoramaConnection**.

IPanoramaMarker

Interface describing markers on the panorama.

Constructor | Fields | Methods

Constructor

```
IPanoramaMarker()
```

Fields

Name	Type	Description
properties	**data.Manager**	Additional marker properties.

Methods

Name	Returns	Description
getIconSet()	**vow.Promise**	Returns a promise object that will be resolved by an object that implements the interface **IPanoramaMarkerIconSet** and contains images for all the marker states.
getPanorama()	**IPanorama**	Returns the panorama the marker belongs to.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getPosition()	Number[]	Returns the marker's position in the coordinate system of the panorama that the graph containing the node belongs to. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` – the height of the marker in meters, set relative to the same level as the height of the panorama.

Fields details

properties

```
{data.Manager} properties
```

Additional marker properties.

Methods details

getIconSet

```
{vow.Promise} getIconSet()
```

Returns a promise object that will be resolved by an object that implements the interface IPanoramaMarkerIconSet and contains images for all the marker states.

getPanorama

```
{IPanorama} getPanorama()
```

Returns the panorama the marker belongs to.

getPosition

```
{Number[]} getPosition()
```

Returns the marker's position in the coordinate system of the panorama that the graph containing the node belongs to. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` – the height of the marker in meters, set relative to the same level as the height of the panorama.

IPanoramaMarkerIcon

Interface describing the marker's icon.

Constructor | Fields

Constructor

```
IPanoramaMarkerIcon()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
image	HTMLCanvasElement|HTMLImageElement	Icon image.
offset	Number[]	Offset of the top left corner of the icon in pixels, relative to the projection of the point that the marker is bound to.

Fields details

image

```
{HTMLCanvasElement|HTMLImageElement} image
```

Icon image.

offset

```
{Number[]} offset
```

Offset of the top left corner of the icon in pixels, relative to the projection of the point that the marker is bound to.

IPanoramaMarkerIconSet

Interface describing a set of marker icons.

Constructor | Fields

Constructor

```
IPanoramaMarkerIconSet()
```

Fields

Name	Type	Description
default	IPanoramaMarkerIcon|null	Default state icon.
expanded	IPanoramaMarkerIcon|null	Icon for the "active" marker state. The marker switches to this state when clicked; when clicked again, the marker becomes inactive.
expandedHovered	IPanoramaMarkerIcon|null	Icon for the "active highlighted" state. The marker switches to this state when it is active and the cursor is on it.
hovered	IPanoramaMarkerIcon|null	Icon for the "highlighted" marker state. The marker switches to this state when the cursor is on it.

Fields details

default

```
{IPanoramaMarkerIcon|null} default
```

Default state icon.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

expanded

```
{IPanoramaMarkerIcon|null} expanded
```

Icon for the "active" marker state. The marker switches to this state when clicked; when clicked again, the marker becomes inactive.

expandedHovered

```
{IPanoramaMarkerIcon|null} expandedHovered
```

Icon for the "active highlighted" state. The marker switches to this state when it is active and the cursor is on it.

hovered

```
{IPanoramaMarkerIcon|null} hovered
```

Icon for the "highlighted" marker state. The marker switches to this state when the cursor is on it.

IPanoramaTileLevel

Interface for describing zoom levels of the panorama image.

Constructor | Methods

Constructor

```
IPanoramaTileLevel()
```

Methods

Name	Returns	Description
getImageSize()	Number[]	Returns the size of the entire image in pixels.
getTileUrl(x, y)	String	Gets the URL of the tile from its position in the panorama image. The position is set by a pair of indexes: horizontal and vertical. The horizontal index starts from the left at zero and increases to the right. The vertical index starts from the top of the image and increases going down.

Methods details

getImageSize

```
{Number[]} getImageSize()
```

Returns the size of the entire image in pixels.

getTileUrl

```
{String} getTileUrl(x, y)
```

Gets the URL of the tile from its position in the panorama image. The position is set by a pair of indexes: horizontal and vertical. The horizontal index starts from the left at zero and increases to the right. The vertical index starts from the top of the image and increases going down.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns the URL of the tile with the passed indexes.

Parameters:

Parameter	Default value	Description
x*	—	Type: Number Horizontal index of the tile.
y*	—	Type: Number Vertical index of the tile.

* Mandatory parameter/option.

IParentOnMap

Interface for a parent object that belongs to a particular map object.

Constructor | Events | Methods

Constructor

```
IParentOnMap()
```

Events

Name	Description
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map.

Events details

mapchange

Map reference changed. Data fields:

- oldMap — Old map.
- newMap — New map.

Methods details

getMap

```
{Map} getMap()
```

Returns reference to the map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

IPixelCircleGeometry

Extends **IPixelGeometry**.

Interface for the "Circle" pixel geometry.

Constructor | Fields | Methods

Constructor

```
IPixelCircleGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getCoordinates()	Number[]	Returns coordinates of the center of the circle.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getRadius()	Number	Returns radius of the circle.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

Methods details

getCoordinates

```
{Number[]} getCoordinates()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns coordinates of the center of the circle.

getRadius

```
{Number} getRadius()
```

Returns radius of the circle.

IPixelGeometry

Extends **IBaseGeometry**.

Interface of a pixel geometry.

Constructor | Fields | Methods

Constructor

```
IPixelGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getMetaData()	Object	Returns metadata of the pixel geometry.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount.

Methods details

equals

```
{Boolean} equals(geometry)
```

Returns true if the passed geometry is equivalent to the given one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
geometry*	—	Type: IPixelGeometry The geometry to compare.

* Mandatory parameter/option.

getMetaData

```
{Object} getMetaData()
```

Returns metadata of the pixel geometry.

scale

```
{IPixelGeometry} scale(factor)
```

Creates a scaled copy of the geometry.

Returns a scaled copy of the geometry.

Parameters:

Parameter	Default value	Description
factor*	—	Type: Number Scaling factor.

* Mandatory parameter/option.

Example:

```
// Reducing the geometry to half its size
var smallCopy = myPixelGeometry.scale(0.5);
```

shift

```
{IPixelGeometry} shift(offset)
```

Creates a copy of the geometry that is shifted by the specified amount.

Returns a shifted copy of the geometry.

Parameters:

Parameter	Default value	Description
offset*	—	Type: Number[] Amount to shift on the axes.

* Mandatory parameter/option.

Example:

```
// Shifting all the geometry's coordinates 200 pixels to the left
var shifted = myPixelGeometry.shift([-200, 0]);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

IPixelLineStringGeometry

Extends **IPixelGeometry**.

Interface for the "Polyline" pixel geometry.

Constructor | Fields | Methods

Constructor

```
IPixelLineStringGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for a point on the polyline closest to the anchorPosition.
getCoordinates()	Number[][]	Returns coordinates of a line.
getLength()	Integer	Returns the number of points in the geometry.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the polyline closest to the anchorPosition.

Returns an object with the following fields:

- position — Point on the polyline closest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".
- closestPointIndex — Index of the vertex closest to "position".
- nextPointIndex — Index of the vertex that follows "position".
- prevPointIndex — Index of the vertex that precedes "position".

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the polyline vertexes.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest polyline vertex is calculated.

* Mandatory parameter/option.

getCoordinates

```
{Number[][]} getCoordinates()
```

Returns coordinates of a line.

getLength

```
{Integer} getLength()
```

Returns the number of points in the geometry.

IPixelMultiLineGeometry

Extends **IPixelGeometry**.

Interface for the "Multiline" pixel geometry.

Constructor | Fields | Methods

Constructor

```
IPixelMultiLineGeometry()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from IPixelGeometry.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from IBaseGeometry.
getClosest(anchorPosition)	Object	Searches for a point on the contour closest to the anchorPosition.
getCoordinates()	Number[][][]	Returns coordinates of a multiline.
getLength()	Integer	Returns the number of lines in the multiline.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from IPixelGeometry.
getType()	String	Returns ID of the geometry type. Inherited from IBaseGeometry.
scale(factor)	IPixelGeometry	Creates a scaled copy of the geometry. Inherited from IPixelGeometry.
shift(offset)	IPixelGeometry	Creates a copy of the geometry that is shifted by the specified amount. Inherited from IPixelGeometry.

Methods details

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the contour closest to the anchorPosition.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns an object with the following fields:

- position — The point on the multipolygon contour that is nearest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".
- closestPointIndex — Index of the multipolygon vertex closest to "position".
- nextPointIndex — Index of the multipolygon vertex that follows "position".
- prevPointIndex — Index of the multipolygon vertex that precedes "position".
- pathIndex — Index of the multipolygon contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the multipolygon vertexes.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest contour vertex is calculated.

* Mandatory parameter/option.

getCoordinates

```
{Number[][][]} getCoordinates()
```

Returns coordinates of a multiline.

getLength

```
{Integer} getLength()
```

Returns the number of lines in the multiline.

IPixelMultiPolygonGeometry

Extends **IPixelGeometry**.

Interface for the "Multipolygon" pixel geometry.

Constructor | Fields | Methods

Constructor

```
IPixelMultiPolygonGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the multipolygon.
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the multipolygon contour.
getCoordinates()	Number[][][][]	Returns coordinates of the multipolygon.
getFillRule()	String	Returns the string ID that defines the multipolygon fill rule. The ID can have one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it. .
getLength()	Integer	Returns the number of polygons in the multipolygon.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

Methods details

contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the multipolygon.

Returns indicator for whether the point belongs to the multipolygon.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the multipolygon contour.

Returns an object with the following fields:

- position — The point on the multipolygon contour that is nearest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".
- closestPointIndex — Index of the multipolygon vertex closest to "position".
- nextPointIndex — Index of the multipolygon vertex that follows "position".
- prevPointIndex — Index of the multipolygon vertex that precedes "position".
- pathIndex — Index of the multipolygon contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the multipolygon vertexes.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest vertex on the polygon contour is calculated.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getCoordinates

```
{Number[][][][]} getCoordinates()
```

Returns coordinates of the multipolygon.

getFillRule

```
{String} getFillRule()
```

Returns the string ID that defines the multipolygon fill rule. The ID can have one of two values:

- evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.
- nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.

.

getLength

```
{Integer} getLength()
```

Returns the number of polygons in the multipolygon.

IPixelPointGeometry

Extends **IPixelGeometry**.

Interface for the "Point" pixel geometry.

Constructor | Fields | Methods

Constructor

```
IPixelPointGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getCoordinates()	Number[]	Returns coordinates of a point.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

Methods details

getCoordinates

```
{Number[]} getCoordinates()
```

Returns coordinates of a point.

IPixelPolygonGeometry

Extends **IPixelGeometry**.

Interface for the "Polygon" pixel geometry.

Constructor | Fields | Methods

Constructor

```
IPixelPolygonGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the polygon.
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the polygon contour.
getCoordinates()	Number[][][]	Returns coordinates of the polygon.
getFillRule()	String	Returns string ID that defines the polygon fill rule. The ID accepts one of two values: • evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it. • nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it. .
getLength()	Integer	Returns the number of contours in the polygon.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

Methods details

contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the polygon.

Returns indicator for whether the point belongs to the polygon.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the polygon contour.

Returns an object with the following fields:

- position — The point on the polygon contour that is nearest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".
- closestPointIndex — Index of the polygon vertex closest to "position".
- nextPointIndex — Index of the polygon vertex that follows "position".
- prevPointIndex — Index of the polygon vertex that precedes "position".
- pathIndex — Index of the polygon contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the polygon vertexes.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest vertex on the polygon contour is calculated.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getCoordinates

```
{Number[][][]} getCoordinates()
```

Returns coordinates of the polygon.

getFillRule

```
{String} getFillRule()
```

Returns string ID that defines the polygon fill rule. The ID accepts one of two values:

- evenOdd — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.
- nonZero — Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.

.

getLength

```
{Integer} getLength()
```

Returns the number of contours in the polygon.

IPixelRectangleGeometry

Extends **IPixelGeometry**.

Interface for the "Rectangle" pixel geometry.

Constructor | Fields | Methods

Constructor

```
IPixelRectangleGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Methods

Name	Returns	Description
equals(geometry)	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from **IPixelGeometry**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IBaseGeometry**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the rectangle.
getCoordinates()	Number[][]	Returns coordinates of two opposite corners of the rectangle.
getMetaData()	Object	Returns metadata of the pixel geometry. Inherited from **IPixelGeometry**.
getType()	String	Returns ID of the geometry type. Inherited from **IBaseGeometry**.
scale(factor)	**IPixelGeometry**	Creates a scaled copy of the geometry. Inherited from **IPixelGeometry**.
shift(offset)	**IPixelGeometry**	Creates a copy of the geometry that is shifted by the specified amount. Inherited from **IPixelGeometry**.

Methods details

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the rectangle.

Returns an object with the following fields:

- position — The point on the rectangle that is nearest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".
- closestPointIndex — Index of the rectangle vertex closest to "position".
- nextPointIndex — Index of the rectangle vertex that follows "position".
- prevPointIndex — Index of the rectangle vertex that precedes "position".
- pathIndex — Index of the rectangle contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the rectangle vertexes.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest rectangle vertex is calculated.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getCoordinates

```
{Number[][]} getCoordinates()
```

Returns coordinates of two opposite corners of the rectangle.

IPointGeometry

Extends **IGeometry**, **IPointGeometryAccess**.

Interface for the "Point" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IPointGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
change	Changed coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. Inherited from **IPointGeometryAccess**.
mapchange	Map reference changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldMap — Old map. • newMap — New map. Inherited from **IGeometry**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
pixelgeometrychange	The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelGeometry — New **IPixelGeometry** pixel geometry. Inherited from **IGeometry**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from IGeometry.
getCoordinates()	Number[]|null	Returns coordinates of a point. Inherited from IPointGeometryAccess.
getMap()	Map|null	Returns the current map. Inherited from IGeometry.
getPixelGeometry([options])	IPixelGeometry	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from IGeometry.
getType()	String	Returns the "Point" string.
setCoordinates(coordinates)	IPointGeometryAccess	Sets coordinates of a point. Inherited from IPointGeometryAccess.
setMap(map)		Sets the map. Inherited from IGeometry.

Methods details

getType

```
{String} getType()
```

Returns the "Point" string.

IPointGeometryAccess

Interface for access to the "Point" geometry.

Constructor | Events | Methods

Constructor

```
IPointGeometryAccess()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates.

Methods

Name	Returns	Description
getCoordinates()	Number[]|null	Returns coordinates of a point.
setCoordinates(coordinates)	IPointGeometryAccess	Sets coordinates of a point.

Events details

change

Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method:

- oldCoordinates — Old coordinates
- newCoordinates — New coordinates.

Methods details

getCoordinates

```
{Number[]|null} getCoordinates()
```

Returns coordinates of a point.

setCoordinates

```
{IPointGeometryAccess} setCoordinates(coordinates)
```

Sets coordinates of a point.

Returns self-reference.

Parameters:

Parameter	Default value	Description
coordinates*	—	Type: Number[]|null Coordinates of a point.

* Mandatory parameter/option.

IPolygonGeometry

Extends IGeometry, IPolygonGeometryAccess.

Interface for the "Polygon" geometry.

Constructor | Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IPolygonGeometry()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. • oldFillRule — Old fill rule. • newFillRule — New fill rule. Inherited from IPolygonGeometryAccess.
mapchange	Map reference changed. Instance of the Event class. Names of fields that are available via the Event.get method: • oldMap — Old map. • newMap — New map. Inherited from IGeometry.
optionschange	Change to the object options. Inherited from ICustomizable.
pixelgeometrychange	The pixel geometry changed. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelGeometry — New IPixelGeometry pixel geometry. Inherited from IGeometry.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the polygon. Inherited from IPolygonGeometryAccess.
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(index)	Number[][]	Returns coordinates of the contour with the specified index.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IPolygonGeometryAccess**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getChildGeometry(index)	**ILinearRingGeometryAccess**	Creates and returns an **ILinearRingGeometryAccess** object for the specified contour. Inherited from **IPolygonGeometryAccess**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the polygon contour. Inherited from **IPolygonGeometryAccess**.
getCoordinates()	Number[][][]	Returns an array of geometry coordinates. Inherited from **IPolygonGeometryAccess**.
getFillRule()	String	Returns ID of the fill rule. Inherited from **IPolygonGeometryAccess**.
getLength()	Integer	Returns the number of contours in the geometry. Inherited from **IPolygonGeometryAccess**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getType()	String	Returns the "Polygon" string.
insert(index, path)	**IPolygonGeometryAccess**	Adds a new contour with the specified index. Inherited from **IPolygonGeometryAccess**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
remove(index)	**ILinearRingGeometryAccess**	Removes the contour with the specified index. Inherited from **IPolygonGeometryAccess**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
set(index, path)	**IPolygonGeometryAccess**	Sets coordinates of the contour with the specified index. Inherited from **IPolygonGeometryAccess**.
setCoordinates(coordinates)	**IPolygonGeometryAccess**	Sets an array of geometry coordinates. Inherited from **IPolygonGeometryAccess**.
setFillRule(fillRule)	**IPolygonGeometryAccess**	Sets the polygon's fill rule. Inherited from **IPolygonGeometryAccess**.
setMap(map)		Sets the map. Inherited from **IGeometry**.
splice(index, number)	**ILinearRingGeometryAccess**[]	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter. Inherited from **IPolygonGeometryAccess**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Methods details

getType

```
{String} getType()
```

Returns the "Polygon" string.

IPolygonGeometryAccess

Extends **IFreezable**.

Interface for accessing the "Polygon" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IPolygonGeometryAccess()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager for the object. Inherited from **IFreezable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
change	Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates • newCoordinates — New coordinates. • oldFillRule — Old fill rule. • newFillRule — New fill rule.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the polygon.
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
get(index)	Number[][]	Returns coordinates of the contour with the specified index.
getChildGeometry(index)	ILinearRingGeometryAccess	Creates and returns an ILinearRingGeometryAccess object for the specified contour.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the polygon contour.
getCoordinates()	Number[][][]	Returns an array of geometry coordinates.
getFillRule()	String	Returns ID of the fill rule.
getLength()	Integer	Returns the number of contours in the geometry.
insert(index, path)	IPolygonGeometryAccess	Adds a new contour with the specified index.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from IFreezable.
remove(index)	ILinearRingGeometryAccess	Removes the contour with the specified index.
set(index, path)	IPolygonGeometryAccess	Sets coordinates of the contour with the specified index.
setCoordinates(coordinates)	IPolygonGeometryAccess	Sets an array of geometry coordinates.
setFillRule(fillRule)	IPolygonGeometryAccess	Sets the polygon's fill rule.
splice(index, number)	ILinearRingGeometryAccess[]	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
unfreeze()	IFreezable	Switches the object to active mode. Inherited from IFreezable.

Events details

change

Changed coordinates. Instance of the Event class. Names of fields that are available via the Event.get method:

- oldCoordinates — Old coordinates
- newCoordinates — New coordinates.
- oldFillRule — Old fill rule.
- newFillRule — New fill rule.

Methods details

contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the polygon.

Returns indicator for whether the point belongs to the polygon.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

Example:

```
var myPolygon = new ymaps.geometry.Polygon([
        [[69, 45], [68, 55], [86, 56]]
    ]);

// This method only works with a map that is set correctly.
myPolygon.options.setParent(myMap.options);
myPolygon.geometry.setMap(myMap);

// Checking whether the click point is inside the polygon with the geometry set
above.
myMap.events.add('click', function (e) {
    alert(myPolygon.geometry.contains(e.get('coords')) ? 'Hit!' : 'Miss!');
});
```

get

```
{Number[][]} get(index)
```

Returns coordinates of the contour with the specified index.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Contour index.

* Mandatory parameter/option.

getChildGeometry

```
{ILinearRingGeometryAccess} getChildGeometry(index)
```

Creates and returns an **ILinearRingGeometryAccess** object for the specified contour.

Returns the geometry object that corresponds to the specified contour.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Contour index.

* Mandatory parameter/option.

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the polygon contour.

Returns an object with the following fields:

- position — The point on the polygon contour that is nearest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".
- closestPointIndex — Index of the polygon vertex closest to "position".
- nextPointIndex — Index of the polygon vertex that follows "position".
- prevPointIndex — Index of the polygon vertex that precedes "position".
- pathIndex — Index of the polygon contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the polygon vertexes.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest vertex on the polygon contour is calculated.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
var myPolygon = new ymaps.Polygon([
        [[69, 45], [68, 55], [86, 56], [87, 43]],
        [[60, 48], [62, 60], [80, 62], [80, 48]]
    ]),
    myPoint = new ymaps.Placemark([74, 52]),
    // Constructing the shortest normal from the polygon contour to the point:
    normalVec = new ymaps.Polyline([

myPolygon.geometry.getClosest(myPoint.geometry.getCoordinates()).position,
        myPoint.geometry.getCoordinates()
    ]));

myMap.geoObjects
    .add(myPolygon)
    .add(myPoint),
    .add(normalVec);
```

getCoordinates

```
{Number[][][]} getCoordinates()
```

Returns an array of geometry coordinates.

getFillRule

```
{String} getFillRule()
```

Returns ID of the fill rule.

Example:

```
var myPolygon = new ymaps.Polygon([
        [[69, 45], [68, 55], [86, 56], [87, 43]],
        [[60, 48], [62, 60], [80, 62], [80, 48]]
    ]),
    middlePoint = [74, 52];

myMap.geoObjects.add(myPolygon);

// Checking a point that falls on the intersection of two contours
// by default (with the value fillRule == 'evenOdd') forms an opening (hole).
alert(myPolygon.geometry.contains(middlePoint)); // =&gt; false

// After setting the value to 'nonZero', all intersections are also "filled in",
// so now the point is in the polygon.
myPolygon.geometry.setFillRule('nonZero');
alert(myPolygon.geometry.contains(middlePoint)); // =&gt; true
```

getLength

```
{Integer} getLength()
```

Returns the number of contours in the geometry.

insert

```
{IPolygonGeometryAccess} insert(index, path)
```

Adds a new contour with the specified index.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Contour index.
path*	—	Type: Number[][] Contour coordinates.

* Mandatory parameter/option.

remove

```
{ILinearRingGeometryAccess} remove(index)
```

Removes the contour with the specified index.

Returns the deleted contour.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Contour index.

* Mandatory parameter/option.

set

```
{IPolygonGeometryAccess} set(index, path)
```

Sets coordinates of the contour with the specified index.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Contour index.
path*	—	Type: Number[][] Contour coordinates.

* Mandatory parameter/option.

setCoordinates

```
{IPolygonGeometryAccess} setCoordinates(coordinates)
```

Sets an array of geometry coordinates.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
coordinates*	—	Type: Number[][][] Geometry coordinates.

* Mandatory parameter/option.

setFillRule

```
{IPolygonGeometryAccess} setFillRule(fillRule)
```

Sets the polygon's fill rule.

Returns self-reference.

Parameters:

Parameter	Default value	Description
fillRule*	—	Type: String ID of the fill rule.

* Mandatory parameter/option.

splice

```
{ILinearRingGeometryAccess[]} splice(index, number)
```

Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter.

Returns the deleted contours.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer The index to start from for removing and adding contours.
number*	—	Type: Integer The number of contours to be deleted.

* Mandatory parameter/option.

IPopup

Extends **ICustomizable**, **IEventEmitter**.

Interface for an info object.

Constructor | Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
IPopup()
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
close	Closing the info object.
open	Opening the info object.
optionschange	Change to the object options. Inherited from ICustomizable.

Methods

Name	Returns	Description
close([force])	vow.Promise	Closes the info object.
getData()		Returns info object data.
getOverlay()	vow.Promise	Returns the promise object to return the overlay.
getOverlaySync()	IOverlay	Returns the overlay, if one exists.
getPosition()		Returns the coordinates of the info object.
isOpen()	Boolean	Returns the info object state: open/closed.
open([position[, data]])	vow.Promise	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the IProjection that is in the options.
setData(data)	vow.Promise	Defines new data for the info object.
setPosition(position)	vow.Promise	Specifies a new position for the info object.

Events details

close

Closing the info object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

open

Opening the info object.

Methods details

close

```
{vow.Promise} close([force])
```

Closes the info object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
force	false	Type: Boolean Instant closure.

getData

```
{} getData()
```

Returns info object data.

getOverlay

```
{vow.Promise} getOverlay()
```

Returns the promise object to return the overlay.

getOverlaySync

```
{IOverlay} getOverlaySync()
```

Returns the overlay, if one exists.

getPosition

```
{} getPosition()
```

Returns the coordinates of the info object.

isOpen

```
{Boolean} isOpen()
```

Returns the info object state: open/closed.

open

```
{vow.Promise} open([position[, data]])
```

Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the **IProjection** that is in the options.

Returns Promise object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
position	—	Type: Number[] The point where you want to place the balloon.
data	—	Type: Object|String|HTMLElement Overlay data.

setData

```
{vow.Promise} setData(data)
```

Defines new data for the info object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
data*	—	Type: Object|String|HTMLElement Info object data.

* Mandatory parameter/option.

setPosition

```
{vow.Promise} setPosition(position)
```

Specifies a new position for the info object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] The coordinates of the info object.

* Mandatory parameter/option.

IPopupManager

Extends **IEventEmitter**.

Interface for the info object manager.

Constructor | Fields | Events | Methods

Constructor

```
IPopupManager()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
close	Closing the info object. Names of fields available via Event.get: • target — Reference to the object where the closing occurred.
open	Opening the info object. Names of fields available via Event.get: • target — Reference to the object where the opening occurred.

Methods

Name	Returns	Description
close([force])	vow.Promise	Closes the info object.
destroy()		Disables the info object manager.
getData()	Object|null	Returns the data of the info object or 'null'.
getOptions()	IOptionManager|null	Returns the options manager or 'null'.
getOverlay()	vow.Promise	Returns the promise object to return the overlay.
getOverlaySync()	IOverlay|null	Returns the overlay, if one exists.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'.
isOpen()	Boolean	Returns the info object state: open/closed.
open([position[, data[, options]]])	vow.Promise	Opens the info object at the specified position.
setData(data)	vow.Promise	Defines new data for the info object.
setOptions(options)	vow.Promise	Defines new options for the info object.
setPosition(position)	vow.Promise	Specifies a new position for the info object.

Events details

close

Closing the info object. Names of fields available via Event.get:

- target — Reference to the object where the closing occurred.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

open

Opening the info object. Names of fields available via **Event.get**:

- target — Reference to the object where the opening occurred.

Methods details

close

```
{vow.Promise} close([force])
```

Closes the info object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
force	false	Type: Boolean Instant closure.

destroy

```
{} destroy()
```

Disables the info object manager.

getData

```
{Object|null} getData()
```

Returns the data of the info object or 'null'.

getOptions

```
{IOptionManager|null} getOptions()
```

Returns the options manager or 'null'.

getOverlay

```
{vow.Promise} getOverlay()
```

Returns the promise object to return the overlay.

getOverlaySync

```
{IOverlay|null} getOverlaySync()
```

Returns the overlay, if one exists.

getPosition

```
{Number[]|null} getPosition()
```

Returns the coordinates of the info object or 'null'.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

isOpen

```
{Boolean} isOpen()
```

Returns the info object state: open/closed.

open

```
{vow.Promise} open([position[, data[, options]]])
```

Opens the info object at the specified position.

Returns Promise object.

Parameters:

Parameter	Default value	Description
position	—	Type: Number[] Coordinates of the point where the hint is opened.
data	—	Type: Object|String|HTMLElement Data.
options	—	Type: Object Options.

setData

```
{vow.Promise} setData(data)
```

Defines new data for the info object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
data*	—	Type: Object|String|HTMLElement Info object data.

* Mandatory parameter/option.

setOptions

```
{vow.Promise} setOptions(options)
```

Defines new options for the info object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
options*	—	Type: Object Info object options.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

setPosition

```
{vow.Promise} setPosition(position)
```

Specifies a new position for the info object.

Returns Promise object.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] The coordinates of the info object.

* Mandatory parameter/option.

IPositioningContext

Interface for the positioning context, an object that allows to position an object inside itself that is defined by global pixel coordinates.

Constructor | Methods

Constructor

```
IPositioningContext()
```

Methods

Name	Returns	Description
fromClientPixels(clientPixelPoint)	Number[]	Converts client pixel coordinates to global coordinates.
getZoom()	Number	Returns the current zoom level at which the positioning context works.
toClientPixels(globalPixelPoint)	Number[]	Converts global pixel coordinates to client coordinates.

Methods details

fromClientPixels

```
{Number[]} fromClientPixels(clientPixelPoint)
```

Converts client pixel coordinates to global coordinates.

Returns global pixel coordinates.

Parameters:

Parameter	Default value	Description
clientPixelPoint*	—	Type: Number[] Client pixel coordinates.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getZoom

```
{Number} getZoom()
```

Returns the current zoom level at which the positioning context works.

toClientPixels

```
{Number[]} toClientPixels(globalPixelPoint)
```

Converts global pixel coordinates to client coordinates.

Returns client pixel coordinates.

Parameters:

Parameter	Default value	Description
globalPixelPoint*	—	Type: Number[] Global pixel coordinates.

* Mandatory parameter/option.

IProjection

Projection. Describes how the real map is projected onto the endless pixel plane. One "world" should be sized 256x256 pixels at the zero zoom level, while the upper left corner of this world has the coordinates (0,0) and the axes coordinates go to the right and down. "Worlds" can be connected along any axis (or both axes at once).

Constructor | Methods

Constructor

```
IProjection()
```

Methods

Name	Returns	Description
fromGlobalPixels(globalPixelPoint, zoom)	Number[]	Converts pixel coordinates to the projection's coordinates at the specified zoom level.
getCoordSystem()	**ICoordSystem**	Returns the coordinate system that is used by the projection.
isCycled()	Boolean[]	Indicator of projection cycling.
toGlobalPixels(coordPoint, zoom)	Number[]	Converts projection coordinates to global pixel coordinates at the specified zoom level.

Methods details

fromGlobalPixels

```
{Number[]} fromGlobalPixels(globalPixelPoint, zoom)
```

Converts pixel coordinates to the projection's coordinates at the specified zoom level.

Returns a point in the projection's coordinates.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
globalPixelPoint*	—	Type: Number[] A point in pixel coordinates.
zoom*	—	Type: Number Zoom level.

* Mandatory parameter/option.

getCoordSystem

```
{ICoordSystem} getCoordSystem()
```

Returns the coordinate system that is used by the projection.

isCycled

```
{Boolean[]} isCycled()
```

Indicator of projection cycling.

Returns a pair of flags that show whether the map is looped along the pixel axes (x/y).

toGlobalPixels

```
{Number[]} toGlobalPixels(coordPoint, zoom)
```

Converts projection coordinates to global pixel coordinates at the specified zoom level.

Returns a pair of pixel coordinates.

Parameters:

Parameter	Default value	Description
coordPoint*	—	Type: Number[] A point in the projection's coordinates.
zoom*	—	Type: Number Zoom level.

* Mandatory parameter/option.

IPromiseProvider

Object fulfilling the promise.

Constructor | Methods

Constructor

```
IPromiseProvider()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
then(onResolve, onReject)	IPromiseProvider	Returns a self-reference or a new Promise object.

Methods details

then

```
{IPromiseProvider} then(onResolve, onReject)
```

Returns a self-reference or a new Promise object.

Parameters:

Parameter	Default value	Description
onResolve*	—	Type: Function Handler function that is called if the promise was fulfilled.
onReject*	—	Type: Function Handler function that is called if the promise was not fulfilled (an error occurred).

* Mandatory parameter/option.

IRatioMap

Interface of an object containing the ratio of custom data to devicePixelRatio. Used when needed to support screens with a ratio of virtual to physical pixels greater than one. As keys for the object, use strings consisting of whole numbers or fractions indicating the pixel density coefficient.

Constructor

Constructor

```
IRatioMap()
```

Example:

```
// Let's say we need to display the image correctly.
var images = {
    // For regular screens we'll display the normal picture.
    "1": "100x100.png",
    // For HTC Desire, Samsung Galaxy S II and others with devicePixelRatio =
1.5.
    "1.5": "150x150.png",
    // For Apple devices with the Retina screen, as well as for Sony Xperia S,
HTC One X and others
    // with the ratio = 2.
    "2": "200x100.png"
}
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

IRectangleGeometry

Extends **IGeometry**, **IRectangleGeometryAccess**.

Interface for the "Rectangle" geometry.

Constructor | Fields | Events | Methods

Constructor

```
IRectangleGeometry()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
change	Change to corner coordinates. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCoordinates — Old coordinates of the corners. • newCoordinates — New coordinates of the corners. Inherited from **IRectangleGeometryAccess**.
mapchange	Map reference changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldMap — Old map. • newMap — New map. Inherited from **IGeometry**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
pixelgeometrychange	The pixel geometry changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelGeometry — New **IPixelGeometry** pixel geometry. Inherited from **IGeometry**.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the rectangle. Inherited from **IRectangleGeometryAccess**.
freeze()	**IFreezable**	Switches the object to "frozen" mode.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IFreezable**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner. Inherited from **IGeometry**.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the rectangle contour. Inherited from **IRectangleGeometryAccess**.
getCoordinates()	Number[][]	Returns coordinates of two opposite corners of the rectangle. Inherited from **IRectangleGeometryAccess**.
getMap()	**Map**|null	Returns the current map. Inherited from **IGeometry**.
getPixelGeometry([options])	**IPixelGeometry**	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from **IGeometry**.
getType()	String	Returns the "Rectangle" string.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
setCoordinates(coordinates)	**IRectangleGeometryAccess**	Sets the coordinates of two opposite corners of the rectangle. Inherited from **IRectangleGeometryAccess**.
setMap(map)		Sets the map. Inherited from **IGeometry**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.

Methods details

getType

```
{String} getType()
```

Returns the "Rectangle" string.

IRectangleGeometryAccess

Extends **IFreezable**.

Interface for accessing the "Rectangle" geometry.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor | Fields | Events | Methods

Constructor

```
IRectangleGeometryAccess()
```

Fields

Name	Type	Description
events	IEventManager	Event manager for the object. Inherited from IFreezable.

Events

Name	Description
change	Change to corner coordinates. Instance of the Event class. Names of fields that are available via the Event.get method: • oldCoordinates — Old coordinates of the corners. • newCoordinates — New coordinates of the corners.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the rectangle.
freeze()	IFreezable	Switches the object to "frozen" mode. Inherited from IFreezable.
getClosest(anchorPosition)	Object	Searches for the point nearest to "anchorPosition" on the rectangle contour.
getCoordinates()	Number[][]	Returns coordinates of two opposite corners of the rectangle.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from IFreezable.
setCoordinates(coordinates)	IRectangleGeometryAccess	Sets the coordinates of two opposite corners of the rectangle.
unfreeze()	IFreezable	Switches the object to active mode. Inherited from IFreezable.

Events details

change

Change to corner coordinates. Instance of the Event class. Names of fields that are available via the Event.get method:

- oldCoordinates — Old coordinates of the corners.
- newCoordinates — New coordinates of the corners.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the rectangle.

Returns indicator for whether the point belongs to the rectangle.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the rectangle contour.

Returns an object with the following fields:

- position — The point on the rectangle contour that is nearest to "anchorPosition".
- distance — Distance from "anchorPosition" to "position".

.

Parameters:

Parameter	Default value	Description
anchorPosition*	—	Type: Number[] Coordinates of a point for which the nearest rectangle vertex is calculated.

* Mandatory parameter/option.

getCoordinates

```
{Number[][]} getCoordinates()
```

Returns coordinates of two opposite corners of the rectangle.

setCoordinates

```
{IRectangleGeometryAccess} setCoordinates(coordinates)
```

Sets the coordinates of two opposite corners of the rectangle.

Returns self-reference.

Parameters:

Parameter	Default value	Description
coordinates*	—	Type: Number[][]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Coordinates of corners.

* Mandatory parameter/option.

IRoutePanel

Extends **IEventEmitter**.

Interface for route panel.

Constructor | Fields | Events | Methods

Constructor

```
IRoutePanel()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Option manager. Names of options: • allowSwitch: Boolean = true — Whether button for switching way points should be shown. • reverseGeocoding: Boolean = true — Whether reverse geocoding should be enabled during routing. • adjustMapMargin: Boolean = false — Whether the panel registers its size in the map margins manager **map.margin.Manager**. • types: Object = { auto: true, masstransit: true, pedestrian: true } — Specifies routing modes available for user to select in routing panel. When set, state.type is automatically adjusted, if current state.type becomes unavailable. Types are shown in panel only if two or more are available for user to select.
state	**IDataManager**	State manager. Names of states: • type: String — Routing type **IMultiRouteParams.routingMode**. • fromEnabled: Boolean — Enables the "from" field for users to enter the route origin. • from: String — Address or coordinates of departure. • toEnabled: Boolean — Enables the "to" field for users to enter the route destination. • to: String — Address or coordinates of arrival.

Events

Name	Description
disable	Route panel is unloaded.
enable	Route panel and it's dependencies are loaded and ready for user interactions.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
enable()		Loads panel dependencies and enables it for usage.
geolocate(name)	vow.Promise	Use user's geolocation as coordinates for 'from' or 'to'.
getRoute()	multiRouter.MultiRoute	**Attention!** This method is deprecated. See IRoutePanel.getRouteAsync. Returns built route.
getRouteAsync()	vow.Promise.<multiRouter.MultiRoute>	Returns vow.Promise, which will be resolved with built route. If an error occurs, the promise object is rejected.
isEnable()	Boolean	Returns whether panel is fully loaded.
switchPoints()		Switches points (and corresponding inputs).

Fields details

options

```
{IOptionManager} options
```

Option manager. Names of options:

- allowSwitch: Boolean = true — Whether button for switching way points should be shown.
- reverseGeocoding: Boolean = true — Whether reverse geocoding should be enabled during routing.
- adjustMapMargin: Boolean = false — Whether the panel registers its size in the map margins manager map.margin.Manager.
- types: Object = { auto: true, masstransit: true, pedestrian: true } — Specifies routing modes available for user to select in routing panel. When set, state.type is automatically adjusted, if current state.type becomes unavailable. Types are shown in panel only if two or more are available for user to select.

state

```
{IDataManager} state
```

State manager. Names of states:

- type: String — Routing type IMultiRouteParams.routingMode.
- fromEnabled: Boolean — Enables the "from" field for users to enter the route origin.
- from: String — Address or coordinates of departure.
- toEnabled: Boolean — Enables the "to" field for users to enter the route destination.
- to: String — Address or coordinates of arrival.

Events details

disable

Route panel is unloaded.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

enable

Route panel and it's dependencies are loaded and ready for user interactions.

Methods details

enable

```
{} enable()
```

Loads panel dependencies and enables it for usage.

geolocate

```
{vow.Promise} geolocate(name)
```

Use user's geolocation as coordinates for 'from' or 'to'.

Returns Promise object. See **geolocation**.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Input to geolocate. Either 'from' or 'to'.

* Mandatory parameter/option.

getRoute

```
{multiRouter.MultiRoute} getRoute()
```

This method is deprecated. See **IRoutePanel.getRouteAsync**.

Returns built route.

getRouteAsync

```
{vow.Promise.<multiRouter.MultiRoute>} getRouteAsync()
```

Returns **vow.Promise** which:

- will be **resolved** with: <**multiRouter.MultiRoute**> — built route;
- either **rejected** with an error.

isEnable

```
{Boolean} isEnable()
```

Returns whether panel is fully loaded.

switchPoints

```
{} switchPoints()
```

Switches points (and corresponding inputs).

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

ISearchControlLayout

Extends **IExpandableControlLayout**.

Interface for the layout of the "Search on map" control.

Constructor | Fields | Events | Methods

Constructor

```
ISearchControlLayout()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
collapse	Event that initiates collapsing an object. Inherited from **IExpandableControlLayout**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty layout indicator. Instance of the **Event** class. Inherited from **ILayout**.
expand	Event that initiates expanding an object. Inherited from **IExpandableControlLayout**.
loadmore	The event that triggers pulling up additional search results. Instance of the **Event** class.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
parentelementchange	Change to the parent element. Instance of the **Event** class. Inherited from **ILayout**.
resultselect	Event that initiates showing the search results on the map. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index — Number of the object in the server response to show on the map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
search	Event that initiates searching for objects. Instance of the Event class. Names of fields that are available via the Event.get method: • request — String containing the request.
shapechange	Change to the shape of the area spanning the layout. Instance of the Event class. Inherited from ILayout.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getData()	Object	Returns layout data object. Inherited from ILayout.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from ILayout.
getShape()	IShape|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from ILayout.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from ILayout.
setData(data)		Sets layout data. Inherited from ILayout.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from ILayout.

Events details

loadmore

The event that triggers pulling up additional search results. Instance of the Event class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

resultselect

Event that initiates showing the search results on the map. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- index — Number of the object in the server response to show on the map.

search

Event that initiates searching for objects. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- request — String containing the request.

ISearchProvider

Search provider interface.

Constructor | Methods

Constructor

```
ISearchProvider()
```

Methods

Name	Returns	Description
search(request[, options])	**vow.Promise**	Sends a request to search for geo objects. A handler function for processing geocoding results can be added via the returned promise object. The object input to the handler function can contain only the following types of fields: geoObjects, layers, mapState, and metaData.
suggest(request[, options])	**vow.Promise**	Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }. The displayName field represents the toponym in a user-friendly way, and the value field represents the value which should be inserted into the search field after the user selects the suggestion. This method is optional.

Methods details

search

```
{vow.Promise} search(request[, options])
```

Sends a request to search for geo objects. A handler function for processing geocoding results can be added via the returned promise object. The object input to the handler function can contain only the following types of fields: geoObjects, layers, mapState, and metaData.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns Promise object.

Parameters:

Parameter	Default value	Description
request*	—	Type: String Request string.
options	—	Type: Object Options.
options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
options.results	—	Type: Number Maximum number of results to be returned.
options.skip	—	Type: Number Number of results that must be skipped.

* Mandatory parameter/option.

suggest

```
{vow.Promise} suggest(request[, options])
```

Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }. The displayName field represents the toponym in a user-friendly way, and the value field represents the value which should be inserted into the search field after the user selects the suggestion.

This method is optional.

Returns a Promise object.

Parameters:

Parameter	Default value	Description
request*	—	Type: String Request string.
options	—	Type: Object Options.
options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
options.results	—	Type: Number

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Maximum number of results to be returned.
options.strictBounds	—	Type: Boolean Search only inside the area defined by the "boundedBy" option.

* Mandatory parameter/option.

ISelectableControl

Extends **IControl**.

Interface for a control that can be toggled and selected.

Constructor | Fields | Events | Methods

Constructor

```
ISelectableControl()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **IControl**.

Events

Name	Description
deselect	The control is not selected.
disable	The control is unavailable.
enable	The control is available.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
select	The control is selected.

Methods

Name	Returns	Description
deselect()		Cancels selection of the control (turns it off).

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
disable()		Makes the control unavailable (user actions are not allowed).
enable()		Makes the control available (user actions are allowed).
getParent()	IControlParent|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IControl.
isEnabled()	Boolean	Returns true if the control is available, or false if it is unavailable.
isSelected()	Boolean	Returns true if the control is selected, or false if it is not selected.
select()		Selects (turns on) the control.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IControl.

Events details

deselect

The control is not selected.

disable

The control is unavailable.

enable

The control is available.

select

The control is selected.

Methods details

deselect

```
{} deselect()
```

Cancels selection of the control (turns it off).

disable

```
{} disable()
```

Makes the control unavailable (user actions are not allowed).

enable

```
{} enable()
```

Makes the control available (user actions are allowed).

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

isEnabled

```
{Boolean} isEnabled()
```

Returns true if the control is available, or false if it is unavailable.

isSelected

```
{Boolean} isSelected()
```

Returns true if the control is selected, or false if it is not selected.

select

```
{} select()
```

Selects (turns on) the control.

ISelectableControlLayout

Extends **ILayout**.

Interface for the button layout.

Constructor | Fields | Events | Methods

Constructor

```
ISelectableControlLayout()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
deselect	Event that initiates removing an item from the "selected" state.
emptinesschange	Change to the empty layout indicator. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **ILayout**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
parentelementchange	Change to the parent element. Instance of the Event class. Inherited from ILayout.
select	Event that initiates putting an item in the "selected" state.
shapechange	Change to the shape of the area spanning the layout. Instance of the Event class. Inherited from ILayout.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getData()	Object	Returns layout data object. Inherited from ILayout.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from ILayout.
getShape()	IShape|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from ILayout.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from ILayout.
setData(data)		Sets layout data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **ILayout**.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from **ILayout**.

Events details

deselect

Event that initiates removing an item from the "selected" state.

select

Event that initiates putting an item in the "selected" state.

IShape

Interface of a shape. A shape represents a collection of pixel geometry as well as mathematical and logical parameters for rendering it (such as the presence of a contour and its width, filling, etc.)

Constructor | Methods

Constructor

```
IShape()
```

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the shape.
equals(shape)	Boolean	Returns true if the passed shape is equivalent to the given one.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.
getGeometry()	**IPixelGeometry**	Returns pixel geometry of a shape.
getType()	String	Returns ID of the shape type.
scale(factor)	**IShape**	Creates a scaled copy of the shape.
shift(offset)	**IShape**	Creates a copy of the shape that is shifted by the specified amount.

Methods details

contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the shape.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns true if the passed point is inside the shape.

Parameters:

Parameter	Default value	Description
position*	—	Type: Number[] Coordinates of a point.

* Mandatory parameter/option.

equals

```
{Boolean} equals(shape)
```

Returns true if the passed shape is equivalent to the given one.

Parameters:

Parameter	Default value	Description
shape*	—	Type: **IShape** The shape to compare.

* Mandatory parameter/option.

getBounds

```
{Number[][]|null} getBounds()
```

Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.

getGeometry

```
{IPixelGeometry} getGeometry()
```

Returns pixel geometry of a shape.

getType

```
{String} getType()
```

Returns ID of the shape type.

scale

```
{IShape} scale(factor)
```

Creates a scaled copy of the shape.

Returns a scaled copy of the shape.

Parameters:

Parameter	Default value	Description
factor*	—	Type: Number Scaling factor.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

shift

```
{IShape} shift(offset)
```

Creates a copy of the shape that is shifted by the specified amount.

Returns the shifted copy of the shape.

Parameters:

Parameter	Default value	Description
offset*	—	Type: Number[] Amount to shift on the axes.

* Mandatory parameter/option.

ISuggestProvider

Interface for a provider of search suggestions.

Constructor | Methods

Constructor

```
ISuggestProvider()
```

Methods

Name	Returns	Description
suggest(request[, options])	**vow.Promise**	Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi", hl: [[0,5]] }. The displayName field is responsible for representing a toponym in a user-friendly format. The value field is the value that must be inserted in the data entry field after the user selects this suggestion. The hl field is an array of ranges for highlighting to show which part of the result matched the query. The range for highlighting is an array of two numbers: the indexes of the starting and ending symbols of the range.

Methods details

suggest

```
{vow.Promise} suggest(request[, options])
```

Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi", hl: [[0,5]] }. The displayName field is responsible for representing a toponym in a user-friendly format. The value field is the value that must be inserted in the data entry field after the user selects this suggestion.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

The hl field is an array of ranges for highlighting to show which part of the result matched the query. The range for highlighting is an array of two numbers: the indexes of the starting and ending symbols of the range.

Returns a Promise object.

Parameters:

Parameter	Default value	Description
request*	—	Type: String Request string.
options	—	Type: Object Options.
options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
options.results	—	Type: Number Maximum number of results to be returned.

* Mandatory parameter/option.

ISuggestViewLayout

Interface for the layout of the search suggestions panel.

Constructor | Events

Constructor

```
ISuggestViewLayout()
```

Events

Name	Description
hover	Event that occurs when the user selects a suggestion, either by hovering over the result with the mouse or using the keyboard. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index: Number|null — Index of an element in the list of search suggestions.
select	Event that occurs when the user selects a suggestion. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • item: Object — Object with the "displayName" and "value" fields.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

hover

Event that occurs when the user selects a suggestion, either by hovering over the result with the mouse or using the keyboard. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- index: Number|null — Index of an element in the list of search suggestions.

select

Event that occurs when the user selects a suggestion. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- item: Object — Object with the "displayName" and "value" fields.

ITrafficControlLayout

Extends **IExpandableControlLayout**.

Interface for the layout of the "Traffic panel" control.

Constructor | Fields | Events | Methods

Constructor

```
ITrafficControlLayout()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
collapse	Event that initiates collapsing an object. Inherited from **IExpandableControlLayout**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty layout indicator. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **ILayout**.
expand	Event that initiates expanding an object. Inherited from **IExpandableControlLayout**.
hide	Event that initiates deleting traffic from the map. Hide traffic.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
parentelementchange	Change to the parent element. Instance of the Event class. Inherited from ILayout.
providerkeychange	Event that initiates changing the provider key. Instance of the Event class. Names of fields that are available via the Event.get method: • newProviderKey — New value for the provider key. • oldProviderKey — Old key value.
shapechange	Change to the shape of the area spanning the layout. Instance of the Event class. Inherited from ILayout.
show	Event that initiates showing traffic on the map.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getData()	Object	Returns layout data object. Inherited from ILayout.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from ILayout.
getShape()	IShape|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from ILayout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **ILayout**.
setData(data)		Sets layout data. Inherited from **ILayout**.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from **ILayout**.

Events details

hide

Event that initiates deleting traffic from the map. Hide traffic.

providerkeychange

Event that initiates changing the provider key. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- newProviderKey — New value for the provider key.
- oldProviderKey — Old key value.

show

Event that initiates showing traffic on the map.

ITrafficProvider

Extends **ICustomizable**, **IEventEmitter**.

Interface for a provider of traffic data.

Constructor | Fields | Events | Methods

Constructor

```
ITrafficProvider()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.

Methods

Name	Returns	Description
getMap()	Map|null	Returns reference to the map.
setMap(Reference)		

Methods details

getMap

```
{Map|null} getMap()
```

Returns reference to the map.

setMap

```
{} setMap(Reference)
```

Parameters:

Parameter	Default value	Description
Reference*	—	Type: Map|null to the map.

* Mandatory parameter/option.

ITransportProperties

This class does not have a constructor and is intended for describing the data object of the vehicle in the transport segment of a public transport route.

Constructor | Fields

Constructor

```
ITransportProperties()
```

Fields

Name	Type	Description
id	String	Transport vehicle identifier.
name	String	Name of the transport vehicle's route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
type	String	Transport vehicle type identifier. Accepts one of the following string values: • "bus" — Bus. • "trolleybus" — Trolley. • "tramway" — Tram. • "minibus" — Minibus. • "underground" — Subway. • "suburban" — Commuter train.

Fields details

id

```
{String} id
```

Transport vehicle identifier.

name

```
{String} name
```

Name of the transport vehicle's route.

type

```
{String} type
```

Transport vehicle type identifier. Accepts one of the following string values:

- "bus" — Bus.
- "trolleybus" — Trolley.
- "tramway" — Tram.
- "minibus" — Minibus.
- "underground" — Subway.
- "suburban" — Commuter train.

IZoomControlLayout

Extends **ILayout**.

Interface for the layout of the "Zoom slider" control.

Constructor | Fields | Events | Methods

Constructor

```
IZoomControlLayout()
```

Fields

Name	Type	Description
events	IEventManager	Event manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
		Inherited from **IDomEventEmitter**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty layout indicator. Instance of the **Event** class. Inherited from **ILayout**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
parentelementchange	Change to the parent element. Instance of the Event class. Inherited from ILayout.
shapechange	Change to the shape of the area spanning the layout. Instance of the Event class. Inherited from ILayout.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
zoomchange	Event that initiates changing the map zoom level. Change the map zoom level. Instance of the Event class. Names of fields that are available via the Event.get method: • newZoom — New zoom level value. • oldZoom — Old zoom level value.

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getData()	Object	Returns layout data object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **ILayout**.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from **ILayout**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from **ILayout**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **ILayout**.
setData(data)		Sets layout data. Inherited from **ILayout**.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from **ILayout**.

Events details

zoomchange

Event that initiates changing the map zoom level. Change the map zoom level. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- newZoom — New zoom level value.
- oldZoom — Old zoom level value.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

layer

layer.storage

Static object.

Instance of **util.Storage**

Storage for layers.

Methods

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

layer.tile

layer.tile.CanvasTile

Extends **ICanvasTile**.

Image canvas tile. Can draw the specified image via the drawImage method for the 2d context of the canvas element.

Constructor | Fields | Events | Methods

Constructor

```
layer.tile.CanvasTile(url[, options[, renderOptions]])
```

Creates an image canvas tile. Accessible in the storage for tile classes by the key "default#canvas".

Parameters:

Parameter	Default value	Description
`url` *	—	Type: String URL of the image.
`options`	—	Type: Object Options.
`options.notFoundTile`	null	Type: String|null

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.
options.tileAnimationDuration	—	Type: Number Duration of animation of tile transparency. The default value depends on the browser.
renderOptions	—	Type: Object Rendering parameters.
renderOptions.tileNumber	—	Type: Number[]
renderOptions.tileZoom	—	Type: Number

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Tile event manager. Inherited from **ICanvasTile**.

Events

Name	Description
ready	Tile ready event. Inherited from **ICanvasTile**.

Methods

Name	Returns	Description
destroy()		Destroys the tile. Inherited from **ICanvasTile**.
isReady()	Boolean	Checks tile readiness. Inherited from **ICanvasTile**.
renderAt(context, canvasSize, bounds[, animate])		Draws an image tile in the canvas object's 2d context. Inherited from **ICanvasTile**.

layer.tile.DomTile

Extends **IDomTile**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Image tile. Can draw the specified image via the CSS "background" property of the DOM element.

Constructor | Fields | Events | Methods

Constructor

```
layer.tile.DomTile(url[, options[, renderOptions]])
```

Creates an image DOM tile. Accessible in the storage for tile classes by the key "default#dom". Creates an image DOM tile.

Parameters:

Parameter	Default value	Description
url *	—	Type: String URL of the image.
options	—	Type: Object Options.
options.notFoundTile	null	Type: String|null Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.
options.tileAnimationDuration	—	Type: Number Duration of animation of image transparency when drawing, in ms (only applies in browsers that support CSS Transition for the "opacity" property). The default value depends on the browser.
renderOptions	—	Type: Object Rendering parameters.
renderOptions.tileNumber	—	Type: Number[]
renderOptions.tileZoom	—	Type: Number

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Tile event manager. Inherited from **IDomTile**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
ready	Tile ready event. Inherited from IDomTile.

Methods

Name	Returns	Description
destroy()		Destroys the tile. Inherited from IDomTile.
isReady()	Boolean	Checks tile readiness. Inherited from IDomTile.
renderAt(context, clientBounds, animate)		Adds a tile to the parent HTML element. Inherited from IDomTile.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

layer.tileContainer

layer.tileContainer.CanvasContainer

Extends **IChildOnMap**.

Container for tiles on canvas.

Constructor | Fields | Events | Methods

Constructor

```
layer.tileContainer.CanvasContainer(layer[, options])
```

Creates a container for tiles on canvas. Accessible in the storage for tile container classes by the key "default#canvas".

Parameters:

Parameter	Default value	Description
layer *	—	Type: **ILayer** Layer.
options	—	Type: Object Container options.
options.notFoundTile	null	Type: String|null Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.
options.tileClass	'default#canvas'	Type: **ICanvasTile** Class of tiles used by the container. Must implement the interface **ICanvasTile**.
options.tileTransparent	false	Type: Boolean Flag showing whether container tiles are transparent.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getTile(tileNumber, tileZoom, priority)	ICanvasTile	Factory function for creating tiles.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Methods details

getMap

```
{Map} getMap()
```

Returns reference to the map.

getTile

```
{ICanvasTile} getTile(tileNumber, tileZoom, priority)
```

Factory function for creating tiles.

Returns tile instance.

Parameters:

Parameter	Default value	Description
tileNumber*	—	Type: Number[] Tile number.
tileZoom*	—	Type: Number Tile scale.
priority*	—	Type: Number Loading priority.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

layer.tileContainer.DomContainer

Extends **IChildOnMap**.

Container for tiles of the IDomTile type.

Constructor | Fields | Events | Methods

Constructor

```
layer.tileContainer.DomContainer(layer[, options])
```

Creates a container for DOM tiles. Accessible in the storage for tile container classes by the key "default#dom".

Parameters:

Parameter	Default value	Description
layer *	—	Type: **ILayer** Layer.
options	—	Type: Object Container options.
options.notFoundTile	null	Type: String|null Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.
options.tileClass	'default#dom'	Type: **IDomTile** Class of tiles used by the container. Must implement the interface **IDomTile**.
options.tileTransparent	false	Type: Boolean Flag showing whether container tiles are transparent.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
ready	Ready event for all tiles.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getTile(tileNumber, tileZoom, priority)	IDomTile	Factory function for creating tiles.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Events details

ready

Ready event for all tiles.

Methods details

getMap

```
{Map} getMap()
```

Returns reference to the map.

getTile

```
{IDomTile} getTile(tileNumber, tileZoom, priority)
```

Factory function for creating tiles.

Returns tile instance.

Parameters:

Parameter	Default value	Description
tileNumber*	—	Type: Number[] Tile number.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
tileZoom*	—	Type: Number Tile scale.
priority*	—	Type: Number Loading priority.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Layer

Extends **ILayer**, **IParentOnMap**, **IPositioningContext**.

Tile layer. Allows to display a layer consisting of tiles on the map.

Constructor | Fields | Events | Methods

Constructor

```
Layer(tileUrlTemplate[, options])
```

Parameters:

Parameter	Default value	Description
tileUrlTemplate *	—	Type: String|Function String template for the tile URL, or a function that generates the tile URL. For the string template, the following substitutions are supported: • %c is replaced with x=number[0]&y=number[1]&z=zoom level. • %x is replaced with number[0]. • %y is replaced with number[1]. • %z is replaced with the zoom level. • %l is replaced with lang=language. • %d or %d|n is replaced with a number from 1 to n, depending on the tile number; n is the number of domains. Used for distributing the load over multiple domains. For n, specify a factor of two (2, 4, 16, and so on). If the template has %d, then n=4. The template function receives three input parameters: • tileNumber — Array of two numbers, the tile numbers on x and y. • tileZoom — Zoom level. • Returns a URL string.
options	—	Type: Object Options.
options.brightness	0.5	Type: Number Layer brightness. Specified as a number from 0 to 1. 0 corresponds to black, and 1 to white.
options.notFoundTile	null	Type: String|null

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.
options.pane	'ground'	Type: **IPane**|String Pointer to the layer pane or key from **map.pane.Manager**.
options.projection	—	Type: Object Layer projection.
options.tileSize	[256, 256]	Type: Number[] Size of tiles on the layer.
options.tileTransparent	false	Type: Boolean Flag showing whether layer tiles are transparent.
options.zIndex	constants.zIndex.layer	Type: Number Z-index of the layer in the layers container.

* Mandatory parameter/option.

Example:

```
// Adds an OSM layer to the map.
map.layers.add(new ymaps.Layer(
'http://tile.openstreetmap.org/%z/%x/%y.png', {
    projection: ymaps.projection.sphericalMercator
}));
map.copyrights.add('&amp;copy; OpenStreetMap contributors, CC-BY-SA');
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
brightnesschange	Layer brightness change event. Inherited from **ILayer**.
copyrightschange	Event for changes to available copyright information. Inherited from **ILayer**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
zoomrangechange	Event for changes to available information about the zoom level range. Inherited from **ILayer**.

Methods

Name	Returns	Description
clientPixelsToNumber(clientPixelPoint, tileZoom)	Number[]	Returns the number of the tile that the specified point falls on for the specified tile zoom level.
fromClientPixels(clientPixelPoint)	Number[]	Converts client pixel coordinates to global coordinates. Inherited from **IPositioningContext**.
getBrightness()	Number	Optional method. Inherited from **ILayer**.
getCopyrights(coords, zoom)	**vow.Promise**	Optional method. Requests information about copyrights at the specified point with the specified zoom. Inherited from **ILayer**.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getPane()	**IPane**	Returns the container that the layer is located in.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getTileSize(zoom)	Number[]	Returns the horizontal and vertical tile dimensions for the specified zoom level.
getTileUrl(tileNumber, tileZoom)	String|null	Returns the tile URL by its number and zoom level, or null if there is no data for the requested section.
getTileUrlTemplate()	String|Function	Returns string template for the tile URL, or a function that generates it.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getZoom()	Number	Returns the current zoom level at which the positioning context works. Inherited from **IPositioningContext**.
getZoomRange(point)	**vow.Promise**	Optional method. Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers — the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error. Inherited from **ILayer**.
numberToClientBounds(tileNumber, tileZoom)	Number[][]	Converts the tile number and zoom level to the area occupied by the tile in client coordinates of the parent container.
restrict(number, tileZoom)	Integer[]|null	Applies restrictions to the visible area for tiles (including map cycling on the x and y axes).
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.
setTileUrlTemplate(tileUrlTemplate)		
toClientPixels(globalPixelPoint)	Number[]	Converts global pixel coordinates to client coordinates. Inherited from **IPositioningContext**.
update()		Deletes the old tiles and requests new ones.

Methods details

clientPixelsToNumber

```
{Number[]} clientPixelsToNumber(clientPixelPoint, tileZoom)
```

Returns the number of the tile that the specified point falls on for the specified tile zoom level.

Parameters:

Parameter	Default value	Description
`clientPixelPoint`*	—	Type: Number A point in client pixel coordinates.
`tileZoom`*	—	Type: Number Tile zoom level.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getPane

```
{IPane} getPane()
```

Returns the container that the layer is located in.

getTileSize

```
{Number[]} getTileSize(zoom)
```

Returns the horizontal and vertical tile dimensions for the specified zoom level.

Parameters:

Parameter	Default value	Description
zoom*	—	Type: Number Zoom value.

* Mandatory parameter/option.

Example:

```
// Show tiles for a larger zoom level,
// stretched to twice their size up to 512x512 pixels.
// For example, to reduce traffic.
var layer = new ymaps.Layer('', {
    projection: ymaps.projection.sphericalMercator
});
layer.getTileUrl = function (tileNumber, zoom) {
    return [
        'http://tile.openstreetmap.org',
        Math.max(zoom - 1, 0), tileNumber[0], tileNumber[1]
    ].join('/') + '.png';
}
layer.getTileSize = function (zoom) {
    if (zoom == 0) {
        return [256, 256];
    }
    return [512, 512];
}
map.copyrights.add('&amp;copy; OpenStreetMap contributors, CC-BY-SA');
```

getTileUrl

```
{String|null} getTileUrl(tileNumber, tileZoom)
```

Returns the tile URL by its number and zoom level, or null if there is no data for the requested section.

Parameters:

Parameter	Default value	Description
tileNumber*	—	Type:
tileZoom*	—	Type:

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Defines the function for generating the tile URL.
var layer = new ymaps.Layer('');
layer.getTileUrl = function (tileNumber, zoom) {
    return [
        'http://tile.openstreetmap.org',
        zoom, tileNumber[0], tileNumber[1]
    ].join('/') + '.png';
}
```

getTileUrlTemplate

```
{String|Function} getTileUrlTemplate()
```

Returns string template for the tile URL, or a function that generates it.

numberToClientBounds

```
{Number[][]} numberToClientBounds(tileNumber, tileZoom)
```

Converts the tile number and zoom level to the area occupied by the tile in client coordinates of the parent container.

Returns the area in client pixel coordinates.

Parameters:

Parameter	Default value	Description
tileNumber*	—	Type: Integer[] Tile number.
tileZoom*	—	Type: Integer Tile zoom level.

* Mandatory parameter/option.

restrict

```
{Integer[]|null} restrict(number, tileZoom)
```

Applies restrictions to the visible area for tiles (including map cycling on the x and y axes).

Returns the new tile number calculated with restrictions, or null if the tile is not in the visible area.

Parameters:

Parameter	Default value	Description
number*	—	Type: Integer[] Tile number.
tileZoom*	—	Type: Integer Tile zoom level.

* Mandatory parameter/option.

setTileUrlTemplate

```
{} setTileUrlTemplate(tileUrlTemplate)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
tileUrlTemplate*	—	Type: String|Function String template for the tile URL, or a function that generates it.

* Mandatory parameter/option.

update

```
{} update()
```

Deletes the old tiles and requests new ones.

Parameters:

Parameter	Default value	Description
updateBounds*	—	Type:

* Mandatory parameter/option.

LayerCollection

Extends **ILayer**, **IMapObjectCollection**.

Collection of layers.

Constructor | Fields | Events | Methods

Constructor

```
LayerCollection([options])
```

Collection of layers.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Layer options.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
add	A child object was added. Inherited from **ICollection**.
brightnesschange	Layer brightness change event. Inherited from **ILayer**.
copyrightschange	Event for changes to available copyright information. Inherited from **ILayer**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
remove	A child object was deleted. Inherited from **ICollection**.
zoomrangechange	Event for changes to available information about the zoom level range. Inherited from **ILayer**.

Methods

Name	Returns	Description
add(child)	**LayerCollection**	Adds a child object to the collection.
each(callback[, context])		Iterates through all the items in the collection and calls a handler function for each of them.
getBrightness()	Number	Returns layer brightness as a number from 0 to 1.
getCopyrights([coords[, zoom]])	**vow.Promise**	Requests information about copyrights at the specified point with the specified zoom. If the point and zoom are omitted, it uses the map center and zoom.
getIterator()	**IIterator**	Returns iterator for the collection. Inherited from **ICollection**.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getZoomRange([coords])	vow.Promise	Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers — the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error. If the collection does not have a single descendant providing information about the zoom level range, the promise will be rejected with the "noProvider" message.
remove(child)	LayerCollection	Removes a child object from the collection.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Methods details

add

```
{LayerCollection} add(child)
```

Adds a child object to the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
child*	—	Type: ILayer|String Layer to add (key from layer.storage or an instance of the ILayer class).

* Mandatory parameter/option.

Example:

```
// Let's say we want to add several layers to our collection.
var layerCollection = new ymaps.LayerCollection(),
    customLayer = new ymaps.Layer(
        'http://tile.openstreetmap.org/%z/%x/%y.png', {
        projection: ymaps.projection.sphericalMercator
    }),
    // We can use a key from layer.storage to set the layer.
    satelliteLayer = 'yandex#satellite';
// Adding layers to our collection.
layerCollection
    .add(customLayer)
    .add(satelliteLayer);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

each

```
{} each(callback[, context])
```

Iterates through all the items in the collection and calls a handler function for each of them.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function.
context	—	Type: Object Context for the function.

* Mandatory parameter/option.

getBrightness

```
{Number} getBrightness()
```

Returns layer brightness as a number from 0 to 1.

getCopyrights

```
{vow.Promise} getCopyrights([coords[, zoom]])
```

Requests information about copyrights at the specified point with the specified zoom. If the point and zoom are omitted, it uses the map center and zoom.

Returns a Promise object that will be resolved and will pass as a result an array of strings or DOM elements with information about copyrights.

Parameters:

Parameter	Default value	Description
coords	—	Type: Number[] The point on the map that copyright information is being requested for.
zoom	—	Type: Number The zoom level that copyright information is being requested for.

Example:

```
// Let's say we have a service that can return copyrights
// using the coordinates and zoom level
myLayer.getCopyrights = function (coords, zoom) {
    var deferred = ymaps.vow.defer();
    $.ajax('url/to/copyrights/provider?ll=' +
        (coords || map.getCenter()).join(',') + '&z=' +
        (zoom || map.getZoom()),
    function (res) {
        deferred.resolve(res || []);
    });
    return deferred.promise();
};
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getZoomRange

```
{vow.Promise} getZoomRange([coords])
```

Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers — the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error. If the collection does not have a single descendant providing information about the zoom level range, the promise will be rejected with the "noProvider" message.

Returns Promise object.

Parameters:

Parameter	Default value	Description
coords	—	Type: Number[] Coordinates of a point. If omitted, the current map center is used.

Example:

```
// Assuming that our layer was drawn for the 2-15 zoom level for the entire
earth
myLayer.getZoomRange = function () {
    return ymaps.vow.resolve([2, 15]);
}
```

remove

```
{LayerCollection} remove(child)
```

Removes a child object from the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
child*	—	Type: ILayer|String Layer to delete (key string from layer.storage or an instance of the ILayer class).

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

layout

layout.Image

Extends **ILayout**.

Class for creating layouts containing a picture.

Constructor | Fields | Events | Methods

Constructor

```
layout.Image(data)
```

Creates an instance of the picture layout.

Parameters:

Parameter	Default value	Description
data *	—	Type: **ILayout** Layout data.
data.options	—	Type: **ILayout** Layout options.
data.options.imageClipRect	—	Type: Number[][] Coordinates of the display area of the original image, in pixels.
data.options.imageHref	—	Type: String URL of the image file.
data.options.imageOffset	—	Type: Number[] Offset of the image relative to the anchor point.
data.options.imageSize	—	Type: Number[] Dimensions of the image layer.
data.options.shape	—	Type: **IShape**|Object|null The hotspot shape. Can be set as an instance of a class that implements the IShape interface or a JSON description of the pixel geometry of the icon. If not set, the rectangular shape based on the size and offset of the icon will be calculated automatically. The coordinates of the figure geometry are counted from the anchor point.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Creating a round placemark with a 20-pixel radius.
var placemark = new ymaps.Placemark([59.936952, 30.343334], null, {
        iconLayout: 'default#image',
        iconImageHref: './images/roundImage.png',
        iconImageSize: [40, 40],
        iconImageOffset: [-20, -20],
        // Defining a hotspot on top of the map.
        iconShape: {
            type: 'Circle',
            coordinates: [0, 0],
            radius: 20
        },
    });
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty layout indicator. Instance of the Event class. Inherited from ILayout.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
parentelementchange	Change to the parent element. Instance of the **Event** class. Inherited from **ILayout**.
shapechange	Change to the shape of the area spanning the layout. Instance of the **Event** class. Inherited from **ILayout**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from **ILayout**.
getData()	Object	Returns layout data object. Inherited from **ILayout**.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from **ILayout**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from **ILayout**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **ILayout**.
setData(data)		Sets layout data. Inherited from **ILayout**.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from **ILayout**.

layout.ImageWithContent

Extends **layout.Image**.

Class for creating layouts consisting of an image and content.

Constructor | Fields | Events | Methods

Constructor

```
layout.ImageWithContent(data)
```

Creates an instance of the layout image with content.

Parameters:

Parameter	Default value	Description
data *	—	Type: **ILayout** Layout data.
data.options	—	Type: **ILayout** Layout options.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
data.options.contentLayout	—	Type: Function|String Content layout. (Type: constructor for an object with the ILayout interface, or its key in the storage).
data.options.contentOffset	—	Type: Number[] Offset of the layer with content relative to the layer with the image.
data.options.contentSize	—	Type: Number[] Dimensions of the content layer.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty layout indicator. Instance of the Event class. Inherited from ILayout.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
parentelementchange	Change to the parent element. Instance of the **Event** class. Inherited from **ILayout**.
shapechange	Change to the shape of the area spanning the layout. Instance of the **Event** class. Inherited from **ILayout**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getData()	Object	Returns layout data object. Inherited from ILayout.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from ILayout.
getShape()	IShape|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from ILayout.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from ILayout.
setData(data)		Sets layout data. Inherited from ILayout.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from ILayout.

layout.PieChart

Extends **layout.templateBased.Base**.

Pie chart layout. Available in the layout storage by the key 'default#pieChart'. The layout can be used as a tool for visualizing any data, or in combination with other visual API components such as placemarks, the clusterer, and the objects manager.

Note:
Because diagrams are drawn using SVG technology, this layout doesn't work in browsers that don't support SVG, including IE8.

See Placemark Clusterer ObjectManager RemoteObjectManager LoadingObjectManager

Constructor | Fields | Events | Methods

Constructor

```
layout.PieChart(data)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
data *	—	Type: Object Layout data.
data.option.pieChartRadius	25 + 2 * Math.log(sum)	Type: Number|Function Radius of the diagram, in pixels. Set as a number, or as a function that accepts the layout's "properties" and "options" parameters and returns the diagram radius as a number. By default, the radius is defined as 25 + 2 * Math.log(sum) pixels, where sum is the total weight of the sectors.
data.options	—	Type: **IOptionManager** Options for rendering the layout.
data.options.pieChartCaptionMaxWidth	200	Type: Number Maximum size of the label (iconCaption), in pixels.
data.options.pieChartCoreFillStyle	white	Type: String Fill style for the core. Set as a string that encodes the fill color.
data.options.pieChartCoreRadius	pieChartRadius — 15	Type: Number|Function The radius of the central part of the layout, where the content is displayed. Set in the same way as pieChartRadius — as a number or a function. By default, it takes a value that is 15 pixels less than pieChartRadius.
data.options.pieChartStrokeStyle	white	Type: String The style for lines between sectors and the outline of the diagram. Set as a string that encodes the line color.
data.options.pieChartStrokeWidth	2	Type: Number Width of the sector dividing lines and diagram outline, in pixels. Set as an integer.
data.properties *	—	Type: **IDataManager**|Object Properties of the geo object that is displayed using the layout.
data.properties.data *	—	Type: Object[]|Function Statistical data to base the layout on. It should be an array of JSON objects with the "weight" and "color" fields or a function that returns this array.
data.properties.geoObjects	—	Type: **IGeoObject**[]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		An array of geo objects in the cluster that needs to be displayed. Used if "properties.data" is missing. In this case, the layout traverses all geo objects, determines the value of the "iconColor" option for each object, and generates a diagram based on this data.
data.properties.iconCaption	""	Type: String Caption for the diagram.
data.properties.iconContent	The sum of all the sectors	Type: String The value that is written in the center of the diagram. If omitted, the sum of all the sectors is shown.

* Mandatory parameter/option.

Example:

```
var geoObject = new ymaps.Placemark([55.25, 37.43], {
        // Data for generating a diagram.
        data: [
            { weight: 5, color: '#224080' },
            { weight: 3, color: '#408022' },
            { weight: 2, color: '#802240' }
        ]
    }, {
        iconLayout: 'default#pieChart',
        // You can also use the "icon" prefix to redefine layout options.
        iconPieChartCoreRadius: 15
    });

myMap.geoObjects.add(geoObject);
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty layout indicator. Instance of the **Event** class. Inherited from **ILayout**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
parentelementchange	Change to the parent element. Instance of the Event class. Inherited from ILayout.
shapechange	Change to the shape of the area spanning the layout. Instance of the Event class. Inherited from ILayout.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
build()		Builds a layout instance based on the template and adds it to the parent HTML element. Inherited from layout.templateBased.Base.
clear()		Removes layout content from DOM. Inherited from layout.templateBased.Base.
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getData()	Object	Returns layout data object. Inherited from ILayout.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from ILayout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getShape()	**IShape**|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. By default, it tries to construct a shape via the "shape" option — this.getData().options.get('shape'). In the option, it can be set as an instance of a class implementing the IShape interface, or as a JSON description of the shape's pixel geometry. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from **layout.templateBased.Base**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **ILayout**.
onSublayoutSizeChange(sublayoutInfo, nodeSizeByContent)		Automatically called when resizing a nested layout, added with the `observeSize` option. Used to override specific classes of layouts to respond to the resizing of the content. Inherited from **layout.templateBased.Base**.
rebuild()		Re-sets the layout. Inherited from **layout.templateBased.Base**.
setData(data)		Sets layout data. Inherited from **ILayout**.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from **ILayout**.

layout.storage

Static object.

Instance of **util.AsyncStorage**

Storage for layout classes.

Methods

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
define(key[, depends, resolveCallback[, context]])	**util.AsyncStorage**	Defines an asynchronous value in storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
isDefined(key)	Boolean	Checking if the key can be accessed in the storage.
remove(key)	util.Storage	Deletes the "key: value" pair from storage.
require(keys[, successCallback[, errorCallback[, context]]])	vow.Promise	Async request to get values from the storage.

layout.templateBased

layout.templateBased.Base

Extends **ILayout**.

Basic layout class based on templates. This class is used by the layout factory as a base for creating user layouts.

See templateLayoutFactory

Constructor | Fields | Events | Methods

Constructor

```
layout.templateBased.Base(data)
```

Parameters:

Parameter	Default value	Description
data *	—	Type: Object Set of different types of data that are used for building a layout.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty layout indicator. Instance of the **Event** class. Inherited from **ILayout**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
parentelementchange	Change to the parent element. Instance of the Event class. Inherited from ILayout.
shapechange	Change to the shape of the area spanning the layout. Instance of the Event class. Inherited from ILayout.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
build()		Builds a layout instance based on the template and adds it to the parent HTML element.
clear()		Removes layout content from DOM.
destroy()		Destructor. Called when activity with the layout is finished. Inherited from ILayout.
getData()	Object	Returns layout data object. Inherited from ILayout.
getParentElement()	HTMLElement	Returns parent HTML element. Inherited from ILayout.
getShape()	IShape|null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. By default, it tries to construct a shape via the "shape" option — this.getData().options.get('shape'). In the option, it can be set as an instance of a class implementing the IShape interface, or as a JSON description of the shape's pixel geometry. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **ILayout**.
onSublayoutSizeChange(sublayoutInfo, nodeSizeByContent)		Automatically called when resizing a nested layout, added with the `observeSize` option. Used to override specific classes of layouts to respond to the resizing of the content.
rebuild()		Re-sets the layout.
setData(data)		Sets layout data. Inherited from **ILayout**.
setParentElement(parent)		Adds the layout to the DOM tree. Inherited from **ILayout**.

Methods details

build

```
{} build()
```

Builds a layout instance based on the template and adds it to the parent HTML element.

clear

```
{} clear()
```

Removes layout content from DOM.

getShape

```
{IShape|null} getShape()
```

Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. By default, it tries to construct a shape via the "shape" option — this.getData().options.get('shape'). In the option, it can be set as an instance of a class implementing the IShape interface, or as a JSON description of the shape's pixel geometry. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.

Example:

```
// Creating a placemark and setting a circular interactive area for it.
var MyLayoutClass = ymaps.templateLayoutFactory.createClass('&lt;div
class="imageIcon"&gt;{{name}}&lt;/div&gt;'),
    myPlacemark = new ymaps.Placemark([22, 34], {name: 'Cafe Mayak'}, {
        iconLayout: MyLayoutClass,
       iconShape: {type: 'Circle', coordinates: [0, 0], radius: 20}
   });
```

onSublayoutSizeChange

```
{} onSublayoutSizeChange(sublayoutInfo, nodeSizeByContent)
```

Automatically called when resizing a nested layout, added with the `observeSize` option. Used to override specific classes of layouts to respond to the resizing of the content.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
sublayoutInfo*	—	Type: Object Information about the nested layout.
nodeSizeByContent*	—	Type: Object The new size of the element considering the size of its content. Available fields: `width`, `height`, `scrollX`, `scrollY`.

* Mandatory parameter/option.

rebuild

```
{} rebuild()
```

Re-sets the layout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

LoadingObjectManager

Extends **ICustomizable**, **IEventEmitter**, **IGeoObject**, **IParentOnMap**.

The object manager that optimizes downloading of objects from the server. Allows optimally downloading, displaying, clustering and managing visibility for objects. The manager sends the data request to the specified url in the JSONP format. This format corresponds to the format of objects added to ObjectManager, **ObjectManager.add**. Note that objects drawn on the map via this manager can't have editing and dragging modes enabled.

Constructor | Fields | Events | Methods

Constructor

```
LoadingObjectManager(urlTemplate[, options])
```

Parameters:

Parameter	Default value	Description
urlTemplate *	—	Type: String URL data template. Supports special constructions similar to **Layer**. Substitutions are also supported: • %b is replaced by an array of geographic coordinates that describes the rectangular region for which you want to load data. • %t is replaced by an array of tile numbers that describes the rectangular region to load data for.
options	—	Type: Object Options. • You can set all the options specified in the **Clusterer** description, except for hasBalloon and hasHint options. • Cluster options are set with the "cluster" prefix. The list of options is specified in the description of **ClusterPlacemark**; • Options for singular objects should be specified with the geoObject prefix. The list of options is specified in **GeoObject**.
options.clusterize	false	Type: Boolean Flag indicating whether the objects should be clusterized. Note that clusterization only works for point objects at this time. When cluster mode is enabled, all non-point objects are ignored.
options.loadTileSize	256	Type: Number Tile size for data loading.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.paddingParamName	'callback'	Type: Boolean Name of the GET parameter that contains the value of the JSONP callback.
options.paddingTemplate	null	Type: String Template for a jsonp callback. Supports the same substitutions as urlTemplate. All characters other than letters and numbers will be replaced with '_'. If the parameter is omitted, the name of the jsonp callback will be generated automatically. Conversion examples for tileNumber=[3, 1], zoom=9: • 'myCallback=%x' => 'myCallback_3' • '%c' => 'x_3_y_1_z_9' • 'callback2_%c' => 'callback2_x_3_y_1_z_9' • 'callback%test' => 'callback_test' • 'callback_%b' => 'callback_85_0841__180_0000_85_0841_180_0000' Note that if substitution options are not used in the value, this may lead to an error. All requests will go to the same callback function.
options.splitRequests	false	Type: Boolean Divide requests for data into requests for individual tiles. By default, requests are made for data for a rectangular region that contains multiple tiles.
options.syncOverlayInit	false	Type: Boolean A flag that allows creating overlays for objects synchronously. Note that when you create an overlay synchronously, you should ensure that the appropriate class, which implements the IOverlay interface, is loaded. By default, the overlays are created asynchronously, and the overlay class is loaded on demand.
options.viewportMargin	128	Type: Number|Number[] Offset for the area where the objects are shown. Use this option to expand the view area of objects relative to the visible area of the map.

* Mandatory parameter/option.

Examples:

1.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
var objectManager = new ymaps.LoadingObjectManager('http://myServer.com/tile?
bbox=%b', {
    // Enabling clustering.
    clusterize: true,
    // Cluster options are set with the "cluster" prefix.
    clusterHasBalloon: false,
    // Geo object options are set with the "geoObject" prefix.
    geoObjectOpenBalloonOnClick: false
});

// You can set options directly for child collections.
objectManager.clusters.options.set({
    preset: 'islands#grayClusterIcons',
    hintContentLayout: ymaps.templateLayoutFactory.createClass('Group of
objects')
});
objectManager.objects.options.set('preset', 'islands#grayIcon');
```

2.

```
An example of LoadingObjectManager response
jsonp_callback({
    // The resonse contains error and data fields. If an error occurs, the "error"
field
    // contains the error code or description.
    error: null,
    data: {
        type: 'FeatureCollection',
        features: [
            {
                type: 'Feature',
                geometry: {
                    type: 'Point',
                    coordinates: [55, 35]
                },
                id: 23,
                properties: {
                    balloonContent: 'Placemark balloon content',
                    iconContent: 'Placemark content'
                },
                options: {
                    preset: 'islands#yellowIcon'
                }
            }
        ]
    }
});
```

Fields

Name	Type	Description
clusters	objectManager.ClusterCollection	Collection of clusters generated by the manager.
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
geometry	IGeometry|null	Geo object geometry. Inherited from IGeoObject.
objects	objectManager.ObjectCollection	Collection of objects added to the layer.
options	IOptionManager	Options manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
		Inherited from **ICustomizable**.
properties	**IDataManager**	Geo object data. Inherited from **IGeoObject**.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Calculates the boundaries in geo coordinates for an area that covers all the loaded objects in the manager.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getObjectState(id)	Object	Getting information about the current state of an object added to the manager.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getPixelBounds()	Number[][]|null	Calculates the boundaries in global pixel coordinates for an area that covers all the loaded objects in the manager.
getTileUrl()	String|null	Returns URL of the tile with data.
getUrlTemplate()	String	Returns URL data template.
reloadData()		Method that deletes all previously loaded data and sends a request for new data.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.
setUrlTemplate(urlTemplate)		

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

clusters

```
{objectManager.ClusterCollection} clusters
```

Collection of clusters generated by the manager.

Example:

```
objectManager.clusters.events.add('click', function (e) {
    var objectId = e.get('objectId');
    objectManager.clusters.balloon.open(objectId);
});
```

objects

```
{objectManager.ObjectCollection} objects
```

Collection of objects added to the layer.

Example:

```
objectManager.objects.events.add('click', function (e) {
    var objectId = e.get('objectId');
    objectManager.objects.balloon.open(objectId);
});
```

Methods details

getBounds

```
{Number[][]|null} getBounds()
```

Calculates the boundaries in geo coordinates for an area that covers all the loaded objects in the manager.

Returns array of the area's coordinates, or null if the manager has not been added to the map.

getObjectState

```
{Object} getObjectState(id)
```

Getting information about the current state of an object added to the manager.

Returns object with following fields:

- found — Attribute that indicates whether an object with the passed ID exists. Type: Boolean.
- isShown — Attribute that indicates whether the object is located in the visible area of the map. Type: Boolean.
- cluster — JSON description of the cluster the object was added to. Besides the required fields, it contains the properties.geoObjects field with an array of objects that are in the cluster. This field is returned only when clusterization is enabled.
- isClustered — Attribute indicating whether an object is in the cluster. This field is returned only when clusterization is enabled. Type: Boolean.
- isFilteredOut — Attribute indicating whether an object passed through filtration. If the filter is not set or the object passed through filtration, the value of the field is "false". Type: Boolean.

.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
id*	—	Type: Object ID of the object to get the state for.

* Mandatory parameter/option.

Example:

```
// Opening the cluster balloon with the selected object.
// Getting data about the state of an object inside the cluster.
var objectState = objectManager.getObjectState(myObjects[1].id);
// Checking whether the object is located in the visible area of the map.
if (objectState.found && objectState.isShown) {
    // If the object is in a cluster, we open the cluster balloon with the
appropriate object selected.
    if (objectState.isClustered) {
        objectManager.clusters.state.set('activeObject', myObjects[1]);
        objectManager.clusters.balloon.open(objectState.cluster.id);
    } else {
        // If the object isn't in a cluster, we open its own balloon.
        objectManager.objects.balloon.open(myObjects[i].id);
    }
}
```

getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Calculates the boundaries in global pixel coordinates for an area that covers all the loaded objects in the manager.

Returns array of the area's coordinates, or null if the manager has not been added to the map.

getTileUrl

```
{String|null} getTileUrl()
```

Returns URL of the tile with data.

Parameters:

Parameter	Default value	Description
parameters*	—	Type:

* Mandatory parameter/option.

Example:

```
var objectManager = new ymaps.LoadingObjectManager('http://myServer.com/tile?
bbox=%b');
objectManager.getTileUrl = function (parameters) {
   var boundingBox = parameters.boundingBox.join('~');
   return this.getUrlTemplate().replace(/%b/g, boundingBox);
};
```

getUrlTemplate

```
{String} getUrlTemplate()
```

Returns URL data template.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

reloadData

```
{} reloadData()
```

Method that deletes all previously loaded data and sends a request for new data.

setUrlTemplate

```
{} setUrlTemplate(urlTemplate)
```

Parameters:

Parameter	Default value	Description
urlTemplate*	—	Type: String URL data template.

* Mandatory parameter/option.

Map

Extends **IDomEventEmitter**.

Class for creating and managing a map.

Constructor | Fields | Events | Methods

Constructor

```
Map(parentElement, state[, options])
```

Creates a map instance.

Parameters:

Parameter	Default value	Description
parentElement *	—	Type: Object|String Reference to an HTML element that contains the map, or the ID of this HTML element.
state *	—	Type: Object Map parameters.
state.behaviors	['default']	Type: String[] Enabled behaviors for the map. By default, the following are enabled: dragging the map and zooming the map by multitouch or double-click on touch screen devices; dragging the map with the mouse, double-click zooming and right-click area selection on all other devices. You can specify any keys supported by behavior.Manager.
state.bounds	—	Type: Number[][]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Geo coordinates of the viewport the map starts with. When initializing the map, you can set either a state.zoom-state.center pair, or this viewport. If state.bounds is set and the parameters state.zoom or state.center are also set, they will be ignored.
state.center	—	Type: Number[] Geo coordinates of the map center. They must be set in conjunction with state.zoom.
state.controls	['default']	Type: String[] The map controls, added by default. You can specify any keys supported by **control.Manager**. The list of available keys and default sets of controls are described in control.Manager.add.
state.margin	—	Type: Number|Number[] Offsets from the map edges. Passed to map.margin.Manager.setDefaultMargin.
state.type	'yandex#map'	Type: String|**MapType** Map type. Can be a key or an instance of the **MapType** class. List of available keys: • 'yandex#map' — "Roadmap" map type. • 'yandex#satellite' — "Satellite" map type. • 'yandex#hybrid' — "Hybrid" map type.
state.zoom	—	Type: Number Map zoom level. It must be set in conjunction with state.center.
options	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Map options. The map options can be used to make settings for the map itself, as well as for objects that are added to it: • Options for map behaviors. • Options for the map balloon with the `balloon` prefix. • Options for the map hint with the `hint` prefix. • Options for geo objects with the `geoObject` prefix. • Options for layers with the `layer` prefix. • Options for hotspot layers with the `hotspotLayer` prefix. Options that are interpreted directly by the map itself are listed below.
`options.autoFitToViewport`	'ifNull'	Type: String Automatic map container tracking. By default, the map is automatically redrawn when it is initialized from a hidden container, or if we programmatically changed its dimensions; otherwise, you must call map.container.fitToViewport. The following values are available: • 'none' — Don't track changes to the container display when initialized from a hidden container or when its size is changed programmatically. • 'ifNull' — As soon as the container gets a CSS "display" value other than "none", map.container.fitToViewport executes automatically in order to fit the map to it. After this, tracking will stop. • 'always' — Always track changes to the display state of the map container.
`options.avoidFractionalZoom`	true	Type: Boolean If true, the map does not stop on fractional values of the zoom level; if false, it does. **Note:** By default, this option is true for desktop browsers and false for mobile browsers.
`options.exitFullscreenByEsc`	true	Type: Boolean Enables to exit full-screen mode using the ESC button. If true, the map exits full-screen mode when the ESC button is pressed; if false, it does not.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.fullscreenZIndex	10000	Type: Number The value of the z-index property of the map container in full-screen mode.
options.mapAutoFocus	true	Type: Boolean If true, clicking on the map will move the focus out of the currently active DOM element; if false, it will keep the focus on the active element.
options.maxAnimationZoomDifference	5	Type: Number The maximum difference between the current map zoom level and the new zoom level so that zooming will look smooth.
options.maxZoom	23	Type: Number Maximum map zoom level.
options.minZoom	0	Type: Number Minimum map zoom level.
options.nativeFullscreen	false	Type: Boolean To use the native fullscreen "mode" if possible. Switching to the native full screen mode means that the browser asks the user if they want to go into "fullscreen mode" and if the user agrees, expands the map to full screen, hides browser controls and allows the user to exit the mode by pressing ESC key. The fullscreenZIndex and exitFullscreenByEsc options have no effect in the native "fullscreen mode".
options.projection	ymaps.projection.wgs84Mercator	Type: **IProjection** Map projection.
options.restrictMapArea	false	Type: Boolean|Number[][] Sets the map viewport so that the user cannot leave the viewport area. True — Set the area to match the current display area; false — disable this option. If you need to restrict an area with known coordinates, specify this rectangular area.
options.suppressMapOpenBlock	false	Type: Boolean Whether to hide the offer to open the current map in Yandex.Maps with all the available map information preserved as completely as possible. True — hide; false — don't hide. The link to Yandex.Maps is displayed in the lower-left corner of the map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.suppressObsoleteBrowserNotifier	false	Type: Boolean Whether to hide the notification suggesting a browser upgrade that is shown in outdated browsers. True — hide; false — don't hide. This notification is displayed in the lower-left corner of the map.
options.yandexMapAutoSwitch	true	Type: Boolean true — Enable automatically switching to the Public Map in those places where the map is not detailed enough; false — disable. This option only works for lang=ru_RU and lang=uk_UA.
options.yandexMapDisablePoiInteractivity	false	Type: boolean True — disable interactivity of POI (points of interest) in the map's base layer, false — enable.

* Mandatory parameter/option.

Examples:

1.

```
// Initializing a map with a known center and zoom level
var myMap = new ymaps.Map('map', {
        center: [55.74954, 37.621587],
         zoom: 10
    });
```

2.

```
// Initializing a map by a known view area
// We assume that jQuery is enabled on the page
var $mapElement = $('#map'),
    myMap = new ymaps.Map(
        $mapElement[0],
        ymaps.util.bounds.getCenterAndZoom(
            [[55.7, 37.6], [55.8, 37.7]],
            [$mapElement.width(), $mapElement.height()]
        )
    );
```

3.

```
// Initializing a map from geocoding results
var myMap;
ymaps.geocode('Moscow').then(function (res) {
    myMap = new ymaps.Map('map', {
        center: res.geoObjects.get(0).geometry.getCoordinates(),
        zoom : 10
    });
});
```

Fields

Name	Type	Description
action	map.action.Manager	Map actions manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
balloon	map.Balloon	Map balloon.
behaviors	map.behavior.Manager	Map behaviors manager. Enables and disables behaviors, and also provides access to their methods and properties.
container	map.Container	Map container.
controls	control.Manager	Map controls.
converter	map.Converter	Converts map pixel points from global to local and back.
copyrights	map.Copyrights	Manager for copyright information on the map.
cursors	util.cursor.Manager	Map cursors manager.
events	event.Manager	Map event manager. Supports subscriptions with priorities. Throws an event of the MapEvent type.
geoObjects	map.GeoObjects	Manager for map geo objects.
hint	map.Hint	Map hint.
layers	map.layer.Manager	Map layers manager.
margin	map.margin.Manager	Map margins manager.
options	option.Manager	Map options.
panes	map.pane.Manager	Manager for map object containers.
zoomRange	map.ZoomRange	Object that provides access to information about available zoom levels at a point.

Events

Name	Description
actionbegin	The start of a new smooth map movement. Instance of the Event class. Names of fields that are available via the Event.get method: • action — Action that has started.
actionbreak	Event that occurs when an action step was prematurely stopped (for example, because another action or another step of the action was performed). Instance of the Event class. Names of fields that are available via the Event.get method: • action — An action.
actionend	The end of smooth map movement. Instance of the Event class. Names of fields that are available via the Event.get method: • action — Action that has stopped.
actiontick	The start of a new step of smooth movement (for example, the user shifting the map, or a step of animating smooth zooming). Instance of the Event class. Names of fields that are available via the Event.get method: • action — The action being performed at this moment. • tick — Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
actiontickcomplete	The end of performing a step of smooth movement. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • action — The action being performed at this moment. • tick — Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".
balloonclose	Closing the balloon. Instance of the **Event** class.
balloonopen	Opening a balloon on a map. Instance of the **Event** class.
boundschange	Event for changing the map viewport (as the result of changing the center or zoom level). Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCenter — The previous map center, in geocoordinates. • newCenter — The new map center, in geocoordinates. • oldZoom — The previous zoom level. • newZoom — The new zoom level. • oldGlobalPixelCenter — The previous map center, in global pixels. • newGlobalPixelCenter — The new map center, in global pixels. • oldBounds — Previous viewport. • newBounds — New viewport.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
destroy	The map was destroyed.
hintclose	Closing the hint. Instance of the **Event** class.
hintopen	Opening a hint on a map. Instance of the **Event** class.
marginchange	Map margins changed. Instance of the **Event** class.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Map options changed.
sizechange	Map size changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldSize — The previous size of the map. • newSize — The new size of the map.
typechange	The map type changed. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
destroy()		Destroys the map.
getBounds([options])	Number[][]	Returns a two-dimensional array of geocoordinates for the lower-left and upper-right corners of the map viewport.
getCenter([options])	Number[]	Returns geographical coordinates of the current map center.
getGlobalPixelCenter([options])	Number[]	Returns global pixel coordinates of the current map center.
getPanoramaManager()	**vow.Promise**.<**panorama.Manager**>	Returns **vow.Promise**, which will be resolved with The panorama manager for the map. If an error occurs, the promise object is rejected.
getType()	String|**MapType**	Returns the current map type.
getZoom()	Number	Returns the current map zoom.
panTo(center[, options])	**vow.Promise**	Sets the map center. If an array of points is passed, the map will move from one point to the other.
setBounds(bounds[, options])	**vow.Promise**	Positions the map for displaying the region that was passed.
setCenter(center[, zoom[, options]])	**vow.Promise**	Sets the map center and zoom level. The center is set in geographical coordinates.
setGlobalPixelCenter(globalPixelCenter[, zoom[, options]])	**vow.Promise**	Sets the map center and zoom level. The center is set in global pixel coordinates.
setType(type[, options])	**vow.Promise**	Sets the map type.
setZoom(zoom[, options])	**vow.Promise**	Sets the map zoom level.

Fields details

action

```
{map.action.Manager} action
```

Map actions manager.

Example:

```
var myAction = new ymaps.map.action.Single({
          center: [0, 0],
          zoom: 4,
          duration: 1000,
          timingFunction: "ease-in"
    });

myMap.action.execute(myAction);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

balloon

```
{map.Balloon} balloon
```

Map balloon.

behaviors

```
{map.behavior.Manager} behaviors
```

Map behaviors manager. Enables and disables behaviors, and also provides access to their methods and properties.

Example:

```
// Enabling mouse wheel zooming
myMap.behaviors.enable('scrollZoom');
```

container

```
{map.Container} container
```

Map container.

Example:

```
// Adjusting the map size to the new size of the container
// (for example, if the page layout changed or the map was initialized
// in the hidden state)
map.container.fitToViewport();
```

controls

```
{control.Manager} controls
```

Map controls.

Example:

```
myMap.controls.add('zoomControl', {
    float: 'none',
    position: {
        right: 40,
        top: 5
    }
});
```

converter

```
{map.Converter} converter
```

Converts map pixel points from global to local and back.

Example:

```
// Converting the mouse coordinates to geographical coordinates
var projection = map.options.get('projection');
$('#map').bind('click', function (e) {
    console.log(projection.fromGlobalPixels(
        map.converter.pageToGlobal([e.pageX, e.pageY]), map.getZoom()
    ));
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

copyrights

```
{map.Copyrights} copyrights
```

Manager for copyright information on the map.

Example:

```
// Adding author information
map.copyrights.add('&amp;copy; Jimmy John');
```

cursors

```
{util.cursor.Manager} cursors
```

Map cursors manager.

Example:

```
// Adding the "Help" cursor to the map
var accessor = map.cursors.push('help');
// ...
// Removing the cursor
accessor.remove();
```

events

```
{event.Manager} events
```

Map event manager. Supports subscriptions with priorities. Throws an event of the **MapEvent** type.

Examples:

1.

```
// Adding a placemark on a click on the map.
map.events.add('click', function (e) {
    // To get the geographical coordinates of the point of click,
    // call .get('coords')
    var position = e.get('coords');
    map.geoObjects.add(new ymaps.Placemark(position));
});
```

2.

```
// Tracking the map's center and zoom during smooth movements.
map.events.add('actiontick', function () {
    var state = map.action.getCurrentState();
    console.log(state.zoom, state.globalPixelCenter);
});
```

geoObjects

```
{map.GeoObjects} geoObjects
```

Manager for map geo objects.

hint

```
{map.Hint} hint
```

Map hint.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

layers

{map.layer.Manager} layers

Map layers manager.

See Layer

Example:

```
// Adding a layer of custom tiles to the map
map.layers.add(new ymaps.Layer('http://some.server/tiles?&amp;c'));
```

margin

{map.margin.Manager} margin

Map margins manager.

options

{option.Manager} options

Map options.

Example:

```
// Forbidding all objects on the map to open balloons on mouse clicks.
map.options.set('openBalloonOnClick', false);
```

panes

{map.pane.Manager} panes

Manager for map object containers.

Example:

```
// Adding a custom element to the map controls container
$('&lt;div&gt;&lt;input type=&quot;button&quot; value=&quot;Click!&quot;/
&gt;&lt;/div&gt;')
    .css({ position: 'absolute', left: '5px', top: '50px'})
    .appendTo(map.panes.get('controls').getElement());
```

zoomRange

{map.ZoomRange} zoomRange

Object that provides access to information about available zoom levels at a point.

Example:

```
// Getting the maximum available map zoom level
// at the point [20, 30]
map.zoomRange.get([20, 30]).then(function (zoomRange) {
    alert(zoomRange[1]);
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

actionbegin

The start of a new smooth map movement. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — Action that has started.

actionbreak

Event that occurs when an action step was prematurely stopped (for example, because another action or another step of the action was performed). Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — An action.

actionend

The end of smooth map movement. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — Action that has stopped.

actiontick

The start of a new step of smooth movement (for example, the user shifting the map, or a step of animating smooth zooming). Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — The action being performed at this moment.
- tick — Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

Example:

```
// Tracks all map movement, even user dragging
// and smooth zooming
map.events.add('actiontick', function (e) {
    var tick = e.get('tick');
    console.log('Now the map is moving to the point (' +
        map.options.get('projection').fromGlobalPixels(tick.globalPixelCenter,
tick.zoom).join(',') +
        ') during ' + e.get('tick').duration + ' milliseconds');
});
```

actiontickcomplete

The end of performing a step of smooth movement. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — The action being performed at this moment.
- tick — Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

balloonclose

Closing the balloon. Instance of the **Event** class.

balloonopen

Opening a balloon on a map. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

boundschange

Event for changing the map viewport (as the result of changing the center or zoom level). Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldCenter — The previous map center, in geocoordinates.
- newCenter — The new map center, in geocoordinates.
- oldZoom — The previous zoom level.
- newZoom — The new zoom level.
- oldGlobalPixelCenter — The previous map center, in global pixels.
- newGlobalPixelCenter — The new map center, in global pixels.
- oldBounds — Previous viewport.
- newBounds — New viewport.

Example:

```
// Tracking changes to the map zoom level
map.events.add('boundschange', function (event) {
    if (event.get('newZoom') != event.get('oldZoom')) {
        alert('Zoom level changed');
    }
});
```

destroy

The map was destroyed.

hintclose

Closing the hint. Instance of the **Event** class.

hintopen

Opening a hint on a map. Instance of the **Event** class.

marginchange

Map margins changed. Instance of the **Event** class.

optionschange

Map options changed.

sizechange

Map size changed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldSize — The previous size of the map.
- newSize — The new size of the map.

typechange

The map type changed. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

destroy

```
{} destroy()
```

Destroys the map.

getBounds

```
{Number[][]} getBounds([options])
```

Returns a two-dimensional array of geocoordinates for the lower-left and upper-right corners of the map viewport.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options.
options.useMapMargin	false	Type: Boolean Whether to use offsets that were calculated in the margins manager **map.margin.Manager**.

getCenter

```
{Number[]} getCenter([options])
```

Returns geographical coordinates of the current map center.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options.
options.useMapMargin	false	Type: Boolean Whether to use offsets that were calculated in the margins manager **map.margin.Manager**.

getGlobalPixelCenter

```
{Number[]} getGlobalPixelCenter([options])
```

Returns global pixel coordinates of the current map center.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.useMapMargin	false	Type: Boolean Whether to use offsets that were calculated in the margins manager **map.margin.Manager**.

Example:

```
// Shifting the map 10 pixels left
var position = map.getGlobalPixelCenter();
map.setGlobalPixelCenter([ position[0] - 10, position[1] ]);
```

getPanoramaManager

```
{vow.Promise.<panorama.Manager>} getPanoramaManager()
```

Returns **vow.Promise** which:

- will be **resolved** with: <**panorama.Manager**> — The panorama manager for the map;
- either **rejected** with an error.

getType

```
{String|MapType} getType()
```

Returns the current map type.

getZoom

```
{Number} getZoom()
```

Returns the current map zoom.

panTo

```
{vow.Promise} panTo(center[, options])
```

Sets the map center. If an array of points is passed, the map will move from one point to the other.

Returns Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

Parameters:

Parameter	Default value	Description
center*	—	Type: Number[]|Object[] The map center or an array of points to move through sequentially.
options	—	Type: Object Options.
options.checkZoomRange	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Checks whether the current zoom can be set after moving the map center. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
options.delay	1000	Type: Number The delay time between moves, in milliseconds.
options.duration	500	Type: Number Animation duration, in milliseconds.
options.flying	true	Type: Boolean Allow decreasing and then increasing the map zoom when moving between points. If the distance between points is less than twice the size of the map container, the map smoothly moves the center to the specified point without changing the zoom level. If the distance is more, the map applies a smaller zoom level, then moves to the destination point and applies the required zoom level (emulates flying over the map).
options.safe	true	Type: Boolean Shifts the map safely. You can only use it if the "flying" option is false. If the distance between points is less than twice the size of the map container, the map smoothly moves the center to the specified point. If the distance exceeds that value, the map instantly moves to the selected point. Use this mode to conveniently move the map without displaying unloaded tiles.
options.timingFunction	'ease-in-out'	Type: String Timing function. The same as the value of the CSS property transition-timing-function. Full list of values: http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag
options.useMapMargin	false	Type: Boolean Whether to use offsets that were calculated in the margins manager **map.margin.Manager**.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Flight from Kaliningrad to Vladivostok via Moscow
map.setCenter([54.704815, 20.466380], 10);
map.panTo([
    [55.751574, 37.573856],
    [43.134091, 131.928478]
]).then(function () {
   alert('Landed!');
}, function (err) {
   alert('An error occurred ' + err);
}, this);
```

setBounds

```
{vow.Promise} setBounds(bounds[, options])
```

Positions the map for displaying the region that was passed.

Returns Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

Parameters:

Parameter	Default value	Description
bounds*	—	Type: Number[][] Boundaries of the viewport.
options	—	Type: Object Options.
options.checkZoomRange	false	Type: Boolean Checks whether the specified zoom level can be set. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
options.duration	0	Type: Number Animation duration, in milliseconds.
options.preciseZoom	false	Type: Boolean The ability to use fractional zoom levels.
options.timingFunction	'linear'	Type: String Timing function. The same as the value of the CSS property transition-timing-function. Full list of values: http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag
options.useMapMargin	false	Type: Boolean Whether to use offsets that were calculated in the margins manager **map.margin.Manager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.zoomMargin	0	Type: Number|Number[] Offset from the borders of the visible area of the map. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" option is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager **map.margin.Manager**.

* Mandatory parameter/option.

Example:

```
/// The new map center and zoom level are calculated based on the current
// map state.
// If the current zoomRange does not match the zoomRange for the new map center,
// grey tiles may be displayed if the viewport is very small.
// To avoid this problem, use the checkZoomRange option.
map.setBounds([[60,-40], [20,60]], {
   checkZoomRange: true,
}).then(function () {
      // Action has completed successfully.
   }, function (err) {
      // Specified region could not be shown
      // ...
   }, this);
```

setCenter

```
{vow.Promise} setCenter(center[, zoom[, options]])
```

Sets the map center and zoom level. The center is set in geographical coordinates.

Returns Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

Parameters:

Parameter	Default value	Description
center*	—	Type: Number[] Geo coordinates of the map center.
zoom	—	Type: Number Map zoom level.
options	—	Type: Object Options.
options.checkZoomRange	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Checks whether the specified zoom level can be set. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
options.duration	0	Type: Number Animation duration, in milliseconds.
options.timingFunction	'linear'	Type: String Timing function. The same as the value of the CSS property transition-timing-function. Full list of values: http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag
options.useMapMargin	false	Type: Boolean Whether to use offsets that were calculated in the margins manager **map.margin.Manager**.

* Mandatory parameter/option.

Example:

```
myMap.setCenter([40, 50], 3, {
    checkZoomRange: true
});
```

setGlobalPixelCenter

```
{vow.Promise} setGlobalPixelCenter(globalPixelCenter[, zoom[, options]])
```

Sets the map center and zoom level. The center is set in global pixel coordinates.

Returns Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

Parameters:

Parameter	Default value	Description
globalPixelCenter*	—	Type: Number[] Pixel coordinates of the new map center.
zoom	—	Type: Number Map zoom level.
options	—	Type: Object Options.
options.checkZoomRange	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Checks whether the specified zoom level can be set. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
options.duration	0	Type: Number Animation duration, in milliseconds.
options.timingFunction	'linear'	Type: String Timing function. The same as the value of the CSS property transition-timing-function. Full list of values: http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag
options.useMapMargin	false	Type: Boolean Whether to use offsets that were calculated in the margins manager **map.margin.Manager**.

* Mandatory parameter/option.

Example:

```
// Shifting the map 10 pixels left
var position = map.getGlobalPixelCenter();
map.setGlobalPixelCenter([ position[0] - 10, position[1] ]);
```

setType

```
{vow.Promise} setType(type[, options])
```

Sets the map type.

Returns Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

Parameters:

Parameter	Default value	Description
type*	—	Type: String|**MapType** Map type. Can be a key or an instance of the **MapType** class. List of available keys: • 'yandex#map' — "Roadmap" map type. • 'yandex#satellite' — "Satellite" map type. • 'yandex#hybrid' — "Hybrid" map type.
options	—	Type: Object Map options.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.checkZoomRange	false	Type: Boolean Checks if the specified map type can be set at the specified zoom level. When set to "true", a request is sent to the server, which returns the range of acceptable zoom values for the given map type. After this, the specified center and appropriate zoom are applied.

* Mandatory parameter/option.

Example:

```
map.setType('yandex#hybrid', {
   checkZoomRange: true
}).then(function () {
   // The map type has been set with the acceptable zoom level.
}, this);
```

setZoom

```
{vow.Promise} setZoom(zoom[, options])
```

Sets the map zoom level.

Returns Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

Parameters:

Parameter	Default value	Description
zoom*	—	Type: Number Map zoom level.
options	—	Type: Object Options.
options.checkZoomRange	false	Type: Boolean Checks whether the specified zoom level can be set.
options.duration	0	Type: Number Animation duration, in milliseconds.
options.useMapMargin	false	Type: Boolean Whether to use offsets that were calculated in the margins manager **map.margin.Manager**.

* Mandatory parameter/option.

Example:

```
myMap.setZoom(4, {duration: 1000});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map

map.action

map.action.Continuous

Extends **IMapAction**.

Map movement consisting of one or more steps. Intended for implementing complex map movements.

Constructor | Fields | Events | Methods

Constructor

```
map.action.Continuous()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
end	Event that notifies the map that movement has finished. Inherited from **IMapAction**.
tick	Event that notifies the map of the next step. Contains the fields: • globalPixelCenter — The new map center, in global pixels. • zoom — The new map zoom. • duration — The time that is allowed for performing the step. • timingFunction — Function describing the type of movement. Inherited from **IMapAction**.

Methods

Name	Returns	Description
begin(mapActionManager)		Starts the movement to be performed by the map. This method is called automatically by the map movement manager. From the moment when **IMapAction.begin** is called, the movement manager listens for **IMapAction.event:tick** and **IMapAction.event:end** and executes them. Inherited from **IMapAction**.
end()		Stops movement. Inherited from **IMapAction**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
isActive()	Boolean	Checks whether map movement is being performed at this moment.
tick(tick)	map.action.Continuous	Performs a single step of the map movement.

Methods details

isActive

```
{Boolean} isActive()
```

Checks whether map movement is being performed at this moment.

Returns true if the movement is currently being performed by the map, otherwise false.

tick

```
{map.action.Continuous} tick(tick)
```

Performs a single step of the map movement.

Returns self-reference.

Parameters:

Parameter	Default value	Description
tick*	—	Type: Object Movement parameters.
tick.duration	0	Type: Number Duration of making the move, in milliseconds.
tick.globalPixelCenter	—	Type: Number[] The new map center in global pixels. One of the parameters must be set: either pixelOffset, or globalPixelCenter.
tick.pixelOffset	—	Type: Number[] The offset in pixels relative to the previous center. One of the parameters must be set: either pixelOffset, or globalPixelCenter.
tick.timingFunction	'linear'	Type: String Timing function.
tick.zoom	—	Type: Number The new map zoom. If omitted, the map zoom does not change.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map.action.Manager

Extends **IEventEmitter**.

Map actions manager. Makes complex movements possible on the map and ensures that complex movements do not overlap each other. Every map already has its own actions manager, available as **Map.action**. Don't create new instances of this class unless necessary.

See Map.action

Constructor | Fields | Events | Methods

Constructor

```
map.action.Manager(map)
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Map.

* Mandatory parameter/option.

Example:

```
// Creating a complex movement: every 100 ms, the map
// center shifts a random amount.

// Creating an instance of the complex movement
var action = new ymaps.map.action.Continuous();
// Executing it on the map
myMap.action.execute(action);

// Recalling the pixel map center and zoom level
var center = myMap.getGlobalPixelCenter(),
    zoom = myMap.getZoom();
// Generate a random shift every 100 milliseconds
var interval = window.setInterval(function () {
    center[0] += Math.round(Math.random() * 100) - 50;
    center[1] += Math.round(Math.random() * 100) - 50;
    // Generating a new shift in the map
  action.tick({
        globalPixelCenter: center,
       zoom: zoom
    });
}, 100);

// As soon as the user moves the map, our movement
// stops being executed and the "end" event occurs.
var listener = action.events.once('end', function () {
    listener.removeAll();
    window.clearInterval(interval);
});
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
begin	Event that occurs when an action has started. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • action — Action that has started.
break	Event that occurs when an action step was prematurely stopped (for example, because another action or another step of the action was performed). Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • action — An action.
end	Event that occurs when an action is stopped. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • action — Action that has stopped.
tick	Event that occurs when the next step of an action begins to be performed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • action — The action being performed at this moment. • tick — Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".
tickcomplete	Event that occurs when an action step has been completed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • action — The action being performed at this moment. • tick — Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

Methods

Name	Returns	Description
breakTick()		Interrupts an action step.
execute(action)		Starts an action on the map. If a different movement is being performed on the map at this time, it is stopped (the "end" method is called). The new movement is started using a "begin" method call.
getCurrentState()	Object	Checks the map state at the time of smooth movement.
getMap()	**Map**	Returns reference to the map.
setCorrection(userFunction)		Used for making user adjustments to complex movements on the map. When the adjustment is finished, the corrected values must be returned.
stop()		Stops an action being performed on the map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

begin

Event that occurs when an action has started. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — Action that has started.

break

Event that occurs when an action step was prematurely stopped (for example, because another action or another step of the action was performed). Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — An action.

end

Event that occurs when an action is stopped. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — Action that has stopped.

tick

Event that occurs when the next step of an action begins to be performed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — The action being performed at this moment.
- tick — Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

Example:

```
// Tracks all map movement, even user dragging
// and smooth zooming
myMap.action.events.add('tick', function (e) {
    var tick = e.get('tick');
    console.log('Now the map is moving to the point (' +

myMap.options.get('projection').fromGlobalPixels(tick.globalPixelCenter,
tick.zoom).join(',') +
        ') during ' + e.get('tick').duration + ' milliseconds');
});
```

tickcomplete

Event that occurs when an action step has been completed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- action — The action being performed at this moment.
- tick — Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

Methods details

breakTick

```
{} breakTick()
```

Interrupts an action step.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

execute

```
{} execute(action)
```

Starts an action on the map. If a different movement is being performed on the map at this time, it is stopped (the "end" method is called). The new movement is started using a "begin" method call.

Parameters:

Parameter	Default value	Description
action*	—	Type: **IMapAction** Action.

* Mandatory parameter/option.

getCurrentState

```
{Object} getCurrentState()
```

Checks the map state at the time of smooth movement.

Returns object with the fields: isTicking — Whether a step of smooth movement is being performed. tickProgress — Which part of the current step has been completed. zoom — The map zoom during the current step. globalPixelCenter — The map center in global pixels during the current step.

Example:

```
// Logs the current map center.
// Even works during smooth zooming or
// while the user is dragging the map.
window.setInterval(function () {
    console.log(myMap.action.getCurrentState().center.join(', '));
}, 100);
```

getMap

```
{Map} getMap()
```

Returns reference to the map.

setCorrection

```
{} setCorrection(userFunction)
```

Used for making user adjustments to complex movements on the map. When the adjustment is finished, the corrected values must be returned.

Parameters:

Parameter	Default value	Description
userFunction*	—	Type: Function Custom function for adjusting steps.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Making it impossible for the user to drag the map center
// outside of the Moscow Ring Road.
var mkad = [
    [55.785017, 37.841576],
    [55.861979, 37.765992],
    [55.898533, 37.635961],
    [55.888897, 37.48861],
    [55.83251, 37.395275],
    [55.744789, 37.370248],
    [55.660424, 37.434424],
    [55.5922, 37.526366],
    [55.574019, 37.683167],
    [55.62913, 37.802473],
    [55.712203, 37.837121]
],
mkadPolygon = new ymaps.Polygon([mkad], {}, {
    fillColor: '#FFFF00',
    opacity: .4
});
myMap.geoObjects.add(mkadPolygon);
myMap.action.setCorrection(function (tick) {
    var projection = myMap.options.get('projection'),
        tickCenter = projection.fromGlobalPixels(tick.globalPixelCenter,
tick.zoom);
    // If the map center is not in our area.
    if (!mkadPolygon.geometry.contains(tickCenter)) {
        tick.globalPixelCenter = projection.toGlobalPixels(
            mkadPolygon.geometry.getClosest(tickCenter).position,
            tick.zoom
        );
        tick.duration = 0;
    }
    return tick;
});
```

stop

```
{} stop()
```

Stops an action being performed on the map.

map.action.Single

Extends **IMapAction**.

Simple map movement. The movement is made immediately after it is passed to map.action.Manager.

Constructor | Fields | Events | Methods

Constructor

```
map.action.Single(tick)
```

Creates a simple (single step) map movement.

Parameters:

Parameter	Default value	Description
tick *	—	Type: Object Movement parameters.
tick.callback	—	Type: Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Function that will be called after performing the action. Accepts an error or null as a parameter, if the action was not successful.
tick.center	—	Type: Number[] The new map center in geo coordinates.
tick.checkZoomRange	false	Type: Boolean Flag showing whether the new map zoom needs to be checked. If the flag is set to true, the range of allowable zoom levels at the new point will be requested before performing the action. If the specified zoom does not fall within the allowable values, it will be corrected. In addition, the value of the new map center in global pixel coordinates will be changed.
tick.duration	0	Type: Number Duration of making the move, in milliseconds.
tick.globalPixelCenter	—	Type: Number[] The new map center in global pixels. When the "center" and "globalPixelCenter" parameters are set simultaneously, the "center" parameter is ignored.
tick.timingFunction	'linear'	Type: String Timing function.
tick.zoom	—	Type: Number The new map zoom.

* Mandatory parameter/option.

Example:

```
var myCallback = function(err) {
        if (err) {
            throw err;
        }
    },
    myAction = new ymaps.map.action.Single({
          center: [0, 0],
          zoom: 4,
          duration: 1000,
          timingFunction: 'ease-in',
          checkZoomRange: true,
          callback: myCallback
    });

// The action is performed immediately after calling "execute".
myMap.action.execute(myAction);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
end	Event that notifies the map that movement has finished. Inherited from IMapAction.
tick	Event that notifies the map of the next step. Contains the fields: • globalPixelCenter — The new map center, in global pixels. • zoom — The new map zoom. • duration — The time that is allowed for performing the step. • timingFunction — Function describing the type of movement. Inherited from IMapAction.

Methods

Name	Returns	Description
begin(mapActionManager)		Starts the movement to be performed by the map. This method is called automatically by the map movement manager. From the moment when IMapAction.begin is called, the movement manager listens for IMapAction.event:tick and IMapAction.event:end and executes them. Inherited from IMapAction.
end()		Stops movement. Inherited from IMapAction.
isActive()	Boolean	Checks whether map movement is being performed at this moment.

Methods details

isActive

```
{Boolean} isActive()
```

Checks whether map movement is being performed at this moment.

Returns true if the movement is currently being performed by the map, otherwise false.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map.addon

map.addon.balloon

Note:
The constructor of the map.addon.balloon class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use a balloon for the map. Adds the **IBalloonOwner** interface to the map (**Map**). When enabling package.full (the standard set of modules), it is available by default. If **Map** is enabled separately, this module must be explicitly specified in the loader. If **map.addon.balloon** if enabled separately after creating **Map**, the **IBalloonOwner** interface won't be added. Then in order to initialize the balloon manager, you will need to use the map.addon.balloon#get method.

Methods

Methods

Name	Returns	Description
get(map)	**IPopupManager**	Returns map balloon manager.

Methods details

get

```
{IPopupManager} get(map)
```

Returns map balloon manager.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** Map

* Mandatory parameter/option.

Example:

```
ymaps.map.addon.balloon.get(myMap)
```

map.addon.hint

Note:
The constructor of the map.addon.hint class is hidden, as this class is not intended for autonomous initialization.

Static object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

The module that makes it possible to use a hint for the map. Adds the **IHintOwner** interface to the map (**Map**). When enabling package.full (the standard set of modules), it is available by default. If **Map** is enabled separately, this module must be explicitly specified in the loader. If **map.addon.hint** if enabled separately after creating **Map**, the **IHintOwner** interface won't be added. Then in order to initialize the balloon manager, you will need to use the map.addon.hint#get method.

Methods

Methods

Name	Returns	Description
get(map)	**IPopupManager**	Returns map hint manager.

Methods details

get

```
{IPopupManager} get(map)
```

Returns map hint manager.

Parameters:

Parameter	Default value	Description
map*	—	Type: **Map** Map

* Mandatory parameter/option.

Example:

```
ymaps.map.addon.hint.get(myMap)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map.Balloon

Extends **IBalloonManager**, IBalloonSharingManager.

Map balloon manager. Each map already has its own balloon manager, available as myMap.balloon. Only one balloon controlled by this manager can be open on the map at a time. Don't create new instances of this class unless necessary.

See Balloon Map.balloon

Constructor | Fields | Events | Methods

Constructor

```
map.Balloon(map)
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Reference to a map object.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
autopanbegin	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
autopanend	End of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
beforeuserclose	The event which precedes **Balloon.event:userclose**. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
close	Closing the info object. Names of fields available via **Event.get**: • target — Reference to the object where the closing occurred. Inherited from **IPopupManager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
open	Opening the info object. Names of fields available via **Event.get**: • target — Reference to the object where the opening occurred. Inherited from **IPopupManager**.
userclose	Balloon closed by the user. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.

Methods

Name	Returns	Description
autoPan()	**vow.Promise**	Moves the map so that the balloon is visible. Inherited from **IBalloonManager**.
close([force])	**vow.Promise**	Closes the info object. Inherited from **IPopupManager**.
destroy()		Disables the info object manager. Inherited from **IPopupManager**.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from **IPopupManager**.
getOptions()	**IOptionManager**|null	Returns the options manager or 'null'. Inherited from **IPopupManager**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopupManager**.
getOverlaySync()	**IOverlay**|null	Returns the overlay, if one exists. Inherited from **IPopupManager**.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from **IPopupManager**.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from **IPopupManager**.
open([position[, data[, options]]])	**vow.Promise**	Opens the info object at the specified position. Inherited from **IPopupManager**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopupManager**.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

map.behavior

map.behavior.Manager

Extends **ICustomizable**, **IEventEmitter**, **IParentOnMap**.

Map behaviors manager. Allows to enable and disable behaviors. Each map already has its own behavior manager, available as map.behaviors. Don't instantiate new instances of this class unless necessary

See Map.behaviors

Constructor | Fields | Events | Methods

Constructor

```
map.behavior.Manager(map[, behaviors[, options]])
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Map.
behaviors	—	Type: String|String[] List of map behaviors that are immediately enabled when a map is created. By default — "drag", "dblClickZoom" and "rightMouseButtonMagnifier" for desktop browsers; "drag", "dblClickZoom" and "multiTouch" for mobile browsers.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Acceptable key values: • "default" — Shortcut for enabling/disabling default map behaviors. • "drag" — Dragging the map when the left mouse button is held down, or by a single touch **behavior.Drag**. • "scrollZoom" — Changing the zoom with the mouse wheel **behavior.ScrollZoom**. • "dblClickZoom" — Zooming the map on a double click **behavior.DblClickZoom**. • "multiTouch" — Zooming the map with a multi touch (i.e. on a touch screen) **behavior.MultiTouch**. • "rightMouseButtonMagnifier" — Magnifying the area that is selected using the right mouse button (for desktop browsers only), **behavior.RightMouseButtonMagnifier**. • "leftMouseButtonMagnifier" — Magnifying the area selected by the left mouse button or a single touch, **behavior.LeftMouseButtonMagnifier**. • "ruler" — Measuring distance **behavior.Ruler**. • "routeEditor" — Route editor **behavior.RouteEditor**. You can add and remove behavior classes via the behaviors storage **behavior.storage**.
`options`	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Behavior options. The following options can be set: • Options for the behavior.Drag behavior with the `drag` prefix. • Options for the behavior.ScrollZoom behavior with the `scrollZoom` prefix. • Options for the behavior.DblClickZoom behavior with the `dblClickZoom` prefix. • Options for the behavior.MultiTouch behavior with the `multiTouch` prefix. • Options for the magnifier.RightMouseButtonMagnifier behavior with the `rightMouseButtonMagnifier` prefix. • Options for the behavior.LeftMouseButtonMagnifier behavior with the `leftMouseButtonMagnifier` prefix. • Options for the behavior.Ruler behavior with the `ruler` prefix.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Methods

Name	Returns	Description
disable(behaviors)	**map.behavior.Manager**	Disables behaviors on the map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
enable(behaviors)	**map.behavior.Manager**	Enables behaviors on the map.
get(behaviorName)	**IBehavior**	Returns instance of the behavior by the key.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
isEnabled(behaviorName)	Boolean	Checks whether a behavior is currently enabled.

Methods details

disable

```
{map.behavior.Manager} disable(behaviors)
```

Disables behaviors on the map.

Returns self-reference.

Parameters:

Parameter	Default value	Description
behaviors*	—	Type: String|String[] List of behaviors that can be disabled.

* Mandatory parameter/option.

Example:

```
myMap.behaviors.disable('drag');
```

enable

```
{map.behavior.Manager} enable(behaviors)
```

Enables behaviors on the map.

Returns self-reference.

Parameters:

Parameter	Default value	Description
behaviors*	—	Type: String|String[] List of behaviors that can be enabled.

* Mandatory parameter/option.

Example:

```
myMap.behaviors.enable(['ruler', 'multiTouch']);
```

get

```
{IBehavior} get(behaviorName)
```

Returns instance of the behavior by the key.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
behaviorName*	—	Type: String Name of the behavior.

* Mandatory parameter/option.

Example:

```
myMap.behaviors.get('drag');
```

isEnabled

```
{Boolean} isEnabled(behaviorName)
```

Checks whether a behavior is currently enabled.

Returns true if the behavior is enabled, otherwise false.

Parameters:

Parameter	Default value	Description
behaviorName*	—	Type: String Behavior ID.

* Mandatory parameter/option.

Example:

```
// If the "drag" behavior is disabled, we enable it
if (!(myMap.behaviors.isEnabled('drag'))) {
    myMap.behaviors.enable('drag');
}
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map.Container

Extends **IDomEventEmitter**.

Map container manager. Each map already has its own container manager, available as map.container. Don't instantiate new instances of this class unless necessary.

See Map.container

Constructor | Fields | Events | Methods

Constructor

```
map.Container(parentElement)
```

Parameters:

Parameter	Default value	Description
parentElement *	—	Type: String|HTMLElement HTML element (or its ID) where the map will be created.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.

Events

Name	Description
beforefullscreenexit	The event preceding the "fullscreenexit" event. If the **Event.preventDefault** method is called for this event, a subsequent fullscreenexit event will be canceled. Instance of the **Event** class.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
fullscreenenter	The map switched to fullscreen mode. Instance of the **Event** class.
fullscreenexit	The map exited full-screen mode. Instance of the **Event** class.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
sizechange	Change to the size of the map container. Instance of the Event class. Names of fields that are available via the Event.get method: • oldSize: Number[]; • newSize: Number[]; • oldOffset: Number[]; • newOffset: Number[]; • preservePixelPosition: Boolean.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
enterFullscreen()		Allows you to switch the map to full-screen mode.
exitFullscreen()		Allows you to take the map out of full-screen mode.
fitToViewport([preservePixelPosition])		Called when the size of the map container is changed so that the map applies the new size.
getElement()	HTMLElement	Returns map HTML element.
getOffset()	Number[]	Returns the shift of the map container in pixels, relative to the upper-left corner of the document.
getParentElement()	HTMLElement	Returns custom HTML element which contains the created map.
getSize()	Number[]	Returns size of the map container, in pixels.
isFullscreen()	Boolean	Returns indicates whether the map is in full-screen mode.

Events details

beforefullscreenexit

The event preceding the "fullscreenexit" event. If the Event.preventDefault method is called for this event, a subsequent fullscreenexit event will be canceled. Instance of the Event class.

fullscreenenter

The map switched to fullscreen mode. Instance of the Event class.

fullscreenexit

The map exited full-screen mode. Instance of the Event class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

sizechange

Change to the size of the map container. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldSize: Number[];
- newSize: Number[];
- oldOffset: Number[];
- newOffset: Number[];
- preservePixelPosition: Boolean.

Methods details

enterFullscreen

```
{} enterFullscreen()
```

Allows you to switch the map to full-screen mode.

exitFullscreen

```
{} exitFullscreen()
```

Allows you to take the map out of full-screen mode.

fitToViewport

```
{} fitToViewport([preservePixelPosition])
```

Called when the size of the map container is changed so that the map applies the new size.

Parameters:

Parameter	Default value	Description
preservePixelPosition	—	Type: Boolean Save the location of the map center.

Example:

```
// Changing the dimensions of the map container
map.container.getElement().style.width = '300px';
// Initializing size recalculation
map.container.fitToViewport();
```

getElement

```
{HTMLElement} getElement()
```

Returns map HTML element.

getOffset

```
{Number[]} getOffset()
```

Returns the shift of the map container in pixels, relative to the upper-left corner of the document.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getParentElement

```
{HTMLElement} getParentElement()
```

Returns custom HTML element which contains the created map.

getSize

```
{Number[]} getSize()
```

Returns size of the map container, in pixels.

isFullscreen

```
{Boolean} isFullscreen()
```

Returns indicates whether the map is in full-screen mode.

map.Converter

Class for converting global pixel coordinates of a point (calculated from the upper-left corner of the world) to local coordinates (calculated from the upper-left corner of the window) and back. Each map already has its own converter, available as map.converter. Don't instantiate new instances of this class unless necessary.

See Map.converter

Constructor | Methods

Constructor

```
map.Converter(map)
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Reference to the map.

* Mandatory parameter/option.

Methods

Name	Returns	Description
globalToPage(globalPixelPoint)	Number[]	Converts global pixel coordinates of a point to local coordinates.
pageToGlobal(pagePixelPoint)	Number[]	Converts local pixel coordinates of a point to global coordinates.

Methods details

globalToPage

```
{Number[]} globalToPage(globalPixelPoint)
```

Converts global pixel coordinates of a point to local coordinates.

Returns the converted coordinates.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
globalPixelPoint*	—	Type: Number[] Pixel coordinates of a point that need to be converted.

* Mandatory parameter/option.

Example:

```
// Converting geographical coordinates to browser window pixels
var projection = map.options.get('projection');
console.log(map.converter.globalToPage(
    projection.toGlobalPixels(
        // geographical coordinates
        [55, 37],
        map.getZoom()
    )
));
```

pageToGlobal

```
{Number[]} pageToGlobal(pagePixelPoint)
```

Converts local pixel coordinates of a point to global coordinates.

Returns the converted coordinates.

Parameters:

Parameter	Default value	Description
pagePixelPoint*	—	Type: Number[] Pixel coordinates of a point that need to be converted.

* Mandatory parameter/option.

Example:

```
// Converting mouse cursor coordinates to geocoordinates
var projection = map.options.get('projection');
$('#map').bind('click', function (e) {
    console.log(projection.fromGlobalPixels(
        map.converter.pageToGlobal([e.pageX, e.pageY]), map.getZoom()
    ).join(', ');
});
```

map.Copyrights

Manager for copyright information on the map. Each map already has its own copyright information manager, available as map.copyrights. Don't instantiate new instances of this class unless necessary.

See Map.copyrights

Constructor | Events | Methods

Constructor

```
map.Copyrights(map)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Map.

* Mandatory parameter/option.

Example:

```
// Adding static information about copyright info to the map
var accessor = map.copyrights.add('&amp;copy; Gerardus Mercator');
// ...
// Removing information about copyrights
accessor.remove();
```

Events

Name	Description
change	Event for changes to copyright information placed on the map. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • oldCopyrights — Old array of copyright information. • newCopyrights — New array of copyright information.

Methods

Name	Returns	Description
add(customCopyrights)	**ICopyrightsAccessor**	Adds static copyright information to the map (independent of the current center and zoom level).
addProvider(provider)	**map.Copyrights**	Adds a new provider of copyright information.
get([point[, zoom]])	**vow.Promise**	Determines copyright information at the specified point.
getPromoLink()	String	Returns external link from the "Open in Yandex.Maps" block.
removeProvider(provider)	**map.Copyrights**	Removes a provider of copyright information.

Events details

change

Event for changes to copyright information placed on the map. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldCopyrights — Old array of copyright information.
- newCopyrights — New array of copyright information.

Methods details

add

```
{ICopyrightsAccessor} add(customCopyrights)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Adds static copyright information to the map (independent of the current center and zoom level).

Returns an object for managing the added information.

Parameters:

Parameter	Default value	Description
customCopyrights*	—	Type: String|HTMLElement|String[]|HTMLElement[] Copyright information in string format, a DOM element, or an array of strings/DOM elements.

* Mandatory parameter/option.

addProvider

```
{map.Copyrights} addProvider(provider)
```

Adds a new provider of copyright information.

Returns self-reference.

Parameters:

Parameter	Default value	Description
provider*	—	Type: ICopyrightsProvider Provider.

* Mandatory parameter/option.

Example:

```
// Creating a dynamic provider of copyright information
// that will get up-to-date information about copyrights for
// a particular point on the map from a remote server.
var myProvider = {
        getCopyrights: function (center, zoom) {
            var deferred = ymaps.vow.defer();
            $.ajax('http://some.server/copyrights/?ll=' +
                center.join(',') = '&z=' + zoom, {
                // The server must return an array of strings
                success: function (res) {
                    deferred.resolve(res);
                }
            });
            return deferred.promise();
        }
    };
// Adding a provider to the map; now when the center or zoom changes,
// the map copyright information manager will
// request updated information from the remote server and display
// the received data automatically.
map.copyrights.addProvider(myProvider);
```

get

```
{vow.Promise} get([point[, zoom]])
```

Determines copyright information at the specified point.

Returns a Promise object that will be resolved and will pass an array of strings or DOM elements as a parameter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
point	—	Type: Number[] The point (in geographical coordinates) that copyright information needs to be determined for. If omitted, the current map center is used.
zoom	—	Type: Number The zoom level that copyright information needs to be determined for. If omitted, the current map zoom level is used.

getPromoLink

```
{String} getPromoLink()
```

Returns external link from the "Open in Yandex.Maps" block.

removeProvider

```
{map.Copyrights} removeProvider(provider)
```

Removes a provider of copyright information.

Returns self-reference.

Parameters:

Parameter	Default value	Description
provider*	—	Type: **ICopyrightsProvider** Provider.

* Mandatory parameter/option.

map.GeoObjects

Extends **IGeoObjectCollection**.

Collection of map geo objects. Each map already has its own geo objects collection, available as map.geoObjects. Don't instantiate new instances of this class unless necessary.

See Map.geoObjects

Constructor | Fields | Events | Methods

Constructor

```
map.GeoObjects(map[, options])
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options	—	Type: Object Options of a collection of geo objects. The map.geoObjects options can be used to make settings for geo objects that have been added to the map: • Options for clusterers with the `clusterer` prefix. • Options for clusters with the `cluster` prefix.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
add	A child geo object has been added (inserted). Instance of the Event class. Names of fields that are available via the Event.get method: • index: Integer — Index of the added geo object. • child: IGeoObject — Reference to the added geo object. Inherited from IGeoObjectCollection.
boundschange	Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the Event class. Inherited from IGeoObjectCollection.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
optionschange	Change to the object options. Inherited from ICustomizable.
pixelboundschange	Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the Event class. Inherited from IGeoObjectCollection.
remove	A child geo object has been removed. Instance of the Event class. Names of fields that are available via the Event.get method: • index: Integer — Index of the deleted geo object. • child: IGeoObject — Reference to the deleted geo object. Inherited from IGeoObjectCollection.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
set	A new child geo object has been added to the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • index: Integer — Index of the geo object. • child: **IGeoObject** — Reference to the new geo object. • prevChild: **IGeoObject** — Reference to the previous value for this index. Inherited from **IGeoObjectCollection**.

Methods

Name	Returns	Description
add(child[, index])	**map.GeoObjects**	Adds (inserts) a child geo object to the collection.
each(callback[, context])		Calls a handler function for each child geo object.
get(index)	**IGeoObject**	Returns a child geo object with the specified index.
getBounds()	Number[][]|null	Returns geographical coordinates of the area that covers the collection and its child geo objects.
getIterator()	**IIterator**	Returns iterator for the collection.
getLength()	Integer	Returns length of the collection.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getPixelBounds()	Number[][]|null	Returns global pixel coordinates of the area that spans the collection and its child geo objects.
indexOf(object)	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.
remove(child)	**map.GeoObjects**	Removes a child geo object from the collection.
removeAll()	**map.GeoObjects**	Clears the collection.
set(index, child)	**map.GeoObjects**	Adds a new child geo object to the collection.
splice(index, number)	**GeoObjectCollection**	Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

Methods details

add

```
{map.GeoObjects} add(child[, index])
```

Adds (inserts) a child geo object to the collection.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
child*	—	Type: **IGeoObject** Child object.
index	—	Type: Integer The index where the new object is added. By default, the object is added to the end of the collection.

* Mandatory parameter/option.

each

```
{} each(callback[, context])
```

Calls a handler function for each child geo object.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function.
context	—	Type: Object Context for the handler function.

* Mandatory parameter/option.

Example:

```
// Displaying a geo object's index in a collection for its icon contents.
myGeoObjects.events.add(["add", "remove", "set"], function () {
    this.each(function (el, i) {
        el.properties.set("iconContent", i);
    })
}, myGeoObjects);
```

get

```
{IGeoObject} get(index)
```

Returns a child geo object with the specified index.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index.

* Mandatory parameter/option.

getBounds

```
{Number[][]|null} getBounds()
```

Returns geographical coordinates of the area that covers the collection and its child geo objects.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getIterator

```
{IIterator} getIterator()
```

Returns iterator for the collection.

getLength

```
{Integer} getLength()
```

Returns length of the collection.

getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Returns global pixel coordinates of the area that spans the collection and its child geo objects.

indexOf

```
{Integer} indexOf(object)
```

Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.

Parameters:

Parameter	Default value	Description
object*	—	Type: Object Child object.

* Mandatory parameter/option.

remove

```
{map.GeoObjects} remove(child)
```

Removes a child geo object from the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
child*	—	Type: IGeoObject Geo object being removed.

* Mandatory parameter/option.

Example:

```
// When a geo object is clicked, we remove it from the collection.
myGeoObjects.events.add("click", function (e) {
    if (e.get("target").getParent() == this) {
        this.remove(e.get("target"));
    }
}, myGeoObjects);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

removeAll

```
{map.GeoObjects} removeAll()
```

Clears the collection.

Returns self-reference.

set

```
{map.GeoObjects} set(index, child)
```

Adds a new child geo object to the collection.

Returns self-reference.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index.
child*	—	Type: **IGeoObject** Child object.

* Mandatory parameter/option.

splice

```
{GeoObjectCollection} splice(index, number)
```

Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

Returns collection of deleted geo objects.

Parameters:

Parameter	Default value	Description
index*	—	Type: Integer Index of the geo object to start deletion from.
number*	—	Type: Integer The number of geo objects to be deleted.

* Mandatory parameter/option.

Example:

```
// Removes the second object.
myGeoObjects.splice(1, 1);
// Puts a new "obj" object in the second position.
myGeoObjects.splice(1, 0, obj);
// Replaces the second object with the new "obj" object.
myGeoObjects.splice(1, 1, obj);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map.Hint

Extends **IHintManager**, IHintSharingManager.

Map hint manager. Each map already has its own hint manager, available as myMap.hint. Only one hint, controlled by this manager, can be open on the map at a time. Don't create new instances of this class unless necessary.

See Hint Map.hint

Constructor | Fields | Events | Methods

Constructor

```
map.Hint(map)
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Reference to a map object.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
close	Closing the info object. Names of fields available via **Event.get**: • target — Reference to the object where the closing occurred. Inherited from **IPopupManager**.
open	Opening the info object. Names of fields available via **Event.get**: • target — Reference to the object where the opening occurred. Inherited from **IPopupManager**.

Methods

Name	Returns	Description
close([force])	**vow.Promise**	Closes the info object. Inherited from **IPopupManager**.
destroy()		Disables the info object manager. Inherited from **IPopupManager**.
getData()	Object|null	Returns the data of the info object or 'null'. Inherited from **IPopupManager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getOptions()	**IOptionManager**|null	Returns the options manager or 'null'. Inherited from **IPopupManager**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopupManager**.
getOverlaySync()	**IOverlay**|null	Returns the overlay, if one exists. Inherited from **IPopupManager**.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from **IPopupManager**.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from **IPopupManager**.
open([position[, data[, options]]])	**vow.Promise**	Opens the info object at the specified position. Inherited from **IPopupManager**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopupManager**.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

map.layer

map.layer.Manager

Extends **ILayer**, **IMapObjectCollection**.

Map layers manager.

See Map.layers

Constructor | Fields | Events | Methods

Constructor

```
map.layer.Manager(map[, options])
```

Creates an instance of the class for working with map layers. Each map already has its own layer manager, available as map.layers. Don't instantiate new instances of this class unless necessary.

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map**

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Map
options	—	Type: Object Map layer options. The map.layers options can be used to make settings for layers that have been added to the map. Options for hotspot layers are set using the "hotspotLayer" prefix.
options.trafficImageZIndex	201	Type: Number The z-index of the traffic picture layer.
options.trafficInfoZIndex	1	Type: Number Priority of the hotspot layer of info points.
options.trafficJamZIndex	0	Type: Number Priority of the hotspot layer for traffic jams.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
add	A child object was added. Inherited from **ICollection**.
brightnesschange	Layer brightness change event. Inherited from **ILayer**.
copyrightschange	Event for changes to available copyright information. Inherited from **ILayer**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
parentchange	The parent object reference changed.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
remove	A child object was deleted. Inherited from **ICollection**.
zoomrangechange	Event for changes to available information about the zoom level range. Inherited from **ILayer**.

Methods

Name	Returns	Description
add(object)	**ICollection**	Adds a child object to the collection. Inherited from **ICollection**.
each(callback[, context])		Iterates through all the items in the collection and calls a handler function for each of them.
getBrightness()	Number	Optional method. Inherited from **ILayer**.
getCopyrights(coords, zoom)	**vow.Promise**	Optional method. Requests information about copyrights at the specified point with the specified zoom. Inherited from **ILayer**.
getIterator()	**IIterator**	Returns iterator for the collection. Inherited from **ICollection**.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getZoomRange(point)	**vow.Promise**	Optional method. Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers — the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error. Inherited from **ILayer**.
remove(object)	**ICollection**	Removes a child object from the collection. Inherited from **ICollection**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Methods details

each

```
{} each(callback[, context])
```

Iterates through all the items in the collection and calls a handler function for each of them.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function.
context	—	Type: Object Context for the function.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map.margin

map.margin.Accessor

An object that provides access to the rectangular area in the margins manager.

Constructor | Fields | Events | Methods

Constructor

```
map.margin.Accessor(screenArea)
```

Parameters:

Parameter	Default value	Description
screenArea *	—	Type: Object The rectangular area, which is set in screen coordinates. This area is defined as an object containing information about the margins from the map edges (left, top, right, bottom) and the dimensions of the area (width, height). The values can be set as percentages of the width/height of the map container. Don't instantiate new instances of this class unless necessary.

* Mandatory parameter/option.

Fields

Name	Type	Description
events		Event manager.

Events

Name	Description
change	The rectangular area was changed.
remove	The accessor was deleted from the margins manager.

Methods

Name	Returns	Description
getArea()	Object	Returns the rectangular area.
remove()	**map.margin.Accessor**	Deletes the rectangular area from the margins manager.
setArea(screenArea)	**map.margin.Accessor**	Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

events

```
events
```

Event manager.

Events details

change

The rectangular area was changed.

remove

The accessor was deleted from the margins manager.

Methods details

getArea

```
{Object} getArea()
```

Returns the rectangular area.

remove

```
{map.margin.Accessor} remove()
```

Deletes the rectangular area from the margins manager.

Returns self-reference.

setArea

```
{map.margin.Accessor} setArea(screenArea)
```

Returns self-reference.

Parameters:

Parameter	Default value	Description
screenArea*	—	Type: Object The rectangular area, which is set in screen coordinates. This area is defined as an object containing information about the margins from the map edges (left, top, right, bottom) and the dimensions of the area (width, height). The values can be set as percentages of the width/height of the map container.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
accessor.setArea({
    top: 50,
    left: 50,
    width: 100,
    height: 100
});
```

map.margin.Manager

Extends **IEventEmitter**.

Map margins manager.

A manager for calculating the optimal margins from the edge of the map container.

As input, the manager accepts a definition of rectangular areas in screen coordinates, which designate the occupied areas of the map container.

Margins can be used when setting the current map viewport, in order to get the best display of data on the map. The data on the map never ends up under the occupied areas of the map container. Map#setBound.

Don't instantiate new instances of this class unless necessary.

Constructor | Fields | Events | Methods

Constructor

```
map.margin.Manager()
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map**

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
change	Changes to map margins.

Methods

Name	Returns	Description
addArea(screenArea)	**map.margin.Accessor**	Adding a new rectangular area.
destroy()	**map.margin.Manager**	Destroying the margins manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getMargin()	Number[]	Returns the current margins from the map edges. Values in the array are ordered as: upper margin, right margin, lower margin, left margin.
getOffset()	Number[]	Returns the difference (in pixels) between the geometric center of the map (without margins) and the logical center (with consideration for the margins).
setDefaultMargin(margin)		Sets the default margins from the map edges.

Events details

change

Changes to map margins.

Methods details

addArea

```
{map.margin.Accessor} addArea(screenArea)
```

Adding a new rectangular area.

Returns the object that provides access to the added area. To delete the added rectangular area, you must use this object.

Parameters:

Parameter	Default value	Description
screenArea*	—	Type: Object The rectangular area, which is set in local coordinates. This area is defined as an object containing information about the margins from the map edges (left, top, right, bottom) and the dimensions of the area (width, height). The values can be set as percentages of the width/height of the map container.

* Mandatory parameter/option.

Examples:

1.

```
// Offset from the upper-left corner.
map.margin.addArea({
    left: 0,
    top: 0,
    width: 78,
    height: 101
});
console.log(map.margin.getMargin()); // [0, 0, 0, 78]
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Offset from the lower-right corner.
map.margin.addArea({
    right: 50,
    bottom: 50,
    width: 26,
    height: 25
});
console.log(map.margin.getMargin()); // [0, 0, 75, 0]
```

3.

```
// Setting the margins as a percentage.
map.margin.addArea({
    left: 50,
    bottom: '1%',
    width: 20,
    height: 20
});
console.log(map.margin.getMargin()); // [0, 0, 26, 0]
```

4.

```
// Setting the sizes as a percentage.
map.margin.addArea({
    left: 0,
    top: 50,
    width: '100%',
    height: 25
});
console.log(map.margin.getMargin()); // [75, 0, 0, 0]
```

5.

```
// Setting multiple areas.
map.margin.addArea({
    top: 10,
    left: 10,
    width: 20,
    height: 20
});
map.margin.addArea({
    top: 20,
    right: 40,
    width: 100,
    height: 100
});
map.margin.addArea({
    bottom: 20,
    left: 30,
    width: 120,
    height: 30
});
console.log(map.margin.getMargin()); // [120, 0, 50, 0]
```

destroy

```
{map.margin.Manager} destroy()
```

Destroying the margins manager.

Returns self-reference.

getMargin

```
{Number[]} getMargin()
```

Returns the current margins from the map edges. Values in the array are ordered as: upper margin, right margin, lower margin, left margin.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getOffset

```
{Number[]} getOffset()
```

Returns the difference (in pixels) between the geometric center of the map (without margins) and the logical center (with consideration for the margins).

setDefaultMargin

```
{} setDefaultMargin(margin)
```

Sets the default margins from the map edges.

Parameters:

Parameter	Default value	Description
margin*	—	Type: Number|Number[] Margin values in the form of one, two, or four numbers (similar to setting margins in CSS).

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map.pane

map.pane.Manager

Map pane manager. Each map already has its own pane manager, available as map.panes. Don't create new instances of this class unless necessary. The list of default keys for map panes and their zIndex values:

- 'ground': **pane.MovablePane** (zIndex: 100) — The lowest pane, intended for the map's base layer.
- 'areas': **pane.MovablePane** (zIndex: 200) — Pane for objects with an area, like a polygon.
- 'shadows': **pane.MovablePane** (zIndex: 300) — Pane for the shadows of map objects that are above it.
- 'places': **pane.MovablePane** (zIndex: 400) — Pane for point objects, such as placemarks.
- 'events': **pane.EventsPane** (zIndex: 500) — Pane for listening to map events.
- 'overlaps': **pane.MovablePane** (zIndex: 600) — Pane for objects that don't require hotspots to make them interactive.
- 'balloon': **pane.MovablePane** (zIndex: 700) — Balloon pane.
- 'outerBalloon': **pane.MovablePane** (zIndex: 800) — External balloon pane.
- 'controls': **pane.StaticPane** (zIndex: 900) — Map controls pane.
- 'hint': **pane.StaticPane** (zIndex: 1100) — Hint pane.
- 'outerHint': **pane.StaticPane** (zIndex: 1200) — External hint pane.

See Map.panes

Constructor | Methods

Constructor

```
map.pane.Manager(map)
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Map.

* Mandatory parameter/option.

Example:

```
// Adding a watermark on top of the map container.
// To do this, we will change the event container's background.
map.panes.get('events').getElement().style.backgroundImage = 'my/background/
image';
```

Methods

Name	Returns	Description
append(key, pane)		Adds a new pane to the map. The key of the pane being added must be unique within the current set of keys for the map panes.
destroy()		Destructor.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
get(key)	IPane|null	Returns either the map pane with the given key, or null, if the requested pane is not available on the map.
getLower()	String	Takes the keys of map panes as arguments and returns the key of the lowest pane in the received set. If no key was specified, the search is performed on the entire set of map panes.
getUpper()	String	Takes the keys of map panes as arguments and returns the key of the top pane in the received set. If no key was specified, the search is performed on the entire set of map panes.
insertBefore(key, pane, referenceKey)		Inserts a new pane in front of another map pane. The key of the pane being added must be unique within the current set of keys for the map panes.
remove(pane)		Deletes a pane from the map.

Methods details

append

```
{} append(key, pane)
```

Adds a new pane to the map. The key of the pane being added must be unique within the current set of keys for the map panes.

Parameters:

Parameter	Default value	Description
key*	—	Type: String The key of the pane being added.
pane*	—	Type: IPane The pane being added.

* Mandatory parameter/option.

destroy

```
{} destroy()
```

Destructor.

get

```
{IPane|null} get(key)
```

Returns either the map pane with the given key, or null, if the requested pane is not available on the map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
key*	—	Type: String The pane key.

* Mandatory parameter/option.

getLower

```
{String} getLower()
```

Takes the keys of map panes as arguments and returns the key of the lowest pane in the received set. If no key was specified, the search is performed on the entire set of map panes.

Returns the key for the lowest map pane.

getUpper

```
{String} getUpper()
```

Takes the keys of map panes as arguments and returns the key of the top pane in the received set. If no key was specified, the search is performed on the entire set of map panes.

Returns the key for the uppermost map pane.

insertBefore

```
{} insertBefore(key, pane, referenceKey)
```

Inserts a new pane in front of another map pane. The key of the pane being added must be unique within the current set of keys for the map panes.

Parameters:

Parameter	Default value	Description
key*	—	Type: String The key of the pane being added.
pane*	—	Type: **IPane** The pane being added.
referenceKey*	—	Type: String The key of the pane to insert the new pane in front of.

* Mandatory parameter/option.

remove

```
{} remove(pane)
```

Deletes a pane from the map.

Parameters:

Parameter	Default value	Description
pane*	—	Type: **IPane**

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The pane being deleted.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

map.ZoomRange

Extends **IEventEmitter**.

Map zoom level manager. Each map already has its own zoom level manager, available as map.zoomRange. Don't instantiate new instances of this class unless necessary.

See Map.zoomRange

Constructor | Fields | Events | Methods

Constructor

```
map.ZoomRange(map, constraints)
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Map.
constraints *	—	Type: Number[] An array containing the minimum and maximum map zooms.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
change	Changes occurred in the map zoom range.

Methods

Name	Returns	Description
get([coords])	**vow.Promise**	Checks the available range of zoom levels at the specified point.
getCurrent()	Number[]	Returns the current (last received) values of the minimum and maximum map zoom levels.

Events details

change

Changes occurred in the map zoom range.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

get

```
{vow.Promise} get([coords])
```

Checks the available range of zoom levels at the specified point.

Returns a Promise object, which will be resolved and will get an array of two numbers as a parameter — the maximum and minimum zoom at the given point.

Parameters:

Parameter	Default value	Description
coords	—	Type: Number[] Coordinates of a point. If omitted, the current map center is used.

Example:

```
// Finding the coordinates of the Yandex office and showing it on the map
// at the maximum possible zoom level.
ymaps.geocode('Moscow, Lev Tolstoy street, 16').then(function (res) {
    var coords = res.geoObjects.get(0).geometry.getCoordinates();
    map.zoomRange.get(coords).then(function (range) {
        map.setCenter(coords, range[1]);
    });
});
```

getCurrent

```
{Number[]} getCurrent()
```

Returns the current (last received) values of the minimum and maximum map zoom levels.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

MapEvent

Extends **Event**.

Object that describes an event that took place on the map. Names of fields that are available via the **Event.get** method:

- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- Coordinates of the event in global pixels from the upper-left corner of the page (also available by the name "position").
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Constructor | Methods

Constructor

```
MapEvent(originalEvent[, sourceEvent])
```

Parameters:

Parameter	Default value	Description
originalEvent *	—	Type: Object Data associated with the event. It should contain the "map" field referencing the map on which the event has occurred.
sourceEvent	—	Type: **IEvent** Source event.

* Mandatory parameter/option.

Example:

```
// Opening a balloon at the point where the map was clicked
map.events.add('click', function (e) {
    map.balloon.open(e.get('coords'), 'Click!');
});
```

Methods

Name	Returns	Description
allowMapEvent()		Allows the propagation of the event to the map. Inherited from **IEvent**.
callMethod(name)	Object	Calls the specified method. The operation is equivalent to searching fields via "get" and making a call that passes originalEvent as context. All arguments after the first one are passed as parameters to the method being called. Inherited from **Event**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
get(name)	Object	Returns the field value from originalEvent. originalEvent always has the following fields: • type — String event type. • target — Reference to the object that generated the event. . Inherited from **Event**.
getSourceEvent()	**IEvent**|null	Returns source event. Inherited from **IEvent**.
isDefaultPrevented()	Boolean	Checks whether the default reaction to the event is canceled in the Yandex.Maps API event system. Inherited from **Event**.
isImmediatePropagationStopped()	Boolean	Checks whether event propagation is stopped in the Yandex.Maps API event system. Inherited from **Event**.
isMapEventAllowed()	Boolean	Returns true if the map event is enabled. Inherited from **IEvent**.
isPropagationStopped()	Boolean	Checks whether event propagation up the hierarchy of objects and collections is stopped in the Yandex.Maps API event system. Inherited from **Event**.
preventDefault()		Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree. Inherited from **Event**.
stopImmediatePropagation()		Stops event propagation in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree. Inherited from **Event**.
stopPropagation()		Stops event propagation up the hierarchy of objects and collections in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree. Inherited from **Event**.

MapType

Map type.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor | Methods

Constructor

```
MapType(name, layers)
```

Creates an instance of the map type.

Parameters:

Parameter	Default value	Description
name *	—	Type: String Type name.
layers *	—	Type: Function[]|String[] Array containing constructors for layers or keys.

* Mandatory parameter/option.

Example:

```
// Creating a custom map type that consists of satellite images from MapQuest
// with the Yandex.Hybrid overlay.

// Class of MapQuest tiles
var MQLayer = function () {
    var layer = new ymaps.Layer('http://oatile%d.mqcdn.com/naip//%z/%x/%y.jpg');
    // Copyrights
    layer.getCopyrights = function () {
        return ymaps.vow.resolve('Data, imagery and map information provided by
MapQuest, Open Street Map and contributors, CC-BY-SA');
    };
    // Range of available zoom levels
    layer.getZoomRange = function () {
        return ymaps.vow.resolve([0, 18]);
    };

    return layer;
};
// Adding a layer with the key
ymaps.layer.storage.add('mq#aerial', MQLayer);
// Creating a map type that consists of the layers 'mq#aerial' and
'yandex#skeleton'
var myMapType = new ymaps.MapType('MQ + Ya', ['mq#aerial', 'yandex#skeleton']);
// Adding it to the map type storage
ymaps.mapType.storage.add('mq_ya#hybrid', myMapType);
// Now we can set our map type for any map
map.setType('mq_ya#hybrid');
```

Methods

Name	Returns	Description
getLayers()	Function[]|String[]	Returns a list of layers for the given map type — an array of constructors or keys for layers.
getName()	String	Returns name of the map type.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

getLayers

```
{Function[]|String[]} getLayers()
```

Returns a list of layers for the given map type — an array of constructors or keys for layers.

getName

```
{String} getName()
```

Returns name of the map type.

Parameters:

Parameter	Default value	Description
map*	—	Type:

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

mapType

mapType.storage

Static object.

Instance of **util.Storage**

Storage for map types.

Methods

Example:

```
// Adding a custom map type

// Let's say we have our own map layer
var myLayer = function () {
    return new ymaps.Layer('http://some.server/?%c');
}
// Adding it to the layer storage
ymaps.layer.storage.add('my#layer', myLayer);
// Creating our own map type, consisting of a single layer
var myType = new ymaps.MapType('My map type', ['my#layer']);
// Adding it to the map type storage
ymaps.mapType.storage.add('my#mapType', myType);
// Now we can assign our map type to a map
map.setType('my#type');
```

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

meta

Static object.

Information about the API.

Fields

Fields

Name	Type	Description
coordinatesOrder	String	The order of coordinates used in the API. Possible values: • latlong — [latitude, longitude] • longlat — [longitude, latitude] Set by the "coordorder" GET parameter when enabling the API. More information about API setup parameters
countryCode	String	Two-letter country code. Returned in ISO 3166-1 format. Set by the "lang" GET parameter when enabling the API. More information about API setup parameters
languageCode	String	Two-letter language code. Returned in ISO 639-1 format. Set by the "lang" GET parameter when enabling the API. More information about API setup parameters
mode	String	Yandex.Maps API mode. Possible values: • release • debug Set by the "mode" GET parameter when enabling the API. More information about API setup parameters
ns	Object	Link to the Yandex.Maps API namespace. It has a value independent of the "ns" parameter value when enabling the API.
version	String	Version of the Yandex.Maps API.

Fields details

coordinatesOrder

```
{String} coordinatesOrder
```

The order of coordinates used in the API. Possible values:

- latlong — [latitude, longitude]
- longlat — [longitude, latitude]

Set by the "coordorder" GET parameter when enabling the API.

More information about API setup parameters

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

countryCode

```
{String} countryCode
```

Two-letter country code. Returned in ISO 3166-1 format.

Set by the "lang" GET parameter when enabling the API.

More information about API setup parameters

languageCode

```
{String} languageCode
```

Two-letter language code. Returned in ISO 639-1 format.

Set by the "lang" GET parameter when enabling the API.

More information about API setup parameters

mode

```
{String} mode
```

Yandex.Maps API mode. Possible values:

- release
- debug

Set by the "mode" GET parameter when enabling the API.

More information about API setup parameters

ns

```
{Object} ns
```

Link to the Yandex.Maps API namespace.

It has a value independent of the "ns" parameter value when enabling the API.

version

```
{String} version
```

Version of the Yandex.Maps API.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

modules

Static object.

The module system that the Yandex.Maps API is based on.

The Yandex.Maps API consists of a large number of interconnected modules. A module is a programming unit. For example, a class, a specific realization of a class, a static object, or a function. The module system guarantees that when initializing a particular module, all the modules it needs will already be initialized.

The module system provides asynchronous access, since it may be necessary to load missing modules.

You can add your own modules to the module system.

modules.define

Static function.

Defining a module in the module system.

Note:
We recommend creating your custom modules in your own namespace, to avoid accidentally replacing the modules that are necessary for the API to work.

Returns self-reference.

```
{ modules } modules.define(module[, depends, resolveCallback[, context]])
```

Parameters:

Parameter	Default value	Description
module*	—	Type: String Module name.
depends	—	Type: String[] Array of names of necessary modules. This argument may be omitted.
resolveCallback*	—	Type: Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Function that defines the module. The first argument in resolveCallback will be a provide function, to which you will need to pass a module. The provide function call can be delayed. Subsequent arguments are the modules specified in the dependencies. The order of the modules will match the order in the depends array. If a module cannot be resolved, the module system must be notified. This can be done by passing a second argument to the provide function. The second argument will be passed to errorCallback and promise as an error in the module request. As a result, the module can be requested again.
context	—	Type: Object Context for the function.

* Mandatory parameter/option.

Examples:

1.

```
// Defining a custom module. 'plugin.*' is a custom namespace.
ymaps.modules.define('CustomModule', function (provide) {
    var CustomModule = function (defValue) {
        this.field = defValue;
    };
    provide(CustomModule);
});
// Requesting modules
ymaps.modules.require(['CustomModule'])
    .spread(
        function (CustomModule) {
            // ...
        },
        this
   );
```

2.

```
// Defining a custom asynchronous module.
ymaps.modules.define('CustomAsyncModule', function (provide) {
    // For the module to work, we must load an external script.
    $.getScript( "ajax/test.js")
        .done(function( script, textStatus ) {
            function CustomAsyncModule () {
                // ...
            }
        // Calling the provide function after loading the script.
        provide(CustomAsyncModule);
    });
});
// Requesting modules
ymaps.modules.require(['CustomAsyncModule'])
    .spread(
        function (CustomAsyncModule) {
            // ...
        },
        this
    );
```

3.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Creating a custom asynchronous module with consideration for an error.
ymaps.modules.define('plugin.CustomModule', function (provide) {
    $.getScript( "ajax/test.js" )
        .done(function( script, textStatus ) {
            // ... Processing successful script loading.
            provide({ a: 1 });
        })
        .fail(function( jqxhr, settings, exception ) {
           // Notifying the module system that the module can't be prepared right
now.
            provide(null, new Error('Error when loading'));
        });
});

// Requesting modules with error handling.
ymaps.modules.require(['plugin.CustomModule'])
    .spread(
        function (CustomModule) {
            // ...
        },
        function (error) {
            console.log(error.messsage); // "Error when loading".
            // Code for handling the module loading error. You can request the
module one more time.
        },
        this
    );
```

4.

```
// Defining a custom module with dependencies.
ymaps.modules.define('CustomLayoutModule', [
     'templateLayoutFactory',
     'layout.storage'
], function (provide, templateLayoutFactory, layoutStorage) {
    var customLayoutClass = templateLayoutFactory.createClass(
        '<div>My Layout</div>',
        {
            build: function () {
                 customLayoutClass.superclass.build.call(this);
                 // ...
            },
            clear: function () {
                 // ...
                 customLayoutClass.superclass.clear.call(this);
            }
        }
    );
    layoutStorage.add('customLayout', customLayoutClass);
    provide(customLayoutClass);
});
// Requesting a custom layout
ymaps.modules.require(['CustomLayoutModule'])
    .spread(
        function (CustomLayoutModule) {
            // ...
        },
        this
   );
```

modules.isDefined

Static function.

Checking the module's availability by name.

Returns true if the module is defined, false if not.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
{ Boolean } modules.isDefined(module)
```

Parameters:

Parameter	Default value	Description
module*	—	Type: String Module

* Mandatory parameter/option.

Example:

```
if (ymaps.modules.isDefined('plugin.CustomModule')) {
    ymaps.modules.require(['plugin.CustomModule'])
        .spread(
            function (CustomModule) {
                // ...
            }
        );
}
```

modules.require

Static function.

Request to get modules.

Returns a promise object that is confirmed by the requested modules. Or it is rejected, if an error occurred. For example, one of the requested modules is missing in the module system.

```
{ vow.Promise } modules.require(modules[, successCallback[, errorCallback[, context]]])
```

Parameters:

Parameter	Default value	Description
modules*	—	Type: String|String[] Module name or array of module names.
successCallback	—	Type: Function The function that is called after receiving all the modules. The requested entities will be passed to the function as arguments. The order of the arguments will match the order in the modules array.
errorCallback	—	Type: Function The function that will be called in case of an error.
context	—	Type: Object The execution context of the callback function.

* Mandatory parameter/option.

Examples:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

1.

```
// Requesting the 'Map' and 'Placemark' modules using a promise object.
ymaps.modules.require(['Map', 'Placemark'])
    .spread(
        function (Map, Placemark) {
            var myMap = new Map("map", {
                center: [55.72, 37.64],
                zoom: 5
            });
            myMap.geoObjects.add(
                new Placemark(myMap.getCenter())
            );
        },
        function (error) {
            // Error handling.
        },
        this
   );
```

2.

```
// Requesting the 'Map' module.
ymaps.modules.require('Map')
    .spread(
        function (Map) {
            var myMap = new Map("map", {
                center: [55.72, 37.64],
                zoom: 5
            });
        },
        this
    );
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Monitor

Object that tracks changes to certain data fields in the specified data manager. It can also be used for tracking changes to options.

Constructor | Methods

Constructor

```
Monitor(dataManager)
```

Parameters:

Parameter	Default value	Description
dataManager *	—	Type: IDataManager| IOptionManager Data manager.

* Mandatory parameter/option.

Example:

```
// Tracking changes to placemark options
var placemark = new ymaps.Placemark([0, 0]),
    optionMonitor = new ymaps.Monitor(placemark.options);
optionMonitor.add("cursor", function (newValue) {
    alert("cursor: " + newValue);
});
myMap.geoObjects.add(placemark);
// Outputs the string "cursor: arrow".
myMap.options.set({
    geoObjectCursor: "arrow"
});
```

Methods

Name	Returns	Description
add(name, changeCallback[, context[, params]])	Monitor	Enables monitoring particular fields or a group of data fields.
forceChange()	Monitor	Initializes checking for changes to values of the monitored data fields.
get(name)	Object	Returns the current value of one of the monitored data fields.
remove(name)	Monitor	Disables monitoring particular fields or a group of data fields.
removeAll()	Monitor	Disables monitoring for all data fields.

Methods details

add

```
{Monitor} add(name, changeCallback[, context[, params]])
```

Enables monitoring particular fields or a group of data fields.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
name*	—	Type: String|String[] Name or array of names of data fields that monitoring is being set up for.
changeCallback*	—	Type: Function Handler for changes to data fields, or one of the data fields from a group.
context	—	Type: Object Context for the handler of data changes and for options handlers.
params	—	Type: Object Optional parameters.
params.compareCallback	—	Type: Function Handler that compares the old and new values of data fields. Takes two arguments: the old value and the new value. Lower priority relative to handlers set using the compareCallbacks parameter.
params.compareCallbacks	—	Type: Object Hash in the format {data field name: reference to handler}. This parameter is for setting individual handlers for comparing values for various data fields in a group.
params.defaultValue	—	Type: Object The default value that is used if the data field is not defined.
params.defaultValues	—	Type: Object Hash in the format {data field name: default value}. This parameter is for setting individual default values for various data fields in a group.
params.resolveCallback	—	Type: Function Handler that allows the data field value. Takes two arguments; the data field name and the reference to the data manager. Lower priority relative to handlers set using the resolveCallbacks parameter.
params.resolveCallbacks	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Hash in the format {data field name: reference to handler}. This parameter is for setting individual handlers for allowing values for various data fields in a group.

* Mandatory parameter/option.

forceChange

```
{Monitor} forceChange()
```

Initializes checking for changes to values of the monitored data fields.

Returns self-reference.

get

```
{Object} get(name)
```

Returns the current value of one of the monitored data fields.

Parameters:

Parameter	Default value	Description
name*	—	Type: String Data field name.

* Mandatory parameter/option.

remove

```
{Monitor} remove(name)
```

Disables monitoring particular fields or a group of data fields.

Returns self-reference.

Parameters:

Parameter	Default value	Description
name*	—	Type: String|String[] Name or array of names of data fields that monitoring is being disabled for.

* Mandatory parameter/option.

removeAll

```
{Monitor} removeAll()
```

Disables monitoring for all data fields.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multiRouter

multiRouter.driving

multiRouter.driving.Path

Note:
The constructor of the multiRouter.driving.Path class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of a path on a multi-stop driving route. A single route can contain several paths, and each path connects two waypoints.

Fields | Events | Methods

Creates a representation to display a path of the multi-stop driving route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.driving.PathModel**	Data model for the multi-stop route's path.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**data.Manager**	Multi-stop route's path data. The following fields are available: • index: Integer — The sequential number of the path in the multi-stop route's corresponding route. • type: String — Route type identifier, which takes the value "driving" for automobile routes. • distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds. • durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the path (in seconds) considering traffic. • coordinates: Number[][] — Coordinates of all points on the path. • encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
update	Updating the path rendering. Instance of the Event class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getSegments()	GeoObjectCollection	Returns the child collection of segments that the path consists of.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

model

```
{multiRouter.driving.PathModel} model
```

Data model for the multi-stop route's path.

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Multi-stop route's path data. The following fields are available:

- index: Integer — The sequential number of the path in the multi-stop route's corresponding route.
- type: String — Route type identifier, which takes the value "driving" for automobile routes.
- distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the path (in seconds) considering traffic.
- coordinates: Number[][] — Coordinates of all points on the path.
- encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.

Events details

update

Updating the path rendering. Instance of the **Event** class.

Methods details

getSegments

```
{GeoObjectCollection} getSegments()
```

Returns the child collection of segments that the path consists of.

multiRouter.driving.PathModel

Note:
The constructor of the multiRouter.driving.PathModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model for the path of a driving route. A single route can contain several paths, and each path connects two waypoints.

Fields | Events | Methods

Creates the data model of a driving route path.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	data.Manager	Multi-stop route's path data. The following fields are available: • index: Integer — The sequential number of the path in the multi-stop route's corresponding route. • type: String — Route type identifier, which takes the value "driving" for automobile routes. • distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds. • durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the path (in seconds) considering traffic. • coordinates: Number[][] — Coordinates of all points on the path. • encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.
route	multiRouter.driving.RouteModel	Reference to the parent model of the route.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class. Names of fields that are available via the Event.get method: • segmentsChange: Boolean — Flag for whether the set of segments is changed

Methods

Name	Returns	Description
destroy()		Destroys a model.
getSegments()	multiRouter.driving.SegmentModel[]	Returns array of path segments.
getType()	String	Returns ID of the route path type. For automobile routes, it returns the string "driving".
update(pathJson)		Updates the state of the model.

Fields details

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Multi-stop route's path data. The following fields are available:

- index: Integer — The sequential number of the path in the multi-stop route's corresponding route.
- type: String — Route type identifier, which takes the value "driving" for automobile routes.
- distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the path (in seconds) considering traffic.
- coordinates: Number[][] — Coordinates of all points on the path.
- encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.

route

```
{multiRouter.driving.RouteModel} route
```

Reference to the parent model of the route.

Events details

update

Updating the model with new data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- segmentsChange: Boolean — Flag for whether the set of segments is changed

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getSegments

```
{multiRouter.driving.SegmentModel[]} getSegments()
```

Returns array of path segments.

getType

```
{String} getType()
```

Returns ID of the route path type. For automobile routes, it returns the string "driving".

update

```
{} update(pathJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
pathJson*	—	Type: Object JSON data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

multiRouter.driving.Route

Note:
The constructor of the multiRouter.driving.Route class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of an individual automobile route. A multi-stop route can consist of several individual routes.

Fields | Events | Methods

Creates a representation to display a single driving route.

Fields

Name	Type	Description
balloon	**IMultiRouterRouteBalloon**	Balloon of a route.
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.driving.RouteModel**	The data model of an individual route.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**data.Manager**	The route data. The following fields are available: • index: Integer — The ordinal number of the route in a multi-stop route. • type: String — Route type identifier, which takes the value "driving" for automobile routes. • blocked: Boolean — Indicates that the route contains blocked sections. • distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the route in seconds. • durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the route (in seconds) considering traffic. • boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
balloonclose	Closing the balloon.
balloonopen	Opening the balloon.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
update	Updating the route rendering. Instance of the Event class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getPaths()	GeoObjectCollection	Returns a child collection of paths that make up the route.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

balloon

```
{IMultiRouterRouteBalloon} balloon
```

Balloon of a route.

model

```
{multiRouter.driving.RouteModel} model
```

The data model of an individual route.

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

The route data. The following fields are available:

- index: Integer — The ordinal number of the route in a multi-stop route.
- type: String — Route type identifier, which takes the value "driving" for automobile routes.
- blocked: Boolean — Indicates that the route contains blocked sections.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the route in seconds.
- durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the route (in seconds) considering traffic.
- boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Events details

balloonclose

Closing the balloon.

balloonopen

Opening the balloon.

update

Updating the route rendering. Instance of the **Event** class.

Methods details

getPaths

```
{GeoObjectCollection} getPaths()
```

Returns a child collection of paths that make up the route.

multiRouter.driving.RouteModel

Note:
The constructor of the multiRouter.driving.RouteModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model for an individual driving route. A multi-stop route can consist of several individual routes.

Fields | Events | Methods

Creates the data model of an individual driving route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
multiRoute	multiRouter.MultiRouteModel	Reference to the parent model of a multi-stop route.
properties	data.Manager	The route data. The following fields are available: • index: Integer — The ordinal number of the route in a multi-stop route. • type: String — Route type identifier, which takes the value "driving" for automobile routes. • blocked: Boolean — Indicates that the route contains blocked sections. • distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the route in seconds. • durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the route (in seconds) considering traffic. • boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class. Names of fields that are available via the Event.get method: • pathsChange: Boolean — Flag for whether the set of paths is changed.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getPaths()	multiRouter.driving.PathModel[]	Returns array of route paths.
getType()	String	Returns ID of the route type. For automobile routes, it returns the string "driving".
update(routeJson)		Updates the state of the model.

Fields details

multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

The route data. The following fields are available:

- index: Integer — The ordinal number of the route in a multi-stop route.
- type: String — Route type identifier, which takes the value "driving" for automobile routes.
- blocked: Boolean — Indicates that the route contains blocked sections.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the route in seconds.
- durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the route (in seconds) considering traffic.
- boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Events details

update

Updating the model with new data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pathsChange: Boolean — Flag for whether the set of paths is changed.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getPaths

```
{multiRouter.driving.PathModel[]} getPaths()
```

Returns array of route paths.

getType

```
{String} getType()
```

Returns ID of the route type. For automobile routes, it returns the string "driving".

update

```
{} update(routeJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
routeJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multiRouter.driving.Segment

Note:
The constructor of the multiRouter.driving.Segment class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of a segment on the automobile route. A segment of a driving route is a part of the path from one manoeuver to another.

Fields | Events | Methods

Creates a representation to display a segment on the driving route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.driving.SegmentModel**	Data model for a segment.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**data.Manager**	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • type: String — Segment type identifier, which takes the value "driving" for automobile segments. • street: String — Text description of the street that the segment goes along. • action: Object — An object with the "text" and "value" fields that describes the final maneuver of the segment. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds. • durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the segment (in seconds) considering traffic. • text: String — Text description of the segment. • viaPoints: Integer[] — Indexes of throughpoints lying on the segment. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
update	Updating the segment rendering. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

model

```
{multiRouter.driving.SegmentModel} model
```

Data model for a segment.

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- type: String — Segment type identifier, which takes the value "driving" for automobile segments.
- street: String — Text description of the street that the segment goes along.
- action: Object — An object with the "text" and "value" fields that describes the final maneuver of the segment.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the segment (in seconds) considering traffic.
- text: String — Text description of the segment.
- viaPoints: Integer[] — Indexes of throughpoints lying on the segment.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the segment rendering. Instance of the **Event** class.

multiRouter.driving.SegmentModel

Note:
The constructor of the multiRouter.driving.SegmentModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model for a segment on the path of a driving route. A segment of a driving route is a part of the path from one manoeuver to another.

Fields | Events | Methods

Creates the data model for a segment on the path of a driving route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**geometry.base.LineString**	Geometry of a segment.
path	**multiRouter.driving.PathModel**	Reference to the parent model of the path.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	data.Manager	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • type: String — Segment type identifier, which takes the value "driving" for automobile segments. • street: String — Text description of the street that the segment goes along. • action: Object — An object with the "text" and "value" fields that describes the final maneuver of the segment. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds. • durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the segment (in seconds) considering traffic. • text: String — Text description of the segment. • viaPoints: Integer[] — Indexes of throughpoints lying on the segment. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getType()	String	Returns ID of the segment type. For segments of automobile routes, it returns the string "driving".
getViaPoints()	multiRouter.ViaPointModel[]	Returns an array of throughpoints on the segment.
update(segmentJson)		Updates the state of the model.

Fields details

geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

path

```
{multiRouter.driving.PathModel} path
```

Reference to the parent model of the path.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- type: String — Segment type identifier, which takes the value "driving" for automobile segments.
- street: String — Text description of the street that the segment goes along.
- action: Object — An object with the "text" and "value" fields that describes the final maneuver of the segment.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- durationInTraffic: Object — An object with the "text" and "value" fields that specifies the travel time of the segment (in seconds) considering traffic.
- text: String — Text description of the segment.
- viaPoints: Integer[] — Indexes of throughpoints lying on the segment.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the model with new data. Instance of the **Event** class.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getType

```
{String} getType()
```

Returns ID of the segment type. For segments of automobile routes, it returns the string "driving".

getViaPoints

```
{multiRouter.ViaPointModel[]} getViaPoints()
```

Returns an array of throughpoints on the segment.

update

```
{} update(segmentJson)
```

Updates the state of the model.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
segmentJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multiRouter.Editor

Extends **ICustomizable**, **IEventEmitter**.

Multi-route editor.

Constructor | Fields | Events | Methods

Constructor

```
multiRouter.Editor(multiRoute[, state[, options]])
```

Creates a multi-route editor.

Parameters:

Parameter	Default value	Description
multiRoute *	—	Type: **multiRouter.MultiRoute** The multi-route being edited.
state	—	Type: Object Object which describes the initial state of the editor. For a list of available fields, see **multiRouter.Editor.state**.
options	—	Type: Object Options.
options.drawCursor	—	Type: Object Name of the cursor to use in waypoint drawing mode.
options.drawOver	true	Type: Object Allows putting points on top of map objects in waypoint drawing mode.
options.midPointsType	"way"	Type: String Specifies the type of points to add while dragging the marker which appears when hovering the mouse cursor over the active route. Accepts one of the following string values: • "way" — Add waypoints. • "via" — Add throughpoints. See also the description of the addMidPoints field for the state manager **multiRouter.Editor.state**.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
		Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**data.Manager**	The state manager of the multi-stop route editor. Available fields: • addWayPoints: Boolean — Allows adding new waypoints by clicking on the map. Default value: false. • dragWayPoints: Boolean — Allows dragging existing waypoints. Default value: true. • removeWayPoints: Boolean — Allows deleting waypoints by double-clicking on them. Default value: false. • dragViaPoints: Boolean — Allows dragging existing throughpoints. Default value: true. • removeViaPoints: Boolean — Allows deleting throughpoints by double-clicking on them. Default value: true. • addMidPoints: Boolean — Allows adding intermediate throughpoints or waypoints by dragging the placemark which appears when you hover the mouse over the active route. The type of point to add is set by the option midPointsType. Default value: true.

Events

Name	Description
beforemidpointadd	Event preceding the "midpointadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • coords: Number[] — Coordinates for adding a midpoint. • pointType: String — Type ID for the midpoint being added. • insertIndex: Integer — The insertion index of the midpoint in the set of reference points for the multi-stop route. Names of methods that are accessible via **Event.callMethod**: • setPointType — The method that allows you to set the type of the added point. Takes a string identifier of the type as the argument (see the description of the midPointsType option). • setInsertIndex — The method that allows you to adjust the insertion index of a midpoint before it is applied. Takes the new index as the argument. If you call the **Event.preventDefault** method of this event, the next midpoint to add, as well as the "midpointadd" event, will be canceled.
beforemidpointdrag	The event preceding the "midpointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, the subsequent "midpointdrag" event will be canceled.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
beforemidpointpinshow	The event preceding the "midpointpinshow" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pin: **Placemark** — Reference to the placemark object. • globalPixels: Number[] — Global pixel coordinates of the placemark. • segment: **multiRouter.driving.Segment** — Reference to the segment of the route where the placemark should appear. If the **Event.preventDefault** method is called for this event, a subsequent "midpointpinshow" event will be canceled and the placemark will be hidden.
beforeviapointdrag	The event preceding the "viapointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. • viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "viapointdrag" event will be canceled.
beforeviapointdragstart	The event preceding the "viapointdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent: **DomEvent** — The source DOM event, if there is one. • viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "viapointdragstart" event, will be canceled.
beforeviapointremove	The event preceding the "viapointremove" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being deleted. If the **Event.preventDefault** method is called for this event, deletion of a throughpoint, as well as a subsequent "viapointremove" event, will be canceled.
beforewaypointadd	Event preceding the "waypointadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • coords: Number[] — Coordinates of the added waypoint. Names of methods that are accessible via **Event.callMethod**: • setCoords — Method that allows you to adjust the coordinates of the waypoint being added. It takes an argument with the new pixel shift in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, addition of a waypoint, as well as a subsequent "waypointadd" event, will be canceled.
beforewaypointdrag	The event preceding the "waypointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. • wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "waypointdrag" event will be canceled.
beforewaypointdragstart	The event preceding the "waypointdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent: **DomEvent** — The source DOM event, if there is one. • wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "waypointdragstart" event, will be canceled.
beforewaypointremove	The event preceding the "waypointremove" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being deleted. If the **Event.preventDefault** method is called for this event, removal of the waypoint, as well as a subsequent "waypointremove" event, will be canceled.
midpointadd	Adding a midpoint (intermediate point) to the route. The point type is determined by the value of the midPointsType option. Instance of the Event class. Names of fields that are available via the **Event.get** method: • coords: Number[] — Coordinates for adding a midpoint. • pointType: String — Type ID for the midpoint being added. • insertIndex: Integer — The insertion index of the midpoint in the set of reference points for the multi-stop route.
midpointdrag	Dragging the added midpoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset: Number[] — Pixel shift along the axes at the given step.
midpointdragend	End of dragging the added midpoint. Instance of the **Event** class.
midpointpinshow	Displaying the draggable placemark when you hover over the active route. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pin: **Placemark** — Reference to the placemark object. • globalPixels: Number[] — Global pixel coordinates of the placemark. • segment: **multiRouter.driving.Segment** — Reference to the segment of the route where the placemark appeared.
optionschange	Change to the object options. Inherited from **ICustomizable**.
viapointdrag	Throughpoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. • viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
viapointdragend	End of throughpoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • viaPoint: multiRouter.ViaPoint — A reference to the throughpoint object being dragged.
viapointdragstart	Start of throughpoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • domEvent: DomEvent — The source DOM event, if there is one. • viaPoint: multiRouter.ViaPoint — A reference to the throughpoint object being dragged.
viapointremove	Deleting a throughpoint. Instance of the Event class. Names of fields that are available via the Event.get method: • viaPoint: multiRouter.ViaPoint — A reference to the deleted throughpoint object.
waypointadd	Adding a waypoint. Instance of the Event class. Names of fields that are available via the Event.get method: • coords: Number[] — Coordinates of the added waypoint.
waypointdrag	Waypoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. • wayPoint: multiRouter.WayPoint — A reference to the waypoint object being dragged.
waypointdragend	End of waypoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • wayPoint: multiRouter.WayPoint — A reference to the waypoint object being dragged.
waypointdragstart	Start of waypoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • domEvent: DomEvent — The source DOM event, if there is one. • wayPoint: multiRouter.WayPoint — A reference to the waypoint object being dragged.
waypointremove	Deleting a waypoint. Instance of the Event class. Names of fields that are available via the Event.get method: • wayPoint: multiRouter.WayPoint — A reference to the deleted waypoint object.

Methods

Name	Returns	Description
destroy()		Destroys the multi-route editor.
getMultiRoute()	multiRouter.MultiRoute	Returns a reference to the multi-route being edited.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

state

```
{data.Manager} state
```

The state manager of the multi-stop route editor.

Available fields:

- addWayPoints: Boolean — Allows adding new waypoints by clicking on the map. Default value: false.
- dragWayPoints: Boolean — Allows dragging existing waypoints. Default value: true.
- removeWayPoints: Boolean — Allows deleting waypoints by double-clicking on them. Default value: false.
- dragViaPoints: Boolean — Allows dragging existing throughpoints. Default value: true.
- removeViaPoints: Boolean — Allows deleting throughpoints by double-clicking on them. Default value: true.
- addMidPoints: Boolean — Allows adding intermediate throughpoints or waypoints by dragging the placemark which appears when you hover the mouse over the active route. The type of point to add is set by the option midPointsType. Default value: true.

Events details

beforemidpointadd

Event preceding the "midpointadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- coords: Number[] — Coordinates for adding a midpoint.
- pointType: String — Type ID for the midpoint being added.
- insertIndex: Integer — The insertion index of the midpoint in the set of reference points for the multi-stop route.

Names of methods that are accessible via **Event.callMethod**:

- setPointType — The method that allows you to set the type of the added point. Takes a string identifier of the type as the argument (see the description of the midPointsType option).
- setInsertIndex — The method that allows you to adjust the insertion index of a midpoint before it is applied. Takes the new index as the argument.

If you call the **Event.preventDefault** method of this event, the next midpoint to add, as well as the "midpointadd" event, will be canceled.

beforemidpointdrag

The event preceding the "midpointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, the subsequent "midpointdrag" event will be canceled.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

beforemidpointpinshow

The event preceding the "midpointpinshow" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pin: **Placemark** — Reference to the placemark object.
- globalPixels: Number[] — Global pixel coordinates of the placemark.
- segment: **multiRouter.driving.Segment** — Reference to the segment of the route where the placemark should appear.

If the **Event.preventDefault** method is called for this event, a subsequent "midpointpinshow" event will be canceled and the placemark will be hidden.

beforeviapointdrag

The event preceding the "viapointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.
- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "viapointdrag" event will be canceled.

beforeviapointdragstart

The event preceding the "viapointdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent: **DomEvent** — The source DOM event, if there is one.
- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "viapointdragstart" event, will be canceled.

beforeviapointremove

The event preceding the "viapointremove" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being deleted.

If the **Event.preventDefault** method is called for this event, deletion of a throughpoint, as well as a subsequent "viapointremove" event, will be canceled.

beforewaypointadd

Event preceding the "waypointadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- coords: Number[] — Coordinates of the added waypoint.

Names of methods that are accessible via **Event.callMethod**:

- setCoords — Method that allows you to adjust the coordinates of the waypoint being added. It takes an argument with the new pixel shift in the form of an array of two numbers.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

If the **Event.preventDefault** method is called for this event, addition of a waypoint, as well as a subsequent "waypointadd" event, will be canceled.

beforewaypointdrag

The event preceding the "waypointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.
- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "waypointdrag" event will be canceled.

beforewaypointdragstart

The event preceding the "waypointdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent: **DomEvent** — The source DOM event, if there is one.
- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "waypointdragstart" event, will be canceled.

beforewaypointremove

The event preceding the "waypointremove" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being deleted.

If the **Event.preventDefault** method is called for this event, removal of the waypoint, as well as a subsequent "waypointremove" event, will be canceled.

midpointadd

Adding a midpoint (intermediate point) to the route. The point type is determined by the value of the midPointsType option. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- coords: Number[] — Coordinates for adding a midpoint.
- pointType: String — Type ID for the midpoint being added.
- insertIndex: Integer — The insertion index of the midpoint in the set of reference points for the multi-stop route.

midpointdrag

Dragging the added midpoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.

midpointdragend

End of dragging the added midpoint. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

midpointpinshow

Displaying the draggable placemark when you hover over the active route. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pin: **Placemark** — Reference to the placemark object.
- globalPixels: Number[] — Global pixel coordinates of the placemark.
- segment: **multiRouter.driving.Segment** — Reference to the segment of the route where the placemark appeared.

viapointdrag

Throughpoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.
- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

viapointdragend

End of throughpoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

viapointdragstart

Start of throughpoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent: **DomEvent** — The source DOM event, if there is one.
- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

viapointremove

Deleting a throughpoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- viaPoint: **multiRouter.ViaPoint** — A reference to the deleted throughpoint object.

waypointadd

Adding a waypoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- coords: Number[] — Coordinates of the added waypoint.

waypointdrag

Waypoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.
- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

waypointdragend

End of waypoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

waypointdragstart

Start of waypoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent: **DomEvent** — The source DOM event, if there is one.
- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

waypointremove

Deleting a waypoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- wayPoint: **multiRouter.WayPoint** — A reference to the deleted waypoint object.

Methods details

destroy

```
{} destroy()
```

Destroys the multi-route editor.

getMultiRoute

```
{multiRouter.MultiRoute} getMultiRoute()
```

Returns a reference to the multi-route being edited.

multiRouter.EditorAddon

Note:
The constructor of the multiRouter.EditorAddon class is hidden, as this class is not intended for autonomous initialization.

Extends **ICustomizable**, **IEventEmitter**.

Add-on for the multi-stop route editor.

Fields | Events | Methods

Creates an add-on for the multi-stop route editor.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**data.Manager**	The state manager of the multi-stop route editor.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
beforemidpointadd	Event preceding the "midpointadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • coords: Number[] — Coordinates for adding a midpoint. • pointType: String — Type ID for the midpoint being added. • insertIndex: Integer — The insertion index of the midpoint in the set of reference points for the multi-stop route. Names of methods that are accessible via **Event.callMethod**: • setPointType — The method that allows you to set the type of the added point. Takes a string identifier of the type as the argument (see the description of the midPointsType option). • setInsertIndex — The method that allows you to adjust the insertion index of a midpoint before it is applied. Takes the new index as the argument. If you call the **Event.preventDefault** method of this event, the next midpoint to add, as well as the "midpointadd" event, will be canceled.
beforemidpointdrag	The event preceding the "midpointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, the subsequent "midpointdrag" event will be canceled.
beforemidpointpinshow	The event preceding the "midpointpinshow" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pin: **Placemark** — Reference to the placemark object. • globalPixels: Number[] — Global pixel coordinates of the placemark. • segment: **multiRouter.driving.Segment** — Reference to the segment of the route where the placemark should appear. If the **Event.preventDefault** method is called for this event, a subsequent "midpointpinshow" event will be canceled and the placemark will be hidden.
beforeviapointdrag	The event preceding the "viapointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. • viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "viapointdrag" event will be canceled.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
beforeviapointdragstart	The event preceding the "viapointdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent: **DomEvent** — The source DOM event, if there is one. • viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "viapointdragstart" event, will be canceled.
beforeviapointremove	The event preceding the "viapointremove" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being deleted. If the **Event.preventDefault** method is called for this event, deletion of a throughpoint, as well as a subsequent "viapointremove" event, will be canceled.
beforewaypointadd	Event preceding the "waypointadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • coords: Number[] — Coordinates of the added waypoint. Names of methods that are accessible via **Event.callMethod**: • setCoords — Method that allows you to adjust the coordinates of the waypoint being added. It takes an argument with the new pixel shift in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, addition of a waypoint, as well as a subsequent "waypointadd" event, will be canceled.
beforewaypointdrag	The event preceding the "waypointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. • wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent "waypointdrag" event will be canceled.
beforewaypointdragstart	The event preceding the "waypointdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • domEvent: **DomEvent** — The source DOM event, if there is one. • wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "waypointdragstart" event, will be canceled.
beforewaypointremove	The event preceding the "waypointremove" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being deleted. If the **Event.preventDefault** method is called for this event, removal of the waypoint, as well as a subsequent "waypointremove" event, will be canceled.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
midpointadd	Adding a midpoint (intermediate point) to the route. The point type is determined by the value of the midPointsType option. Instance of the Event class. Names of fields that are available via the Event.get method: • coords: Number[] — Coordinates for adding a midpoint. • pointType: String — Type ID for the midpoint being added. • insertIndex: Integer — The insertion index of the midpoint in the set of reference points for the multi-stop route.
midpointdrag	Dragging the added midpoint. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelOffset: Number[] — Pixel shift along the axes at the given step.
midpointdragend	End of dragging the added midpoint. Instance of the Event class.
midpointpinshow	Displaying the draggable placemark when you hover over the active route. Instance of the Event class. Names of fields that are available via the Event.get method: • pin: Placemark — Reference to the placemark object. • globalPixels: Number[] — Global pixel coordinates of the placemark. • segment: multiRouter.driving.Segment — Reference to the segment of the route where the placemark appeared.
optionschange	Change to the object options. Inherited from ICustomizable.
start	Enabling the editor. Instance of the Event class.
stop	Disabling the editor. Instance of the Event class.
viapointdrag	Throughpoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. • viaPoint: multiRouter.ViaPoint — A reference to the throughpoint object being dragged.
viapointdragend	End of throughpoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • viaPoint: multiRouter.ViaPoint — A reference to the throughpoint object being dragged.
viapointdragstart	Start of throughpoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • domEvent: DomEvent — The source DOM event, if there is one. • viaPoint: multiRouter.ViaPoint — A reference to the throughpoint object being dragged.
viapointremove	Deleting a throughpoint. Instance of the Event class. Names of fields that are available via the Event.get method: • viaPoint: multiRouter.ViaPoint — A reference to the deleted throughpoint object.
waypointadd	Adding a waypoint. Instance of the Event class. Names of fields that are available via the Event.get method: • coords: Number[] — Coordinates of the added waypoint.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
waypointdrag	Waypoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • pixelOffset: Number[] — Pixel shift along the axes at the given step. • wayPoint: multiRouter.WayPoint — A reference to the waypoint object being dragged.
waypointdragend	End of waypoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • wayPoint: multiRouter.WayPoint — A reference to the waypoint object being dragged.
waypointdragstart	Start of waypoint dragging. Instance of the Event class. Names of fields that are available via the Event.get method: • domEvent: DomEvent — The source DOM event, if there is one. • wayPoint: multiRouter.WayPoint — A reference to the waypoint object being dragged.
waypointremove	Deleting a waypoint. Instance of the Event class. Names of fields that are available via the Event.get method: • wayPoint: multiRouter.WayPoint — A reference to the deleted waypoint object.

Methods

Name	Returns	Description
isActive()	Boolean	Returns a flag for whether the editor is currently enabled.
start(state)		Enables the editor.
stop()		Disables the editor.

Fields details

state

```
{data.Manager} state
```

The state manager of the multi-stop route editor.

Events details

beforemidpointadd

Event preceding the "midpointadd" event. Instance of the Event class. Names of fields that are available via the Event.get method:

- coords: Number[] — Coordinates for adding a midpoint.
- pointType: String — Type ID for the midpoint being added.
- insertIndex: Integer — The insertion index of the midpoint in the set of reference points for the multi-stop route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Names of methods that are accessible via **Event.callMethod**:

- setPointType — The method that allows you to set the type of the added point. Takes a string identifier of the type as the argument (see the description of the midPointsType option).
- setInsertIndex — The method that allows you to adjust the insertion index of a midpoint before it is applied. Takes the new index as the argument.

If you call the **Event.preventDefault** method of this event, the next midpoint to add, as well as the "midpointadd" event, will be canceled.

beforemidpointdrag

The event preceding the "midpointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, the subsequent "midpointdrag" event will be canceled.

beforemidpointpinshow

The event preceding the "midpointpinshow" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pin: **Placemark** — Reference to the placemark object.
- globalPixels: Number[] — Global pixel coordinates of the placemark.
- segment: **multiRouter.driving.Segment** — Reference to the segment of the route where the placemark should appear.

If the **Event.preventDefault** method is called for this event, a subsequent "midpointpinshow" event will be canceled and the placemark will be hidden.

beforeviapointdrag

The event preceding the "viapointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.
- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "viapointdrag" event will be canceled.

beforeviapointdragstart

The event preceding the "viapointdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent: **DomEvent** — The source DOM event, if there is one.
- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "viapointdragstart" event, will be canceled.

beforeviapointremove

The event preceding the "viapointremove" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being deleted.

If the **Event.preventDefault** method is called for this event, deletion of a throughpoint, as well as a subsequent "viapointremove" event, will be canceled.

beforewaypointadd

Event preceding the "waypointadd" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- coords: Number[] — Coordinates of the added waypoint.

Names of methods that are accessible via **Event.callMethod**:

- setCoords — Method that allows you to adjust the coordinates of the waypoint being added. It takes an argument with the new pixel shift in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, addition of a waypoint, as well as a subsequent "waypointadd" event, will be canceled.

beforewaypointdrag

The event preceding the "waypointdrag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.
- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

Names of methods that are accessible via **Event.callMethod**:

- setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the **Event.preventDefault** method is called for this event, a subsequent "waypointdrag" event will be canceled.

beforewaypointdragstart

The event preceding the "waypointdragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent: **DomEvent** — The source DOM event, if there is one.
- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "waypointdragstart" event, will be canceled.

beforewaypointremove

The event preceding the "waypointremove" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being deleted.

If the **Event.preventDefault** method is called for this event, removal of the waypoint, as well as a subsequent "waypointremove" event, will be canceled.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

midpointadd

Adding a midpoint (intermediate point) to the route. The point type is determined by the value of the midPointsType option. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- coords: Number[] — Coordinates for adding a midpoint.
- pointType: String — Type ID for the midpoint being added.
- insertIndex: Integer — The insertion index of the midpoint in the set of reference points for the multi-stop route.

midpointdrag

Dragging the added midpoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.

midpointdragend

End of dragging the added midpoint. Instance of the **Event** class.

midpointpinshow

Displaying the draggable placemark when you hover over the active route. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pin: **Placemark** — Reference to the placemark object.
- globalPixels: Number[] — Global pixel coordinates of the placemark.
- segment: **multiRouter.driving.Segment** — Reference to the segment of the route where the placemark appeared.

start

Enabling the editor. Instance of the **Event** class.

stop

Disabling the editor. Instance of the **Event** class.

viapointdrag

Throughpoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.
- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

viapointdragend

End of throughpoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

viapointdragstart

Start of throughpoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent: **DomEvent** — The source DOM event, if there is one.
- viaPoint: **multiRouter.ViaPoint** — A reference to the throughpoint object being dragged.

viapointremove

Deleting a throughpoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- viaPoint: **multiRouter.ViaPoint** — A reference to the deleted throughpoint object.

waypointadd

Adding a waypoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- coords: Number[] — Coordinates of the added waypoint.

waypointdrag

Waypoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pixelOffset: Number[] — Pixel shift along the axes at the given step.
- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

waypointdragend

End of waypoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

waypointdragstart

Start of waypoint dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- domEvent: **DomEvent** — The source DOM event, if there is one.
- wayPoint: **multiRouter.WayPoint** — A reference to the waypoint object being dragged.

waypointremove

Deleting a waypoint. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- wayPoint: **multiRouter.WayPoint** — A reference to the deleted waypoint object.

Methods details

isActive

```
{Boolean} isActive()
```

Returns a flag for whether the editor is currently enabled.

start

```
{} start(state)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Enables the editor.

Parameters:

Parameter	Default value	Description
state*	—	Type: Object The initial state of the editor.

* Mandatory parameter/option.

stop

```
{} stop()
```

Disables the editor.

multiRouter.masstransit

multiRouter.masstransit.Path

Note:
The constructor of the multiRouter.masstransit.Path class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of the route path for public transport. A single route can contain several paths, and each path connects two waypoints.

Fields | Events | Methods

Creates a representation of the public transport route path.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.masstransit.PathModel**	Data model for the multi-stop route's path.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	data.Manager	Multi-stop route's path data. The following fields are available: • index: Integer — The sequential number of the path in the multi-stop route's corresponding route. • type: String — Route type identifier, which takes the value "masstransit" for public transport routes. • distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds. • coordinates: Number[][] — Coordinates of all points on the path. • encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
update	Updating the path rendering. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getSegmentMarkers()	**GeoObjectCollection**	Returns a child collection of segment markers.
getSegments()	**GeoObjectCollection**	Returns the child collection of segments that the path consists of.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

model

```
{multiRouter.masstransit.PathModel} model
```

Data model for the multi-stop route's path.

properties

```
{data.Manager} properties
```

Multi-stop route's path data. The following fields are available:

- index: Integer — The sequential number of the path in the multi-stop route's corresponding route.
- type: String — Route type identifier, which takes the value "masstransit" for public transport routes.
- distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- coordinates: Number[][] — Coordinates of all points on the path.
- encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.

Events details

update

Updating the path rendering. Instance of the **Event** class.

Methods details

getSegmentMarkers

```
{GeoObjectCollection} getSegmentMarkers()
```

Returns a child collection of segment markers.

getSegments

```
{GeoObjectCollection} getSegments()
```

Returns the child collection of segments that the path consists of.

multiRouter.masstransit.PathModel

Note:
The constructor of the multiRouter.masstransit.PathModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model of a path on a public transport route. A single route can contain several paths, and each path connects two waypoints.

Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Creates a data model for a path on a public transport route.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
properties	data.Manager	Multi-stop route's path data. The following fields are available: • index: Integer — The sequential number of the path in the multi-stop route's corresponding route. • type: String — Route type identifier, which takes the value "masstransit" for public transport routes. • distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds. • coordinates: Number[][] — Coordinates of all points on the path. • encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.
route	multiRouter.masstransit.RouteModel	Reference to the parent model of the route.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class. Names of fields that are available via the Event.get method: • segmentsChange: Boolean — Flag for whether the set of segments is changed

Methods

Name	Returns	Description
destroy()		Destroys a model.
getSegments()	(multiRouter.masstransit.TransferSegmentModel|multiRouter.masstransit.TransportSegmentModel|multiRouter.masstransit.WalkSegmentModel)[]	Returns array of path segments.
getType()	String	Returns ID of the route type. For public transport routes, the string "masstransit" is returned.
update(pathJson)		Updates the state of the model.

Fields details

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Multi-stop route's path data. The following fields are available:

- index: Integer — The sequential number of the path in the multi-stop route's corresponding route.
- type: String — Route type identifier, which takes the value "masstransit" for public transport routes.
- distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- coordinates: Number[][] — Coordinates of all points on the path.
- encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.

route

```
{multiRouter.masstransit.RouteModel} route
```

Reference to the parent model of the route.

Events details

update

Updating the model with new data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- segmentsChange: Boolean — Flag for whether the set of segments is changed

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getSegments

```
{(multiRouter.masstransit.TransferSegmentModel|
multiRouter.masstransit.TransportSegmentModel|
multiRouter.masstransit.WalkSegmentModel)[]} getSegments()
```

Returns array of path segments.

getType

```
{String} getType()
```

Returns ID of the route type. For public transport routes, the string "masstransit" is returned.

update

```
{} update(pathJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
pathJson*	—	Type: Object JSON data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

multiRouter.masstransit.Route

Note:
The constructor of the multiRouter.masstransit.Route class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Displaying an individual route of public transport. A multi-stop route can consist of several individual routes.

Fields | Events | Methods

Creates the representation to display an individual route of public transport.

Fields

Name	Type	Description
balloon	**IMultiRouterRouteBalloon**	Balloon of a route.
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.masstransit.RouteModel**	The data model of an individual route.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**data.Manager**	The route data. The following fields are available: • index: Integer — The ordinal number of the route in a multi-stop route. • type: String — Route type identifier, which takes the value "masstransit" for public transport routes. • distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the route in seconds. • boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Events

Name	Description
balloonclose	Closing the balloon.
balloonopen	Opening the balloon.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
update	Updating the route rendering. Instance of the Event class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getPaths()	GeoObjectCollection	Returns a child collection of paths that make up the route.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

balloon

```
{IMultiRouterRouteBalloon} balloon
```

Balloon of a route.

model

```
{multiRouter.masstransit.RouteModel} model
```

The data model of an individual route.

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

The route data. The following fields are available:

- index: Integer — The ordinal number of the route in a multi-stop route.
- type: String — Route type identifier, which takes the value "masstransit" for public transport routes.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the route in seconds.
- boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Events details

balloonclose

Closing the balloon.

balloonopen

Opening the balloon.

update

Updating the route rendering. Instance of the **Event** class.

Methods details

getPaths

```
{GeoObjectCollection} getPaths()
```

Returns a child collection of paths that make up the route.

multiRouter.masstransit.RouteModel

Note:
The constructor of the multiRouter.masstransit.RouteModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model of an individual route of public transport. A multi-stop route can consist of several individual routes.

Fields | Events | Methods

Creates the data model for an individual route of public transport.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
multiRoute	**multiRouter.MultiRouteModel**	Reference to the parent model of a multi-stop route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	data.Manager	The route data. The following fields are available: • index: Integer — The ordinal number of the route in a multi-stop route. • type: String — Route type identifier, which takes the value "masstransit" for public transport routes. • distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the route in seconds. • boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class. Names of fields that are available via the Event.get method: • pathsChange: Boolean — Flag for whether the set of paths is changed.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getPaths()	multiRouter.masstransit.PathModel[]	Returns array of route paths.
getType()	String	Returns ID of the route type. For public transport routes, the string "masstransit" is returned.
update(routeJson)		Updates the state of the model.

Fields details

multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

properties

```
{data.Manager} properties
```

The route data. The following fields are available:

- index: Integer — The ordinal number of the route in a multi-stop route.
- type: String — Route type identifier, which takes the value "masstransit" for public transport routes.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the route in seconds.
- boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

update

Updating the model with new data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pathsChange: Boolean — Flag for whether the set of paths is changed.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getPaths

```
{multiRouter.masstransit.PathModel[]} getPaths()
```

Returns array of route paths.

getType

```
{String} getType()
```

Returns ID of the route type. For public transport routes, the string "masstransit" is returned.

update

```
{} update(routeJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
routeJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

multiRouter.masstransit.StopModel

Note:
The constructor of the multiRouter.masstransit.StopModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model for a stop on a transportation segment of a public transport route.

Fields | Events | Methods

Creates the data model for a stop of a transport segment.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
geometry	geometry.base.Point	Geometry of the stop.
properties	data.Manager	Stop data. The following fields are available: • Integer — The ordinal number of the stop in the set of stops corresponding to the transit segment. • id: String — The stop ID. • name: String — The name of the stop. • lodIndex: Integer — The ordinal number of the stop point in the full set of coordinates of the corresponding route path.
segment	multiRouter.masstransit.TransportSegmentModel	Reference to the parent model of the transit segment.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class.

Methods

Name	Description
update(stopJson)	Updates the state of the model.

Fields details

geometry

```
{geometry.base.Point} geometry
```

Geometry of the stop.

properties

```
{data.Manager} properties
```

Stop data. The following fields are available:

- Integer — The ordinal number of the stop in the set of stops corresponding to the transit segment.
- id: String — The stop ID.
- name: String — The name of the stop.
- lodIndex: Integer — The ordinal number of the stop point in the full set of coordinates of the corresponding route path.

segment

```
{multiRouter.masstransit.TransportSegmentModel} segment
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Reference to the parent model of the transit segment.

Events details

update

Updating the model with new data. Instance of the **Event** class.

Methods details

update

```
{} update(stopJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
stopJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

multiRouter.masstransit.TransferSegment

Note:
The constructor of the multiRouter.masstransit.TransferSegment class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of a transfer segment on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

Fields | Events | Methods

Creates a representation to display a transfer segment on a public transport route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.masstransit.TransferSegmentModel**	Data model for a segment.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	data.Manager	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • type: String — Segment type identifier, which takes the value "transfer" for transfer segments. • text: String — Text description of the segment. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
update	Updating the segment rendering. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Fields details

model

```
{multiRouter.masstransit.TransferSegmentModel} model
```

Data model for a segment.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

properties

`{data.Manager} properties`

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- type: String — Segment type identifier, which takes the value "transfer" for transfer segments.
- text: String — Text description of the segment.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the segment rendering. Instance of the **Event** class.

multiRouter.masstransit.TransferSegmentModel

Note:
The constructor of the multiRouter.masstransit.TransferSegmentModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model for a transfer segment of a path on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

Fields | Events | Methods

Creates the data model for a transfer segment of a path on a public transport route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**geometry.base.LineString**	Geometry of a segment.
path	**multiRouter.masstransit.PathModel**	Reference to the parent model of the path.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	data.Manager	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • type: String — Segment type identifier, which takes the value "transfer" for transfer segments. • text: String — Text description of the segment. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class.

Methods

Name	Returns	Description
destroy(segmentJson)		Updates the state of the model.
getType()	String	Returns ID of the segment type. For transfer segments on public transport routes, it returns the string "transfer".

Fields details

geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

path

```
{multiRouter.masstransit.PathModel} path
```

Reference to the parent model of the path.

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- type: String — Segment type identifier, which takes the value "transfer" for transfer segments.
- text: String — Text description of the segment.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the model with new data. Instance of the **Event** class.

Methods details

destroy

```
{} destroy(segmentJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
segmentJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

getType

```
{String} getType()
```

Returns ID of the segment type. For transfer segments on public transport routes, it returns the string "transfer".

multiRouter.masstransit.TransportSegment

Note:
The constructor of the multiRouter.masstransit.TransportSegment class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of a transport segment on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

Fields | Events | Methods

Creates the representation to display a transport segment on a public transport route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
geometry	IGeometry|null	Geo object geometry. Inherited from IGeoObject.
model	multiRouter.masstransit.TransportSegmentModel	Data model for a segment.
options	IOptionManager	Options manager. Inherited from ICustomizable.
properties	data.Manager	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • String — Segment type identifier, which takes the value "transport" for transport segments. • text: String — Text description of the segment. • transports: ITransportProperties[] — An array describing the specific forms of transport available for traveling on this segment. • stops: Object — Description of stops in the format GeoJson:FeatureCollection. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
update	Updating the segment rendering. Instance of the Event class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

model

```
{multiRouter.masstransit.TransportSegmentModel} model
```

Data model for a segment.

properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- String — Segment type identifier, which takes the value "transport" for transport segments.
- text: String — Text description of the segment.
- transports: ITransportProperties[] — An array describing the specific forms of transport available for traveling on this segment.
- stops: Object — Description of stops in the format GeoJson:FeatureCollection.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the segment rendering. Instance of the Event class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multiRouter.masstransit.TransportSegmentModel

Note:
The constructor of the multiRouter.masstransit.TransportSegmentModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model for a transportation segment of a path on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

Fields | Events | Methods

Creates a data model for a transportation segment of a path on a public transport route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**geometry.base.LineString**	Geometry of a segment.
path	**multiRouter.masstransit.PathModel**	Reference to the parent model of the path.
properties	**data.Manager**	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • String — Segment type identifier, which takes the value "transport" for transport segments. • text: String — Text description of the segment. • transports: **ITransportProperties**[] — An array describing the specific forms of transport available for traveling on this segment. • stops: Object — Description of stops in the format GeoJson:FeatureCollection. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events

Name	Description
update	Updating the model with new data. Instance of the **Event** class.

Methods

Name	Returns	Description
destroy()		Destroys a model.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getStops()	multiRouter.masstransit.StopModel[]	Returns array of stops on a transport route.
getType()	String	Returns ID of the segment type. For transport segments on public transport routes, it returns the string "transport".
update(segmentJson)		Updates the state of the model.

Fields details

geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

path

```
{multiRouter.masstransit.PathModel} path
```

Reference to the parent model of the path.

properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- String — Segment type identifier, which takes the value "transport" for transport segments.
- text: String — Text description of the segment.
- transports: ITransportProperties[] — An array describing the specific forms of transport available for traveling on this segment.
- stops: Object — Description of stops in the format GeoJson:FeatureCollection.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the model with new data. Instance of the Event class.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getStops

```
{multiRouter.masstransit.StopModel[]} getStops()
```

Returns array of stops on a transport route.

getType

```
{String} getType()
```

Returns ID of the segment type. For transport segments on public transport routes, it returns the string "transport".

update

```
{} update(segmentJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
segmentJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

multiRouter.masstransit.WalkSegment

Note:
The constructor of the multiRouter.masstransit.WalkSegment class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of a walking segment on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

Fields | Events | Methods

Creates the representation to display a walking segment on a public transport route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.masstransit.WalkSegmentModel**	Data model for a segment.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
options	IOptionManager	Options manager. Inherited from ICustomizable.
properties	data.Manager	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • type: String — Segment type identifier, which takes the value "walk" for walking segments. • text: String — Text description of the segment. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
update	Updating the segment rendering. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

model

```
{multiRouter.masstransit.WalkSegmentModel} model
```

Data model for a segment.

properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- type: String — Segment type identifier, which takes the value "walk" for walking segments.
- text: String — Text description of the segment.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the segment rendering. Instance of the **Event** class.

multiRouter.masstransit.WalkSegmentModel

Note:
The constructor of the multiRouter.masstransit.WalkSegmentModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Data model for a walking segment of a path on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

Fields | Events | Methods

Creates the data model for a walking segment of a path on a public transport route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**geometry.base.LineString**	Geometry of a segment.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
path	multiRouter.masstransit.PathModel	Reference to the parent model of the path.
properties	data.Manager	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • type: String — Segment type identifier, which takes the value "walk" for walking segments. • text: String — Text description of the segment. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getType()	String	Returns ID of the segment type. For walking segments on public transport routes, it returns the string "walk".

Fields details

geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

path

```
{multiRouter.masstransit.PathModel} path
```

Reference to the parent model of the path.

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- type: String — Segment type identifier, which takes the value "walk" for walking segments.
- text: String — Text description of the segment.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the model with new data. Instance of the **Event** class.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getType

```
{String} getType()
```

Returns ID of the segment type. For walking segments on public transport routes, it returns the string "walk".

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multiRouter.MultiRoute

Extends **IGeoObject**.

Multi-route on the map. Displays the route on the map along with one or more alternatives. Provides the interface for selecting the active route.

Constructor | Fields | Events | Methods

Constructor

```
multiRouter.MultiRoute(model[, options])
```

Creates a multi-route on the map.

Parameters:

Parameter	Default value	Description
`model` *	—	Type: **multiRouter.MultiRouteModel**|**IMultiRouteModelJson** The data model of a multi-route, or the model description object. If you pass a description object, the model (which is based on it) is created automatically.
`options`	—	Type: Object Multi-route options. To define options for the parts of a multi-route, use the following prefixes: • `wayPoint` — Waypoint options. • `wayPointStart` — Display options for the starting waypoint. • `wayPointFinish` — Display options for the end waypoint. • `viaPoint` — Throughpoint options. • `pin` — Options for point markers on the route. • `editor` — Options for the multi-route editor (see **multiRouter.Editor**). To define options for individual routes within a multi-route, use the following prefixes: • `route` — Options for routes, including inactive ones. • `routeActive` — Options for the active route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Note that options specified with the "routeActive" prefix have a higher priority than the options with the "route" prefix. To define options for segments of public transport routes, use the following prefixes: • `routeMarker` — Options for route segment markers. • `routeWalkMarker` — Options for markers of walking segments on the route. • `routeTransferMarker` — Options for transfer segment markers. • `routeTransportMarker` — Options for transit segment markers on the route. • `routeWalkSegment` — Options for walking segment lines on the route. • `routeTransferSegment` — Options for transfer segment lines. • `routeTransportSegment` — Options for transit segment lines on the route. • `routePedestrianSegment` — Options for pedestrian route segment lines. These prefixes are also available in the "routeActive*" version.
`options.activeRouteAutoSelection`	true	Type: Boolean After the data is refreshed, automatically use the route with the shortest travel time as the active route.
`options.boundsAutoApply`	false	Type: Boolean When adding a multi-route to the map, you can automatically set the center and the zoom level so that the multi-route is entirely visible.
`options.dragUpdateInterval`	—	Type: String|Number Time interval of rebuilding the route while reference points are being dragged. Can be set in milliseconds, or the optimal value can be calculated automatically. The value of this option is translated to the **IMultiRouteParams.requestSendInterval** parameter of the multi-route.
`options.preventDragUpdate`	false	Type: Boolean Allows to disable the route update while reference points are being dragged.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.useMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager**.
options.zoomMargin	0	Type: Number|Number[] Offset from the map viewport borders when changing the zoom level. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" option is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager **map.margin.Manager**.

* Mandatory parameter/option.

Examples:

1.

```
// Creating a multi-route and adding it to the map.
var multiRoute = new ymaps.multiRouter.MultiRoute({
    referencePoints: ['Moscow, Leninsky Prospekt', 'Moscow, Kulakov pereulok'],
}, {
    editorDrawOver: false,
    wayPointDraggable: true,
    viaPointDraggable: true,
    // Setting a custom design for multi-route lines.
    routeStrokeColor: "000088",
    routeActiveStrokeColor: "ff0000",
    pinIconFillColor: "ff0000",
    boundsAutoApply: true,
    zoomMargin: 30
});
myMap.geoObjects.add(multiRoute);
```

2.

```
// Creating a multi-route and adding it to the map.
var multiRoute = new ymaps.multiRouter.MultiRoute({
    referencePoints: ['Moscow, Leninsky Prospekt', 'Moscow, Kulakov pereulok',
'Zelenograd'],
});
myMap.geoObjects.add(multiRoute);

// Once multi-route is loaded.
multiRoute.events.once('update', function () {
    // Set first non-blocked route as active and open it's balloon.
    var routes = multiRoute.getRoutes();
    for (var i = 0, l = routes.getLength(); i < l; i++) {
        var route = routes.get(i);
        if (!route.properties.get('blocked')) {
            multiRoute.setActiveRoute(route);
            route.balloon.open();
            break;
        }
    }
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
editor	multiRouter.EditorAddon	Multi-stop route editor.
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
geometry	IGeometry|null	Geo object geometry. Inherited from IGeoObject.
model	multiRouter.MultiRouteModel	The data model of a multi-route.
options	IOptionManager	Options manager. Inherited from ICustomizable.
properties	IDataManager	Geo object data. Inherited from IGeoObject.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
activeroutechange	Change to the active route. Instance of the Event class. Names of fields that are available via the Event.get method: • oldActiveRoute: multiRouter.driving.Route|multiRouter.masstransit.Route|null — previous active route.
balloonclose	Closing the balloon. Names of fields that are available via the Event.get method: • originalTarget: multiRouter.pedestrian.Route|multiRouter.driving.Route|multiRouter.masstransit.Route|null — route on which balloon was closed.
balloonopen	Opening the balloon. Names of fields that are available via the Event.get method: • originalTarget: multiRouter.pedestrian.Route|multiRouter.driving.Route|multiRouter.masstransit.Route|null — route on which balloon was opened.
boundsautoapply	The event occurs at the time of setting the map center and its zoom level for optimal display of the multi-route. Also see the description of the boundsAutoApply option. Instance of the Event class.
boundschange	Changing coordinates of the geographical area covering the multi-route. Instance of the Event class.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
pixelboundschange	Changing pixel coordinates of the area covering the multi-route. Instance of the **Event** class.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
update	Updating the multi-route. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getActiveRoute()	**multiRouter.driving.Route**| **multiRouter.masstransit.Route**|null	Returns active route.
getBounds()	Number[][]|null	Returns the geographical coordinates of the area covering the multi-route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getPixelBounds()	Number[][]|null	Returns the global pixel coordinates of the area covering the multi-route.
getRoutes()	GeoObjectCollection	Returns child collection of individual routes on the multi-route.
getViaPoints()	GeoObjectCollection	Returns child collection of throughpoints for the multi-route.
getWayPoints()	GeoObjectCollection	Returns child collection of waypoints for the multi-route.
setActiveRoute(route)		Sets the active route. The previous active route is deactivated and the multiRouter.MultiRoute.event:activeroutechange event is generated.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

editor

```
{multiRouter.EditorAddon} editor
```

Multi-stop route editor.

Example:

```
// Start of route editing.
ymaps.route(['Moscow', 'Petersburg'], { multiRoute: true })
    .done(function (multiRoute) {
        myMap.geoObjects.add(multiRoute);
        multiRoute.editor.start({
            addWayPoints: true,
            removeWayPoints: true
        });
        // ...
        // End of route editing.
        multiRoute.editor.stop();
    });
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

model

```
{multiRouter.MultiRouteModel} model
```

The data model of a multi-route.

Events details

activeroutechange

Change to the active route. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldActiveRoute: multiRouter.driving.Route|multiRouter.masstransit.Route|null — previous active route.

balloonclose

Closing the balloon. Names of fields that are available via the **Event.get** method:

- originalTarget: multiRouter.pedestrian.Route|multiRouter.driving.Route|multiRouter.masstransit.Route|null — route on which balloon was closed.

balloonopen

Opening the balloon. Names of fields that are available via the **Event.get** method:

- originalTarget: multiRouter.pedestrian.Route|multiRouter.driving.Route|multiRouter.masstransit.Route|null — route on which balloon was opened.

boundsautoapply

The event occurs at the time of setting the map center and its zoom level for optimal display of the multi-route. Also see the description of the boundsAutoApply option. Instance of the **Event** class.

boundschange

Changing coordinates of the geographical area covering the multi-route. Instance of the **Event** class.

pixelboundschange

Changing pixel coordinates of the area covering the multi-route. Instance of the **Event** class.

update

Updating the multi-route. Instance of the **Event** class.

Methods details

getActiveRoute

```
{multiRouter.driving.Route|multiRouter.masstransit.Route|null} getActiveRoute()
```

Returns active route.

getBounds

```
{Number[][]|null} getBounds()
```

Returns the geographical coordinates of the area covering the multi-route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Returns the global pixel coordinates of the area covering the multi-route.

getRoutes

```
{GeoObjectCollection} getRoutes()
```

Returns child collection of individual routes on the multi-route.

getViaPoints

```
{GeoObjectCollection} getViaPoints()
```

Returns child collection of throughpoints for the multi-route.

getWayPoints

```
{GeoObjectCollection} getWayPoints()
```

Returns child collection of waypoints for the multi-route.

setActiveRoute

```
{} setActiveRoute(route)
```

Sets the active route. The previous active route is deactivated and the **multiRouter.MultiRoute.event:activeroutechange** event is generated.

Parameters:

Parameter	Default value	Description
route*	—	Type: **multiRouter.driving.Route**|**multiRouter.masstransit.Route**|null Route to activate.

* Mandatory parameter/option.

multiRouter.MultiRouteModel

Extends **IEventEmitter**.

The data model of a multi-stop route.

Constructor | Fields | Events | Methods

Constructor

```
multiRouter.MultiRouteModel(referencePoints[, params])
```

Creates the data model of a multi-stop route.

Parameters:

Parameter	Default value	Description
referencePoints *	—	Type: **IMultiRouteReferencePoint**[]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The description of the reference points on the multi-stop route.
params	—	Type: **IMultiRouteParams** Routing options.

* Mandatory parameter/option.

Example:

```
// Creating a model of the multi-stop route.
var multiRouteModel = new ymaps.multiRouter.MultiRouteModel(['Moscow', 'Tver',
'Peterburg'], {
    avoidTrafficJams: true,
    viaIndexes: [1]
});

// Creating a representation of a multi-stop route based on the model.
var multiRouteView = new ymaps.multiRouter.MultiRoute(multiRouteModel);
myMap.geoObjects.add(multiRouteView);

// Subscribing to the events of the multi-stop route model.
multiRouteView.model.events
    .add("requestsuccess", function (event) {
        var routes = event.get("target").getRoutes();
        console.log("Found routes: " + routes.length);
        for (var i = 0, l = routes.length; i &lt; l; i++) {
            console.log("Route length " + (i + 1) + ": " +
routes[i].properties.get("distance").text);
        }
    })
    .add("requestfail", function (event) {
        console.log("Error: " + event.get("error").message);
    });
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
properties	**data.Manager**	Multi-stop route data.

Events

Name	Description
requestcancel	The data request has been canceled. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • referencePoints: Object[] — Array describing the set of reference points. • params: Object — Routing options.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
requestchange	The reference data model of a multi-stop route (anchor points or routing options) has been changed. The result is a new data request to the routing service. Instance of the Event class. Names of fields that are available via the Event.get method: • referencePoints: Object[] — Array describing the set of reference points. • params: Object — Routing parameters. • oldReferencePoints: Object[] — Array describing the previous array of reference points. • oldParams: Object — Previous routing options.
requestfail	The request for data has failed. Instance of the Event class. Names of fields that are available via the Event.get method: • error: Error — error object.
requestsend	A new request for the multi-stop route data model has been sent. Instance of the Event class. Names of fields that are available via the Event.get method: • referencePoints: Object[] — Array describing the set of reference points. • params: Object — Routing options.
requestsuccess	The request for data has completed successfully and the data model has been updated. Instance of the Event class. Names of fields that are available via the Event.get method: • init: Boolean — Indicates an initialization request. • rough: Boolean — Flag for whether it is an intermediate request (used to optimize the size of the server response when editing a route). • wayPointsChange: Boolean — Flag for whether the set of waypoints is changed. • viaPointsChange: Boolean — Flag for whether the set of throughpoints is changed. • routesChange: Boolean — Flag for whether the set of routes is changed.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getAllPoints()	(multiRouter.WayPointModel|multiRouter.ViaPointModel)[]	Returns the combined array of models of throughpoints and waypoints, in the same order as the corresponding reference points.
getJson()	Object	Returns JSON data for the multi-stop route model.
getParams()	IMultiRouteParams	Returns the object with the current values of routing options.
getPoints()	(multiRouter.WayPointModel|multiRouter.ViaPointModel)[]	Deprecated name of the multiRouter.MultiRouteModel.getAllPoints method. Not recommended for use.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getReferencePointIndexes()	Object	Returns an object containing the following data fields: • way: Integer[] — Indexes of reference points corresponding to the set of the model's waypoints. • via: Integer[] — Indexes of reference points corresponding to the set of the model's throughpoints. .
getReferencePoints()	IMultiRouteReferencePoint[]	Returns array of reference points on the multi-stop route.
getRoutes()	multiRouter.driving.RouteModel[]| multiRouter.masstransit.RouteModel[]	Returns an array of models for child routes.
getViaPoints()	multiRouter.ViaPointModel[]	Returns an array of models for child throughpoints.
getWayPoints()	multiRouter.WayPointModel[]	Returns an array of models for child waypoints.
setParams(params[, extend[, clearRequests]])		Sets routing options.
setReferencePoints(referencePoints[, viaIndexes[, clearRequests]])		Sets the reference points on the multi-stop route.

Fields details

properties

```
{data.Manager} properties
```

Multi-stop route data.

Events details

requestcancel

The data request has been canceled. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- referencePoints: Object[] — Array describing the set of reference points.
- params: Object — Routing options.

requestchange

The reference data model of a multi-stop route (anchor points or routing options) has been changed. The result is a new data request to the routing service. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- referencePoints: Object[] — Array describing the set of reference points.
- params: Object — Routing parameters.
- oldReferencePoints: Object[] — Array describing the previous array of reference points.
- oldParams: Object — Previous routing options.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

requestfail

The request for data has failed. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- error: Error — error object.

requestsend

A new request for the multi-stop route data model has been sent. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- referencePoints: Object[] — Array describing the set of reference points.
- params: Object — Routing options.

requestsuccess

The request for data has completed successfully and the data model has been updated. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- init: Boolean — Indicates an initialization request.
- rough: Boolean — Flag for whether it is an intermediate request (used to optimize the size of the server response when editing a route).
- wayPointsChange: Boolean — Flag for whether the set of waypoints is changed.
- viaPointsChange: Boolean — Flag for whether the set of throughpoints is changed.
- routesChange: Boolean — Flag for whether the set of routes is changed.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getAllPoints

```
{(multiRouter.WayPointModel|multiRouter.ViaPointModel)[]} getAllPoints()
```

Returns the combined array of models of throughpoints and waypoints, in the same order as the corresponding reference points.

getJson

```
{Object} getJson()
```

Returns JSON data for the multi-stop route model.

getParams

```
{IMultiRouteParams} getParams()
```

Returns the object with the current values of routing options.

getPoints

```
{(multiRouter.WayPointModel|multiRouter.ViaPointModel)[]} getPoints()
```

Deprecated name of the multiRouter.MultiRouteModel.getAllPoints method. Not recommended for use.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns the combined array of models of throughpoints and waypoints, in the same order as the corresponding reference points.

getReferencePointIndexes

```
{Object} getReferencePointIndexes()
```

Returns an object containing the following data fields:

- way: Integer[] — Indexes of reference points corresponding to the set of the model's waypoints.
- via: Integer[] — Indexes of reference points corresponding to the set of the model's throughpoints.

.

getReferencePoints

```
{IMultiRouteReferencePoint[]} getReferencePoints()
```

Returns array of reference points on the multi-stop route.

getRoutes

```
{multiRouter.driving.RouteModel[]|multiRouter.masstransit.RouteModel[]}
getRoutes()
```

Returns an array of models for child routes.

getViaPoints

```
{multiRouter.ViaPointModel[]} getViaPoints()
```

Returns an array of models for child throughpoints.

getWayPoints

```
{multiRouter.WayPointModel[]} getWayPoints()
```

Returns an array of models for child waypoints.

setParams

```
{} setParams(params[, extend[, clearRequests]])
```

Sets routing options.

Parameters:

Parameter	Default value	Description
params*	—	Type: IMultiRouteParams Routing options.
extend	false	Type: Boolean Allows you to change only the specified set of parameters keeping all the rest of the values the same.
clearRequests	false	Type: Boolean Allows you to clear the queue of previous requests to the server.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

setReferencePoints

```
{} setReferencePoints(referencePoints[, viaIndexes[, clearRequests]])
```

Sets the reference points on the multi-stop route.

Parameters:

Parameter	Default value	Description
referencePoints*	—	Type: IMultiRouteReferencePoint[] Array of reference points.
viaIndexes	—	Type: Integer[] Indexes of throughpoints in the array of reference points.
clearRequests	false	Type: Boolean Allows you to clear the queue of previous requests to the server.

* Mandatory parameter/option.

multiRouter.pedestrian

multiRouter.pedestrian.Path

Note:
The constructor of the multiRouter.pedestrian.Path class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of the pedestrian route path. A single route can contain several paths, and each path connects two waypoints.

Fields | Events | Methods

Creates a representation of the pedestrian route path.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
geometry	IGeometry|null	Geo object geometry. Inherited from IGeoObject.
model	multiRouter.pedestrian.PathModel	Data model for the multiroute path.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
options	IOptionManager	Options manager. Inherited from ICustomizable.
properties	data.Manager	Multi-stop route's path data. The following fields are available: • index: Integer — The sequential number of the path in the multi-stop route's corresponding route. • type: String — The route type identifier; takes the value "pedestrian" for pedestrian routes. • distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds. • coordinates: Number[][] — Coordinates of all points on the path. • encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
update	Updating the path rendering. Instance of the **Event** class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getSegmentMarkers()	**GeoObjectCollection**	Returns a child collection of segment markers.
getSegments()	**GeoObjectCollection**	Returns the child collection of segments that the path consists of.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

model

```
{multiRouter.pedestrian.PathModel} model
```

Data model for the multiroute path.

properties

```
{data.Manager} properties
```

Multi-stop route's path data. The following fields are available:

- index: Integer — The sequential number of the path in the multi-stop route's corresponding route.
- type: String — The route type identifier; takes the value "pedestrian" for pedestrian routes.
- distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- coordinates: Number[][] — Coordinates of all points on the path.
- encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.

Events details

update

Updating the path rendering. Instance of the **Event** class.

Methods details

getSegmentMarkers

```
{GeoObjectCollection} getSegmentMarkers()
```

Returns a child collection of segment markers.

getSegments

```
{GeoObjectCollection} getSegments()
```

Returns the child collection of segments that the path consists of.

multiRouter.pedestrian.PathModel

Note:
The constructor of the multiRouter.pedestrian.PathModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

A data model of the pedestrian route. A single route can contain several paths, and each path connects two waypoints.

Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Creates a data model of the pedestrian route.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
properties	data.Manager	Multi-stop route's path data. The following fields are available: • index: Integer — The sequential number of the path in the multi-stop route's corresponding route. • type: String — The route type identifier; takes the value "pedestrian" for pedestrian routes. • distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds. • coordinates: Number[][] — Coordinates of all points on the path. • encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.
route	multiRouter.pedestrian.RouteModel	Reference to the parent model of the route.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class. Names of fields that are available via the Event.get method: • segmentsChange: Boolean — Flag for whether the set of segments is changed

Methods

Name	Returns	Description
destroy()		Destroys a model.
getSegments()	multiRouter.pedestrian.SegmentModel[]	Returns array of path segments.
getType()	String	Returns ID of the route type. For pedestrian routes, returns the string "pedestrian".
update(pathJson)		Updates the state of the model.

Fields details

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Multi-stop route's path data. The following fields are available:

- index: Integer — The sequential number of the path in the multi-stop route's corresponding route.
- type: String — The route type identifier; takes the value "pedestrian" for pedestrian routes.
- distance: Object — An object with the "text" and "value" fields that describes the length of the path in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- coordinates: Number[][] — Coordinates of all points on the path.
- encodedCoordinates: String — A string of base64-encoded coordinates for all points on the path.

route

```
{multiRouter.pedestrian.RouteModel} route
```

Reference to the parent model of the route.

Events details

update

Updating the model with new data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- segmentsChange: Boolean — Flag for whether the set of segments is changed

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getSegments

```
{multiRouter.pedestrian.SegmentModel[]} getSegments()
```

Returns array of path segments.

getType

```
{String} getType()
```

Returns ID of the route type. For pedestrian routes, returns the string "pedestrian".

update

```
{} update(pathJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
pathJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multiRouter.pedestrian.Route

Note:
The constructor of the multiRouter.pedestrian.Route class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Displaying an individual pedestrian route. A multi-stop route can consist of several individual routes.

Fields | Events | Methods

Creates the representation to display an individual pedestrian route.

Fields

Name	Type	Description
balloon	**IMultiRouterRouteBalloon**	Balloon of a route.
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.pedestrian.RouteModel**	The data model of an individual route.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**data.Manager**	The route data. The following fields are available: • index: Integer — The ordinal number of the route in a multi-stop route. • type: String — The route type identifier; takes the value "pedestrian" for pedestrian routes. • distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the route in seconds. • boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Events

Name	Description
balloonclose	Closing the balloon.
balloonopen	Opening the balloon.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
update	Updating the route rendering. Instance of the Event class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getPaths()	GeoObjectCollection	Returns a child collection of paths that make up the route.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

balloon

```
{IMultiRouterRouteBalloon} balloon
```

Balloon of a route.

model

```
{multiRouter.pedestrian.RouteModel} model
```

The data model of an individual route.

properties

```
{data.Manager} properties
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

The route data. The following fields are available:

- index: Integer — The ordinal number of the route in a multi-stop route.
- type: String — The route type identifier; takes the value "pedestrian" for pedestrian routes.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the route in seconds.
- boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Events details

balloonclose

Closing the balloon.

balloonopen

Opening the balloon.

update

Updating the route rendering. Instance of the **Event** class.

Methods details

getPaths

```
{GeoObjectCollection} getPaths()
```

Returns a child collection of paths that make up the route.

multiRouter.pedestrian.RouteModel

Note:
The constructor of the multiRouter.pedestrian.RouteModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

A data model of a single pedestrian route. A multi-stop route can consist of several individual routes.

Fields | Events | Methods

Creates a data model of a single pedestrian route.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
multiRoute	**multiRouter.MultiRouteModel**	Reference to the parent model of a multi-stop route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	data.Manager	The route data. The following fields are available: • index: Integer — The ordinal number of the route in a multi-stop route. • type: String — The route type identifier; takes the value "pedestrian" for pedestrian routes. • distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the route in seconds. • boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class. Names of fields that are available via the Event.get method: • pathsChange: Boolean — Flag for whether the set of paths is changed.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getPaths()	multiRouter.pedestrian.PathModel[]	Returns array of route paths.
getType()	String	Returns ID of the route type. For pedestrian routes, returns the string "pedestrian".
update(routeJson)		Updates the state of the model.

Fields details

multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

properties

```
{data.Manager} properties
```

The route data. The following fields are available:

- index: Integer — The ordinal number of the route in a multi-stop route.
- type: String — The route type identifier; takes the value "pedestrian" for pedestrian routes.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the route in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the route in seconds.
- boundedBy: Number[][] — Coordinates of the upper and lower corners of the rectangle that bounds the route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

update

Updating the model with new data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- pathsChange: Boolean — Flag for whether the set of paths is changed.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getPaths

```
{multiRouter.pedestrian.PathModel[]} getPaths()
```

Returns array of route paths.

getType

```
{String} getType()
```

Returns ID of the route type. For pedestrian routes, returns the string "pedestrian".

update

```
{} update(routeJson)
```

Updates the state of the model.

Parameters:

Parameter	Default value	Description
routeJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

multiRouter.pedestrian.SegmentModel

Note:
The constructor of the multiRouter.pedestrian.SegmentModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

A data model of a pedestrian route segment.

Fields | Events | Methods

Creates a data model of a pedestrian route segment.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
geometry	geometry.base.LineString	Geometry of a segment.
path	multiRouter.pedestrian.PathModel	Reference to the parent model of the path.
properties	data.Manager	Segment data. The following fields are available: • index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path. • type: String — Segment type identifier, which takes the value "pedestrian" for pedestrian segments. • text: String — Text description of the segment. • distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters. • duration: Object — An object with the "text" and "value" fields that describes the travel time of the segment in seconds. • lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events

Name	Description
update	Updating the model with new data. Instance of the Event class.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getType()	String	Returns ID of the segment type. For walking segments of pedestrian routes, returns the string "pedestrian".

Fields details

geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

path

```
{multiRouter.pedestrian.PathModel} path
```

Reference to the parent model of the path.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- index: Integer — The ordinal number of the segment in the array of segments of the corresponding route path.
- type: String — Segment type identifier, which takes the value "pedestrian" for pedestrian segments.
- text: String — Text description of the segment.
- distance: Object — An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object — An object with the "text" and "value" fields that describes the travel time of the segment in seconds.
- lodIndex: Integer — The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

Events details

update

Updating the model with new data. Instance of the **Event** class.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getType

```
{String} getType()
```

Returns ID of the segment type. For walking segments of pedestrian routes, returns the string "pedestrian".

multiRouter.pedestrian.WalkSegment

Note:
The constructor of the multiRouter.pedestrian.WalkSegment class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of a segment on a pedestrian route.

Fields | Events | Methods

Creates the representation to display a segment on a pedestrian route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
geometry	IGeometry|null	Geo object geometry. Inherited from IGeoObject.
options	IOptionManager	Options manager. Inherited from ICustomizable.
properties	IDataManager	Geo object data. Inherited from IGeoObject.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multiRouter.ViaPoint

Note:
The constructor of the multiRouter.ViaPoint class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of a throughpoint. Throughpoints do not mean a stop. Thus, when passing via a throughpoint, the segment of the route path doesn't break.

Fields | Events | Methods

Creates a representation for displaying the throughpoint.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
model	**multiRouter.ViaPointModel**	The data model of a throughpoint.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**data.Manager**	Data for a multi-stop route throughpoint. The following fields are available: • index: Integer — The sequential number of the point. • lodIndex: Integer — The ordinal number of the throughpoint in the full set of coordinates of the corresponding route path.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
update	Updating a representation for displaying the throughpoint. Instance of the Event class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Fields details

model

```
{multiRouter.ViaPointModel} model
```

The data model of a throughpoint.

properties

```
{data.Manager} properties
```

Data for a multi-stop route throughpoint. The following fields are available:

- index: Integer — The sequential number of the point.
- lodIndex: Integer — The ordinal number of the throughpoint in the full set of coordinates of the corresponding route path.

Events details

update

Updating a representation for displaying the throughpoint. Instance of the **Event** class.

multiRouter.ViaPointModel

Note:
The constructor of the multiRouter.ViaPointModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

The data model of a throughpoint on the multi-stop route. Throughpoints do not signify a stop. When passing via a throughpoint, the segment of the route path doesn't break.

Fields | Events | Methods

Creates the data model of a multi-stop route throughpoint.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
geometry	geometry.base.Point	Geometry of a multi-stop route throughpoint.
multiRoute	multiRouter.MultiRouteModel	Reference to the parent model of a multi-stop route.
properties	data.Manager	Data for a multi-stop route throughpoint. The following fields are available: • index: Integer — The sequential number of the point. • lodIndex: Integer — The ordinal number of the throughpoint in the full set of coordinates of the corresponding route path.

Events

Name	Description
referencepointchange	Change to the reference point. Instance of the Event class. Names of fields that are available via the Event.get method: • oldReferencePoint: Object — Description of the previous reference point.
update	Updating the model with new data. Instance of the Event class.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getReferencePoint()	Object	Returns the corresponding reference point.
getReferencePointIndex()	Integer	Returns the index corresponding to a reference point in the set of reference points for the parent multi-stop route.
setReferencePoint(referencePoint)		Sets the description for the corresponding reference point. A reference point can be set using one of the following ways: • A string containing the postal address of the reference point. • An array containing the latitude and longitude of the reference point. • A geometry.Point geometry describing the reference point.
update(viaPointJson)		Updates the model with new data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

geometry

```
{geometry.base.Point} geometry
```

Geometry of a multi-stop route throughpoint.

multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

properties

```
{data.Manager} properties
```

Data for a multi-stop route throughpoint. The following fields are available:

- index: Integer — The sequential number of the point.
- lodIndex: Integer — The ordinal number of the throughpoint in the full set of coordinates of the corresponding route path.

Events details

referencepointchange

Change to the reference point. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldReferencePoint: Object — Description of the previous reference point.

update

Updating the model with new data. Instance of the **Event** class.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getReferencePoint

```
{Object} getReferencePoint()
```

Returns the corresponding reference point.

getReferencePointIndex

```
{Integer} getReferencePointIndex()
```

Returns the index corresponding to a reference point in the set of reference points for the parent multi-stop route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

setReferencePoint

```
{} setReferencePoint(referencePoint)
```

Sets the description for the corresponding reference point. A reference point can be set using one of the following ways:

- A string containing the postal address of the reference point.
- An array containing the latitude and longitude of the reference point.
- A geometry.Point geometry describing the reference point.

Parameters:

Parameter	Default value	Description
referencePoint*	—	Type: Object Description of the reference point.

* Mandatory parameter/option.

update

```
{} update(viaPointJson)
```

Updates the model with new data.

Parameters:

Parameter	Default value	Description
viaPointJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

multiRouter.WayPoint

Note:
The constructor of the multiRouter.WayPoint class is hidden, as this class is not intended for autonomous initialization.

Extends **IGeoObject**.

Representation of a waypoint. A waypoint means a stop, and waypoints divide the route into so-called paths.

Fields | Events | Methods

Creates a representation for displaying the waypoint.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
geometry	IGeometry|null	Geo object geometry. Inherited from IGeoObject.
model	multiRouter.WayPointModel	The data model of a waypoint.
options	IOptionManager	Options manager. Inherited from ICustomizable.
properties	data.Manager	Data for a multi-stop route waypoint. The following fields are available: • index: Integer — The sequential number of the point. • request: String — The request correpsonding to the point of the request. • address: String — The postal address of the point. • description: String — Description of the point. • name: String — Name of the point. • geocoderMetaData: Object — Geocoder metadata.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
update	Updating a representation for displaying the waypoint. Instance of the Event class.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

model

```
{multiRouter.WayPointModel} model
```

The data model of a waypoint.

properties

```
{data.Manager} properties
```

Data for a multi-stop route waypoint. The following fields are available:

- index: Integer — The sequential number of the point.
- request: String — The request correpsonding to the point of the request.
- address: String — The postal address of the point.
- description: String — Description of the point.
- name: String — Name of the point.
- geocoderMetaData: Object — Geocoder metadata.

Events details

update

Updating a representation for displaying the waypoint. Instance of the **Event** class.

multiRouter.WayPointModel

Note:
The constructor of the multiRouter.WayPointModel class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

The data model of a multi-stop route waypoint. A waypoint means a stop, and waypoints divide the route into so-called paths.

Fields | Events | Methods

Creates the data model of a multi-stop route waypoint.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
geometry	**geometry.base.Point**	Geometry of a multi-stop route waypoint.
multiRoute	**multiRouter.MultiRouteModel**	Reference to the parent model of a multi-stop route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
properties	data.Manager	Data for a multi-stop route waypoint. The following fields are available: • index: Integer — The sequential number of the point. • request: String — The request correpsonding to the point of the request. • address: String — The postal address of the point. • description: String — Description of the point. • name: String — Name of the point. • geocoderMetaData: Object — Geocoder metadata.

Events

Name	Description
referencepointchange	Change to the reference point. Instance of the Event class. Names of fields that are available via the Event.get method: • oldReferencePoint: Object — Description of the previous reference point.
update	Updating the model with new data. Instance of the Event class.

Methods

Name	Returns	Description
destroy()		Destroys a model.
getReferencePoint()	Object	Returns the corresponding reference point.
getReferencePointIndex()	Integer	Returns the index corresponding to a reference point in the set of reference points for the parent multi-stop route.
setReferencePoint(referencePoint)		Sets the description for the corresponding reference point. A reference point can be set using one of the following ways: • A string containing the postal address of the reference point. • An array containing the latitude and longitude of the reference point. • A geometry.Point geometry describing the reference point.
update(wayPointJson)		Updates the model with new data.

Fields details

geometry

```
{geometry.base.Point} geometry
```

Geometry of a multi-stop route waypoint.

multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Reference to the parent model of a multi-stop route.

properties

```
{data.Manager} properties
```

Data for a multi-stop route waypoint. The following fields are available:

- index: Integer — The sequential number of the point.
- request: String — The request correpsonding to the point of the request.
- address: String — The postal address of the point.
- description: String — Description of the point.
- name: String — Name of the point.
- geocoderMetaData: Object — Geocoder metadata.

Events details

referencepointchange

Change to the reference point. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- oldReferencePoint: Object — Description of the previous reference point.

update

Updating the model with new data. Instance of the **Event** class.

Methods details

destroy

```
{} destroy()
```

Destroys a model.

getReferencePoint

```
{Object} getReferencePoint()
```

Returns the corresponding reference point.

getReferencePointIndex

```
{Integer} getReferencePointIndex()
```

Returns the index corresponding to a reference point in the set of reference points for the parent multi-stop route.

setReferencePoint

```
{} setReferencePoint(referencePoint)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Sets the description for the corresponding reference point. A reference point can be set using one of the following ways:

- A string containing the postal address of the reference point.
- An array containing the latitude and longitude of the reference point.
- A geometry.Point geometry describing the reference point.

Parameters:

Parameter	Default value	Description
referencePoint*	—	Type: Object Description of the reference point.

* Mandatory parameter/option.

update

```
{} update(wayPointJson)
```

Updates the model with new data.

Parameters:

Parameter	Default value	Description
wayPointJson*	—	Type: Object JSON data.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

ObjectManager

Extends **ICustomizable**, **IEventEmitter**, **IGeoObject**, **IParentOnMap**.

Object manager. Allows optimally displaying, clustering, and managing visibility for objects. Note that objects drawn on the map via this manager can't have editing and dragging modes enabled.

Constructor | Fields | Events | Methods

Constructor

```
ObjectManager([options])
```

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options. • You can set all the options specified in the **Clusterer** description, except for hasBalloon and hasHint options. • Cluster options are set with the "cluster" prefix. The list of options is specified in the description of **ClusterPlacemark**; • Options for singular objects should be specified with the geoObject prefix. The list of options is specified in **GeoObject**.
options.clusterize	false	Type: Boolean Flag indicating whether the objects should be clusterized. Note that clusterization only works for point objects at this time. When cluster mode is enabled, all non-point objects are ignored.
options.syncOverlayInit	false	Type: Boolean A flag that allows creating overlays for objects synchronously. Note that when you create an overlay synchronously, you should ensure that the appropriate class, which implements the IOverlay interface, is loaded. By default, the overlays are created asynchronously, and the overlay class is loaded on demand.
options.viewportMargin	128	Type: Number|Number[] Offset for the area where the objects are shown. Use this option to expand the view area of objects relative to the visible area of the map.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
var objectManager = new ymaps.ObjectManager({
    // Enabling clustering.
    clusterize: true,
    // Cluster options are set with the "cluster" prefix.
    clusterHasBalloon: false,
    // Geo object options are set with the "geoObject" prefix.
    geoObjectOpenBalloonOnClick: false
});

// You can set options directly for child collections.
objectManager.clusters.options.set({
    preset: 'islands#grayClusterIcons',
    hintContentLayout: ymaps.templateLayoutFactory.createClass('Group of
objects')
});
objectManager.objects.options.set('preset', 'islands#grayIcon');
```

Fields

Name	Type	Description
clusters	objectManager.ClusterCollection	Collection of clusters generated by the manager.
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
geometry	IGeometry|null	Geo object geometry. Inherited from IGeoObject.
objects	objectManager.ObjectCollection	Collection of objects added to the layer.
options	IOptionManager	Options manager. Inherited from ICustomizable.
properties	IDataManager	Geo object data. Inherited from IGeoObject.
state	IDataManager	State of the geo object. Inherited from IGeoObject.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
geometrychange	Change to the geo object geometry. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the geometry. Inherited from IGeoObject.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IParentOnMap.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
add(objects)	**ObjectManager**	Returns reference to the object manager.
getBounds()	Number[][]|null	Calculates the boundaries in geo coordinates for an area that covers all the objects in the manager.
getFilter()	String|Function|null	Returns the set filter function.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getObjectState(id)	Object	Getting information about the current state of an object added to the manager.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getPixelBounds()	Number[][]|null	Calculates the boundaries in global pixel coordinates for an area that covers all the objects in the manager.
remove(objects)	**ObjectManager**	Returns reference to the object manager.
removeAll()	**ObjectManager**	Deleting all objects from the manager.
setFilter(filterFunction)		Sets a filter function for objects.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.

Fields details

clusters

```
{objectManager.ClusterCollection} clusters
```

Collection of clusters generated by the manager.

Example:

```
objectManager.clusters.events.add('click', function (e) {
    var objectId = e.get('objectId');
    objectManager.clusters.balloon.open(objectId);
});
```

objects

```
{objectManager.ObjectCollection} objects
```

Collection of objects added to the layer.

Example:

```
objectManager.objects.events.add('click', function (e) {
    var objectId = e.get('objectId');
    objectManager.objects.balloon.open(objectId);
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

add

```
{ObjectManager} add(objects)
```

Returns reference to the object manager.

Parameters:

Parameter	Default value	Description
objects*	—	Type: Object|Object[]|String String or object with a JSON description of the objects. JSON object descriptions are formed using the following approach (see the example below). An object may be an entity or a collection of entities. A collection of entities is made up of an object with the following fields: • type — Type of object. The value of the field must be "FeatureCollection". • features — Array of child entities in the collection. Child objects may be entities or nested collections of entities. An entity is made up of an object with the following fields: • id — Unique object ID. Mandatory field. • type — Type of object. The value of the field must be "Feature". • geometry — Object geometry. Contains the "type" and "coordinates" fields. The value corresponds to the **GeoObject** passed to the constructor. • options — Options for the geo object. • properties — Geo object data.

* Mandatory parameter/option.

Examples:

1.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
var myObjectManager = new ymaps.ObjectManager({ clusterize: true }),
    currentId = 0;

// Adding a single object.
myObjectManager.add({
    type: 'Feature',
    id: currentId++,
    geometry: {
        type: 'Point',
        coordinates: [56.23, 34.79]
    },
    properties: {
        hintContent: 'Popup hint text',
        balloonContent: 'Balloon content'
    }
});
map.geoObjects.add(objectManager);
```

2.

```
// Adding an array of point objects.
var myObjects = [];
for (var i = 0, l = coordinates.length; i < l; i++) {
    myObjects.push({
        type: 'Feature',
        id: currentId++,
        geometry: {
            type: 'Point',
            coordinates: coordinates[i]
        }
    });
}
myObjectManager.add(myObjects);
map.geoObjects.add(objectManager);
```

3.

```
// Adding collections of objects.
var collection = {
    type: 'FeatureCollection',
    features: [{
            type: 'Feature',
            id: currentId++,
            geometry: {
                type: 'Point',
                coordinates: [24.34, 65.24]
            }
        }, {
            type: 'Feature',
            id: currentId++,
            geometry: {
                type: 'Point',
                coordinates: [25.34, 63.24]
            }
        }
   ]
}
myObjectManager.add(collection);
map.geoObjects.add(objectManager);
```

4.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Adding non-point objects
var myObjects = [];

// Adding a circle
myObjects.push({
    type: 'Feature',
    id: 1,
    geometry: {
        type: 'Circle',
        coordinates: [55.755381, 37.619044],
        radius: 300
    }
});

// Adding a rectangle
myObjects.push({
    type: 'Feature',
    id: 2,
    geometry: {
        type: 'Rectangle',
        coordinates: [
            [55.764286, 37.606169],
            [55.759688, 37.620588]
        ]
    },
    options: {
        fillColor: '#FFFFFF',
        opacity: 0.8
    }
});

// Adding a polyline
myObjects.push({
    type: 'Feature',
    id: 3,
    geometry: {
        type: 'LineString',
        coordinates: [
            [55.75901100, 37.6308886],
            [55.7516538, 37.6299444],
            [55.74603822, 37.6380125]
        ]
    },
    options: {
        strokeColor: "#FF0000",
        strokeWidth: 5
    }
});

// Adding a polygon
myObjects.push({
    type: 'Feature',
    id: 4,
    geometry: {
        type: 'Polygon',
        coordinates: [[
            [55.75175065, 37.6041094],
            [55.75005637, 37.6137224],
            [55.742891186, 37.6166407],
            [55.74153546, 37.60342281],
            [55.74700649, 37.59775798]
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
        ]]
    },
    options: {
        opacity: 0.2,
        strokeWidth: 2,
        fillColor: '#00FF00'
    }
});

objectManager.add(myObjects);
```

getBounds

```
{Number[][]|null} getBounds()
```

Calculates the boundaries in geo coordinates for an area that covers all the objects in the manager.

Returns array of the area's coordinates, or null if the manager has not been added to the map.

getFilter

```
{String|Function|null} getFilter()
```

Returns the set filter function.

Parameters:

Parameter	Default value	Description
*	—	Type:

* Mandatory parameter/option.

getObjectState

```
{Object} getObjectState(id)
```

Getting information about the current state of an object added to the manager.

Returns object with following fields:

- found — Attribute that indicates whether an object with the passed ID exists. Type: Boolean.
- isShown — Attribute that indicates whether the object is located in the visible area of the map. Type: Boolean.
- cluster — JSON description of the cluster the object was added to. Besides the required fields, it contains the properties.geoObjects field with an array of objects that are in the cluster. This field is returned only when clusterization is enabled.
- isClustered — Attribute indicating whether an object is in the cluster. This field is returned only when clusterization is enabled. Type: Boolean.
- isFilteredOut — Attribute indicating whether an object passed through filtration. If the filter is not set or the object passed through filtration, the value of the field is "false". Type: Boolean.

.

Parameters:

Parameter	Default value	Description
id*	—	Type: Object ID of the object to get the state for.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Example:

```
// Opening the cluster balloon with the selected object.
// Getting data about the state of an object inside the cluster.
var objectState = objectManager.getObjectState(myObjects[1].id);
// Checking whether the object is located in the visible area of the map.
if (objectState.found && objectState.isShown) {
    // If the object is in a cluster, we open the cluster balloon with the
appropriate object selected.
    if (objectState.isClustered) {
        objectManager.clusters.state.set('activeObject', myObjects[1]);
        objectManager.clusters.balloon.open(objectState.cluster.id);
    } else {
        // If the object isn't in a cluster, we open its own balloon.
        objectManager.objects.balloon.open(myObjects[i].id);
    }
}
```

getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Calculates the boundaries in global pixel coordinates for an area that covers all the objects in the manager.

Returns array of the area's coordinates, or null if the manager has not been added to the map.

remove

```
{ObjectManager} remove(objects)
```

Returns reference to the object manager.

Parameters:

Parameter	Default value	Description
objects*	—	Type: Object|Object[]|String A string object with a JSON description of objects or an array of IDs of the objects being deleted. For the object description format, see the description of the method **ObjectManager.add**

* Mandatory parameter/option.

Examples:

1.

```
// Removing objects with IDs 0 and 56.
myObjectManager.remove([0, 56]);
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Deleting an array of objects
myObjectManager.remove([
    {
        type: 'Feature',
        id: 23,
        geometry: {
            type: 'Point',
            coordinates: [45.33, 24.33]
        }
    },
    {
        type: 'Feature',
        id: 52,
        geometry: {
            type: 'Point',
            coordinates: [45.23, 24.34]
        }
    },
]);
```

3.

```
// Deleting a collection of objects.
myObjectManager.remove({
    type: 'FeatureCollection',
    features: [
         // Describing objects in the collection
         ...
    ]
});
```

removeAll

```
{ObjectManager} removeAll()
```

Deleting all objects from the manager.

Returns self-reference.

Example:

```
myObjectManager.removeAll();
```

setFilter

```
{} setFilter(filterFunction)
```

Sets a filter function for objects.

Parameters:

Parameter	Default value	Description
filterFunction*	—	Type: Function|String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Filter function. Takes a single object added to ObjectManager. If the function returns true, the object will be processed. If false, the object will be excluded from further processing. A string can also be passed as a filter. The following keywords are available in the string filter: • options — Accessing the object's options. • properties — Accessing the object's data. • geometry — Accessing the object's geometry. • id — Accessing the object's identifier. For a filter, you can specify an expression that returns true or false.

* Mandatory parameter/option.

Examples:

1.

```
// Skipping objects with an ID under 100.
objectManager.setFilter('id &gt; 100');
```

2.

```
// Only objects of the specified types will be displayed on the map.
objectManager.setFilter('properties.type == "cafe" || properties.type ==
"pharmacy"');
```

3.

```
// You can define a filter function.
objectManager.setFilter(function (object) {
    return object.properties.name != 'The one who cannot be shown.';
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

objectManager

objectManager.addon

objectManager.addon.clustersBalloon

Note:
The constructor of the objectManager.addon.clustersBalloon class is hidden, as this class is not intended for autonomous initialization.

Static object.

Methods

Example:

```
&lt;script src=&quot;http://api-maps.yandex.ru/2.1/?
lang=ru_RU&load=Map,ObjectManager,objectManager.addon.clustersBalloon&quot;type=
&quot;text/javascript&quot;&gt;&lt;/script&gt;
```

Methods

Name	Returns	Description
get()	IPopupManager	Returns balloon manager of the object manager.

Methods details

get

```
{IPopupManager} get()
```

Returns balloon manager of the object manager.

Example:

```
ymaps.objectManager.addon.clustersBalloon.get(myMap)
```

objectManager.addon.clustersHint

Note:
The constructor of the objectManager.addon.clustersHint class is hidden, as this class is not intended for autonomous initialization.

Static object.

Methods

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
&lt;script src=&quot;http://api-maps.yandex.ru/2.1/?
lang=ru_RU&load=Map,ObjectManager,objectManager.addon.clustersHint&quot;type=&qu
ot;text/javascript&quot;&gt;&lt;/script&gt;
```

Methods

Name	Returns	Description
get()	IPopupManager	Returns hint manager ObjectManager.

Methods details

get

```
{IPopupManager} get()
```

Returns hint manager **ObjectManager**.

Example:

```
ymaps.objectManager.addon.clustersHint.get(myMap)
```

objectManager.addon.objectsBalloon

Note:
The constructor of the objectManager.addon.objectsBalloon class is hidden, as this class is not intended for autonomous initialization.

Static object.

Methods

Example:

```
&lt;script src=&quot;http://api-maps.yandex.ru/2.1/?
lang=ru_RU&load=Map,ObjectManager,objectManager.addon.objectsBalloon&quot;type=&
quot;text/javascript&quot;&gt;&lt;/script&gt;
```

Methods

Name	Returns	Description
get(collection)	IPopupManager	Returns balloon manager of the object manager.

Methods details

get

```
{IPopupManager} get(collection)
```

Returns balloon manager of the object manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
collection*	—	Type: objectManager.ObjectCollection Collection

* Mandatory parameter/option.

Example:

```
ymaps.objectManager.addon.objectsBalloon.get(myMap)
```

objectManager.addon.objectsHint

Note:
The constructor of the objectManager.addon.objectsHint class is hidden, as this class is not intended for autonomous initialization.

Static object.

Methods

Example:

```
&lt;script src=&quot;http://api-maps.yandex.ru/2.1/?
lang=ru_RU&load=Map,ObjectManager,objectManager.addon.objectsHint&quot;type=&quo
t;text/javascript&quot;&gt;&lt;/script&gt;
```

Methods

Name	Returns	Description
get()	IPopupManager	Returns hint manager ObjectManager.

Methods details

get

```
{IPopupManager} get()
```

Returns hint manager ObjectManager.

Example:

```
ymaps.objectManager.addon.objectsHint.get(myMap)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

objectManager.Balloon

Extends **IBalloonManager**.

Manager for the collection balloon ObjectManager. Allows to manage the object's balloon by opening it and hiding it. It uses the map balloon manager **map.Balloon**. Collections of objects in **ObjectManager** contain an instance of this class, accessible as myObjectManager.objects.balloon and myObjectsLayer.clusters.balloon. Don't create new instances of this class unless necessary.

See Balloon

Constructor | Fields | Events | Methods

Constructor

```
objectManager.Balloon(collection)
```

Parameters:

Parameter	Default value	Description
collection *	—	Type: IReadOnlyCollection Collection of objects.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
autopanbegin	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
autopanend	End of automatic shifting of the map center initiated by the autoPan method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
beforeuserclose	The event which precedes **Balloon.event:userclose**. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.
close	Closing the info object. Names of fields available via **Event.get**: • target — Reference to the object where the closing occurred.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IPopupManager**.
open	Opening the info object. Names of fields available via **Event.get**: • target — Reference to the object where the opening occurred. Inherited from **IPopupManager**.
userclose	Balloon closed by the user. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • target — Reference to the **IBalloonOwner** object. Inherited from **IBalloonManager**.

Methods

Name	Returns	Description
autoPan()	**vow.Promise**	Moves the map so that the balloon is visible. Inherited from **IBalloonManager**.
close([force])	**vow.Promise**	Closes an open balloon.
destroy()		Disables the info object manager. Inherited from **IPopupManager**.
getData()	Object|null	Returns hash describing the object the balloon is open on, or null if the balloon was not opened.
getOptions()	**IOptionManager**|null	Returns the options manager or 'null'. Inherited from **IPopupManager**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopupManager**.
getOverlaySync()	**IOverlay**|null	Returns the overlay, if one exists. Inherited from **IPopupManager**.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from **IPopupManager**.
isOpen(id)	Boolean	A method that detects whether the balloon is open on the object with the passed identifier.
open(objectId, anchorPixelPosition)	**vow.Promise**	Opens the balloon on the object with the passed identifier.
setData(objectData)	**vow.Promise**	Sets new data for displaying the balloon.
setOptions(options)	**vow.Promise**	Defines new options for the info object. Inherited from **IPopupManager**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopupManager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

close

```
{vow.Promise} close([force])
```

Closes an open balloon.

Returns Promise object.

Parameters:

Parameter	Default value	Description
force	false	Type: Boolean Instant closure.

Example:

```
// Closing all balloons on the layer.
objectManager.objects.balloon.close();
objectManager.clusters.balloon.close();
```

getData

```
{Object|null} getData()
```

Returns hash describing the object the balloon is open on, or null if the balloon was not opened.

Example:

```
var cluster = objectManager.clusters.balloon.getData();
if (cluster) {
    alert('The balloon for a cluster of  ' + cluster.properties.geoObjects.length
+ ' objects is currently opened.');
}
```

isOpen

```
{Boolean} isOpen(id)
```

A method that detects whether the balloon is open on the object with the passed identifier.

Returns balloon state: open/closed.

Parameters:

Parameter	Default value	Description
id*	—	Type: Object Object ID.

* Mandatory parameter/option.

Example:

```
// Closing the ballon after the second click on the placemark.
objectManager.objects.options.set('hideIconOnBalloonOpen', false);
objectManager.objects.events.add('click', function (e) {
    var objectId = e.get('objectId');
    if (objectManager.objects.balloon.isOpen(objectId)) {
        objectManager.objects.balloon.close();
    }
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

open

```
{vow.Promise} open(objectId, anchorPixelPosition)
```

Opens the balloon on the object with the passed identifier.

Returns Promise object.

Parameters:

Parameter	Default value	Description
objectId*	—	Type: Object ID of the object to open the balloon on.
anchorPixelPosition*	—	Type:

* Mandatory parameter/option.

Example:

```
// Loading the object data before opening the balloon.
// Disabling automatic balloon opening by clicking on the object.
objectManager.options.set('geoObjectOpenBalloonOnClick', false);
objectManager.objects.events.add('click', function (e) {
    var objectId = e.get('objectId');
    // Preloading the object data after the object was clicked.
    loadObjectData(objectId).then(function (data) {
        // As soon as the data is ready, we're assigning it to the object and
opening the balloon.
        objectManager.objects.getById(objectId).properties = data;
        objectManager.objects.balloon.open(objectId);
    });
});
```

setData

```
{vow.Promise} setData(objectData)
```

Sets new data for displaying the balloon.

Returns Promise object.

Parameters:

Parameter	Default value	Description
objectData*	—	Type: Object Hash with a description of the object to open the balloon on. Corresponds to the object description that is input to ObjectManager.add.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Preloading the object data after the ballon is opened on the object.
objectManager.objects.events.add('balloonopen', function (e) {
    var objectId = e.get('objectId'),
        object = objectManager.objects.getById(objectId);
    if (!object.properties) {
        loadObjectData(objectId).then(function (data) {
            if (objectManager.balloon.isOpen(objectid)) {
                object.properties = data;
                objectManager.objects.balloon.setData(object);
            }
        });
    }
});
```

objectManager.ClusterCollection

Extends **ICustomizable**, **IEventEmitter**.

Collection of clusters generated by **ObjectManager**. Clusters are added to and deleted from the collection automatically, and are read-only. The cluster object is a JSON structure the same as the objects in the layer. Cluster object field:

- id — Unique cluster ID.
- geometry — Description of the cluster geometry.
- properties — Description of the cluster data. The properties.geoObjects field stores an array of objects that are included in the cluster.
- options — Cluster options. Optional field.

Constructor | Fields | Events | Methods

Constructor

```
objectManager.ClusterCollection()
```

Fields

Name	Type	Description
balloon	**objectManager.Balloon**	Cluster balloon in the manager.
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
hint	**objectManager.Hint**	Object hint in the **ObjectManager**. Names of fields available via **Event.get**: • objectId — ID of the object where the hint was shown.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
options	option.Manager	Options manager. Names of fields that are available via the option.Manager#get method: • hasBalloon — Indicates whether the collection has the .balloon field. If a balloon doesn't need to be opened when clicking the cluster, we recommend setting this option to the "false" value to avoid unnecessary initializations. • hasHint — Indicates whether the collection has the .hint field. If a popup hint doesn't need to be displayed when the cluster is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations. • hideIconOnBalloonOpen — Hide the icon when opening the balloon. Default value: true. • openBalloonOnClick — Option that allows you to forbid opening the balloon when clicking on a cluster. By default, opening the balloon is allowed. • openHintOnHover — Option that allows you to forbid displaying the popup hint when the cluster is pointed at. By default, showing hints is allowed.
overlays	objectManager.OverlayCollection	Collection of cluster overlays. All events, with the exception of "add" and "remove" events, propagate from the collection of overlays to the collection of clusters.
state	data.Manager	State of the collection of clusters. Defined by the following fields: • activeObject — JSON description of the object selected in the cluster balloon.

Events

Name	Description
add	Adds a cluster to the collection. Instance of the Event class. Names of fields that are available via the Event.get method: • objectId — ID of the added object. • child — The added object.
clusteroptionschange	Modification of cluster options via the objectManager.ClusterCollection.setClusterOptions method. Instance of the Event class. Names of fields that are available via the Event.get method: • objectId — ID of the cluster that had options modified.
optionschange	Change to the object options. Inherited from ICustomizable.
remove	Deletes a cluster from the collection. Instance of the Event class. Names of fields that are available via the Event.get method: • objectId — ID of the deleted object. • child — The deleted object.

Methods

Name	Returns	Description
each(callback, context)		
getAll()	Object[]	Returns array of objects contained in the collection.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getById(id)	Object|null	Returns cluster object with the specified ID, or null if this cluster does not exist.
getIterator()	IIterator	Returns iterator for the collection.
getLength()	Number	Returns the number of objects in the collection.
getObjectManager()	ObjectManager	Returns the parent layer of objects in the collection.
setClusterOptions(objectId, options)	objectManager.ObjectCollection	Returns self-reference.

Fields details

balloon

```
{objectManager.Balloon} balloon
```

Cluster balloon in the manager.

hint

```
{objectManager.Hint} hint
```

Object hint in the ObjectManager. Names of fields available via Event.get:

- objectId — ID of the object where the hint was shown.

options

```
{option.Manager} options
```

Options manager. Names of fields that are available via the option.Manager#get method:

- hasBalloon — Indicates whether the collection has the .balloon field. If a balloon doesn't need to be opened when clicking the cluster, we recommend setting this option to the "false" value to avoid unnecessary initializations.
- hasHint — Indicates whether the collection has the .hint field. If a popup hint doesn't need to be displayed when the cluster is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations.
- hideIconOnBalloonOpen — Hide the icon when opening the balloon. Default value: true.
- openBalloonOnClick — Option that allows you to forbid opening the balloon when clicking on a cluster. By default, opening the balloon is allowed.
- openHintOnHover — Option that allows you to forbid displaying the popup hint when the cluster is pointed at. By default, showing hints is allowed.

Example:

```
objectManager.objects.options.set({
    preset: 'islands#greenDotIcon',
    hintContentLayout:
ymaps.templateLayoutFactory.createClass('{{propeties.name}}')
});
```

overlays

```
{objectManager.OverlayCollection} overlays
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Collection of cluster overlays. All events, with the exception of "add" and "remove" events, propagate from the collection of overlays to the collection of clusters.

Example:

```
// Changing the color of the cluster icon when moused over.
objectManager.clusters.events.add(['mouseenter', 'mouseleave'], function (e) {
    var objectId = e.get('objectId'),
        overlay = objectManager.clusters.overlays.getById(objectId);
    if (e.get('type') == 'mouseenter') {
        setRedColor(objectId);
        overlay.events.add('mapchange', onMapChange);
    } else {
        setGreenColor(objectId);
        overlay.events.remove('mapchange', onMapChange);
    }
});

function onMapChange (e) {
    setGreenColor(objectManager.clusters.overlays.getId(e.get('target')));
}

function setGreenColor (objectId) {
    objectManager.clusters.setClusterOptions(objectId, {
        preset: 'islands#greenClusterIcons'
    });
}

function setRedColor (objectId) {
    objectManager.clusters.setClusterOptions(objectId, {
        preset: 'islands#redClusterIcons'
    });
}
```

state

```
{data.Manager} state
```

State of the collection of clusters. Defined by the following fields:

- activeObject — JSON description of the object selected in the cluster balloon.

Example:

```
// Opening the cluster balloon with the selected object.
var objectState = objectManager.getObjectState(myObjects[i]);
if (objectState.isClustered) {
    objectManager.clusters.state.set('activeObject', myObjects[i]);
    objectManager.clusters.balloon.open(objectState.cluster.id);
}
```

Events details

add

Adds a cluster to the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- objectId — ID of the added object.
- child — The added object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

clusteroptionschange

Modification of cluster options via the **objectManager.ClusterCollection.setClusterOptions** method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- objectId — ID of the cluster that had options modified.

remove

Deletes a cluster from the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- objectId — ID of the deleted object.
- child — The deleted object.

Methods details

each

```
{} each(callback, context)
```

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Callback function that the collection objects are passed to.
context*	—	Type: Object Context for the callback.

* Mandatory parameter/option.

Example:

```
var clusterizedObjectsCounter = 0;
objectManager.clusters.each(function (cluster) {
    clusterizedObjectsCounter += cluster.properties.geoObjects.length;
});
alert('The map shows ' + clusterizedObjectsCounter + ' clusterized objects.');
```

getAll

```
{Object[]} getAll()
```

Returns array of objects contained in the collection.

Example:

```
var clusterArray = objectManager.clusters.getAll();
```

getById

```
{Object|null} getById(id)
```

Returns cluster object with the specified ID, or null if this cluster does not exist.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
id*	—	Type: String Cluster ID.

* Mandatory parameter/option.

Example:

```
// Making the cluster color change if it has more than 20 objects.
objectManager.clusters.events.add('add', function (e) {
    var cluster = objectManager.clusters.getById(e.get('objectId')),
        objects = cluster.properties.geoObjects;
    if (objects.length &gt; 20) {
        objectManager.clusters.setClusterOptions(cluster.id, {
            preset: 'islands#redClusterIcons'
        });
    }
});
```

getIterator

```
{IIterator} getIterator()
```

Returns iterator for the collection.

Example:

```
var clusterizedObjectsCounter = 0,
    it = objectManager.clusters.getIterator(),
    cluster;
while ((cluster = it.getNext()) != it.STOP_ITERATION)
    clusterizedObjectsCounter += cluster.properties.geoObjects.length;
});
alert('The map displays ' + clusterizedObjectsCounter + ' clusterized objects.');
```

getLength

```
{Number} getLength()
```

Returns the number of objects in the collection.

Example:

```
alert('The map displays ' + objectManager.clusters.getLength() + ' clusters.');
```

getObjectManager

```
{ObjectManager} getObjectManager()
```

Returns the parent layer of objects in the collection.

setClusterOptions

```
{objectManager.ObjectCollection} setClusterOptions(objectId, options)
```

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
objectId*	—	Type: String Cluster ID.
options*	—	Type: Object Object with cluster options.

* Mandatory parameter/option.

Example:

```
// Making the cluster color change if it has more than 20 objects.
objectManager.clusters.events.add('add', function (e) {
    var cluster = objectManager.clusters.getById(e.get('objectId')),
        objects = cluster.properties.geoObjects;
    if (objects.length &gt; 20) {
        objectManager.clusters.setClusterOptions(cluster.id, {
            preset: 'islands#redClusterIcons'
        });
    }
});
```

objectManager.Hint

Extends **IHintManager**.

Manager of the hint on an object layer. Allows to manage the hint on an object layer by opening it and hiding it. It uses the map hint manager **map.Hint** inside itself. Object layers contains instances of this class, accessible as myObjectManager.objects.hint and myObjectManager.clusters.hint. Don't create new instances of this class unless necessary.

See Hint

Constructor | Fields | Events | Methods

Constructor

```
objectManager.Hint(collection)
```

Parameters:

Parameter	Default value	Description
collection *	—	Type: IReadOnlyCollection Object layer.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
close	Closing the info object. Names of fields available via Event.get: • target — Reference to the object where the closing occurred. Inherited from IPopupManager.
open	Opening the info object. Names of fields available via Event.get: • target — Reference to the object where the opening occurred. Inherited from IPopupManager.

Methods

Name	Returns	Description
close([force])	vow.Promise	Hides the popup hint.
destroy()		Disables the info object manager. Inherited from IPopupManager.
getData()	Object|null	Returns hash describing the object the hint is shown on, or null if the hint was not shown.
getOptions()	IOptionManager|null	Returns the options manager or 'null'. Inherited from IPopupManager.
getOverlay()	vow.Promise	Returns the promise object to return the overlay. Inherited from IPopupManager.
getOverlaySync()	IOverlay|null	Returns the overlay, if one exists. Inherited from IPopupManager.
getPosition()	Number[]|null	Returns the coordinates of the info object or 'null'. Inherited from IPopupManager.
isOpen(id)	Boolean	A method that determines whether the popup hint was shown on the object with the passed ID.
open(objectId[, position])	vow.Promise	Displays the popup hint on the object with the passed ID.
setData(objectData)	vow.Promise	Sets new data for displaying the popup hint.
setOptions(options)	vow.Promise	Defines new options for the info object. Inherited from IPopupManager.
setPosition(position)	vow.Promise	Specifies a new position for the info object. Inherited from IPopupManager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

close

```
{vow.Promise} close([force])
```

Hides the popup hint.

Returns Promise object.

Parameters:

Parameter	Default value	Description
force	false	Type: Boolean Instant closure.

Example:

```
// Closing all hints on a layer.
objectManager.objects.hint.close();
objectManager.clusters.hint.close();
```

getData

```
{Object|null} getData()
```

Returns hash describing the object the hint is shown on, or null if the hint was not shown.

Example:

```
var cluster = objectManager.clusters.hint.getData();
if (cluster) {
    alert('Cluster displays a hint.');
}
```

isOpen

```
{Boolean} isOpen(id)
```

A method that determines whether the popup hint was shown on the object with the passed ID.

Returns hint state: shown/hidden.

Parameters:

Parameter	Default value	Description
id*	—	Type: Object Object ID.

* Mandatory parameter/option.

Example:

```
// Closing the hint when the object is clicked.
objectManager.objects.add('click', function (e) {
    var objectId = e.get('objectId');
    if (objectManager.objects.hint.isOpen(objectId)) {
        objectManager.objects.hint.close();
    }
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

open

```
{vow.Promise} open(objectId[, position])
```

Displays the popup hint on the object with the passed ID.

Returns Promise object.

Parameters:

Parameter	Default value	Description
objectId*	—	Type: Object ID of the object to open the hint on.
position	—	Type: Number[] The position for showing the popup hint, in global pixel coordinates. If the value is not specified, the hint will appear at the geometric center of the object.

* Mandatory parameter/option.

Example:

```
objectManager.clusters.hint.open(objectId);
```

setData

```
{vow.Promise} setData(objectData)
```

Sets new data for displaying the popup hint.

Returns Promise object.

Parameters:

Parameter	Default value	Description
objectData*	—	Type: Object Hash with a description of the object to open the hint on. Corresponds to the object description that is input to ObjectManager.add.

* Mandatory parameter/option.

Example:

```
objectManager.objects.hint.setData(objectManager.objects.getById(objectId));
```

objectManager.ObjectCollection

Extends **ICollection**, **ICustomizable**.

Collection of objects added to the layer.

Constructor | Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
objectManager.ObjectCollection()
```

Fields

Name	Type	Description
balloon	objectManager.Balloon	Object balloon in the manager.
events	IEventManager	Event manager. Inherited from IEventEmitter.
hint	objectManager.Hint	Object hint in the ObjectManager.
options	option.Manager	Options manager. Names of fields that are available via the option.Manager#get method: • hasBalloon — Indicates whether the collection has the .balloon field. If a balloon doesn't need to be opened when clicking the object, we recommend setting this option to the "false" value to avoid unnecessary initializations. • hasHint — Indicates whether the collection has the .hint field. If a popup hint doesn't need to be displayed when the object is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations. • hideIconOnBalloonOpen — Hide the icon when opening the balloon. Default value: true. • openBalloonOnClick — Show the balloon when clicking the object. Default value: true.
overlays	objectManager.OverlayCollection	Collection of overlays of singular objects. All events, with the exception of "add" and "remove" events, propagate from the collection of overlays to the collection of objects.

Events

Name	Description
add	Adds an object to the collection. Instance of the Event class. Names of fields that are available via the Event.get method: • objectId — ID of the added object. • child — The added object.
objectoptionschange	Modification of object options via the objectManager.ObjectCollection.setObjectOptions method. Instance of the Event class. Names of fields that are available via the Event.get method: • objectId — ID of the object that had options modified.
optionschange	Change to the object options. Inherited from ICustomizable.
remove	Removes an object from the collection. Instance of the Event class. Names of fields that are available via the Event.get method: • objectId — ID of the deleted object. • child — The deleted object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
add(data)	objectManager.ObjectCollection	This method completely duplicates the logic of ObjectManager.add.
each(callback, context)		A method that calls the passed handler function for all the items in the collection.
getAll()	Object[]	Returns array of objects contained in the collection.
getById(id)	Object|null	Returns the object, or null if an object with the passed ID does not exist.
getIterator()	IIterator	Returns iterator for the collection. Inherited from ICollection.
getLength()	Number	Returns the number of objects in the collection.
getObjectManager()	ObjectManager	Returns the parent layer of objects in the collection.
remove(data)	objectManager.ObjectCollection	This method completely duplicates the logic of ObjectManager.remove.
removeAll()	objectManager.ObjectCollection	Deletes all the items from a collection.
setObjectOptions(objectId, options)	objectManager.ObjectCollection	A method that allows to dynamically update object options. This method should be used when you want new options to be immediately applied to the object representation on the map. Note that the manager ignores the 'visible' option.

Fields details

balloon

```
{objectManager.Balloon} balloon
```

Object balloon in the manager.

hint

```
{objectManager.Hint} hint
```

Object hint in the ObjectManager.

options

```
{option.Manager} options
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Options manager. Names of fields that are available via the option.Manager#get method:

- hasBalloon — Indicates whether the collection has the .balloon field. If a balloon doesn't need to be opened when clicking the object, we recommend setting this option to the "false" value to avoid unnecessary initializations.
- hasHint — Indicates whether the collection has the .hint field. If a popup hint doesn't need to be displayed when the object is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations.
- hideIconOnBalloonOpen — Hide the icon when opening the balloon. Default value: true.
- openBalloonOnClick — Show the balloon when clicking the object. Default value: true.

Example:

```
// Creating styles for individual objects within the collection.
objectManager.objects.options.set({
    preset: 'islands#redIcon',
    hasBalloon: false,
    zIndex: 500
});
```

overlays

```
{objectManager.OverlayCollection} overlays
```

Collection of overlays of singular objects. All events, with the exception of "add" and "remove" events, propagate from the collection of overlays to the collection of objects.

Example:

```
objectManager.objects.overlays.events.add('add', function (e) {
    alert('The object ' + e.get('objectId') + ' is shown on the map.');
});
```

Events details

add

Adds an object to the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- objectId — ID of the added object.
- child — The added object.

objectoptionschange

Modification of object options via the **objectManager.ObjectCollection.setObjectOptions** method. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- objectId — ID of the object that had options modified.

remove

Removes an object from the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- objectId — ID of the deleted object.
- child — The deleted object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

add

```
{objectManager.ObjectCollection} add(data)
```

This method completely duplicates the logic of **ObjectManager.add**.

Returns self-reference.

Parameters:

Parameter	Default value	Description
data*	—	Type: Object|Object[]|String Objects to add to the layer.

* Mandatory parameter/option.

Example:

```
objectManager.objects.add({
    type: 'Feature',
    geometry: {
        type: 'Point',
        coordinates: [55.33, 36.64]
    }
});
```

each

```
{} each(callback, context)
```

A method that calls the passed handler function for all the items in the collection.

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Handler function that the collection objects are passed to.
context*	—	Type: Object Context for the handler function.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
var singleCounter = 0,
    clusterCounter = 0;
objectManager.objects.each(function (object) {
    var objectState = objectManager.getObjectState(object.id);
    if (objectState.isClustered) {
        clusterCounter++;
    } else {
        if (objectState.isShown) {
            singleCounter++;
        }
    }
});
alert('Number of single placemarks shown: ' + singleCounter);
alert('Number of placemarks shown in clusters: ' + clusterCounter);
```

getAll

```
{Object[]} getAll()
```

Returns array of objects contained in the collection.

getById

```
{Object|null} getById(id)
```

Returns the object, or null if an object with the passed ID does not exist.

Parameters:

Parameter	Default value	Description
id*	—	Type: Number Object ID.

* Mandatory parameter/option.

Example:

```
objectManager.objects.add('click', function (e) {
    var objectId = e.get('objectId'),
        object = objectManager.objects.getById(objectId);
});
```

getLength

```
{Number} getLength()
```

Returns the number of objects in the collection.

Example:

```
alert('Number of objects in the layer: ' + objectManager.objects.getLength());
```

getObjectManager

```
{ObjectManager} getObjectManager()
```

Returns the parent layer of objects in the collection.

remove

```
{objectManager.ObjectCollection} remove(data)
```

This method completely duplicates the logic of **ObjectManager.remove**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns self-reference.

Parameters:

Parameter	Default value	Description
data*	—	Type: Object|Object[]|String The objects to delete.

* Mandatory parameter/option.

Example:

```
// Removing the objects with the IDs 34 and 25.
objectManager.objects.remove([34, 25]);
```

removeAll

```
{objectManager.ObjectCollection} removeAll()
```

Deletes all the items from a collection.

Returns self-reference.

Example:

```
objectManager.objects.removeAll();
```

setObjectOptions

```
{objectManager.ObjectCollection} setObjectOptions(objectId, options)
```

A method that allows to dynamically update object options. This method should be used when you want new options to be immediately applied to the object representation on the map. Note that the manager ignores the 'visible' option.

Returns self-reference.

Parameters:

Parameter	Default value	Description
objectId*	—	Type: Object ID of the object to set options for.
options*	—	Type: Object New object options.

* Mandatory parameter/option.

Example:

```
// Changing the icon color when moused over.
objectManager.objects.events.add('mouseenter', function (e) {
    var objectId = e.get('objectId'),
        overlay = objectManager.objects.overlays.getById(objectId);
    setRedColor(objectId);
    overlay.events.add('mapchange', setGreenColor);
});

objectManager.objects.events.add('mouseleave', function (e) {
    var objectId = e.get('objectId'),
        overlay = objectManager.objects.overlays.getById(objectId);
    setGreenColor(objectId);
    overlay.events.remove('mapchange', setGreenColor);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
});

function setGreenColor (objectId) {
    objectManager.objects.setObjectOptions(objectId, {
        preset: 'islands#greenIcon'
    });
}

function setRedColor (objectId) {
    objectManager.objects.setClusterOptions(objectId, {
        preset: 'islands#redIcon'
    });
}
```

objectManager.OverlayCollection

Extends **ICustomizable**, **IEventEmitter**.

Collection of overlays.

Constructor | Fields | Events | Methods

Constructor

```
objectManager.OverlayCollection()
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
add	Adds an overlay to the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — The added overlay.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. Instance of the Event class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.
optionschange	Change to the object options. Inherited from **ICustomizable**.
remove	Removes an overlay from the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • objectId — The ID of the object the overlay belongs to. • overlay — The deleted overlay.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Instance of the Event class. Names of fields that are available via the Event.get method: • objectId — The ID of the object the overlay belongs to. • overlay — IOverlay — The overlay at which the event occurred. • coords — Geographical coordinates of the point at which the event occurred. • globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world. • pagePixels — Coordinates of the event in pixels from the upper-left corner of the page. • clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window. • domEvent — Source DOM event (as a DomEvent object), if there is one.

Methods

Name	Returns	Description
each(callback, context)		
getAll()	Object[]	Returns array of objects contained in the collection.
getById(id)	Object|null	Returns overlay, or null if an overlay with the passed ID does not exist.
getId(overlay)	Number|null	Returns the object ID, or null if the overlay is not contained in the collection.
getIterator()	IIterator	Returns iterator for the collection.
getLength()	Number	Returns the number of objects in the collection.

Events details

add

Adds an overlay to the collection. Instance of the Event class. Names of fields that are available via the Event.get method:

- objectId — The ID of the object the overlay belongs to.
- overlay — The added overlay.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

click

Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

contextmenu

Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

dblclick

Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

mousedown

Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

mouseenter

Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

mouseleave

Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

mousemove

Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

mouseup

Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

multitouchend

End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

multitouchmove

Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

multitouchstart

Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Names of fields that are available via the IMultiTouchEvent#get method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

remove

Removes an overlay from the collection. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — The deleted overlay.

wheel

Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- objectId — The ID of the object the overlay belongs to.
- overlay — IOverlay — The overlay at which the event occurred.
- coords — Geographical coordinates of the point at which the event occurred.
- globalPixels — Coordinates of the event in global pixels from the upper-left corner of the world.
- pagePixels — Coordinates of the event in pixels from the upper-left corner of the page.
- clientPixels — Coordinates of the event in pixels from the upper-left corner of the browser window.
- domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

each

```
{} each(callback, context)
```

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Callback function that the collection objects are passed to.
context*	—	Type: Object Context for the callback.

* Mandatory parameter/option.

Example:

```
objectManager.clusters.overlays.each(function (overlay) {
     overlay.options.set('cursor', 'help');
});
```

getAll

```
{Object[]} getAll()
```

Returns array of objects contained in the collection.

Example:

```
var clusterOverlayArray = objectManager.clusters.overlays.getAll();
```

getById

```
{Object|null} getById(id)
```

Returns overlay, or null if an overlay with the passed ID does not exist.

Parameters:

Parameter	Default value	Description
id*	—	Type: Number ID of the object the overlay belongs to.

* Mandatory parameter/option.

Example:

```
objectManager.objects.add('mouseenter', function (e) {
    var objectId = e.get('objectId'),
        overlay = objectManager.objects.overlays.getById(objectId);
    overlay.options.set('zIndex', 100);
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getId

```
{Number|null} getId(overlay)
```

Returns the object ID, or null if the overlay is not contained in the collection.

Parameters:

Parameter	Default value	Description
overlay*	—	Type: **IOverlay** Overlay.

* Mandatory parameter/option.

Example:

```
objectManager.objects.overlays.each(function (overlay) {
    var objectId = objectManager.objects.overlays.getId(overlay);
    objectManager.objects.setObjectOptions(objectId, {
        preset: 'islands#redIcon'
    });
});
```

getIterator

```
{IIterator} getIterator()
```

Returns iterator for the collection.

Example:

```
var it = objectManager.objects.overlays.getIterator(),
    overlay;
while ((overlay = it.getNext()) != it.STOP_ITERATION) {
    overlay.options.set('zIndex', 10);
}
```

getLength

```
{Number} getLength()
```

Returns the number of objects in the collection.

Example:

```
var objectsNumber = objectManager.objects.getLength(),
    overlaysNumber = objectManager.objects.overlays.getNumber();
alert('Now the map shows ' + overlaysNumber + ' of ' + objectsNumber.);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

option

option.Manager

Extends **IOptionManager**.

Options manager. For setting and getting option values by a string key, as well as allowing option values in the context of the existing hierarchy of options managers.

The special "preset" key is for making a set of options by default for this manager. The value of the "preset" option can be a hash with the format {"option name": "option value"}, or a string ID for a hash of options in the **option.presetStorage** storage. This hash of options can also contain a field named "preset", which allows for inheritance of option values from other sets of options.

When searching for values in the hierarchy, first the options themselves are checked, then the options set using the "preset" key, and after that the parent is accessed, if there is one.

To track changes to certain options, you can use **Monitor**.

Constructor | Fields | Events | Methods

Constructor

```
option.Manager([options[, parent[, name]]])
```

Creates an options manager.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Hash of options.
parent	—	Type: **IOptionManager** Parent options manager.
name	—	Type: String Name of the options manager.

Examples:

1.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Example of building a hierarchy of options managers.
var parentManager = new ymaps.option.Manager({
        key1: '123'
    }),
    childManager = new ymaps.option.Manager({
        key2: '234'
    }, parentManager);
// Outputs 123. The value is taken from manager1.
alert(childManager.get('key1'));
// Outputs 234. The value is taken from manager2.
alert(childManager.get('key2'));
// Overriding the option.
childManager.set('key1', '345');
// Outputs 345. The value is taken from manager2.
alert(childManager.get('key1'));
// Outputs 123. The value is taken from manager1.
alert(parentManager.get('key1'));
```

2.

```
// Example using the "preset" option.
var optionManager = new ymaps.option.Manager({
        preset: 'islands#blueIcon'
    }),
    subOptionManager = new ymaps.option.Manager();
// There is no data, because subOptionManager is empty.
alert(subOptionManager.get('iconImageSize');
// Binding two managers.
subOptionManager.setParent(optionManager);
// [37, 42] - value is taken from the preset in the parent manager.
alert(subOptionManager.get('iconImageSize');
// Overriding the value of iconImageSize on the level of subOptionManager.
subOptionManager.set('iconImageSize', [10, 12]);
// [10, 12] - value is taken from subOptionManager.
alert(subOptionManager.get('iconImageSize');
// Canceling override of iconImageSize.
subOptionManager.unset('iconImageSize');
// [37, 42] - value is again taken from the preset in the parent manager.
alert(subOptionManager.get('iconImageSize'));
```

Fields

Name	Type	Description
events	IEventManager	Event manager for the object. Inherited from IFreezable.

Events

Name	Description
change	Changes occurred either in the options values, or in the options inheritance hierarchy. Instance of the Event class.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
freeze()	**IFreezable**	Switches the object to "frozen" mode. Inherited from **IFreezable**.
get(key[, defaultValue])		Returns the value of the specified option in the context of the existing options inheritance hierarchy. When this method is called, first values are searched for in the current options manager, then, if the value is not defined, the search continues in the hierarchy of parent managers. Inherited from **IOptionManager**.
getAll()	Object	Returns a reference to the internal hash that stores option values. Inherited from **IOptionManager**.
getName()	String	Returns name of the options manager. Inherited from **IOptionManager**.
getNative(key)	Object	Returns the value of the specified option that is defined for the given level of the options hierarchy, i.e. in this manager. Inherited from **IOptionManager**.
getParent()	**IOptionManager**|null	Returns parent options manager. Inherited from **IOptionManager**.
isFrozen()	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from **IFreezable**.
resolve(key[, name])	Object	Method intended to be called by child options managers. Inherited from **IOptionManager**.
set(key[, value])	**option.Manager**	Sets option values for this manager. Two signatures are supported: • A single argument consisting of a hash in the format {"option name": "option value"}. • Two arguments, the first of which is the option name, and the second the value.
setName(name)		Sets the name of the options manager. Inherited from **IOptionManager**.
setParent(parent)	**IChild**	Sets the parent options manager. Inherited from **IOptionManager**.
unfreeze()	**IFreezable**	Switches the object to active mode. Inherited from **IFreezable**.
unset(keys)	**option.Manager**	Clears the values for the set options in this manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
unsetAll()	option.Manager	Clears the values for all options in this manager.

Events details

change

Changes occurred either in the options values, or in the options inheritance hierarchy. Instance of the **Event** class.

Methods details

set

```
{option.Manager} set(key[, value])
```

Sets option values for this manager. Two signatures are supported:

- A single argument consisting of a hash in the format {"option name": "option value"}.
- Two arguments, the first of which is the option name, and the second the value.

Returns self-reference.

Parameters:

Parameter	Default value	Description
key*	—	Type: Object|String The option name, or a hash in the format {"option name": "option value"}.
value	—	Type: Object The option value, if the name was passed as the first argument.

* Mandatory parameter/option.

Examples:

1.

```
// Setting multiple options via hash.
myMap.options.set({
    dblClickZoomCentering: true,
    dblClickFloatZoom: true
});
// Generates a single event option.Manager.event:change.
```

2.

```
// Setting options separately.
myMap.options
    .set("dblClickZoomCentering", true)
    .set("dblClickFloatZoom", true);
// Gernerates the event option.Manager.event:change.
```

3.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Using "freeze" to minimize the number of option.Manager.event:change events.
myMap.options.freeze();
myMap.options.set({
    dblClickZoomCentering: true,
    dblClickFloatZoom: true
});
myMap.options.set('cursor', 'zoom');
myMap.unfreeze();
// Generates a single option.Manager.event:change event.
```

unset

```
{option.Manager} unset(keys)
```

Clears the values for the set options in this manager.

Returns self-reference.

Parameters:

Parameter	Default value	Description
keys*	—	Type: String|String[] Option name or array of option names whose values should be canceled.

* Mandatory parameter/option.

Example:

```
map.options.unset(['dblClickZoomCentering', 'dblClickFloatZoom']);
```

unsetAll

```
{option.Manager} unsetAll()
```

Clears the values for all options in this manager.

Returns self-reference.

Example:

```
var geoObject = new ymaps.Placemark([37, 55], {}, {preset:'islands#blueIcon'});
myMap.geoObjects.add(geoObject);
// Changing the style however we want.
geoObject.options.set({
    iconLayout: 'default#image',
    iconImageHref: 'http://mysite.ru/icon.png',
    iconImageSize: [16, 16]
});
// Restoring the initial appearance.
geoObject.options
    // To avoid a double reaction of the geo object
    // to the option changes, first we call "freeze", then after
    // setting all values, we call "unfreeze".
    .freeze()
    .unsetAll()
    .set('preset','islands#blueIcon')
    .unfreeze();
```

option.presetStorage

Static object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Instance of **util.Storage**

Storage for preset options. List of available keys:

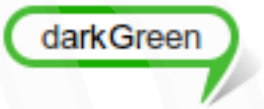
Placemarks with text

Placemark content is set via properties.iconContent.

Icon	Key	Icon	Key
	'islands#blueIcon'		'islands#darkGreenIcon'
	'islands#redIcon'		'islands#violetIcon'
	'islands#darkOrangeIcon'		'islands#blackIcon'
	'islands#nightIcon'		'islands#yellowIcon'
	'islands#darkBlueIcon'		'islands#greenIcon'
	'islands#pinkIcon'		'islands#orangeIcon'
	'islands#grayIcon'		'islands#lightBlueIcon'
	'islands#brownIcon'		'islands#oliveIcon'

Placemarks with text (icons stretch to fit the content)

Placemark content is set via properties.iconContent.

Icon	Key	Icon	Key
	'islands#blueStretchyIcon'		'islands#darkGreenStretchyIcon'
	'islands#redStretchyIcon'		'islands#violetStretchyIcon'

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Icon	Key	Icon	Key
	'islands#darkOrangeStretchyIcon'		'islands#blackStretchyIcon'
	'islands#nightStretchyIcon'		'islands#yellowStretchyIcon'
	'islands#darkBlueStretchyIcon'		'islands#greenStretchyIcon'
	'islands#pinkStretchyIcon'		'islands#orangeStretchyIcon'
	'islands#grayStretchyIcon'		'islands#lightBlueStretchyIcon'
	'islands#brownStretchyIcon'		'islands#oliveStretchyIcon'

Placemarks without content with a dot in the center

Icon	Key	Icon	Key
	'islands#blueDotIcon'		'islands#darkGreenDotIcon'
	'islands#redDotIcon'		'islands#violetDotIcon'
	'islands#darkOrangeDotIcon'		'islands#blackDotIcon'
	'islands#nightDotIcon'		'islands#yellowDotIcon'
	'islands#darkBlueDotIcon'		'islands#greenDotIcon'
	'islands#pinkDotIcon'		'islands#orangeDotIcon'
	'islands#grayDotIcon'		'islands#lightBlueDotIcon'

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Icon	Key	Icon	Key
	'islands#brownDotIcon'		'islands#oliveDotIcon'

Placemarks in the style of a circle with text

Placemark content is set via properties.iconContent.

Icon	Key	Icon	Key
	'islands#blueCircleIcon'		'islands#darkGreenCircleIcon'
	'islands#redCircleIcon'		'islands#violetCircleIcon'
	'islands#darkOrangeCircleIcon'		'islands#blackCircleIcon'
	'islands#nightCircleIcon'		'islands#yellowCircleIcon'
	'islands#darkBlueCircleIcon'		'islands#greenCircleIcon'
	'islands#pinkCircleIcon'		'islands#orangeCircleIcon'
	'islands#grayCircleIcon'		'islands#lightBlueCircleIcon'
	'islands#brownCircleIcon'		'islands#oliveCircleIcon'

Placemarks in the style of circles with a dot in the center

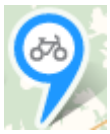
Icon	Key	Icon	Key
	'islands#blueCircleDotIcon'		'islands#darkGreenCircleDotIcon'
	'islands#redCircleDotIcon'		'islands#violetCircleDotIcon'
	'islands#darkOrangeCircleDotIcon'		'islands#blackCircleDotIcon'
	'islands#nightCircleDotIcon'		'islands#yellowCircleDotIcon'
	'islands#darkBlueCircleDotIcon'		'islands#greenCircleDotIcon'
	'islands#pinkCircleDotIcon'		'islands#orangeCircleDotIcon'
	'islands#grayCircleDotIcon'		'islands#lightBlueCircleDotIcon'

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

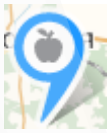
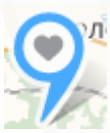
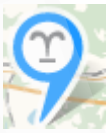
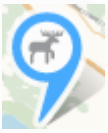
Icon	Key	Icon	Key
	'islands#brownCircleDotIcon'		'islands#oliveCircleDotIcon'

Placemarks with an icon

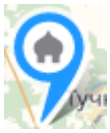
For this type of placemark, the key uses the format: 'islands#{color}{icon}Icon'. For example, 'islands#blueHomeIcon'. The list below shows available icons and corresponding keys. For the list of available colors, see any other table.

Icon	Key	Icon	Key
	'islands#blueAirportIcon'		'islands#blueAttentionIcon'
	'islands#blueAutoIcon'		'islands#blueBarIcon'
	'islands#blueBarberIcon'		'islands#blueBeachIcon'
	'islands#blueBicycleIcon'		'islands#blueBicycle2Icon'
	'islands#blueBookIcon'		'islands#blueCarWashIcon'
	'islands#blueChristianIcon'		'islands#blueCinemaIcon'
	'islands#blueCircusIcon'		'islands#blueCourtIcon'
	'islands#blueDeliveryIcon'		'islands#blueDiscountIcon'
	'islands#blueDogIcon'		'islands#blueEducationIcon'
	'islands#blueEntertainmentCenterIcon'		'islands#blueFactoryIcon'

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Icon	Key	Icon	Key
	'islands#blueFamilyIcon'		'islands#blueFashionIcon'
	'islands#blueFoodIcon'		'islands#blueFuelStationIcon'
	'islands#blueGardenIcon'		'islands#blueGovernmentIcon'
	'islands#blueHeartIcon'		'islands#blueHomeIcon'
	'islands#blueHotelIcon'		'islands#blueHydroIcon'
	'islands#blueInfoIcon'		'islands#blueLaundryIcon'
	'islands#blueLeisureIcon'		'islands#blueMassTransitIcon'
	'islands#blueMedicalIcon'		'islands#blueMoneyIcon'
	'islands#blueMountainIcon'		'islands#blueNightClubIcon'
	'islands#blueObservationIcon'		'islands#blueParkIcon'
	'islands#blueParkingIcon'		'islands#bluePersonIcon'
	'islands#bluePocketIcon'		'islands#bluePoolIcon'

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Icon	Key	Icon	Key
	'islands#bluePostIcon'		'islands#blueRailwayIcon'
	'islands#blueRapidTransitIcon'		'islands#blueRepairShopIcon'
	'islands#blueRunIcon'		'islands#blueScienceIcon'
	'islands#blueShoppingIcon'		'islands#blueSouvenirsIcon'
	'islands#blueSportIcon'		'islands#blueStarIcon'
	'islands#blueTheaterIcon'		'islands#blueToiletIcon'
	'islands#blueUnderpassIcon'		'islands#blueVegetationIcon'
	'islands#blueVideoIcon'		'islands#blueWasteIcon'
	'islands#blueWaterParkIcon'		'islands#blueWaterwayIcon'
	'islands#blueWorshipIcon'		'islands#blueZooIcon'

Placemarks in the style of a circle with an icon

For this type of placemark, the key uses the format: 'islands#{color}{icon}CircleIcon'. For example, 'islands#blueHomeCircleIcon'. The list below shows available icons and corresponding keys. For the list of available colors, see any other table.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Icon	Key	Icon	Key
	'islands#blueHomeCircleIcon'		'islands#blueScienceCircleIcon'
	'islands#blueAirportCircleIcon'		'islands#blueAttentionCircleIcon'
	'islands#blueAutoCircleIcon'		'islands#blueBarCircleIcon'
	'islands#blueBarberCircleIcon'		'islands#blueBeachCircleIcon'
	'islands#blueBicycleCircleIcon'		'islands#blueBicycle2CircleIcon'
	'islands#blueBookCircleIcon'		'islands#blueCarWashCircleIcon'
	'islands#blueChristianCircleIcon'		'islands#blueCinemaCircleIcon'
	'islands#blueCircusCircleIcon'		'islands#blueCourtCircleIcon'
	'islands#blueDeliveryCircleIcon'		'islands#blueDiscountCircleIcon'
	'islands#blueDogCircleIcon'		'islands#blueEducationCircleIcon'
	'islands#blueEntertainmentCenterCircleIcon'		'islands#blueFactoryCircleIcon'
	'islands#blueFamilyCircleIcon'		'islands#blueFashionCircleIcon'
	'islands#blueFoodCircleIcon'		'islands#blueFuelStationCircleIcon'
	'islands#blueGardenCircleIcon'		'islands#blueGovernmentCircleIcon'
	'islands#blueHeartCircleIcon'		'islands#blueHomeCircleIcon'
	'islands#blueHotelCircleIcon'		'islands#blueHydroCircleIcon'
	'islands#blueInfoCircleIcon'		'islands#blueLaundryCircleIcon'

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Icon	Key	Icon	Key
	'islands#blueLeisureCircleIcon'		'islands#blueMassTransitCircleIcon'
	'islands#blueMedicalCircleIcon'		'islands#blueMoneyCircleIcon'
	'islands#blueMountainCircleIcon'		'islands#blueNightClubCircleIcon'
	'islands#blueObservationCircleIcon'		'islands#blueParkCircleIcon'
	'islands#blueParkingCircleIcon'		'islands#bluePersonCircleIcon'
	'islands#bluePocketCircleIcon'		'islands#bluePoolCircleIcon'
	'islands#bluePostCircleIcon'		'islands#blueRailwayCircleIcon'
	'islands#blueRapidTransitCircleIcon'		'islands#blueRepairShopCircleIcon'
	'islands#blueRunCircleIcon'		'islands#blueScienceCircleIcon'
	'islands#blueShoppingCircleIcon'		'islands#blueSouvenirsCircleIcon'
	'islands#blueSportCircleIcon'		'islands#blueStarCircleIcon'
	'islands#blueTheaterCircleIcon'		'islands#blueToiletCircleIcon'
	'islands#blueUnderpassCircleIcon'		'islands#blueVegetationCircleIcon'
	'islands#blueVideoCircleIcon'		'islands#blueWasteCircleIcon'
	'islands#blueWaterParkCircleIcon'		'islands#blueWaterwayCircleIcon'
	'islands#blueWorshipCircleIcon'		'islands#blueZooCircleIcon'

Pictograms

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Icon	Key
	'islands#geolocationIcon'

Cluster icons

Icon	Key	Icon	Key
	'islands#blueClusterIcons'		'islands#invertedBlueClusterIcons'
	'islands#redClusterIcons'		'islands#invertedRedClusterIcons'
	'islands#darkOrangeClusterIcons'		'islands#invertedDarkOrangeClusterIcons'
	'islands#nightClusterIcons'		'islands#invertedNightClusterIcons'
	'islands#darkBlueClusterIcons'		'islands#invertedDarkBlueClusterIcons'
	'islands#pinkClusterIcons'		'islands#invertedPinkClusterIcons'
	'islands#grayClusterIcons'		'islands#invertedGrayClusterIcons'
	'islands#brownClusterIcons'		'islands#invertedBrownClusterIcons'

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Icon	Key	Icon	Key
	'islands#darkGreenClusterIcons'		'islands#invertedDarkGreenClusterIcons'
	'islands#violetClusterIcons'		'islands#invertedVioletClusterIcons'
	'islands#blackClusterIcons'		'islands#invertedBlackClusterIcons'
	'islands#yellowClusterIcons'		'islands#invertedYellowClusterIcons'
	'islands#greenClusterIcons'		'islands#invertedGreenClusterIcons'
	'islands#orangeClusterIcons'		'islands#invertedOrangeClusterIcons'
	'islands#lightBlueClusterIcons'		'islands#invertedLightBlueClusterIcons'
	'islands#oliveClusterIcons'		'islands#invertedOliveClusterIcons'

Methods

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

overlay

overlay.Circle

Extends **IOverlay**.

Interactive overlay for a circle. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.Circle(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelCircleGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.fill	—	Type: Boolean Whether there is fill **graphics.style.color**.
options.fillColor	—	Type: String Fill color.
options.fillImageHref	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image repeats without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	—	Type: Number Fill transparency.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.outline	—	Type: Boolean Whether there is an outline.
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style (not supported in Canvas mode) **graphics.style.stroke**.
options.strokeWidth	—	Type: Number Line width.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class. Inherited from **IOverlay**.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from **IOverlay**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IOverlay**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from **IOverlay**.
getGeometry()	**IPixelGeometry**	Returns the current pixel geometry. Inherited from **IOverlay**.
getMap()	**Map**|null	Returns reference to the current map. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getShape()	IShape|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from IOverlay.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from IOverlay.
setData(data)		Sets the overlay data. Inherited from IOverlay.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from IOverlay.
setMap(map)		Sets the map on which to display the overlay. Inherited from IOverlay.

overlay.hotspot

overlay.hotspot.Circle

Extends **IOverlay**.

Round hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.hotspot.Circle(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelCircleGeometry** Geometry.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.fill	—	Type: Boolean Whether there is fill **graphics.style.color**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.fillColor	—	Type: String Fill color.
options.fillImageHref	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image repeats without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	—	Type: Number Fill transparency.
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.outline	—	Type: Boolean Whether there is an outline.
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style (not supported in Canvas mode) **graphics.style.stroke**.
options.strokeWidth	—	Type: Number Line width.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from IOverlay.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty overlay flag. Instance of the Event class. Inherited from IOverlay.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from IOverlay.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IOverlay.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from **IOverlay**.
getGeometry()	**IPixelGeometry**	Returns the current pixel geometry. Inherited from **IOverlay**.
getMap()	**Map**|null	Returns reference to the current map. Inherited from **IOverlay**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from **IOverlay**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **IOverlay**.
setData(data)		Sets the overlay data. Inherited from **IOverlay**.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from **IOverlay**.
setMap(map)		Sets the map on which to display the overlay. Inherited from **IOverlay**.

overlay.hotspot.Placemark

Extends **IOverlay**.

Point hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.hotspot.Placemark(geometry[, data[, options]])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelPointGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.cursor	—	Type: String Cursor when the mouse is hovering.
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.interactivityModel	'default#geoObject'	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.layout	—	Type: Function|String Layout. (Type: constructor for an object with the ILayout interface).
options.offset	[0,0]	Type: Array Offset in pixels.
options.pane	'places'	Type: String Container where the placemark layout will be placed.
options.shadow	false	Type: Boolean Flag for whether there is a shadow.
options.shadowLayout	—	Type: Function|String Layout for the shadow. (Type: constructor for an object with the ILayout interface).
options.shadowOffset	[0,0]	Type: Array Shadow offset in pixels.
options.shadowsPane	'shadows'	Type: Array Container where the placemark shadow layout will be placed.
options.zIndex	—	Type: Number

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The z-index of the element.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from IOverlay.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty overlay flag. Instance of the Event class. Inherited from IOverlay.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from IOverlay.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IOverlay.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
shapechange	Change to the shape of the area spanning the overlay. Instance of the Event class. Inherited from IOverlay.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from IOverlay.
getGeometry()	IPixelGeometry	Returns the current pixel geometry. Inherited from IOverlay.
getMap()	Map|null	Returns reference to the current map. Inherited from IOverlay.
getShape()	IShape|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from IOverlay.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from IOverlay.
setData(data)		Sets the overlay data. Inherited from IOverlay.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from IOverlay.
setMap(map)		Sets the map on which to display the overlay. Inherited from IOverlay.

overlay.hotspot.Polygon

Extends IOverlay.

Polygon hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use overlay.storage.

Constructor | Fields | Events | Methods

Constructor

```
overlay.hotspot.Polygon(geometry[, data[, options]])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelPolygonGeometry** The geometry of the shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.fill	—	Type: String Whether there is fill.
options.fillColor	—	Type: String Fill color **graphics.style.color**.
options.fillImageHref	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	—	Type: Number Fill transparency.
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.outline	—	Type: String Whether there is an outline.
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style **graphics.style.stroke**.
options.strokeWidth	—	Type: Number Line width.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from IOverlay.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IOverlay.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from IDomEventEmitter.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from **IOverlay**.
getGeometry()	**IPixelGeometry**	Returns the current pixel geometry. Inherited from **IOverlay**.
getMap()	**Map**|null	Returns reference to the current map. Inherited from **IOverlay**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from **IOverlay**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **IOverlay**.
setData(data)		Sets the overlay data. Inherited from **IOverlay**.
setGeometry(geometry)		Sets the overlay pixel geometry.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IOverlay**.
setMap(map)		Sets the map on which to display the overlay. Inherited from **IOverlay**.

overlay.hotspot.Polyline

Extends **IOverlay**.

Polyline hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.hotspot.Polyline(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelLineStringGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style **graphics.style.stroke**.
options.strokeWidth	—	Type: Number

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Line width.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from IOverlay.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty overlay flag. Instance of the Event class. Inherited from IOverlay.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from IOverlay.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IOverlay.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
shapechange	Change to the shape of the area spanning the overlay. Instance of the Event class. Inherited from IOverlay.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from IOverlay.
getGeometry()	IPixelGeometry	Returns the current pixel geometry. Inherited from IOverlay.
getMap()	Map|null	Returns reference to the current map. Inherited from IOverlay.
getShape()	IShape|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from IOverlay.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from IOverlay.
setData(data)		Sets the overlay data. Inherited from IOverlay.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from IOverlay.
setMap(map)		Sets the map on which to display the overlay. Inherited from IOverlay.

overlay.hotspot.Rectangle

Extends IOverlay.

Square hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use overlay.storage.

Constructor | Fields | Events | Methods

Constructor

```
overlay.hotspot.Rectangle(geometry[, data[, options]])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelRectangleGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.borderRadius	—	Type: Number Radius of rounded corners.
options.fill	—	Type: String Whether there is fill.
options.fillColor	—	Type: String Fill color **graphics.style.color**.
options.fillImageHref	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	—	Type: Number Fill transparency.
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.outline	—	Type: String Whether there is an outline.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style (not supported in Canvas mode) **graphics.style.stroke**.
options.strokeWidth	—	Type: Number Line width.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IOverlay**.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from **IOverlay**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IOverlay**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from **IOverlay**.
getGeometry()	**IPixelGeometry**	Returns the current pixel geometry. Inherited from **IOverlay**.
getMap()	**Map**|null	Returns reference to the current map. Inherited from **IOverlay**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from **IOverlay**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **IOverlay**.
setData(data)		Sets the overlay data. Inherited from **IOverlay**.
setGeometry(geometry)		Sets the overlay pixel geometry.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IOverlay**.
setMap(map)		Sets the map on which to display the overlay. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

overlay.html

overlay.html.Balloon

Extends **IOverlay**.

HTML overlay for the balloon. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.html.Balloon(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelPointGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.cursor	—	Type: String Cursor when the mouse is hovering.
options.interactivityModel	"default#opaque"	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.layout	—	Type: Function|String Layout. (Type: constructor for an object with the ILayout interface).
options.offset	[0,0]	Type: Array Offset in pixels.
options.pane	"balloon"	Type: String Container where the balloon layout will be placed.
options.shadow	true	Type: Boolean Flag for whether there is a shadow.
options.shadowLayout	—	Type: Function|String Shadow layout (type: constructor for an object with the ILayout interface).

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.shadowOffset	[0,0]	Type: Array Shadow offset in pixels.
options.shadowsPane	"shadows"	Type: Array Container where the balloon shadow layout will be placed.
options.zIndex	—	Type: Number The z-index of the element.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class. Inherited from **IOverlay**.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IOverlay**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
shapechange	Change to the shape of the area spanning the overlay. Instance of the Event class. Inherited from IOverlay.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getBalloonElement()	HTMLElement	Returns parent element of the balloon layout.
getBalloonLayout()	vow.Promise	Returns Promise object to return the balloon layout.
getBalloonLayoutSync()	ILayout	Returns balloon layout.
getData()	Object	Returns the overlay data object. Inherited from IOverlay.
getElement()	HTMLElement	Returns parent element of the balloon layout.
getGeometry()	IPixelGeometry	Returns the current pixel geometry. Inherited from IOverlay.
getLayout()	vow.Promise	Returns Promise object to return the balloon layout.
getLayoutSync()	ILayout|null	Returns balloon layout.
getMap()	Map|null	Returns reference to the current map. Inherited from IOverlay.
getMode()	String	Returns the current mode of the balloon: "panel" — panel mode, "standard" — standard display.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getShadowElement()	HTMLElement	Returns parent element of the balloon shadow layout.
getShadowLayout()	vow.Promise	Returns Promise object to return the balloon shadow layout.
getShadowLayoutSync()	ILayout|null	Returns balloon shadow layout.
getShape()	IShape|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from IOverlay.
isEmpty()	Boolean	Returns true if the layout is empty or if the layout has not yet been loaded, i.e. it has no content.
setData(data)		Sets the overlay data. Inherited from IOverlay.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from IOverlay.
setMap(map)		Sets the map on which to display the overlay. Inherited from IOverlay.

Methods details

getBalloonElement

```
{HTMLElement} getBalloonElement()
```

Returns parent element of the balloon layout.

getBalloonLayout

```
{vow.Promise} getBalloonLayout()
```

Returns Promise object to return the balloon layout.

getBalloonLayoutSync

```
{ILayout} getBalloonLayoutSync()
```

Returns balloon layout.

getElement

```
{HTMLElement} getElement()
```

Returns parent element of the balloon layout.

getLayout

```
{vow.Promise} getLayout()
```

Returns Promise object to return the balloon layout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getLayoutSync

```
{ILayout|null} getLayoutSync()
```

Returns balloon layout.

getMode

```
{String} getMode()
```

Returns the current mode of the balloon: "panel" — panel mode, "standard" — standard display.

getShadowElement

```
{HTMLElement} getShadowElement()
```

Returns parent element of the balloon shadow layout.

getShadowLayout

```
{vow.Promise} getShadowLayout()
```

Returns Promise object to return the balloon shadow layout.

getShadowLayoutSync

```
{ILayout|null} getShadowLayoutSync()
```

Returns balloon shadow layout.

isEmpty

```
{Boolean} isEmpty()
```

Returns true if the layout is empty or if the layout has not yet been loaded, i.e. it has no content.

overlay.html.Hint

Extends **IOverlay**.

Simple HTML overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.html.Hint(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelPointGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options	—	Type: Object Options.
options.cursor	—	Type: String Cursor when the mouse is hovering.
options.interactivityModel	'default#opaque'	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.layout	—	Type: **ILayout**|String Layout.
options.pane	"outerHint"	Type: String Container where the overlay will be placed.
options.zIndex	—	Type: Number The z-index of the element.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class. Inherited from **IOverlay**.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from **IOverlay**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IOverlay**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
shapechange	Change to the shape of the area spanning the overlay. Instance of the Event class. Inherited from IOverlay.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from IOverlay.
getElement()	HTMLElement	Returns parent element of the icon layout.
getGeometry()	IPixelGeometry	Returns the current pixel geometry. Inherited from IOverlay.
getLayout()	vow.Promise	Returns Promise object to return the icon layout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getLayoutSync()	ILayout|null	Returns icon layout.
getMap()	Map|null	Returns reference to the current map. Inherited from IOverlay.
getShape()	IShape|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from IOverlay.
isEmpty()	Boolean	Returns true if the layout is empty or if the layout has not yet been loaded, i.e. it has no content.
setData(data)		Sets the overlay data. Inherited from IOverlay.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from IOverlay.
setMap(map)		Sets the map on which to display the overlay. Inherited from IOverlay.

Methods details

getElement

```
{HTMLElement} getElement()
```

Returns parent element of the icon layout.

getLayout

```
{vow.Promise} getLayout()
```

Returns Promise object to return the icon layout.

getLayoutSync

```
{ILayout|null} getLayoutSync()
```

Returns icon layout.

isEmpty

```
{Boolean} isEmpty()
```

Returns true if the layout is empty or if the layout has not yet been loaded, i.e. it has no content.

overlay.html.Placemark

Extends **IOverlay**.

HTML overlay for the layout. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
overlay.html.Placemark(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelPointGeometry** Pixel geometry.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.cursor	—	Type: String Cursor when the mouse is hovering.
options.interactivityModel	'default#geoObject'	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.layout	—	Type: Function|String Layout. (Type: constructor for an object with the ILayout interface).
options.offset	[0,0]	Type: Array Offset in pixels.
options.pane	'places'	Type: String Container where the placemark layout will be placed.
options.shadow	false	Type: Boolean Flag for whether there is a shadow.
options.shadowLayout	—	Type: Function|String Layout for the shadow. (Type: constructor for an object with the ILayout interface).
options.shadowOffset	[0,0]	Type: Array Shadow offset in pixels.
options.shadowsPane	'shadows'	Type: Array Container where the placemark shadow layout will be placed.
options.zIndex	—	Type: Number

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The z-index of the element.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from IOverlay.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty overlay flag. Instance of the Event class. Inherited from IOverlay.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from IOverlay.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IOverlay.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
shapechange	Change to the shape of the area spanning the overlay. Instance of the Event class. Inherited from IOverlay.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from IOverlay.
getElement()	HTMLElement	Returns parent element of the icon layout.
getGeometry()	IPixelGeometry	Returns the current pixel geometry. Inherited from IOverlay.
getIconElement()	HTMLElement	Returns parent element of the icon layout.
getIconLayout()	vow.Promise	Returns Promise object to return the icon layout.
getIconLayoutSync()	ILayout|null	Returns icon layout.
getLayout()	vow.Promise	Returns Promise object to return the icon layout.
getLayoutSync()	ILayout|null	Returns icon layout.
getMap()	Map|null	Returns reference to the current map. Inherited from IOverlay.
getShadowElement()	HTMLElement	Returns parent element of the icon shadow layout.
getShadowLayout()	vow.Promise	Returns Promise object to return the icon shadow layout.
getShadowLayoutSync()	ILayout|null	Returns layout for the icon shadow.
getShape()	IShape|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from IOverlay.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from IOverlay.
setData(data)		Sets the overlay data. Inherited from IOverlay.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from IOverlay.
setMap(map)		Sets the map on which to display the overlay. Inherited from IOverlay.

Methods details

getElement

```
{HTMLElement} getElement()
```

Returns parent element of the icon layout.

getIconElement

```
{HTMLElement} getIconElement()
```

Returns parent element of the icon layout.

getIconLayout

```
{vow.Promise} getIconLayout()
```

Returns Promise object to return the icon layout.

getIconLayoutSync

```
{ILayout|null} getIconLayoutSync()
```

Returns icon layout.

getLayout

```
{vow.Promise} getLayout()
```

Returns Promise object to return the icon layout.

getLayoutSync

```
{ILayout|null} getLayoutSync()
```

Returns icon layout.

getShadowElement

```
{HTMLElement} getShadowElement()
```

Returns parent element of the icon shadow layout.

getShadowLayout

```
{vow.Promise} getShadowLayout()
```

Returns Promise object to return the icon shadow layout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getShadowLayoutSync

```
{ILayout|null} getShadowLayoutSync()
```

Returns layout for the icon shadow.

overlay.html.Rectangle

Extends **IOverlay**.

HTML overlay for a rectangle. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.html.Rectangle(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelRectangleGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Overlay options.
options.cursor	—	Type: String Cursor when the mouse is hovering.
options.fillColor	—	Type: String Fill color **graphics.style.color**. An option of the standard rectangle layout.
options.fillImageHref	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored. An option of the standard rectangle layout.
options.fillMethod	'stretch'	Type: String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template. An option of the standard rectangle layout.
options.fillOpacity	—	Type: Number Fill transparency. An option of the standard rectangle layout.
options.interactivityModel	'default#geoObject'	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.opacity	—	Type: Number Overall transparency.
options.pane	"areas"	Type: String Container where the overlay will be placed.
options.strokeColor	—	Type: String Line color **graphics.style.color**. An option of the standard rectangle layout.
options.strokeStyle	—	Type: Number[]|String The outline style supported by the standard CSS *border-style*. An option of the standard rectangle layout.
options.strokeWidth	—	Type: Number Line width. An option of the standard rectangle layout.
options.zIndex	—	Type: Number The z-index of the element.
dataSet.options.borderRadius	—	Type: Number Radius of rounded corners. An option of the standard rectangle layout.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from IOverlay.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty overlay flag. Instance of the Event class. Inherited from IOverlay.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from IOverlay.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IOverlay.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from IOverlay.
getElement()	HTMLElement	Returns parent element of the rectangle layout.
getGeometry()	IPixelGeometry	Returns the current pixel geometry. Inherited from IOverlay.
getLayout()	vow.Promise	Returns a promise object to return the rectangle layout.
getLayoutSync()	ILayout|null	Returns rectangle layout.
getMap()	Map|null	Returns reference to the current map. Inherited from IOverlay.
getShape()	IShape|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from IOverlay.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from IOverlay.
setData(data)		Sets the overlay data. Inherited from IOverlay.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from IOverlay.
setMap(map)		Sets the map on which to display the overlay. Inherited from IOverlay.

Methods details

getElement

```
{HTMLElement} getElement()
```

Returns parent element of the rectangle layout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getLayout

```
{vow.Promise} getLayout()
```

Returns a promise object to return the rectangle layout.

getLayoutSync

```
{ILayout|null} getLayoutSync()
```

Returns rectangle layout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

overlay.Pin

Extends **IOverlay**.

Interactive overlay for a circle placemark. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.Pin(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelPointGeometry** Pixel geometry of a placemark.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.fill	—	Type: Boolean Whether there is fill **graphics.style.color**.
options.fillColor	—	Type: String Fill color.
options.fillImageHref	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image repeats without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	—	Type: Number Fill transparency.
options.interactive	true	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.outline	—	Type: Boolean Whether there is an outline.
options.radius	—	Type: Number Radius of the placemark in pixels.
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style (not supported in Canvas mode) **graphics.style.stroke**.
options.strokeWidth	—	Type: Number Line width.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class. Inherited from **IOverlay**.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from **IOverlay**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IOverlay**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
shapechange	Change to the shape of the area spanning the overlay. Instance of the Event class. Inherited from IOverlay.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from IOverlay.
getGeometry()	IPixelGeometry	Returns the current pixel geometry. Inherited from IOverlay.
getMap()	Map|null	Returns reference to the current map. Inherited from IOverlay.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getShape()	**IShape**|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from **IOverlay**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **IOverlay**.
setData(data)		Sets the overlay data. Inherited from **IOverlay**.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from **IOverlay**.
setMap(map)		Sets the map on which to display the overlay. Inherited from **IOverlay**.

overlay.Placemark

Extends **IOverlay**.

Placemark overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.Placemark(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
`geometry` *	—	Type: **IPixelPointGeometry** Pixel geometry.
`data`	—	Type: Object Data.
`options`	—	Type: Object Options.
`options.cursor`	—	Type: String Cursor when the mouse is hovering.
`options.interactive`	true	Type: Boolean Disables the object's reaction to DOM events.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.interactivityModel	'default#geoObject'	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.layout	—	Type: Function|String Layout. (Type: constructor for an object with the ILayout interface).
options.offset	[0,0]	Type: Array Offset in pixels.
options.pane	'places'	Type: String Container where the placemark layout will be placed.
options.shadow	false	Type: Boolean Flag for whether there is a shadow.
options.shadowLayout	—	Type: Function|String Layout for the shadow. (Type: constructor for an object with the ILayout interface).
options.shadowOffset	[0,0]	Type: Array Shadow offset in pixels.
options.shadowsPane	'shadows'	Type: Array Container where the placemark shadow layout will be placed.
options.zIndex	—	Type: Number The z-index of the element.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class. Inherited from **IOverlay**.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from **IOverlay**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IOverlay**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getElement()	HTMLElement	Returns parent element of the icon layout.
getGeometry()	**IPixelGeometry**	Returns the current pixel geometry. Inherited from **IOverlay**.
getIconElement()	HTMLElement	Returns parent element of the icon layout.
getIconLayout()	**vow.Promise**	Returns Promise object to return the icon layout.
getIconLayoutSync()	**ILayout**|null	Returns icon layout.
getLayout()	**vow.Promise**	Returns Promise object to return the icon layout.
getLayoutSync()	**ILayout**|null	Returns icon layout.
getMap()	**Map**|null	Returns reference to the current map. Inherited from **IOverlay**.
getShadowElement()	HTMLElement	Returns parent element of the icon shadow layout.
getShadowLayout()	**vow.Promise**	Returns Promise object to return the icon shadow layout.
getShadowLayoutSync()	**ILayout**|null	Returns layout for the icon shadow.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from **IOverlay**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **IOverlay**.
setData(data)		Sets the overlay data. Inherited from **IOverlay**.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from **IOverlay**.
setMap(map)		Sets the map on which to display the overlay. Inherited from **IOverlay**.

Methods details

getElement

```
{HTMLElement} getElement()
```

Returns parent element of the icon layout.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getIconElement

```
{HTMLElement} getIconElement()
```

Returns parent element of the icon layout.

getIconLayout

```
{vow.Promise} getIconLayout()
```

Returns Promise object to return the icon layout.

getIconLayoutSync

```
{ILayout|null} getIconLayoutSync()
```

Returns icon layout.

getLayout

```
{vow.Promise} getLayout()
```

Returns Promise object to return the icon layout.

getLayoutSync

```
{ILayout|null} getLayoutSync()
```

Returns icon layout.

getShadowElement

```
{HTMLElement} getShadowElement()
```

Returns parent element of the icon shadow layout.

getShadowLayout

```
{vow.Promise} getShadowLayout()
```

Returns Promise object to return the icon shadow layout.

getShadowLayoutSync

```
{ILayout|null} getShadowLayoutSync()
```

Returns layout for the icon shadow.

overlay.Polygon

Extends **IOverlay**.

Polygon overlay.

Constructor | Fields | Events | Methods

Constructor

```
overlay.Polygon(geometry[, data[, options]])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelPolygonGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.fill	—	Type: String Whether there is fill.
options.fillColor	—	Type: String Fill color **graphics.style.color**.
options.fillImageHref	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	—	Type: Number Fill transparency.
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.outline	—	Type: String Whether there is an outline.
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style **graphics.style.stroke**.
options.strokeWidth	—	Type: Number Line width.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class. Inherited from **IOverlay**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from **IOverlay**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IOverlay**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from **IOverlay**.
getGeometry()	**IPixelGeometry**	Returns the current pixel geometry. Inherited from **IOverlay**.
getMap()	**Map**|null	Returns reference to the current map. Inherited from **IOverlay**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from **IOverlay**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **IOverlay**.
setData(data)		Sets the overlay data. Inherited from **IOverlay**.
setGeometry(geometry)		Sets the overlay pixel geometry.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IOverlay**.
setMap(map)		Sets the map on which to display the overlay. Inherited from **IOverlay**.

overlay.Polyline

Extends **IOverlay**.

Line overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.Polyline(geometry[, data[, options]])
```

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelLineStringGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style **graphics.style.stroke**.
options.strokeWidth	—	Type: Number

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Line width.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IDomEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from IOverlay.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
emptinesschange	Change to the empty overlay flag. Instance of the Event class. Inherited from IOverlay.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from IOverlay.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from IOverlay.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from **IOverlay**.
getGeometry()	**IPixelGeometry**	Returns the current pixel geometry. Inherited from **IOverlay**.
getMap()	**Map**|null	Returns reference to the current map. Inherited from **IOverlay**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from **IOverlay**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **IOverlay**.
setData(data)		Sets the overlay data. Inherited from **IOverlay**.
setGeometry(geometry)		Sets the overlay pixel geometry. Inherited from **IOverlay**.
setMap(map)		Sets the map on which to display the overlay. Inherited from **IOverlay**.

overlay.Rectangle

Extends **IOverlay**.

Polygon overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use **overlay.storage**.

Constructor | Fields | Events | Methods

Constructor

```
overlay.Rectangle(geometry[, data[, options]])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
geometry *	—	Type: **IPixelRectangleGeometry** Pixel geometry of a shape.
data	—	Type: Object Data.
options	—	Type: Object Options.
options.borderRadius	—	Type: Number Radius of rounded corners.
options.fill	—	Type: String Whether there is fill.
options.fillColor	—	Type: String Fill color **graphics.style.color**.
options.fillImageHref	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	—	Type: Number Fill transparency.
options.interactive	true	Type: Boolean Disables the object's reaction to DOM events.
options.opacity	—	Type: Number Overall transparency.
options.outline	—	Type: String Whether there is an outline.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.separateContainer	—	Type: Boolean It is drawn on a separate layer.
options.strokeColor	—	Type: String Line color **graphics.style.color**.
options.strokeOpacity	—	Type: Number Contour transparency.
options.strokeStyle	—	Type: Number[]|String Contour style (not supported in Canvas mode) **graphics.style.stroke**.
options.strokeWidth	—	Type: Number Line width.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
datachange	Data change. Data fields: • oldData — Old data. • newData — New data. Inherited from **IOverlay**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
emptinesschange	Change to the empty overlay flag. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IOverlay**.
geometrychange	Changed geometry. Data fields: • oldGeometry — Old pixel geometry. • newGeometry — New pixel geometry. Inherited from **IOverlay**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IOverlay**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
shapechange	Change to the shape of the area spanning the overlay. Instance of the **Event** class. Inherited from **IOverlay**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getData()	Object	Returns the overlay data object. Inherited from **IOverlay**.
getGeometry()	**IPixelGeometry**	Returns the current pixel geometry. Inherited from **IOverlay**.
getMap()	**Map**|null	Returns reference to the current map. Inherited from **IOverlay**.
getShape()	**IShape**|null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from **IOverlay**.
isEmpty()	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from **IOverlay**.
setData(data)		Sets the overlay data. Inherited from **IOverlay**.
setGeometry(geometry)		Sets the overlay pixel geometry.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		Inherited from **IOverlay**.
setMap(map)		Sets the map on which to display the overlay. Inherited from **IOverlay**.

overlay.storage

Static object.

Instance of **util.AsyncStorage**

Storage for overlays. By default, the overlays have not been added to package.full (the standard set of modules). Overlays are loaded on demand when geo objects are added to the map. To get the overlay class, use the require method for this storage. By default, the storage declares the following keys for asynchronous access:

- 'default#placemark' — Placemark image overlay **overlay.Placemark**.
- 'default#pin' — Overlay of the circle **overlay.Pin** placemark
- 'default#circle' — Circle overlay **overlay.Circle**.
- 'default#rectangle' — Rectangle overlay **overlay.Rectangle**.
- 'default#polyline' — Polyline overlay **overlay.Polyline**.
- 'default#polygon' — Polygon overlay **overlay.Polygon**.
- 'hotspot#placemark' — Hotspot placemark overlay **overlay.hotspot.Placemark**.
- 'hotspot#circle' — Hotspot circle overlay **overlay.hotspot.Circle**.
- 'hotspot#rectangle' — Hotspot rectangle overlay **overlay.hotspot.Rectangle**.
- 'hotspot#polyline' — Hotspot polyline overlay **overlay.hotspot.Polyline**.
- 'hotspot#polygon' — Hotspot polygon overlay **overlay.hotspot.Polygon**.
- 'html#balloon' — HTML balloon overlay **overlay.html.Balloon**.
- 'html#hint' — Basic HTML overlay **overlay.html.Hint**.
- 'html#placemark' — HTML placemark overlay **overlay.html.Placemark**.
- 'html#rectangle' — HTML rectangle overlay **overlay.html.Rectangle**.

Methods

Example:

```
ymaps.overlay.storage.require(['hotspot#circle'], function (HotspotOverlayClass)
{
    // Creating an instance of the received class.
    var overlay = new HotspotOverlayClass(
        new ymaps.geometry.Circle([30, 50], 10), {}, {}
    );
});
```

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
define(key[, depends, resolveCallback[, context]])	**util.AsyncStorage**	Defines an asynchronous value in storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
isDefined(key)	Boolean	Checking if the key can be accessed in the storage.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.
require(keys[, successCallback[, errorCallback[, context]]])	**vow.Promise**	Async request to get values from the storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

pane

pane.EventsPane

Extends **IEventPane**.

Pane of events.

Constructor | Fields | Events | Methods

Constructor

```
pane.EventsPane(map, params)
```

Parameters:

Parameter	Default value	Description
map *	—	Type: **Map** Map.
params *	—	Type: Object Parameters.
params.css	—	Type: Object CSS styles of the DOM element of the pane.
params.zIndex	0	Type: Number The zIndex of the pane.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
overflowchange	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of IEvent. Inherited from IPane.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
zindexchange	Change to the zIndex value of a pane. Instance of IEvent. Inherited from IPane.

Methods

Name	Returns	Description
destroy()		Destroys the pane. Inherited from IPane.
getElement()	HTMLElement	Returns a reference to the pane's DOM container. Inherited from IPane.
getMap()	Map	Returns the map that the pane belongs to. Inherited from IPane.
getOverflow()	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: • "visible" — When you go off the map container, the content of the pane remains visible. • "hidden" — The viewport for the pane content is limited by the map container. . Inherited from IPane.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getZIndex()	Number	Returns the zIndex of the pane. Inherited from IPane.

pane.MovablePane

Extends IContainerPane.

Movable map panes.

Constructor | Fields | Events | Methods

Constructor

```
pane.MovablePane(map, params)
```

Parameters:

Parameter	Default value	Description
map *	—	Type: Map Map.
params *	—	Type: Object Parameters.
params.css	—	Type: Object CSS styles of the DOM element of the pane.
params.margin	0	Type: Number Extra padding around the edges of the map container, extending the viewport of the pane.
params.overflow	"hidden"	Type: String This parameter defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: • "visible" — When you go off the map container, the content of the pane remains visible. • "hidden" — The viewport for the pane content is limited by the map container. .
params.zIndex	0	Type: Number The zIndex of the pane.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
clientpixelschange		Change to the pane's local coordinate system. This event means that objects that calculate their positions inside the pane from the map's global pixel coordinates must recalculate it and update their positions inside the pane's DOM element. Instance of IEvent. Inherited from IContainerPane.
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
actionbegin	The start of pane movement. Instance of IEvent. Inherited from IContainerPane.
actionend	The end of pane movement. Instance of IEvent. Inherited from IContainerPane.
clientpixelschange	Change to the coordinate system of the client pixels. Inherited from IContainerPane.
overflowchange	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of IEvent. Inherited from IPane.
viewportchange	Change to a pane's viewport. Instance of IEvent. Inherited from IContainerPane.
zindexchange	Change to the zIndex value of a pane. Instance of IEvent. Inherited from IPane.
zoomchange	Change to the current zoom level of the pane. Instance of IEvent. Inherited from IContainerPane.

Methods

Name	Returns	Description
destroy()		Destroys the pane. Inherited from IPane.
fromClientPixels(clientPixelPoint)	Number[]	Converts client pixel coordinates to global coordinates. Inherited from IPositioningContext.
getElement()	HTMLElement	Returns a reference to the pane's DOM container. Inherited from IPane.
getMap()	Map	Returns the map that the pane belongs to. Inherited from IPane.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getOverflow()	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: • "visible" — When you go off the map container, the content of the pane remains visible. • "hidden" — The viewport for the pane content is limited by the map container. . Inherited from **IPane**.
getViewport()	Number[][]	Returns the viewport for the pane, in client coordinates. Inherited from **IContainerPane**.
getZIndex()	Number	Returns the zIndex of the pane. Inherited from **IPane**.
getZoom()	Number	Returns the current zoom level at which the positioning context works. Inherited from **IPositioningContext**.
toClientPixels(globalPixelPoint)	Number[]	Converts global pixel coordinates to client coordinates. Inherited from **IPositioningContext**.

pane.StaticPane

Extends **IContainerPane**.

A static map pane.

Constructor | Fields | Events | Methods

Constructor

```
pane.StaticPane(map, params)
```

Parameters:

Parameter	Default value	Description
`map` *	—	Type: **Map** Map.
`params` *	—	Type: Object Parameters.
`params.css`	—	Type: Object CSS styles of the DOM element of the pane.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
params.margin	0	Type: Number Extra padding around the edges of the map container, extending the viewport of the pane.
params.overflow	"hidden"	Type: String This parameter defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: • "visible" — When you go off the map container, the content of the pane remains visible. • "hidden" — The viewport for the pane content is limited by the map container. .
params.zIndex	0	Type: Number The zIndex of the pane.

* Mandatory parameter/option.

Fields

Name	Type	Description
clientpixelschange		Change to the pane's local coordinate system. This event means that objects that calculate their positions inside the pane from the map's global pixel coordinates must recalculate it and update their positions inside the pane's DOM element. Instance of IEvent. Inherited from IContainerPane.
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
actionbegin	The start of pane movement. Instance of IEvent. Inherited from IContainerPane.
actionend	The end of pane movement. Instance of IEvent. Inherited from IContainerPane.
clientpixelschange	Change to the coordinate system of the client pixels. Inherited from IContainerPane.
overflowchange	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of IEvent. Inherited from IPane.
viewportchange	Change to a pane's viewport. Instance of IEvent. Inherited from IContainerPane.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
zindexchange	Change to the zIndex value of a pane. Instance of **IEvent**. Inherited from **IPane**.
zoomchange	Change to the current zoom level of the pane. Instance of **IEvent**. Inherited from **IContainerPane**.

Methods

Name	Returns	Description
destroy()		Destroys the pane. Inherited from **IPane**.
fromClientPixels(clientPixelPoint)	Number[]	Converts client pixel coordinates to global coordinates. Inherited from **IPositioningContext**.
getElement()	HTMLElement	Returns a reference to the pane's DOM container. Inherited from **IPane**.
getMap()	**Map**	Returns the map that the pane belongs to. Inherited from **IPane**.
getOverflow()	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: • "visible" — When you go off the map container, the content of the pane remains visible. • "hidden" — The viewport for the pane content is limited by the map container. . Inherited from **IPane**.
getViewport()	Number[][]	Returns the viewport for the pane, in client coordinates. Inherited from **IContainerPane**.
getZIndex()	Number	Returns the zIndex of the pane. Inherited from **IPane**.
getZoom()	Number	Returns the current zoom level at which the positioning context works. Inherited from **IPositioningContext**.
toClientPixels(globalPixelPoint)	Number[]	Converts global pixel coordinates to client coordinates. Inherited from **IPositioningContext**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

panorama

panorama.Base

Extends **IPanorama**.

Constructor | Methods

Constructor

```
panorama.Base()
```

Initializes the panorama with default parameters.

Example:

```
function Panorama () {
    ymaps.panorama.Base.call(this);
    // Making sure that everything is all right with our panorama.
    this.validate();
}

ymaps.util.defineClass(Panorama, ymaps.panorama.Base, {
    getPosition: function () {
        // Let's put our panorama at the center of the coordinate system.
        return [0, 0, 0];
    },

    getCoordSystem: function () {
        return ymaps.coordSystem.cartesian;
    },

    getAngularBBox: function () {
        // We'll make our panorama fully spherical.
        return [0.5 * Math.PI, 2 * Math.PI, -0.5 * Math.PI, 0];
    },

    getTileSize: function () {
        return [512, 512];
    },

    getTileLevels: funciton () {
        // Our panorama will have just one image.
        return [{
            getTileUrl: function (x, y) {
                return '/' + x + '/' + y + '.jpg';
            },

            getImageSize: function () {
                return [4096, 2048];
            }
        }];
    }
});
```

Methods

Name	Returns	Description
getAngularBBox()		Overriding this method is required.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getConnectionArrows()	IPanoramaConnectionArrow[]	Returns an empty array, as if the panorama doesn't have any connection arrows.
getConnectionMarkers()	IPanoramaConnectionMarker[]	Returns an empty array, as if the panorama doesn't have any connection markers.
getConnections()	IPanoramaConnectionMarker[]	**Attention!** This method is deprecated. This method is **deprecated**. Override panorama.Base.getConnectionMarkers.
getCoordSystem()	ICoordSystem	Returns the geographical coordinate system.
getDefaultDirection()	Number[]	Returns the direction of "north" to the horizon.
getDefaultSpan()	Number[]	Returns the field of view is 130 by 80 degrees in radians.
getGraph()	null	Returns `null`, as if there aren't any quick transitions for the panorama graph.
getMarkers()	IPanoramaMarker[]	Returns an empty array, as if the panorama doesn't have any markers.
getName()	String	Returns empty string.
getPosition()		Overriding this method is required.
getThoroughfares()	IPanoramaConnectionArrow[]	**Attention!** This method is deprecated. This method is **deprecated**. Override panorama.Base.getConnectionArrows.
getTileLevels()		Overriding this method is required.
getTileSize()		Overriding this method is required.
validate()		Checks the consistency and validity of data returned by methods of the panorama object. Conditions that this method verifies: • positions of all objects have three components (including height); • the tile size must be a power of two (for example, 128, 256, or 512 pixels); • the panorama is a full circle (i.e. the width of the angular area must be 2π); • each zoom level for the panorama image contains an integral number of tiles horizontally (this isn't required vertically).

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
		If calling this method generates an error for the panorama object, we cannot guarantee the stability of the panorama player with this panorama.

Methods details

getAngularBBox

```
{} getAngularBBox()
```

Overriding this method is required.

getConnectionArrows

```
{IPanoramaConnectionArrow[]} getConnectionArrows()
```

Returns an empty array, as if the panorama doesn't have any connection arrows.

getConnectionMarkers

```
{IPanoramaConnectionMarker[]} getConnectionMarkers()
```

Returns an empty array, as if the panorama doesn't have any connection markers.

getConnections

```
{IPanoramaConnectionMarker[]} getConnections()
```

This method is **deprecated**. Override panorama.Base.getConnectionMarkers.

This method is deprecated.

Returns an empty array, as if the panorama doesn't have any connections.

getCoordSystem

```
{ICoordSystem} getCoordSystem()
```

Returns the geographical coordinate system.

getDefaultDirection

```
{Number[]} getDefaultDirection()
```

Returns the direction of "north" to the horizon.

getDefaultSpan

```
{Number[]} getDefaultSpan()
```

Returns the field of view is 130 by 80 degrees in radians.

getGraph

```
{null} getGraph()
```

Returns `null`, as if there aren't any quick transitions for the panorama graph.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getMarkers

```
{IPanoramaMarker[]} getMarkers()
```

Returns an empty array, as if the panorama doesn't have any markers.

getName

```
{String} getName()
```

Returns empty string.

getPosition

```
{} getPosition()
```

Overriding this method is required.

getThoroughfares

```
{IPanoramaConnectionArrow[]} getThoroughfares()
```

This method is **deprecated**. Override panorama.Base.getConnectionArrows.

This method is deprecated.

Returns an empty array, as if the panorama doesn't have any transitions.

getTileLevels

```
{} getTileLevels()
```

Overriding this method is required.

getTileSize

```
{} getTileSize()
```

Overriding this method is required.

validate

```
{} validate()
```

Checks the consistency and validity of data returned by methods of the panorama object. Conditions that this method verifies:

- positions of all objects have three components (including height);
- the tile size must be a power of two (for example, 128, 256, or 512 pixels);
- the panorama is a full circle (i.e. the width of the angular area must be 2π);
- each zoom level for the panorama image contains an integral number of tiles horizontally (this isn't required vertically).

If calling this method generates an error for the panorama object, we cannot guarantee the stability of the panorama player with this panorama.

panorama.Base.createPanorama

Static function.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns panorama instance.

```
{ IPanorama } panorama.Base.createPanorama(params)
```

Parameters:

Parameter	Default value	Description
params*	—	Type: Object Panorama parameters.
params.angularBBox*	—	Type: Number[] Angular
params.coordSystem	—	Type: **ICoordSystem** Coordinate system that the panorama position is set in. By default, it uses **coordSystem.geo**.
params.name	''	Type: String Name of the panorama.
params.position*	—	Type: Number[] The position of the panorama.
params.tileLevels*	—	Type: **IPanoramaTileLevel**[] Array of panorama image detail levels.
params.tileSize*	—	Type: Number[] Size of the panorama image tiles.

* Mandatory parameter/option.

Example:

```
var player = new ymaps.panorama.Player(
    'player',
    ymaps.panorama.Base.createPanorama({
        coordSystem: ymaps.coordSystem.cartesian,
        // Let's put our panorama in the center of the coordinate system.
        position: [0, 0],
        name: 'My panorama',
        // We'll make our panorama fully spherical.
        angularBBox: [0.5 * Math.PI, 2 * Math.PI, -0.5 * Math.PI, 0],
        tileSize: [512, 512],
        tileLevels: [{
            getTileUrl: function (x, y) {
                return '/' + x + '/' + y + '.jpg';
           },

            getImageSize: function () {
                return [4096, 2048];
            }
        }]
    })
);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

panorama.Base.getMarkerPositionFromDirection

Static function.

Calculates marker coordinates based on two values: the direction of view to the marker and the distance to it. The coordinates are calculated in the same coordinate system that is used in the panorama.

Returns the position of the marker in the coordinate system used in the panorama. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` is the height of the marker in meters, set relative to the same level as the height of the panorama.

```
{ Number[] } panorama.Base.getMarkerPositionFromDirection(panorama, direction, distance)
```

Parameters:

Parameter	Default value	Description
`panorama`*	—	Type: **IPanorama** Object describing the panorama.
`direction`*	—	Type: Number[] The direction of view to the marker in the `[bearing, pitch]` format, where: • `bearing` is the direction to the marker in the horizontal plane; for the geographic coordinate system it represents azimuth (in radians), for the Cartesian — an angle (in radians) starting from the X-axis counterclockwise. • `pitch` is the angle of elevation to the marker (in radians).
`distance`*	—	Type: Number Distance to the marker.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

panorama.createPlayer

Static function.

Searches for a panorama near the specified point. If at least one is found, it creates a panorama player with this panorama.

Returns a promise object that will be resolved by the instance of the class **panorama.Player** or rejected with the error description.

```
{ vow.Promise } panorama.createPlayer(element, point[, options])
```

Parameters:

Parameter	Default value	Description
element*	—	Type: HTMLElement|String A reference to the HTML element that will contain the player, or the ID of this HTML element.
point*	—	Type: Number[] The point for searching for nearby panoramas.
options	—	Type: Object Options.
options.direction	'auto'	Type: Number[]|String Direction of view in the format `[bearing, pitch]`, where `bearing` – is the azimuth of the direction in degrees, `pitch` – angle of elevation above the horizon in degrees. A special string value `auto` means that after opening of the panorama, the direction specified in the panorama's metadata will be applied.
options.layer	'yandex#panorama'	Type: String The layer to search for panoramas. There are two layers available: • yandex#panorama — Land panoramas, • yandex#airPanorama — Aerial panoramas.
options.span	'auto'	Type: Number[]|String The angular dimensions of the field of view in the format `[horizontalSpan, verticalSpan]`, where `horizontalSpan` – the horizontal field size in degrees, `verticalSpan` – the vertical field size in degrees.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

panorama.isSupported

Static function.

Checks if the panorama player supports the user's platform.

Returns `true` if the player supports the browser, otherwise `false`.

```
{ Boolean } panorama.isSupported()
```

panorama.locate

Static function.

Searches for a panorama at the specified point and layer. The result of the query is an array of found panoramas, represented as **Panorama** objects.

Returns the promise object that will be resolved by an array of found panoramas (if no panoramas found, the array will be empty) or rejected with the error description.

```
{ vow.Promise } panorama.locate(point[, options])
```

Parameters:

Parameter	Default value	Description
point*	—	Type: Number[] The point for searching for nearby panoramas.
options	—	Type: Object Options.
options.layer	'yandex#panorama'	Type: String The layer to search for panoramas. There are two layers available: • yandex#panorama — Land panoramas, • yandex#airPanorama — Aerial panoramas.

* Mandatory parameter/option.

panorama.Manager

Note:
The constructor of the panorama.Manager class is hidden, as this class is not intended for autonomous initialization.

Extends **IEventEmitter**.

Manager of the panorama player linked to the map.

Fields | Events | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
closeplayer	The panorama player is closed. Instance of the Event class.
disablelookup	Search mode is disabled for panoramas. Instance of the Event class.
enablelookup	Search mode is enabled for panoramas. Instance of the Event class.
locate	Started searching for panoramas at the given point. Instance of the Event class. Names of fields that are available via the Event.get method: • point — Search for panoramas around this point. • options — Panorama search options.
locatefail	The panorama search failed with an error. Names of fields that are available via the Event.get method: • point — Search for panoramas around this point. • options — Panorama search options. • error — The error that occurred while searching.
locatesuccess	The panorama search finished successfully. Names of fields that are available via the Event.get method: • point — Search for panoramas around this point. • options — Panorama search options. • result — The list of the found panoramas.
openplayer	The panorama player is opened. Instance of the Event class.

Methods

Name	Returns	Description
closePlayer()		Hides the panorama player.
disableLookup()		Disables panorama search mode on the map.
enableLookup()		Enables panorama search mode on the map.
getPlayer()	panorama.Player	Returns the current panorama player or null if the player is closed.
isLookupEnabled()	Boolean	Checks whether panorama search mode is enabled on the map.
openPlayer(panorama[, locateOptions[, options]])	vow.Promise	Opens the panorama player.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

closeplayer

The panorama player is closed. Instance of the **Event** class.

disablelookup

Search mode is disabled for panoramas. Instance of the **Event** class.

enablelookup

Search mode is enabled for panoramas. Instance of the **Event** class.

locate

Started searching for panoramas at the given point. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- point — Search for panoramas around this point.
- options — Panorama search options.

locatefail

The panorama search failed with an error. Names of fields that are available via the **Event.get** method:

- point — Search for panoramas around this point.
- options — Panorama search options.
- error — The error that occurred while searching.

locatesuccess

The panorama search finished successfully. Names of fields that are available via the **Event.get** method:

- point — Search for panoramas around this point.
- options — Panorama search options.
- result — The list of the found panoramas.

openplayer

The panorama player is opened. Instance of the **Event** class.

Methods details

closePlayer

```
{} closePlayer()
```

Hides the panorama player.

disableLookup

```
{} disableLookup()
```

Disables panorama search mode on the map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

enableLookup

```
{} enableLookup()
```

Enables panorama search mode on the map.

getPlayer

```
{panorama.Player} getPlayer()
```

Returns the current panorama player or null if the player is closed.

isLookupEnabled

```
{Boolean} isLookupEnabled()
```

Checks whether panorama search mode is enabled on the map.

openPlayer

```
{vow.Promise} openPlayer(panorama[, locateOptions[, options]])
```

Opens the panorama player.

Returns the promise that will be rejected with an error if the panorama has failed to open or if opening has been canceled by the closePlayer request.

Parameters:

Parameter	Default value	Description
panorama*	—	Type: IPanorama|Number[] Panorama or the coordinates of the panorama.
locateOptions	—	Type: Object Options for panorama.locate.
options	—	Type: Object
options.direction	'auto'	Type: Number[] Direction of view for panorama.

* Mandatory parameter/option.

panorama.Player

Extends IEventEmitter.

Class for creating and controlling the panorama player.

Constructor | Fields | Events | Methods

Constructor

```
panorama.Player(element, panorama[, options])
```

Creates an instance of the panorama player.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
element *	—	Type: HTMLElement|String A reference to the HTML element that will contain the player, or the ID of this HTML element.
panorama *	—	Type: **IPanorama** The panorama that will be opened in the created panorama player.
options	—	Type: Object Options.
options.autoFitToViewport	"always"	Type: String The option to disable automatic tracking of the size of the player container. By default, the player always follows the size of its container, and reconstructs the image if the size has changed. Available values: • none — Do not track the size of the container. • ifNull — As soon as the container gets a CSS "display" value other than "None", the player automatically adjusts its size to the value. After that, tracking stops. • always — Always follow the size of the container.
options.controls	—	Type: String[] Set of player controls. Available controls: • closeControl – The button that closes the player. • fullscreenControl – The button that switches the player to full screen mode. • panoramaName – The name of the panorama (automatically hidden in the small player). • zoomControl – Zoom control on the panorama. The set of controls includes all of the above by default.
options.direction	'auto'	Type: Number[]|String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Direction of view in the format `[bearing, pitch]`, where `bearing` – is the azimuth of the direction in degrees, and `pitch` is the angle of elevation above the horizon in degrees. The special string value `auto` means that the direction specified in the panorama's metadata is applied when opening the panorama.
`options.hotkeysEnabled`	false	Type: Boolean Enables keyboard control of the player. Please note that when you open the player, it begins to intercept some keys (e.g. arrow keys), thus canceling the default reaction of the browser, which may prevent the user from interacting with your page. This is why keyboard control is disabled by default.
`options.scrollZoomBehavior`	true	Type: Boolean Disables zooming of the panorama by scrolling. Enabled by default, and the player intercepts the mouse wheel events.
`options.span`	'auto'	Type: Number[]|String The angular dimensions of the field of view in the format `[horizontalSpan, verticalSpan]`, where `horizontalSpan` is the horizontal field size in degrees, and `verticalSpan` is the vertical field size in degrees.
`options.suppressMapOpenBlock`	false	Type: Boolean Whether to hide the offer to open the current panorama in Yandex.Maps with all the available map information preserved as completely as possible. `true` — hide, `false` — don't hide. The link to Yandex.Maps is displayed in the lower-left corner of the player.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
destroy	The player was closed by the user or destroyed using the panorama.Player.destroy method.
directionchange	The view direction changed.
error	An error occurred during operation of the player. The user will be shown the appropriate screen.
fullscreenenter	The panorama player switched to full-screen mode.
fullscreenexit	The panorama player exited full-screen mode.
markercollapse	The user clicked on an expanded marker. Names of fields that are available via the Event.get method: • marker — The marker that was collapsed.
markerexpand	The user clicked on a collapsed marker. Names of fields that are available via the Event.get method: • marker — The marker that was expanded.
markermouseenter	The user's cursor hovered over a marker. Names of fields that are available via the Event.get method: • marker — The marker that the cursor hovered over.
markermouseleave	The user's cursor left a marker. Names of fields that are available via the Event.get method: • marker — The marker that the cursor moved away from.
panoramachange	The open panorama changed (for example, as the result of calling the panorama.Player.setPanorama function or a user action).
spanchange	The size of the viewport has been changed.

Methods

Name	Returns	Description
destroy()		Destroys the player.
fitToViewport()		Checks the size of the player container and if it has changed since the last inspection, rebuilds the image.
getDirection()	Number[]	Returns the current viewing direction, in the format `[bearing, pitch]`, where `bearing` is the azimuth of the direction in degrees, and `pitch` is the angle of elevation above the horizon in degrees.
getPanorama()	IPanorama	Returns the panorama that is currently open in the player.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getSpan()	Number[]	Returns the current angular dimensions of the field of view, in the format `[horizontalSpan, verticalSpan]`, where `horizontalSpan` is the horizontal field size in degrees, and `verticalSpan` is the vertical field size in degrees.
lookAt(point)	**panorama.Player**	Rotates the view so that the passed point is in the center of the field of view.
moveTo(point[, options])	**vow.Promise**	Searches for a panorama with the specified parameters and opens it.
setDirection(direction)	**panorama.Player**	Sets a new viewing direction.
setPanorama(panorama)	**panorama.Player**	Opens the passed panorama in the player.
setSpan(span)	**panorama.Player**	Sets new dimensions for the field of view.

Events details

destroy

The player was closed by the user or destroyed using the panorama.Player.destroy method.

directionchange

The view direction changed.

error

An error occurred during operation of the player. The user will be shown the appropriate screen.

fullscreenenter

The panorama player switched to full-screen mode.

fullscreenexit

The panorama player exited full-screen mode.

markercollapse

The user clicked on an expanded marker. Names of fields that are available via the **Event.get** method:

- marker — The marker that was collapsed.

markerexpand

The user clicked on a collapsed marker. Names of fields that are available via the **Event.get** method:

- marker — The marker that was expanded.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

markermouseenter

The user's cursor hovered over a marker. Names of fields that are available via the **Event.get** method:

- marker — The marker that the cursor hovered over.

markermouseleave

The user's cursor left a marker. Names of fields that are available via the **Event.get** method:

- marker — The marker that the cursor moved away from.

panoramachange

The open panorama changed (for example, as the result of calling the panorama.Player.setPanorama function or a user action).

spanchange

The size of the viewport has been changed.

Methods details

destroy

```
{} destroy()
```

Destroys the player.

fitToViewport

```
{} fitToViewport()
```

Checks the size of the player container and if it has changed since the last inspection, rebuilds the image.

getDirection

```
{Number[]} getDirection()
```

Returns the current viewing direction, in the format `[bearing, pitch]`, where `bearing` is the azimuth of the direction in degrees, and `pitch` is the angle of elevation above the horizon in degrees.

getPanorama

```
{IPanorama} getPanorama()
```

Returns the panorama that is currently open in the player.

getSpan

```
{Number[]} getSpan()
```

Returns the current angular dimensions of the field of view, in the format `[horizontalSpan, verticalSpan]`, where `horizontalSpan` is the horizontal field size in degrees, and `verticalSpan` is the vertical field size in degrees.

lookAt

```
{panorama.Player} lookAt(point)
```

Rotates the view so that the passed point is in the center of the field of view.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns self-reference.

Parameters:

Parameter	Default value	Description
point*	—	Type: Number[] The point to rotate the view at. It can be an array of two or three coordinates. The first two coordinates are interpreted as the geographical coordinates of the point. If three coordinates are passed, the third is interpreted as the height of the point relative to the panorama in meters.

* Mandatory parameter/option.

moveTo

```
{vow.Promise} moveTo(point[, options])
```

Searches for a panorama with the specified parameters and opens it.

Returns a promise object that will be resolved if the panorama is found and successfully opened in the player, or rejected with an error message otherwise.

Parameters:

Parameter	Default value	Description
point*	—	Type: Number[] The point for searching for nearby panoramas.
options	—	Type: Object Options.
options.direction	'auto'	Type: Number[]|String Direction of view in the format `[bearing, pitch]`, where `bearing` is the azimuth of the direction in degrees, and `pitch` is the angle of elevation above the horizon in degrees. The special string value `auto` means that the direction specified in the panorama's metadata is applied when opening the panorama.
options.layer	'yandex#panorama'	Type: String The layer to search for panoramas. There are two layers available: • yandex#panorama — Land panoramas. • yandex#airPanorama — Aerial panoramas.
options.span	'auto'	Type: Number[]|String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The angular dimensions of the field of view in the format `[horizontalSpan, verticalSpan]`, where `horizontalSpan` is the horizontal field size, and `verticalSpan` is the vertical field size.

* Mandatory parameter/option.

setDirection

```
{panorama.Player} setDirection(direction)
```

Sets a new viewing direction.

Returns self-reference.

Parameters:

Parameter	Default value	Description
`direction`*	—	Type: Number[]|String Direction of view in the format `[bearing, pitch]`, where `bearing` is the azimuth of the direction in degrees, and `pitch` is the angle of elevation above the horizon in degrees. The special string value `auto` means that the direction specified in the panorama's metadata is applied when opening the panorama.

* Mandatory parameter/option.

setPanorama

```
{panorama.Player} setPanorama(panorama)
```

Opens the passed panorama in the player.

Returns self-reference.

Parameters:

Parameter	Default value	Description
`panorama`*	—	Type: **IPanorama** Panorama.

* Mandatory parameter/option.

setSpan

```
{panorama.Player} setSpan(span)
```

Sets new dimensions for the field of view.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
span*	—	Type: Number[]|String The angular dimensions of the field of view in the format `[horizontalSpan, verticalSpan]`, where `horizontalSpan` is the horizontal field size in degrees, and `verticalSpan` is the vertical field size in degrees.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Panorama

Note:
The constructor of the Panorama class is hidden, as this class is not intended for autonomous initialization.

Extends **IPanorama**.

Object describing the panorama.

See panorama.locate

Methods

Methods

Name	Returns	Description
createPlayer(element[, options])	**vow.Promise**	Creates a new instance of the **panorama.Player** class and opens the panorama in it.
getAngularBBox()	Number[]	Returns spherical coordinates that define the area covered by the image on the panoramic sphere. Coordinates are specified in the format `[thetaTop, phiRight, thetaBottom, phiLeft]` (the same as CSS). Inherited from **IPanorama**.
getConnectionArrows()	**IPanoramaConnectionArrow**[]	Returns array of connection arrows on the panorama. Inherited from **IPanorama**.
getConnectionMarkers()	**IPanoramaConnectionMarker**[]	Returns array of connection markers on the panorama. Inherited from **IPanorama**.
getCoordSystem()	**ICoordSystem**	Returns the coordinate system that is used for defining positions of the panorama and all the associated markers and connections. Inherited from **IPanorama**.
getDefaultDirection()	Number[]	Returns default direction of view. It will be used by the player when opening the panorama. Inherited from **IPanorama**.
getDefaultSpan()	Number[]	Returns default size of the viewing area. It will be used by the player when opening the panorama. Inherited from **IPanorama**.
getGraph()	**IPanoramaGraph**|null	Returns the graph of panoramas connected to the current panorama for making quick transitions. Inherited from **IPanorama**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getLayer()	String	Returns the layer the panorama belongs to.
getMarkers()	IPanoramaMarker[]	Returns array of markers on the panorama. Inherited from IPanorama.
getName()	String	Returns the name of the panorama displayed by the player in the interface. Inherited from IPanorama.
getPosition()	Number[]	Returns the location of the panorama in the coordinate system set in the options. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` – the height of the panorama in meters, relative to some level (not necessarily sea level). Inherited from IPanorama.
getTileLevels()	IPanoramaTileLevel[]	Returns array of zoom levels for the panorama image. Inherited from IPanorama.
getTileSize()	Number[]	Returns the size of tiles that the panorama image is divided into. Inherited from IPanorama.

Methods details

createPlayer

```
{vow.Promise} createPlayer(element[, options])
```

Creates a new instance of the panorama.Player class and opens the panorama in it.

Returns a promise object that will be resolved by the instance of the class panorama.Player with the current panorama open.

Parameters:

Parameter	Default value	Description
element*	—	Type: HTMLElement|String A reference to the HTML element that will contain the player, or the ID of this HTML element.
options	—	Type: Object Options.
options.direction	'auto'	Type: Number[]|String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Direction of view in the format `[bearing, pitch]`, where `bearing` – is the azimuth of the direction in degrees, `pitch` – angle of elevation above the horizon in degrees. A special string value `auto` means that after opening of the panorama, the direction specified in the panorama's metadata will be applied.
`options.span`	'auto'	Type: Number[]|String The angular dimensions of the field of view in the format `[horizontalSpan, verticalSpan]`, where `horizontalSpan` – the horizontal field size, `verticalSpan` – the vertical field size.

* Mandatory parameter/option.

getLayer

```
{String} getLayer()
```

See panorama.locate

Returns the layer the panorama belongs to.

Placemark

Extends **GeoObject**.

Placemark. A geo object with the geometry **geometry.Point**.

See GeoObject geometry.Point

Constructor | Fields | Events | Methods

Constructor

```
Placemark(geometry[, properties[, options]])
```

Creates a placemark instance.

Parameters:

Parameter	Default value	Description
`geometry` *	—	Type: Number[]|Object|**IPointGeometry** Coordinates of the placemark, or a hash describing the geometry, or a reference to the point geometry object.
`properties`	—	Type: Object|**IDataManager**

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Placemark data. Can be set as a class instance implementing the **IDataManager** interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object: • iconContent — Content of the geo object's icon. • hintContent — Content of the geo object's popup hint. • balloonContent — Content of the geo object's balloon. • balloonContentHeader — Content of the geo object balloon title. • balloonContentBody — Content of the main part of the geo object's balloon. • balloonContentFooter — Content of the lower part of the geo object's balloon. The balloonContent field is a shortcut for the balloonContentBody field, but if they are both set simultaneously, balloonContentBody takes priority. You can also add your own custom fields to the placemark data and use them wherever possible. For example, in the placemark layout or balloon layout.

`options`	—	Type: Object
		Placemark options.
		To change the style and color of an icon, use the following options:
		• `preset` — Key for the placemark's preset options. The list of available keys is stored in the **option.presetStorage** description. You can set the `preset` option together with the `iconColor` option only if it takes one of the following values:
		  • 'islands#icon'
		  • 'islands#dotIcon'
		  • 'islands#circleIcon'
		  • 'islands#circleDotIcon'
		• `iconColor` — The color of the placemark icon. You can specify it in any format that is acceptable in CSS (for example, by name or in the RGB format). This option is used for standard icons in browsers that support SVG.
		Note:
		This option does not work in IE8.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description

		If you want to create your own icon layout, you should specify the following options:
		• iconLayout — Icon layout. Type: constructor for an object with the ILayout interface, or its key in the storage. List of available layouts:
		  • 'default#image' — Custom icon image.
		  • 'default#imageWithContent' — A custom image for an icon with content.
		• Additional options for the **layout.Image** and **layout.ImageWithContent** classes with the "icon" prefix.
		
		Note:
		To specify the layout of the icon shadow, use the same set of options, but with the "iconShadow" prefix. For example, iconShadowLayout.
		
		You can also set options for individual objects using the Placemark class:
		• Options for the placemark's balloon with the `balloon` prefix.
		• Options for the placemark's popup hint with the `hint` prefix.
		• Options for the polygon's geometry editor with the `editor` prefix. See the description of the **geometryEditor.Point** class.
		• Geometry options can be set without a prefix. See the description of the **IGeometry** class for the **geometry.Point** geometry.
		
		Below are the other options of this class.

`options.cursor`	"pointer"	Type: String Type of cursor over a placemark.
`options.draggable`	false	Type: Boolean Checks whether the placemark can be dragged.
`options.hasBalloon`	true	Type: Boolean Checks whether the placemark has the "balloon" field.
`options.hasHint`	true	Type: Boolean Checks whether the placemark has the "hint" field.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.hideIconOnBalloonOpen	true	Type: Boolean Hide the placemark when opening the balloon.
options.iconOffset	—	Type: Number[] The pixel offset of the icon relative to its set position.
options.iconShape	—	Type: **IGeometryJson**|null The hotspot shape of the placemark. Specified as a JSON description of the pixel geometry of the icon. Use this option when creating your HTML layouts. The coordinates of the figure geometry are counted from the anchor point.
options.interactiveZIndex	true	Type: Boolean Enables automatically modifying the z-index of the placemark depending on its state.
options.interactivityModel	"default#geoObject"	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.openBalloonOnClick	true	Type: Boolean Checks whether to show the balloon when the placemark is clicked on.
options.openEmptyBalloon	false	Type: Boolean Checks whether to show an empty balloon when the placemark is clicked on.
options.openEmptyHint	false	Type: Boolean Checks whether to show an empty hint when the mouse pointer hovers over the placemark.
options.openHintOnHover	true	Type: Boolean Checks whether to show the hint when the mouse pointer hovers over the placemark.
options.pane	"places"	Type: String The key of the pane where the placemark overlay is placed.
options.pointOverlay	"default#placemark"	Type: String|Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelPointGeometry** — The pixel geometry itself. • data: **IDataManager** or Object — Overlay data. • options: Object — Overlay options. And returns **vow.Promise**.
`options.syncOverlayInit`	false	Type: Boolean Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a placemark to the map.
`options.useMapMarginInDragging`	true	Type: Boolean When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with **map.margin.Manager**.
`options.visible`	true	Type: Boolean Checks placemark visibility.
`options.zIndex`	—	Type: Number The z-index of a placemark in its normal state. Lowest priority.
`options.zIndexActive`	—	Type: Number The z-index of a placemark icon with an open balloon. Highest priority.
`options.zIndexDrag`	—	Type: Number The z-index of a placemark that is being dragged.
`options.zIndexHover`	—	Type: Number The z-index of a placemark when the mouse pointer is hovering over it.

* Mandatory parameter/option.

Examples:

1.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Creating a placemark.
var placemark = new ymaps.Placemark([55.75, 37.61], {
    balloonContent: '&lt;img src="http://img-fotki.yandex.ru/get/
6114/82599242.2d6/0_88b97_ec425cf5_M" /&gt;',
    iconContent: "Azerbaijan"
}, {
    preset: "islands#yellowStretchyIcon",
    // Disabling the close balloon button.
    balloonCloseButton: false,
    // The balloon will open and close when the placemark icon is clicked.
    hideIconOnBalloonOpen: false
});
geoMap.geoObjects.add(placemark);
```

2.

```
var placemark = new ymaps.Placemark([55.75, 37.61], {}, {
    // Setting the placemark style (circle).
    preset: "islands#circleDotIcon",
    // Setting the placemark color (in RGB format).
    iconColor: '#ff0000'
});
geoMap.geoObjects.add(placemark);
```

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object. Inherited from **GeoObject**.
editor	**geometryEditor.Point**	The "Point" geometry editor.
events	**event.Manager**	Event manager. Inherited from **GeoObject**.
geometry	**geometry.Point**	The "Point" type of geometery.
hint	**geoObject.Hint**	Geo object hint. Inherited from **GeoObject**.
options	**option.Manager**	Geo object options manager. Inherited from **GeoObject**.
properties	**data.Manager**	Geo object data manager. Inherited from **GeoObject**.
state	**data.Manager**	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged Inherited from **GeoObject**.

Events

Name	Description
balloonclose	Closing the balloon. Instance of the **Event** class. Inherited from **GeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
balloonopen	Opening a balloon on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
beforedrag	Event preceding the "drag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent drag event will be canceled. Inherited from **GeoObject**.
beforedragstart	Event preceding the "dragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from **GeoObject**.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
drag	Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
dragend	End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragstart	Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent — Original event of the geometry editor. Inherited from **GeoObject**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
hintclose	Closing the hint. Instance of the **Event** class. Inherited from **GeoObject**.
hintopen	Opening a hint on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

editor

```
{geometryEditor.Point} editor
```

The "Point" geometery editor.

geometry

```
{geometry.Point} geometry
```

The "Point" type of geometery.

Polygon

Extends GeoObject.

Polygon. A geo object with the geometry geometry.Polygon.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

See GeoObject geometry.Polygon

Constructor | Fields | Events | Methods

Constructor

```
Polygon(geometry[, properties[, options]])
```

Creates a polygon instance.

Parameters:

Parameter	Default value	Description
geometry *	—	Type: Number[][][]|Object|**IPolygonGeometry** Coordinates of polyline vertexes that define the outer and inner boundaries of the polygon, a hash object with geometry parameters, or a reference to the geometry object. The inner boundary can be omitted.
properties	—	Type: Object|**IDataManager** Polygon data. Can be set as a class instance implementing the **IDataManager** interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object: • hintContent — Content of the polygon's popup hint. • balloonContent — Content of the polygon's balloon. • balloonContentHeader — Content of the polygon balloon title. • balloonContentBody — Content of the main part of the polygon's balloon. • balloonContentFooter — Content of the lower part of the polygon's balloon. The balloonContent field is a shortcut for the balloonContentBody field, but if they are both set simultaneously, balloonContentBody takes priority. You can also add your own custom fields to the polygon data and use them wherever possible. For example, in the polygon balloon layout.
options	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Polygon options. Using this parameter, you can set options for the polygon itself, as well as for its parts: • Options for the polygon's balloon with the `balloon` prefix. • Options for the polygon's popup hint with the `hint` prefix. • Options for the polygon's geometry editor with the `editor` prefix. See the description of the **geometryEditor.Polygon** class. • Geometry options can be set without a prefix. See the description of the **IGeometry** class for the **geometry.Polygon** geometry.
`options.cursor`	"pointer"	Type: String Type of cursor over a polygon.
`options.draggable`	false	Type: Boolean Checks whether the polygon can be dragged.
`options.fill`	true	Type: Boolean Whether the shape is filled.
`options.fillColor`	"0066ff99"	Type: String Fill color.
`options.fillImageHref`	—	Type: String Background image. When this option is enabled in **stretch** mode, the "fillColor" value is ignored.
`options.fillMethod`	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
`options.fillOpacity`	1	Type: Number Fill transparency.
`options.hasBalloon`	true	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Checks if the polygon has the "balloon" field.
options.hasHint	true	Type: Boolean Checks if the polygon has the "hint" field.
options.interactiveZIndex	false	Type: Boolean Enables to automatically modify z-index of the polygon depending on its state.
options.interactivityModel	"default#geoObject"	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.opacity	1	Type: Number Transparency.
options.openBalloonOnClick	true	Type: Boolean Checks whether to show the balloon when the polygon is clicked on.
options.openEmptyBalloon	false	Type: Boolean Checks whether to show an empty balloon when the polygon is clicked on.
options.openEmptyHint	false	Type: Boolean Checks whether to show an empty hint when the mouse pointer hovers over the polygon.
options.openHintOnHover	true	Type: Boolean Checks whether to show an empty hint when the mouse pointer hovers over the polygon.
options.outline	true	Type: Boolean Whether the shape has an outline.
options.pane	"areas"	Type: String The key of the pane where the polygon overlay is placed.
options.polygonOverlay	"default#polygon"	Type: String|Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelPolygonGeometry** — The pixel geometry itself. • data: **IDataManager** or Object — Overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
options.strokeColor	"0066ffff"	Type: String|String[] Color of the line or outline. You can set multiple values for a multistroke outline.
options.strokeOpacity	1	Type: Number|Number[] Transparency of the line or outline. You can set multiple values for a multistroke outline.
options.strokeStyle	—	Type: String|Object|String[]|Object[] Style of the line or outline. You can set multiple values for a multistroke outline.
options.strokeWidth	1	Type: Number|Number[] Thickness of the line or outline. You can set multiple values for a multistroke outline.
options.syncOverlayInit	false	Type: Boolean Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a polygon to the map.
options.useMapMarginInDragging	true	Type: Boolean When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with **map.margin.Manager**.
options.visible	true	Type: Boolean Checks the visibility of the polygon.
options.zIndex	—	Type: Number

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The z-index of the polygon in the normal state. Lowest priority.
options.zIndexActive	—	Type: Number The z-index of the polygon with an opened balloon. Highest priority.
options.zIndexDrag	—	Type: Number The z-index of the polygon while dragging.
options.zIndexHover	—	Type: Number The z-index of a polygon when the mouse pointer is hovering over it.

* Mandatory parameter/option.

Example:

```
var polygon = new ymaps.Polygon([
    // Coordinates of the outer contour.
    [[-80, 60], [-90, 50], [-60, 40], [-80, 60]],
    // Coordinates of the inner contour.
    [[-90, 80], [-90, 30], [-20, 40], [-90, 80]]
], {
    hintContent: "Polygon"
}, {
    fillColor: '#6699ff',
    // Making the polygon transparent for map events.
    interactivityModel: 'default#transparent',
    strokeWidth: 8,
    opacity: 0.5
});
myMap.geoObjects.add(polygon);
myMap.setBounds(polygon.geometry.getBounds());
```

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object. Inherited from **GeoObject**.
editor	**geometryEditor.Polygon**	The "Polygon" geometry editor.
events	**event.Manager**	Event manager. Inherited from **GeoObject**.
geometry	**geometry.Polygon**	The "Polygon" type of geometry.
hint	**geoObject.Hint**	Geo object hint. Inherited from **GeoObject**.
options	**option.Manager**	Geo object options manager. Inherited from **GeoObject**.
properties	**data.Manager**	Geo object data manager. Inherited from **GeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
state	data.Manager	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged Inherited from GeoObject.

Events

Name	Description
balloonclose	Closing the balloon. Instance of the Event class. Inherited from GeoObject.
balloonopen	Opening a balloon on a geo object. Instance of the Event class. Inherited from GeoObject.
beforedrag	Event preceding the "drag" event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a DomEvent object), if there is one. Names of methods that are accessible via Event.callMethod: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the Event.preventDefault method is called for this event, a subsequent drag event will be canceled. Inherited from GeoObject.
beforedragstart	Event preceding the "dragstart" event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a DomEvent object), if there is one. If the Event.preventDefault method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from GeoObject.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
drag	Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragend	End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragstart	Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent — Original event of the geometry editor. Inherited from **GeoObject**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
hintclose	Closing the hint. Instance of the **Event** class. Inherited from **GeoObject**.
hintopen	Opening a hint on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

editor

```
{geometryEditor.Polygon} editor
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

The "Polygon" geometry editor.

geometry

```
{geometry.Polygon} geometry
```

The "Polygon" type of geometry.

Polyline

Extends **GeoObject**.

Polyline. A geo object with the geometry **geometry.LineString**.

See GeoObject geometry.LineString

Constructor | Fields | Events | Methods

Constructor

```
Polyline(geometry[, properties[, options]])
```

Creates a polyline instance.

Parameters:

Parameter	Default value	Description
geometry *	—	Type: Number[][]|Object|**ILineStringGeometry** Coordinates of the vertexes, a hash object with geometery parameters, or a reference to the polyline geometry object.
properties	—	Type: Object|**IDataManager** Polyline data. Can be set as a class instance implementing the **IDataManager** interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object: • hintContent — Content of the polyline's popup hint. • balloonContent — Content of the polyline's balloon. • balloonContentHeader — Content of the polyline balloon title. • balloonContentBody — Content of the main part of the polyline's balloon. • balloonContentFooter — Content of the lower part of the polyline's balloon.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The balloonContent field is a shortcut for the balloonContentBody field, but if they are both set simultaneously, balloonContentBody takes priority. You can also add your own custom fields to the polyline data and use them, for example, in the popup hint layout.
options	—	Type: Object Polyline options. Using this parameter, you can set options for the polyline itself, as well as for its parts: • Options for the polyline balloon with the balloon prefix. • Options for the polyline's popup hint with the hint prefix. • Options for the polyline's geometry editor with the editor prefix. See the description of the geometryEditor.LineString class. • Geometry options can be set without a prefix. See the description of the IGeometry class for the geometry.LineString geometry.
options.cursor	"pointer"	Type: String Type of cursor over a polyline.
options.draggable	false	Type: Boolean Checks whether the polyline can be dragged.
options.hasBalloon	true	Type: Boolean Checks whether the polyline has the "balloon" field.
options.hasHint	true	Type: Boolean Checks whether the polyline has the "hint" field.
options.interactiveZIndex	false	Type: Boolean Enables automatically modifying the z-index of the polyline depending on its state.
options.interactivityModel	"default#geoObject"	Type: String Interactivity model. Available keys and their values are listed in the description of interactivityModel.storage.
options.lineStringOverlay	"default#polyline"	Type: String|Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelLineStringGeometry** — The pixel geometry itself. • data: **IDataManager** or Object — Overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
options.opacity	1	Type: Number Transparency.
options.openBalloonOnClick	true	Type: Boolean Checks whether to show the balloon when the polyline is clicked on.
options.openEmptyBalloon	false	Type: Boolean Checks whether to show an empty balloon when the polyline is clicked on.
options.openEmptyHint	false	Type: Boolean Checks whether to show an empty hint when the mouse pointer hovers over the polyline.
options.openHintOnHover	true	Type: Boolean Checks whether to show an empty hint when the mouse pointer hovers over the polyline.
options.pane	"areas"	Type: **IPane**|String The key of the pane where the polyline overlay is placed.
options.strokeColor	"0066ffff"	Type: String|String[] Color of the line or outline. You can set multiple values for a multistroke outline.
options.strokeOpacity	1	Type: Number|Number[] Transparency of the line or outline. You can set multiple values for a multistroke outline.
options.strokeStyle	—	Type: String|Object|String[]|Object[] Style of the line or outline. You can set multiple values for a multistroke outline.
options.strokeWidth	1	Type: Number|Number[]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Thickness of the line or outline. You can set multiple values for a multistroke outline.
options.syncOverlayInit	false	Type: Boolean Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of polylines. However, adding asynchronously does not allow accessing the overlay immediately after adding a polyline to the map.
options.useMapMarginInDragging	true	Type: Boolean When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with **map.margin.Manager**.
options.visible	true	Type: Boolean Checks polyline visibility.
options.zIndex	—	Type: Number The z-index of a polyline in its normal state. Lowest priority.
options.zIndexActive	—	Type: Number The z-index of a polyline with an open balloon. Highest priority.
options.zIndexDrag	—	Type: Number The z-index of a polyline that is being dragged.
options.zIndexHover	—	Type: Number The z-index of a polyline when the mouse pointer is hovering over it.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Creating a polyline
var polyline = new ymaps.Polyline([
    [-80, 60], [-90, 50], [-60, 40], [-80, 60]
], {
    hintContent: "Polyline"
}, {
    draggable: true,
    strokeColor: '#000000',
    strokeWidth: 4,
    // The first number sets the stroke length. The second number sets the gap
length.
    strokeStyle: '1 5'
});
// Adding the polyline to the map.
myMap.geoObjects.add(polyline);
// Setting the polyline borders for the map.
myMap.setBounds(polyline.geometry.getBounds());
```

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object. Inherited from **GeoObject**.
editor	**geometryEditor.LineString**	The "Polyline" geometry editor.
events	**event.Manager**	Event manager. Inherited from **GeoObject**.
geometry	**geometry.LineString**	"Polyline" type of geometry.
hint	**geoObject.Hint**	Geo object hint. Inherited from **GeoObject**.
options	**option.Manager**	Geo object options manager. Inherited from **GeoObject**.
properties	**data.Manager**	Geo object data manager. Inherited from **GeoObject**.
state	**data.Manager**	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged Inherited from **GeoObject**.

Events

Name	Description
balloonclose	Closing the balloon. Instance of the **Event** class. Inherited from **GeoObject**.
balloonopen	Opening a balloon on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
beforedrag	Event preceding the "drag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent drag event will be canceled. Inherited from **GeoObject**.
beforedragstart	Event preceding the "dragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from **GeoObject**.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
drag	Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragend	End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **GeoObject**.
dragstart	Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent — Original event of the geometry editor. Inherited from **GeoObject**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
hintclose	Closing the hint. Instance of the **Event** class. Inherited from **GeoObject**.
hintopen	Opening a hint on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

editor

```
{geometryEditor.LineString} editor
```

The "Polyline" geometry editor.

geometry

```
{geometry.LineString} geometry
```

"Polyline" type of geometry.

Popup

Extends IPopup.

A class for creating an info object.

Constructor | Fields | Events | Methods

Constructor

```
Popup(map[, options])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Info object.

Parameters:

Parameter	Default value	Description
`map` *	—	Type: **Map** Reference to the map.
`options`	—	Type: Object Options.
`options.closeTimeout`	700	Type: Number Delay before closing (in ms).
`options.interactivityModel`	—	Type: String Key for the interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
`options.openTimeout`	150	Type: Number Delay before opening (in ms).
`options.pane`	—	Type: **IPane**|String Info object pane. For the list of available default panes, see **map.pane.Manager**.
`options.projection`	—	Type: **IProjection** The projection of the coordinates to global pixels.
`options.zIndex`	—	Type: String The z-index of the info object.

* Mandatory parameter/option.

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Events

Name	Description
close	Closing the info object. Inherited from **IPopup**.
open	Opening the info object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IPopup**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Methods

Name	Returns	Description
close([force])	**vow.Promise**	Closes the info object. Inherited from **IPopup**.
getData()		Returns info object data. Inherited from **IPopup**.
getOverlay()	**vow.Promise**	Returns the promise object to return the overlay. Inherited from **IPopup**.
getOverlaySync()	**IOverlay**	Returns the overlay, if one exists. Inherited from **IPopup**.
getPosition()		Returns the coordinates of the info object. Inherited from **IPopup**.
isOpen()	Boolean	Returns the info object state: open/closed. Inherited from **IPopup**.
open([position[, data]])	**vow.Promise**	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the **IProjection** that is in the options. Inherited from **IPopup**.
setData(data)	**vow.Promise**	Defines new data for the info object. Inherited from **IPopup**.
setPosition(position)	**vow.Promise**	Specifies a new position for the info object. Inherited from **IPopup**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

projection

projection.Cartesian

Extends **IProjection**.

Cartesian projection of a rectangular area. Uses the "coordorder" parameter for loading the API — when the value is "latlong", in an array of point coordinates y should be in the first position, and x in the second.

Constructor | Methods

Constructor

```
projection.Cartesian(bounds[, cycled[, scale]])
```

Creates a projection of a rectangular coordinate area in pixels. The size of the area in pixels is always NxN, where N = 256 * 2^zoom.

Parameters:

Parameter	Default value	Description
bounds *	—	Type: Number[][] Array of two points, the coordinates of the lower-left and upper-right corners of a rectangular coordinate area.
cycled	[false, false]	Type: Boolean[] Array of indicators of map cycling on x and y.
scale	1	Type: Number|Number[] The increment of a division on an axis. Can be a number or pair of numbers for each of the axes.

* Mandatory parameter/option.

Methods

Name	Returns	Description
fromGlobalPixels(globalPixelPoint, zoom)	Number[]	Converts pixel coordinates to the projection's coordinates at the specified zoom level. Inherited from **IProjection**.
getCoordSystem()	**ICoordSystem**	Returns the coordinate system that is used by the projection. Inherited from **IProjection**.
isCycled()	Boolean[]	Indicator of projection cycling. Inherited from **IProjection**.
toGlobalPixels(coordPoint, zoom)	Number[]	Converts projection coordinates to global pixel coordinates at the specified zoom level. Inherited from **IProjection**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

projection.sphericalMercator

Static object.

Instance of **IProjection**

Mercator projection on a sphere. It is used by many cartographic services, including OpenStreetMap.

Methods

Example:

```
// Creating the map in the spherical Mercator projection
var map = new ymaps.Map('YMapsID', {
    center: [55, 37],
    zoom: 6
}, {
    projection: ymaps.projection.sphericalMercator
});
map.layers.add(new ymaps.Layer('http://tile.openstreetmap.org/%z/%x/%y.png'));
```

Methods

Name	Returns	Description
fromGlobalPixels(globalPixelPoint, zoom)	Number[]	Converts pixel coordinates to the projection's coordinates at the specified zoom level.
getCoordSystem()	**ICoordSystem**	Returns the coordinate system that is used by the projection.
isCycled()	Boolean[]	Indicator of projection cycling.
toGlobalPixels(coordPoint, zoom)	Number[]	Converts projection coordinates to global pixel coordinates at the specified zoom level.

projection.wgs84Mercator

Static object.

Instance of **IProjection**

Mercator projection on a WGS84 reference ellipsoid. Used by Yandex.Maps by default.

Methods

Methods

Name	Returns	Description
fromGlobalPixels(globalPixelPoint, zoom)	Number[]	Converts pixel coordinates to the projection's coordinates at the specified zoom level.
getCoordSystem()	**ICoordSystem**	Returns the coordinate system that is used by the projection.
isCycled()	Boolean[]	Indicator of projection cycling.
toGlobalPixels(coordPoint, zoom)	Number[]	Converts projection coordinates to global pixel coordinates at the specified zoom level.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

ready

Static function.

Performs the passed function when the API and DOM are ready for use.

Returns the promise object that is verified by the API namespace, or rejected if an error occurred when loading.

```
{ vow.Promise } ready([successCallback[, errorCallback[, context]]])
```

Parameters:

Parameter	Default value	Description
successCallback	—	Type: Function|Object Function that will be called after successfully loading and initializing the API and DOM, or an object with parameters if the extended syntax is used. Available parameters: • require — Array of additional modules that must be loaded with the API. • successCallback — Function that will be called when the API is loaded successfully. • errorCallback — Function that will be called if an error occurs. • context — Execution context for functions. All parameters are optional. The API namespace will be passed to successCallback.
errorCallback	—	Type: Function The function that will be called if an error occurred during initialization. The error will be passed to the function.
context	—	Type: Object Context for the function.

Examples:

1.

```
<!DOCTYPE html>
<html>
<head>
    <title>Example</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <script src="http://api-maps.yandex.ru/2.1/?lang=ru_RU" type="text/
javascript"></script>
    <script type="text/javascript">
    ymaps.ready(function () {
        var map = new ymaps.Map('map', {
            center: [55.7, 37.6],
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
            zoom: 10
        });
        // ...
    });
    </script>
</head>
<body>
    <div id="map" style="width: 500px; height: 500px;"></div>
</body>
</html>
```

2.

```
// Example using the extended syntax.
ymaps.ready({
    // successCallback will be called when the API and the "myModule1" module are
loaded.
    require: ['myModule1'],
    successCallback: function (ym) {
        var map = new ymaps.Map('map', {
            center: [55.7, 37.6],
            zoom: 10
        });
        var obj = new ymaps.myModule1();
        // ...
    }
})
```

Rectangle

Extends **GeoObject**.

Rectangle. A geo object with the geometry **geometry.Rectangle**.

See GeoObject geometry.Rectangle

Constructor | Fields | Events | Methods

Constructor

```
Rectangle(geometry[, properties[, options]])
```

Creates a rectangle instance.

Parameters:

Parameter	Default value	Description
geometry *	—	Type: Number[][]|Object|**IRectangleGeometry** Coordinates of two opposite corners, a hash object with geometery parameters, or a reference to the rectangle geometry object.
properties	—	Type: Object|**IDataManager**

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Rectangle data. Can be set as a class instance implementing the **IDataManager** interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object: • hintContent — Content of the rectangle's popup hint. • balloonContent — Content of the rectangle's balloon. • balloonContentHeader — Content of the rectangle balloon title. • balloonContentBody — Content of the main part of the rectangle's balloon. • balloonContentFooter — Content of the lower part of the rectangle's balloon. The balloonContent field is a shortcut for the balloonContentBody field, but if they are both set simultaneously, balloonContentBody takes priority. You can also add your own custom fields to the rectangle data and use them, for example, in the balloon layout.
`options`	—	Type: Object Rectangle options. Using this parameter, you can set options for the rectangle itself, as well as for its parts: • Options for the rectangle's balloon with the `balloon` prefix. • Options for the rectangle's popup hint with the `hint` prefix. • Geometry options can be set without a prefix. See the description of the **IGeometry** class for the **geometry.Rectangle** geometry.
`options.cursor`	"pointer"	Type: String Type of cursor over a rectangle.
`options.draggable`	false	Type: Boolean Checks whether the rectangle can be dragged.
`options.fill`	true	Type: Boolean Whether the shape is filled.
`options.fillColor`	"0066ff99"	Type: String Fill color.
`options.fillImageHref`	—	Type: String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Background image. When this option is enabled in **stretch** mode, the "fillColor" value is ignored.
options.fillMethod	'stretch'	Type: String Type of background fill. Accepts one of two values: • stretch — The background image stretches to fit the size of the overlay. • tile — The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.
options.fillOpacity	1	Type: Number Fill transparency.
options.hasBalloon	true	Type: Boolean Checks if the rectangle has the "balloon" field.
options.hasHint	true	Type: Boolean Checks if the rectangle has the "hint" field.
options.interactiveZIndex	false	Type: Boolean Enables automatically modifying the z-index of the rectangle depending on its state.
options.interactivityModel	"default#geoObject"	Type: String Interactivity model. Available keys and their values are listed in the description of **interactivityModel.storage**.
options.opacity	1	Type: Number Transparency.
options.openBalloonOnClick	true	Type: Boolean Checks whether to show the balloon when the rectangle is clicked on.
options.openEmptyBalloon	false	Type: Boolean Checks whether to show an empty balloon when the rectangle is clicked on.
options.openEmptyHint	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Checks whether to show an empty hint when the mouse pointer hovers over the rectangle.
options.openHintOnHover	true	Type: Boolean Checks whether to show an empty hint when the mouse pointer hovers over the rectangle.
options.outline	true	Type: Boolean Whether the shape has an outline.
options.pane	"places"	Type: String The key of the pane where the rectangle overlay is placed.
options.rectangleOverlay	"default#rectangle"	Type: String|Function Key identifier from **overlay.storage** or the overlay class. The generator function accepts three parameters: • geometry: **IPixelCircleGeometry** — The pixel geometry itself. • data: **IDataManager** or Object — Overlay data. • options: Object — The overlay options. And returns **vow.Promise**.
options.strokeColor	"0066ffff"	Type: String|String[] Color of the line or outline. You can set multiple values for a multistroke outline.
options.strokeOpacity	1	Type: Number|Number[] Transparency of the line or outline. You can set multiple values for a multistroke outline.
options.strokeStyle	—	Type: String|Object|String[]|Object[] Style of the line or outline. You can set multiple values for a multistroke outline.
options.strokeWidth	1	Type: Number|Number[] Thickness of the line or outline. You can set multiple values for a multistroke outline.
options.syncOverlayInit	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a rectangle to the map.
options.useMapMarginInDragging	true	Type: Boolean When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with **map.margin.Manager**.
options.visible	true	Type: Boolean Checks the visibility of the rectangle.
options.zIndex	—	Type: Number The z-index of the rectangle in the normal state. Lowest priority.
options.zIndexActive	—	Type: Number The z-index of the rectangle with an opened balloon. Highest priority.
options.zIndexDrag	—	Type: Number The z-index of the rectangle while dragging.
options.zIndexHover	—	Type: Number The z-index of a rectangle when the mouse pointer is hovering over it.

* Mandatory parameter/option.

Example:

```
// Creating a geodesic circle with a radius of 1000 kilometers.
var circle = new ymaps.Circle([[50, 50], 1000000], {}, {
    draggable: true
});
// Adding the circle to the map.
myMap.geoObjects.add(circle);

// Creating a rectangle based on the rectangle's boundaries.
var rectangle = new ymaps.Rectangle(circle.geometry.getBounds(), {}, {
    fill: false,
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
    coordRendering: "boundsPath",
    strokeWidth: 4
});
// Adding the rectangle to the map.
myMap.geoObjects.add(rectangle);

// Updating the rectangle's coordinates when changing the rectangle geometry.
circle.geometry.events.add("change", function (event) {
    this.geometry.setCoordinates(event.get("target").getBounds());
}, rectangle);
```

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object. Inherited from **GeoObject**.
editor	null	An editor for the "Rectangle" geometry has not yet been implemented.
events	**event.Manager**	Event manager. Inherited from **GeoObject**.
geometry	**geometry.Rectangle**	The "Rectangle" type of geometry.
hint	**geoObject.Hint**	Geo object hint. Inherited from **GeoObject**.
options	**option.Manager**	Geo object options manager. Inherited from **GeoObject**.
properties	**data.Manager**	Geo object data manager. Inherited from **GeoObject**.
state	**data.Manager**	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged Inherited from **GeoObject**.

Events

Name	Description
balloonclose	Closing the balloon. Instance of the **Event** class. Inherited from **GeoObject**.
balloonopen	Opening a balloon on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
beforedrag	Event preceding the "drag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent drag event will be canceled. Inherited from **GeoObject**.
beforedragstart	Event preceding the "dragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from **GeoObject**.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
drag	Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragend	End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **GeoObject**.
dragstart	Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent — Original event of the geometry editor. Inherited from **GeoObject**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
hintclose	Closing the hint. Instance of the **Event** class. Inherited from **GeoObject**.
hintopen	Opening a hint on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

editor

```
{null} editor
```

An editor for the "Rectangle" geometry has not yet been implemented.

geometry

```
{geometry.Rectangle} geometry
```

The "Rectangle" type of geometry.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

regions

regions.load

Attention!
This function is deprecated. Use **borders.load**.

Static function.

Provides access to the geometry of various regions and countries.

Returns Promise object.

```
{ vow.Promise } regions.load(region[, options])
```

Parameters:

Parameter	Default value	Description
region*	—	Type: String The ISO_3166-1 country code (RU, UA, BY, KZ) for loading the regional area, or '001' for loading the geometry of country borders.
options	—	Type: Object Display options.
options.disputedBorders	—	Type: String Two-letter code of the country to use as the official reference for determining the administrative subordination of disputed territories. Accepted values: 'RU', 'UA', 'BY', 'KZ'. By default, it coincides with the country code that is specified when loading the API. Unsupported country codes are reset to RU. For the region '001' (borders of countries), the code 'UN' is supported — world borders according to the United Nations.
options.lang	—	Type: String Language (ru, uk, en, be).
options.quality	1	Type: Number Quality level. Available values: • 0 — minimal quality • 1 — standard quality • 2 — improved quality • 3 — high quality

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The quality level affects how accurately curves are represented, as well as the volume of the data file.

* Mandatory parameter/option.

Example:

```
ymaps.regions.load('RU', {
    lang: 'en'
}).then(function (result) {
    geoMap.geoObjects.add(result.geoObjects);
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

RemoteObjectManager

Extends **ICustomizable**, **IEventEmitter**, **IGeoObject**, **IParentOnMap**.

The object manager that optimizes downloading of objects from the server. The manager sends data requests to the specified url in the JSONP format. This format corresponds to the format of objects added to ObjectManager, **ObjectManager.add**. Objects of type "Cluster" with the following fields are also supported:

- type — The object type, always "Cluster" for clusters.
- id — Unique cluster ID.
- geometry — The cluster geometry in JSON format.
- features — An array of cluster objects. Optional field.
- bbox — An array of coordinates describing a rectangular area that includes all the cluster objects.
- number — Number of objects in the cluster.
- properties — Cluster data.

This module is designed for downloading and displaying the data which was previously processed on the server. Particularly, it is recommended to use the module for displaying results of server-side clustering. Data is requested again if the map zoom level is changed. The module does not provide clustering or filtering possibilities for managing visibility on the client side. To cluster objects on the client side after loading, use **LoadingObjectManager**. Note that objects drawn on the map via this manager can't have editing and dragging modes enabled.

Constructor | Fields | Events | Methods

Constructor

```
RemoteObjectManager(urlTemplate[, options])
```

Parameters:

Parameter	Default value	Description
urlTemplate *	—	Type: String URL data template. Supports special constructions similar to **Layer**. Substitutions are also supported: • %b is replaced by an array of geographic coordinates that describes the rectangular region for which you want to load data. • %t is replaced by an array of tile numbers that describes the rectangular region to load data for.
options	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Options. • You can set all the options specified in the **Clusterer** description, except for hasBalloon and hasHint options. • Cluster options are set with the "cluster" prefix. The list of options is specified in the description of **ClusterPlacemark**; • Options for singular objects should be specified with the geoObject prefix. The list of options is specified in **GeoObject**.
options.loadTileSize	256	Type: Number Tile size for data loading.
options.paddingParamName	'callback'	Type: Boolean Name of the GET parameter that contains the value of the JSONP callback.
options.paddingTemplate	null	Type: String Template for a jsonp callback. Supports the same substitutions as urlTemplate. All characters other than letters and numbers will be replaced with '_'. If the parameter is omitted, the name of the jsonp callback will be generated automatically. Conversion examples for tileNumber=[3, 1], zoom=9: • 'myCallback=%x' => 'myCallback_3' • '%c' => 'x_3_y_1_z_9' • 'callback2_%c' => 'callback2_x_3_y_1_z_9' • 'callback%test' => 'callback_test' • 'callback_%b' => 'callback_85_0841__180_0000_85_0841_180_0000' Note that if substitution options are not used in the value, this may lead to an error. All requests will go to the same callback function.
options.splitRequests	false	Type: Boolean Divide requests for data into requests for individual tiles. By default, requests are made for data for a rectangular region that contains multiple tiles.
options.syncOverlayInit	false	Type: Boolean

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		A flag that allows creating overlays for objects synchronously. Note that when you create an overlay synchronously, you should ensure that the appropriate class, which implements the IOverlay interface, is loaded. By default, the overlays are created asynchronously, and the overlay class is loaded on demand.

* Mandatory parameter/option.

Examples:

1.

```
var objectManager = new ymaps.RemoteObjectManager('http://myServer.com/tile?bbox=
%b', {
    // Cluster options are set with the "cluster" prefix.
    clusterHasBalloon: false,
    // Geo object options are set with the "geoObject" prefix.
    geoObjectOpenBalloonOnClick: false
});

// You can set options directly for child collections.
objectManager.clusters.set({
    preset: 'islands#grayClusterIcons',
    hintContentLayout: ymaps.templateLayoutFactory.createClass('Group of
objects')
});
objectManager.objects.set('preset', 'islands#grayIcon');
```

2.

```
An example of RemoteObjectManager response
jsonp_callback({
    // The response contains the error and data fields. If an error occurs, the
"error" field
    // contains the error code or description.
    error: null,
    data: {
        type: 'FeatureCollection',
        features: [
             {
                 type: 'Feature',
                 geometry: {
                     type: 'Point',
                     coordinates: [55, 35]
                 },
                 id: 23,
                 properties: {
                     balloonContent: 'Placemark balloon content',
                     iconContent: 'Placemark content'
                 },
                 options: {
                     preset: 'islands#yellowIcon'
                 }
             },
             {
                 type: 'Cluster',
                 id: 24,
                 bbox: [[35, 46], [46, 57]],
                 number: 34,
                 // Array describing the 34 objects in the cluster.
                 // Optional field.
                 // If omitted, an empty balloon opens when the cluster is
clicked.
                 features: [{
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
                    type: 'Feature',
                    id: 512,
                    properties: {
                        balloonContent: 'Placemark balloon content',
                        clusterCaption: 'Placemark title in the cluster
balloon'
                    },
                    ...
                ],
                geometry: {
                    type: 'Point',
                    coordinates: [40.5, 51]
                },
                properties: {
                    iconContent: 34
                }
            }
        ]
    }
});
```

Fields

Name	Type	Description
clusters	**objectManager.ClusterCollection**	Collection of clusters generated by the manager.
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
objects	**objectManager.ObjectCollection**	Collection of objects added to the layer.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**IDataManager**	Geo object data. Inherited from **IGeoObject**.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Events

Name	Description
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dataloaderror	An error occured when loading data. Instance of the **Event** class.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object. Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getBounds()	Number[][]|null	Calculates the boundaries in geo coordinates for an area that covers all the loaded objects in the manager.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getObjectState(id)	Object	Getting information about the current state of an object added to the manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getPixelBounds()	Number[][]|null	Calculates the boundaries in global pixel coordinates for an area that covers all the loaded objects in the manager.
getTileUrl()	String|null	Returns URL of the tile with data.
getUrlTemplate()	String	Returns URL data template.
reloadData()		Method that deletes all previously loaded data and sends a request for new data.
setFilter(filterFunction)		Sets a filter function for objects. Filters both individual objects and clusters.
setParent(parent)	**IChildOnMap**	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from **IChildOnMap**.
setUrlTemplate(urlTemplate)		

Fields details

clusters

```
{objectManager.ClusterCollection} clusters
```

Collection of clusters generated by the manager.

Example:

```
objectManager.clusters.events.add('click', function (e) {
    var objectId = e.get('objectId');
    objectManager.clusters.balloon.open(objectId);
});
```

objects

```
{objectManager.ObjectCollection} objects
```

Collection of objects added to the layer.

Example:

```
objectManager.objects.events.add('click', function (e) {
    var objectId = e.get('objectId');
    objectManager.objects.balloon.open(objectId);
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

dataloaderror

An error occured when loading data. Instance of the **Event** class.

Methods details

getBounds

```
{Number[][]|null} getBounds()
```

Calculates the boundaries in geo coordinates for an area that covers all the loaded objects in the manager.

Returns array of the area's coordinates, or null if the manager has not been added to the map.

getObjectState

```
{Object} getObjectState(id)
```

Getting information about the current state of an object added to the manager.

Returns object with following fields:

- found — Attribute that indicates whether an object with the passed ID exists in the loaded data. Type: Boolean.
- isShown — Attribute that indicates whether the object is located in the visible area of the map. Type: Boolean.
- isFilteredOut — Attribute indicating whether an object passed through filtration. If the filter is not set or the object passed through filtration, the value of the field is "false". Type: Boolean.

.

Parameters:

Parameter	Default value	Description
id*	—	Type: Object ID of the object to get the state for.

* Mandatory parameter/option.

Example:

```
remoteObjectManager.setFilter('properties.type == "shop"');
...
if (!remoteObjectManager.getObjectState(7).isFilteredOut) {
    remoteObjectManager.objects.balloon.open(7);
}
```

getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Calculates the boundaries in global pixel coordinates for an area that covers all the loaded objects in the manager.

Returns array of the area's coordinates, or null if the manager has not been added to the map.

getTileUrl

```
{String|null} getTileUrl()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns URL of the tile with data.

Parameters:

Parameter	Default value	Description
parameters*	—	Type:

* Mandatory parameter/option.

Example:

```
var objectManager = new ymaps.RemoteObjectManager('http://myServer.com/tile?bbox=
%b');
objectManager.getTileUrl = function (parameters) {
   var boundingBox = parameters.boundingBox.join('~');
   return this.getUrlTemplate().replace(/%b/g, boundingBox);
};
```

getUrlTemplate

```
{String} getUrlTemplate()
```

Returns URL data template.

reloadData

```
{} reloadData()
```

Method that deletes all previously loaded data and sends a request for new data.

setFilter

```
{} setFilter(filterFunction)
```

Sets a filter function for objects. Filters both individual objects and clusters.

Parameters:

Parameter	Default value	Description
filterFunction*	—	Type: Function|String Filter function. Takes a single object added to ObjectManager. If the function returns true, the object will be processed. If false, the object will be excluded from further processing. A string can also be passed as a filter. The following keywords are available in the string filter: • options — Accessing the object's options. • properties — Accessing the object's data. • geometry — Accessing the object's geometry. • id — Accessing the object's identifier. For a filter, you can specify an expression that returns true or false.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Examples:

1.

```
// Select clusters with id > 100.
objectManager.setFilter('object.type == "Cluster" && id > 100');
```

2.

```
// Only objects of the specified types will be displayed on the map.
objectManager.setFilter('properties.type == "cafe" || properties.type ==
"pharmacy"');
```

3.

```
// You can define a filter function.
objectManager.setFilter(function (object) {
    return object.properties.name != 'The one who cannot be shown.';
});
```

setUrlTemplate

```
{} setUrlTemplate(urlTemplate)
```

Parameters:

Parameter	Default value	Description
urlTemplate*	—	Type: String URL data template.

* Mandatory parameter/option.

route

Static function.

Plots a route through the specified points.

Returns a promise object that is confirmed by the route object **router.Route**, or the multi-stop route object **multiRouter.MultiRoute**, depending on the value of the multiRoute parameter. If an error occurs, the promise object is rejected.

```
{ vow.Promise } route(points[, params])
```

Parameters:

Parameter	Default value	Description
points*	—	Type: Object[]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Array of points that the route should go through. Each point can be set by a string containing the address, an array of the coordinates, and a JSON object with the following fields: • type: String — Type of point. The 'wayPoint' value sets the route waypoint. Use the "viaPoint" value to define a throughpoint, i.e. a point that must be driven through without stopping. • point: Number[]|String — Array of the coordinates of a point, or its address as a string.
params	—	Type: Object Routing options.
params.avoidTrafficJams	false	Type: Boolean Enables constructing a route with consideration for traffic. When using the options, keep in mind that it is not always possible to detour around traffic jams.
params.boundedBy	—	Type: Number[][] Area on the map where the objects being searched for are presumably located. This is used if the route points are set using the mailing address instead of coordinates.
params.mapStateAutoApply	false	Type: Boolean Flag that allows to automatically set the center and map zoom so that the route will be entirely visible.
params.multiRoute	false	Type: Boolean Allows to construct multi-stop routes.
params.reverseGeocoding	false	Type: Boolean Whether to use reverse geocoding for points specified as coordinates.
params.routingMode	"auto"	Type: String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Routing type. Accepts one of three string values: • "auto" — Driving route. • "masstransit" — Routing using public transit. Only for multi-stop routes (the multiRoute option must be set to true). • "pedestrian" — Walking route. Only for multi-stop routes (the multiRoute option must be set to true).
params.searchCoordOrder	—	Type: String Defines how to interpret the coordinates in the request. This is used if the route points are set using coordinates, and not the mailing address.
params.strictBounds	false	Type: Boolean Search only inside the area defined by the "boundedBy" option.
params.useMapMargin	true	Type: Boolean Whether to account for map margins **map.margin.Manager**.
params.viaIndexes	[]	Type: Integer[] Indexes of the multi-stop route throughpoints.
params.zoomMargin	0	Type: Number|Number[] Offset from the map viewport borders when changing the zoom level. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" parameter is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager **map.margin.Manager**.

* Mandatory parameter/option.

Examples:

1.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Building the route from Korolev to Krasnogorsk via Khimki and Mytischi,
// where Mytischi is a throughpoint. Setting coordinates for Krasnogorsk.
ymaps.route([
    'Korolev',
    { type: 'viaPoint', point: 'Mytischi' },
    'Himki',
    { type: 'wayPoint', point: [55.811511, 37.312518] }
], {
    mapStateAutoApply: true
}).then(function (route) {
    route.getPaths().options.set({
        // the balloon only shows information about the travel time with traffic
        balloonContentLayout:
ymaps.templateLayoutFactory.createClass('{{ properties.humanJamsTime }}'),
        // you can make settings for route graphics
        strokeColor: '0000ffff',
        opacity: 0.9
    });
    // adding the route to the map
    map.geoObjects.add(route);
});
```

2.

```
// Building a multi-stop route and adding it to the map using autoscaling.
ymaps.route(['Southern Butovo', 'Moscow, metro Park Kultury'], {
    multiRoute: true
}).done(function (route) {
    route.options.set("mapStateAutoApply", true);
    myMap.geoObjects.add(route);
}, function (err) {
    throw err;
}, this);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

router

router.addon

router.addon.editor

router.addon.editor.get

Static function.

Returns router editor.

```
{ router.Route } router.addon.editor.get()
```

Example:

```
ymaps.router.addon.editor.get(myMap)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

router.Editor

Note:
The constructor of the router.Editor class is hidden, as this class is not intended for autonomous initialization.

Extends **ICustomizable**, **IEventEmitter**.

Route editor. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Events | Methods

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**IDataManager**	The state manager of the route editor. Data fields that are available via the "get" and "set" methods: • routeloading: Boolean — Flag for whether the data is currently being downloaded from the routing service. • waypointsdrag: Boolean — Flag for whether the waypoint is currently being dragged. • viapointsdrag: Boolean — Flag for whether the throughpoint is currently being dragged.

Events

Name	Description
optionschange	Change to the object options. Inherited from **ICustomizable**.
routeupdate	Updating the route. Using the value of the e.get('rough') flag, you can determine whether the event was thrown after editing was completed or during it. If you want your application to update information associated with the route, you need to make a check for e.get('rough') == false in order to avoid handling this event too often.
start	Enabling the editor.
stop	Disabling the editor.
viapointadd	Adding a throughpoint. Use e.get('viaPoint') to get a throughpoint to be added.
viapointdragend	End of throughpoint dragging. Use e.get('viaPoint') to get a throughpoint.
viapointdragstart	Start of throughpoint dragging. Use e.get('viaPoint') to get a throughpoint.
viapointremove	Deleting a throughpoint. Use e.get('viaPoint') to get a throughpoint to be deleted.
waypointadd	Adding a waypoint. Use e.get('wayPoint') to get a waypoint to be added.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
waypointdragend	End of waypoint dragging. Use e.get('wayPoint') to get a waypoint.
waypointdragstart	Start of waypoint dragging. Use e.get('wayPoint') to get a waypoint.
waypointremove	Deleting a waypoint. Use e.get('wayPoint') to get a waypoint to be deleted.

Methods

Name	Description
start([options])	Enables the route editor.
stop()	Disables the route editor.

Fields details

state

```
{IDataManager} state
```

The state manager of the route editor.

Data fields that are available via the "get" and "set" methods:

- routeloading: Boolean — Flag for whether the data is currently being downloaded from the routing service.
- waypointsdrag: Boolean — Flag for whether the waypoint is currently being dragged.
- viapointsdrag: Boolean — Flag for whether the throughpoint is currently being dragged.

Events details

routeupdate

Updating the route. Using the value of the e.get('rough') flag, you can determine whether the event was thrown after editing was completed or during it. If you want your application to update information associated with the route, you need to make a check for e.get('rough') == false in order to avoid handling this event too often.

start

Enabling the editor.

stop

Disabling the editor.

viapointadd

Adding a throughpoint. Use e.get('viaPoint') to get a throughpoint to be added.

viapointdragend

End of throughpoint dragging. Use e.get('viaPoint') to get a throughpoint.

viapointdragstart

Start of throughpoint dragging. Use e.get('viaPoint') to get a throughpoint.

viapointremove

Deleting a throughpoint. Use e.get('viaPoint') to get a throughpoint to be deleted.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

waypointadd

Adding a waypoint. Use e.get('wayPoint') to get a waypoint to be added.

waypointdragend

End of waypoint dragging. Use e.get('wayPoint') to get a waypoint.

waypointdragstart

Start of waypoint dragging. Use e.get('wayPoint') to get a waypoint.

waypointremove

Deleting a waypoint. Use e.get('wayPoint') to get a waypoint to be deleted.

Methods details

start

```
{} start([options])
```

Enables the route editor.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Options.
options.addViaPoints	true	Type: Boolean If true, adding throughpoints is allowed; if false, it is prohibited.
options.addWayPoints	false	Type: Boolean If true, waypoints can be added when clicking on the map; if false, this is prohibited.
options.editViaPoints	true	Type: Boolean If true, editing (moving) throughpoints is allowed; if false, it is prohibited.
options.editWayPoints	true	Type: Boolean If true, editing (moving) waypoints is allowed; if false, it is prohibited.
options.removeViaPoints	true	Type: Boolean If true, deleting throughpoints by double-click is allowed; if false, it is prohibited.
options.removeWayPoints	false	Type: Boolean If true, deleting waypoints by double-click is allowed; if false, it is prohibited.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

stop

```
{} stop()
```

Disables the route editor.

router.Path

Note:
The constructor of the router.Path class is hidden, as this class is not intended for autonomous initialization.

Extends **GeoObject**.

Object that describes part of the route (a path). The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand. The route can contain several paths, and each path connects two waypoints.

See route

Fields | Events | Methods

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object. Inherited from **GeoObject**.
editor	**IGeometryEditor**	Editor for the geo object geometry. Inherited from **GeoObject**.
events	**event.Manager**	Event manager. Inherited from **GeoObject**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **GeoObject**.
hint	**geoObject.Hint**	Geo object hint. Inherited from **GeoObject**.
options	**option.Manager**	Geo object options manager. Inherited from **GeoObject**.
properties	**data.Manager**	Geo object data manager. Inherited from **GeoObject**.
state	**data.Manager**	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged Inherited from **GeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events

Name	Description
balloonclose	Closing the balloon. Instance of the **Event** class. Inherited from **GeoObject**.
balloonopen	Opening a balloon on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
beforedrag	Event preceding the "drag" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Names of methods that are accessible via **Event.callMethod**: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the **Event.preventDefault** method is called for this event, a subsequent drag event will be canceled. Inherited from **GeoObject**.
beforedragstart	Event preceding the "dragstart" event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. If the **Event.preventDefault** method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from **GeoObject**.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
drag	Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragend	End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragstart	Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent — Original event of the geometry editor. Inherited from **GeoObject**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
hintclose	Closing the hint. Instance of the **Event** class. Inherited from **GeoObject**.
hintopen	Opening a hint on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.
overlaychange	Change to the geo object overlay. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • overlay: **IOverlay**|null — Reference to the overlay. • oldOverlay: **IOverlay**|null — Previous overlay of the geo object.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IGeoObject**.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from **IChild**.
propertieschange	Change to the geo object data. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the data manager. Inherited from **IGeoObject**.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Methods

Name	Returns	Description
getHumanJamsTime()	String	Returns a string representation of path driving time with measurement units, with consideration for traffic.
getHumanLength()	String	Returns a string representation of path length with measurement units.
getHumanTime()	String	Returns a string representation of path driving time with measurement units.
getJamsTime()	Integer	Returns the path driving time in seconds, with consideration for traffic.
getLength()	Number	Returns path length in meters.
getMap()	**Map**	Returns reference to the map. Inherited from **IParentOnMap**.
getOverlay()	**vow.Promise**	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from **IGeoObject**.
getOverlaySync()	**IOverlay**|null	The method provides synchronous access to the overlay. Inherited from **IGeoObject**.
getParent()	**IParentOnMap**|null	Returns link to the parent object, or null if the parent element was not set. Inherited from **IChildOnMap**.
getSegments()	**router.Segment**[]	Returns path segments.
getTime()	Integer	Returns the path driving time in seconds.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Methods details

getHumanJamsTime

```
{String} getHumanJamsTime()
```

Returns a string representation of path driving time with measurement units, with consideration for traffic.

getHumanLength

```
{String} getHumanLength()
```

Returns a string representation of path length with measurement units.

getHumanTime

```
{String} getHumanTime()
```

Returns a string representation of path driving time with measurement units.

getJamsTime

```
{Integer} getJamsTime()
```

Returns the path driving time in seconds, with consideration for traffic.

getLength

```
{Number} getLength()
```

Returns path length in meters.

getSegments

```
{router.Segment[]} getSegments()
```

Returns path segments.

getTime

```
{Integer} getTime()
```

Returns the path driving time in seconds.

router.Route

Note:
The constructor of the router.Route class is hidden, as this class is not intended for autonomous initialization.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Extends **IGeoObject**.

Object that describes the plotted route. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

See route

Fields | Events | Methods

Fields

Name	Type	Description
editor	**router.Editor**	Route editor.
events	**IEventManager**	Event manager. Inherited from **IDomEventEmitter**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **IGeoObject**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
properties	**IDataManager**	Geo object data. Inherited from **IGeoObject**.
state	**IDataManager**	State of the geo object. Inherited from **IGeoObject**.

Events

Name	Description
boundsapply	Event for applying the route boundaries to the map with the options.mapStateAutoApply option set.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from IDomEventEmitter.
optionschange	Change to the object options. Inherited from ICustomizable.
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
update	Route updating event when the route editor is enabled.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getHumanJamsTime()	String	Returns a string representation of route driving time with measurement units, with consideration for traffic.
getHumanLength()	String	Returns a string representation of route length with measurement units.
getHumanTime()	String	Returns a string representation of route driving time with measurement units.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getJamsTime()	Integer	Returns the route driving time in seconds, with consideration for traffic.
getLength()	Number	Returns the length of the route in meters.
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getPaths()	GeoObjectCollection	Returns a collection of paths that make up the route.
getTime()	Integer	Returns the route driving time in seconds.
getViaPoints()	GeoObjectCollection	Returns a collection of throughpoints on the route.
getWayPoints()	GeoObjectCollection	Returns a collection of waypoints on the route.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Fields details

editor

```
{router.Editor} editor
```

Route editor.

Example:

```
// Start of route editing.
ymaps.route(['Moscow', 'Petersburg'], function (route) {
    route.editor.start();
    // ...
    // End of route editing.
    route.editor.stop();
});
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Events details

boundsapply

Event for applying the route boundaries to the map with the options.mapStateAutoApply option set.

update

Route updating event when the route editor is enabled.

Methods details

getHumanJamsTime

```
{String} getHumanJamsTime()
```

Returns a string representation of route driving time with measurement units, with consideration for traffic.

getHumanLength

```
{String} getHumanLength()
```

Returns a string representation of route length with measurement units.

getHumanTime

```
{String} getHumanTime()
```

Returns a string representation of route driving time with measurement units.

getJamsTime

```
{Integer} getJamsTime()
```

Returns the route driving time in seconds, with consideration for traffic.

getLength

```
{Number} getLength()
```

Returns the length of the route in meters.

getPaths

```
{GeoObjectCollection} getPaths()
```

Returns a collection of paths that make up the route.

getTime

```
{Integer} getTime()
```

Returns the route driving time in seconds.

getViaPoints

```
{GeoObjectCollection} getViaPoints()
```

Returns a collection of throughpoints on the route.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getWayPoints

```
{GeoObjectCollection} getWayPoints()
```

Returns a collection of waypoints on the route.

router.Segment

Note:
The constructor of the router.Segment class is hidden, as this class is not intended for autonomous initialization.

Object that describes a segment of the route. A segment is a part of the route from one maneuver to the next. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Methods

Methods

Name	Returns	Description
getAction()	String	Returns the direction the route turns at the end of the segment. Possible values: • left — Left. • slight left — Slightly left. • hard left — Sharp left turn. • right — Right. • slight right — Slightly right. • hard right — Sharp right turn. • none — Straight. • back — Turn around. • enter roundabout — Entrance to a roundabout intersection. • leave roundabout [N] — Exit from a roundabout intersection. The number N is the number of the turn on the circle. This number can be omitted. For example, "leave roundabout" or "leave roundabout 2". • merge — Entrance to a highway. Signifies merging with traffic. • board ferry — Ferry crossing. .
getAngle()	Number	Checks the angle at which the route turns at the end of the segment.
getCoordinates()	Number[][]	Returns coordinates of a polyline that describes the segment geometry.
getHumanAction()	String	Returns the direction of the turn in the form of a localized human-readable string.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getHumanJamsTime()	String	Returns a string representation of segment driving time with measurement units, with consideration for traffic.
getHumanLength()	String	Returns a string representation of segment length with measurement units.
getHumanTime()	String	Returns a string representation of segment driving time with measurement units.
getIndex()	Integer	Returns the index of the given segment in the array of all the segments in the path.
getJamsTime()	Integer	Returns the segment driving time in seconds, with consideration for traffic.
getLength()	Number	Returns the length of the segment in meters.
getPolylineEndIndex()	Integer	Returns the index of the point in the path geometry at which the segment ends.
getPolylineStartIndex()	Integer	Returns the index of the point in the path geometry from which the segment begins.
getStreet()	String	Returns the name of the street that the segment of the route goes along.
getTime()	Integer	Returns the segment driving time in seconds.

Methods details

getAction

```
{String} getAction()
```

Returns the direction the route turns at the end of the segment. Possible values:

- left — Left.
- slight left — Slightly left.
- hard left — Sharp left turn.
- right — Right.
- slight right — Slightly right.
- hard right — Sharp right turn.
- none — Straight.
- back — Turn around.
- enter roundabout — Entrance to a roundabout intersection.
- leave roundabout [N] — Exit from a roundabout intersection. The number N is the number of the turn on the circle. This number can be omitted. For example, "leave roundabout" or "leave roundabout 2".
- merge — Entrance to a highway. Signifies merging with traffic.
- board ferry — Ferry crossing.

.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getAngle

```
{Number} getAngle()
```

Checks the angle at which the route turns at the end of the segment.

Returns the angle of the turn (in degrees).

getCoordinates

```
{Number[][]} getCoordinates()
```

Returns coordinates of a polyline that describes the segment geometry.

getHumanAction

```
{String} getHumanAction()
```

Returns the direction of the turn in the form of a localized human-readable string.

getHumanJamsTime

```
{String} getHumanJamsTime()
```

Returns a string representation of segment driving time with measurement units, with consideration for traffic.

getHumanLength

```
{String} getHumanLength()
```

Returns a string representation of segment length with measurement units.

getHumanTime

```
{String} getHumanTime()
```

Returns a string representation of segment driving time with measurement units.

getIndex

```
{Integer} getIndex()
```

Returns the index of the given segment in the array of all the segments in the path.

getJamsTime

```
{Integer} getJamsTime()
```

Returns the segment driving time in seconds, with consideration for traffic.

getLength

```
{Number} getLength()
```

Returns the length of the segment in meters.

getPolylineEndIndex

```
{Integer} getPolylineEndIndex()
```

Returns the index of the point in the path geometry at which the segment ends.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

getPolylineStartIndex

```
{Integer} getPolylineStartIndex()
```

Returns the index of the point in the path geometry from which the segment begins.

getStreet

```
{String} getStreet()
```

Returns the name of the street that the segment of the route goes along.

getTime

```
{Integer} getTime()
```

Returns the segment driving time in seconds.

router.ViaPoint

Note:
The constructor of the router.ViaPoint class is hidden, as this class is not intended for autonomous initialization.

Extends **GeoObject**.

Object that describes a throughpoint on the route. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

Fields | Events | Methods

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object. Inherited from **GeoObject**.
editor	**IGeometryEditor**	Editor for the geo object geometry. Inherited from **GeoObject**.
events	**event.Manager**	Event manager. Inherited from **GeoObject**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **GeoObject**.
hint	**geoObject.Hint**	Geo object hint. Inherited from **GeoObject**.
options	**option.Manager**	Geo object options manager. Inherited from **GeoObject**.
properties	**data.Manager**	Geo object data manager. Inherited from **GeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
state	data.Manager	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged Inherited from GeoObject.

Events

Name	Description
balloonclose	Closing the balloon. Instance of the Event class. Inherited from GeoObject.
balloonopen	Opening a balloon on a geo object. Instance of the Event class. Inherited from GeoObject.
beforedrag	Event preceding the "drag" event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a DomEvent object), if there is one. Names of methods that are accessible via Event.callMethod: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the Event.preventDefault method is called for this event, a subsequent drag event will be canceled. Inherited from GeoObject.
beforedragstart	Event preceding the "dragstart" event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a DomEvent object), if there is one. If the Event.preventDefault method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from GeoObject.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
	Inherited from **IDomEventEmitter**.
drag	Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragend	End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragstart	Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent — Original event of the geometry editor. Inherited from **GeoObject**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
hintclose	Closing the hint. Instance of the **Event** class. Inherited from **GeoObject**.
hintopen	Opening a hint on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
getPathIndex()	Integer	Returns the index of the path that the point is on.
getSegmentIndex()	Integer	Returns the index of the segment of the path where the point is located.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

getPathIndex

```
{Integer} getPathIndex()
```

Returns the index of the path that the point is on.

getSegmentIndex

```
{Integer} getSegmentIndex()
```

Returns the index of the segment of the path where the point is located.

router.WayPoint

Extends **GeoObject**.

Object describing a waypoint on the route.

Constructor | Fields | Events | Methods

Constructor

```
router.WayPoint(feature[, options])
```

Parameters:

Parameter	Default value	Description
feature *	—	Type: Object Properties and geometry.
options	—	Type: Object Options.

* Mandatory parameter/option.

Fields

Name	Type	Description
balloon	**geoObject.Balloon**	Balloon for a geo object. Inherited from **GeoObject**.
editor	**IGeometryEditor**	Editor for the geo object geometry. Inherited from **GeoObject**.
events	**event.Manager**	Event manager. Inherited from **GeoObject**.
geometry	**IGeometry**|null	Geo object geometry. Inherited from **GeoObject**.
hint	**geoObject.Hint**	Geo object hint. Inherited from **GeoObject**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
options	option.Manager	Geo object options manager. Inherited from GeoObject.
properties	data.Manager	Data manager for a waypoint. If the waypoint was set as an address, the GeocoderMetaData field will contain the geocoder metadata. See geocode.
state	data.Manager	State of the geo object. Defined by the following fields: • active: Boolean — Indicates that a balloon is open on the geo object. • hover: Boolean — Indicates that the mouse is currently pointed at the geo object. • drag: Boolean — Indicates that the geo object is being dragged Inherited from GeoObject.

Events

Name	Description
balloonclose	Closing the balloon. Instance of the Event class. Inherited from GeoObject.
balloonopen	Opening a balloon on a geo object. Instance of the Event class. Inherited from GeoObject.
beforedrag	Event preceding the "drag" event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a DomEvent object), if there is one. Names of methods that are accessible via Event.callMethod: • setPixelOffset — This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers. If the Event.preventDefault method is called for this event, a subsequent drag event will be canceled. Inherited from GeoObject.
beforedragstart	Event preceding the "dragstart" event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a DomEvent object), if there is one. If the Event.preventDefault method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from GeoObject.
click	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
contextmenu	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
dblclick	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
drag	Dragging a geo object. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • pixelOffset — Array of two numbers that describe the pixel offset at this step. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragend	End of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
dragstart	Start of geo object dragging. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • domEvent — Source DOM event (as a **DomEvent** object), if there is one. Inherited from **GeoObject**.
editorstatechange	Change in the state of the editor for the geo object's geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent — Original event of the geometry editor. Inherited from **GeoObject**.
geometrychange	Change to the geo object geometry. Instance of the **Event** class. Names of fields that are available via the **Event.get** method: • originalEvent: **IEvent** — Original event of the geometry. Inherited from **IGeoObject**.
hintclose	Closing the hint. Instance of the **Event** class. Inherited from **GeoObject**.
hintopen	Opening a hint on a geo object. Instance of the **Event** class. Inherited from **GeoObject**.
mapchange	Map reference changed. Data fields: • oldMap — Old map. • newMap — New map. Inherited from **IParentOnMap**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
mousedown	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseenter	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseleave	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mousemove	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
mouseup	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in **domEvent.manager**. Inherited from **IDomEventEmitter**.
multitouchend	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface. Inherited from **IDomEventEmitter**.
multitouchmove	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.
multitouchstart	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: • clientX — X coordinate of the touch relative to the viewable area of the browser. • clientY — Y coordinate of the touch relative to the viewable area of the browser. • pageX — X coordinate of the touch relative to the beginning of the document. • pageY — Y coordinate of the touch relative to the beginning of the document. Inherited from **IDomEventEmitter**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.
overlaychange	Change to the geo object overlay. Instance of the Event class. Names of fields that are available via the Event.get method: • overlay: IOverlay|null — Reference to the overlay. • oldOverlay: IOverlay|null — Previous overlay of the geo object. Inherited from IGeoObject.
parentchange	The parent object reference changed. Data fields: • oldParent — Old parent. • newParent — New parent. Inherited from IChild.
propertieschange	Change to the geo object data. Instance of the Event class. Names of fields that are available via the Event.get method: • originalEvent: IEvent — Original event of the data manager. Inherited from IGeoObject.
wheel	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in domEvent.manager. Inherited from IDomEventEmitter.

Methods

Name	Returns	Description
getMap()	Map	Returns reference to the map. Inherited from IParentOnMap.
getOverlay()	vow.Promise	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from IGeoObject.
getOverlaySync()	IOverlay|null	The method provides synchronous access to the overlay. Inherited from IGeoObject.
getParent()	IParentOnMap|null	Returns link to the parent object, or null if the parent element was not set. Inherited from IChildOnMap.
setParent(parent)	IChildOnMap	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from IChildOnMap.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

properties

```
{data.Manager} properties
```

Data manager for a waypoint. If the waypoint was set as an address, the GeocoderMetaData field will contain the geocoder metadata. See **geocode**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

shape

shape.Circle

Extends **IShape**.

"Circle" pixel shape.

Constructor | Methods

Constructor

```
shape.Circle(pixelGeometry[, params])
```

Creates the shape.

Parameters:

Parameter	Default value	Description
pixelGeometry *	—	Type: **IPixelCircleGeometry** Pixel geometry of a shape.
params	—	Type: Object Shape parameters.
params.fill	true	Type: Boolean Flag for filling the shape.
params.outline	true	Type: Boolean Flag for an outline.
params.strokeWidth	0	Type: Number The outline width, in pixels.

* Mandatory parameter/option.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the shape. Inherited from **IShape**.
equals(shape)	Boolean	Returns true if the passed shape is equivalent to the given one. Inherited from **IShape**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IShape**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getGeometry()	**IPixelGeometry**	Returns pixel geometry of a shape. Inherited from **IShape**.
getType()	String	Returns ID of the shape type. Inherited from **IShape**.
scale(factor)	**IShape**	Creates a scaled copy of the shape. Inherited from **IShape**.
shift(offset)	**IShape**	Creates a copy of the shape that is shifted by the specified amount. Inherited from **IShape**.

shape.LineString

Extends **IShape**.

The "Polyline" pixel shape.

Constructor | Methods

Constructor

```
shape.LineString(pixelGeometry[, params])
```

Creates the shape.

Parameters:

Parameter	Default value	Description
`pixelGeometry` *	—	Type: **IPixelLineStringGeometry** Pixel geometry of a shape.
`params`	—	Type: Object Shape parameters.
`params.strokeWidth`	1	Type: Number The line width, in pixels.

* Mandatory parameter/option.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the shape. Inherited from **IShape**.
equals(shape)	Boolean	Returns true if the passed shape is equivalent to the given one. Inherited from **IShape**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IShape**.
getGeometry()	**IPixelGeometry**	Returns pixel geometry of a shape. Inherited from **IShape**.
getType()	String	Returns ID of the shape type. Inherited from **IShape**.
scale(factor)	**IShape**	Creates a scaled copy of the shape. Inherited from **IShape**.
shift(offset)	**IShape**	Creates a copy of the shape that is shifted by the specified amount. Inherited from **IShape**.

shape.MultiGeometry

Extends **IShape**.

"Multipolygon" pixel shape.

Constructor | Methods

Constructor

```
shape.MultiGeometry(pixelGeometry[, params])
```

Creates the shape.

Parameters:

Parameter	Default value	Description
`pixelGeometry` *	—	Type: IPixelMultiGeometry Pixel geometry of a shape.
`params`	—	Type: Object Shape parameters.
`params.fill`	true	Type: Boolean Flag for filling the shape.
`params.outline`	true	Type: Boolean Flag for an outline.
`params.strokeWidth`	0	Type: Number The outline width, in pixels.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the shape. Inherited from **IShape**.
equals(shape)	Boolean	Returns true if the passed shape is equivalent to the given one. Inherited from **IShape**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IShape**.
getGeometry()	**IPixelGeometry**	Returns pixel geometry of a shape. Inherited from **IShape**.
getType()	String	Returns ID of the shape type. Inherited from **IShape**.
scale(factor)	**IShape**	Creates a scaled copy of the shape. Inherited from **IShape**.
shift(offset)	**IShape**	Creates a copy of the shape that is shifted by the specified amount. Inherited from **IShape**.

shape.MultiPolygon

Extends **IShape**.

"Multipolygon" pixel shape.

Constructor | Methods

Constructor

```
shape.MultiPolygon(pixelGeometry[, params])
```

Creates the shape.

Parameters:

Parameter	Default value	Description
`pixelGeometry` *	—	Type: **IPixelMultiPolygonGeometry** Pixel geometry of a shape.
`params`	—	Type: Object Shape parameters.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
params.fill	true	Type: Boolean Flag for filling the shape.
params.outline	true	Type: Boolean Flag for an outline.
params.strokeWidth	0	Type: Number The outline width, in pixels.

* Mandatory parameter/option.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the shape. Inherited from **IShape**.
equals(shape)	Boolean	Returns true if the passed shape is equivalent to the given one. Inherited from **IShape**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IShape**.
getGeometry()	**IPixelGeometry**	Returns pixel geometry of a shape. Inherited from **IShape**.
getType()	String	Returns ID of the shape type. Inherited from **IShape**.
scale(factor)	**IShape**	Creates a scaled copy of the shape. Inherited from **IShape**.
shift(offset)	**IShape**	Creates a copy of the shape that is shifted by the specified amount. Inherited from **IShape**.

shape.Polygon

Extends **IShape**.

"Polygon" pixel shape.

Constructor | Methods

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor

```
shape.Polygon(pixelGeometry[, params])
```

Creates the shape.

Parameters:

Parameter	Default value	Description
pixelGeometry *	—	Type: **IPixelPolygonGeometry** Pixel geometry of a shape.
params	—	Type: Object Shape parameters.
params.fill	true	Type: Boolean Flag for filling the shape.
params.outline	true	Type: Boolean Flag for an outline.
params.strokeWidth	0	Type: Number The outline width, in pixels.

* Mandatory parameter/option.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the shape. Inherited from **IShape**.
equals(shape)	Boolean	Returns true if the passed shape is equivalent to the given one. Inherited from **IShape**.
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IShape**.
getGeometry()	**IPixelGeometry**	Returns pixel geometry of a shape. Inherited from **IShape**.
getType()	String	Returns ID of the shape type. Inherited from **IShape**.
scale(factor)	**IShape**	Creates a scaled copy of the shape. Inherited from **IShape**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
shift(offset)	IShape	Creates a copy of the shape that is shifted by the specified amount. Inherited from **IShape**.

shape.Rectangle

Extends **IShape**.

"Rectangle" pixel shape.

Constructor | Methods

Constructor

```
shape.Rectangle(pixelGeometry[, params])
```

Creates the shape.

Parameters:

Parameter	Default value	Description
pixelGeometry *	—	Type: **IPixelRectangleGeometry** Pixel geometry of a shape.
params	—	Type: Object Shape parameters.
params.fill	true	Type: Boolean Flag for filling the shape.
params.outline	true	Type: Boolean Flag for an outline.
params.strokeWidth	0	Type: Number The outline width, in pixels.

* Mandatory parameter/option.

Methods

Name	Returns	Description
contains(position)	Boolean	Checks whether the passed point is located inside the shape. Inherited from **IShape**.
equals(shape)	Boolean	Returns true if the passed shape is equivalent to the given one. Inherited from **IShape**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
getBounds()	Number[][]|null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from **IShape**.
getGeometry()	**IPixelGeometry**	Returns pixel geometry of a shape. Inherited from **IShape**.
getType()	String	Returns ID of the shape type. Inherited from **IShape**.
scale(factor)	**IShape**	Creates a scaled copy of the shape. Inherited from **IShape**.
shift(offset)	**IShape**	Creates a copy of the shape that is shifted by the specified amount. Inherited from **IShape**.

shape.storage

Static object.

Instance of **util.Storage**

Storage for hotspot shape geometries.

Methods

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

suggest

Static function.

Processes requests for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }. The displayName field represents the toponym in a user-friendly way, and the value field represents the value which should be inserted into the search field after the user selects the suggestion.

Returns a Promise object.

```
{ vow.Promise } suggest(request[, options])
```

Parameters:

Parameter	Default value	Description
request*	—	Type: String Request string.
options	—	Type: Object Options.
options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
options.provider	'yandex#map'	Type: **ISuggestProvider**|String Search suggestion provider. You can use the 'yandex#map' built-in search suggestion provider for map objects, or specify your own.
options.results	—	Type: Number Maximum number of results to be returned.

* Mandatory parameter/option.

Example:

```
ymaps.suggest('myt').then(function (items) {
    // items - Array of search suggestions.
});
```

SuggestView

Extends **ICustomizable**, **IEventEmitter**.

Creates a drop-down list with search suggestions and attaches it to the HTML element <input type="text">.

Constructor | Fields | Events | Methods

Constructor

```
SuggestView(element[, options])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
element *	—	Type: HTMLElement|String HTML element or its ID.
options	—	Type: Object Options.
options.boundedBy	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
options.container	—	Type: HTMLElement HTML element for placing the display of the suggestions panel. If omitted, the suggestions panel is added to the parent of the HTML "input" element that it is created for.
options.layout	'islands#suggestView'	Type: String|**ISuggestViewLayout** Panel layout. The layout constructor is passed an object containing the fields: • suggestView — Link to the panel with search suggestions. • options — Options manager for the control suggestView.options. • state — State manager for the control suggestView.state. The layout adapts its appearance based on the state and options of the suggestions panel. The control, in turn, reacts to layout interface events and changes the values of fields for suggestView.state depending on the commands received.
options.offset	—	Type: Number[] Offsets of the suggestions panel from its default location (by default, the suggestions panel is attached to the lower edge of the "input" element and has the same width as it). Set as horizontal and vertical offsets relative to the lower-left corner of the "input" element.
options.provider	"yandex#map"	Type: String|**ISuggestProvider** Provider for search suggestions. May be set as an object implementing the **ISuggestProvider** interface, or the standard value "yandex#map".

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
options.results	5	Type: Number Maximum quantity of suggestions to display.
options.width	—	Type: Number Width of the suggestions panel. By default, the same as the width of the HTML "input" element to which the panel is attached.
options.zIndex	40000	Type: Number The z-index for the dom element of the suggestions panel.

* Mandatory parameter/option.

Examples:

1.

```
&lt;input type="text" id="suggest"/&gt;
&lt;script src="//api-maps.yandex.ru/2.1/?
lang=ru_RU&amp;load=SuggestView&amp;onload=onLoad"&gt;&lt;/script&gt;
&lt;script&gt;
function onLoad (ymaps) {
    var suggestView = new ymaps.SuggestView('suggest');
}
&lt;/script&gt;
```

2.

```
&lt;input type="text" id="suggest"/&gt;
&lt;script src="//api-maps.yandex.ru/2.1/?
lang=ru_RU&amp;load=SuggestView&amp;onload=onLoad"&gt;&lt;/script&gt;
&lt;script&gt;
function onLoad (ymaps) {
    var suggestView = new ymaps.SuggestView('suggest', {results: 1, offset: [20,
30]});
}
&lt;/script&gt;
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Type	Description
state	data.Manager	State of the search suggestions panel. Names of fields that are available via the data.Manager.get method: • request: String — Current active request. • items: Object[] — Array of search suggestions (objects with the value and displayName fields). • activeIndex: Number|null — Index of the currently active suggestion selected by the user with the mouse or keyboard, or null, if none of the suggestions is active. • panelClosed: Boolean — Indicates whether the user closed the panel by pressing ESC or choosing one of the suggestions.

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.
select	The user selected one of the search suggestions. Names of fields that are available via the Event.get method • `item` – A search suggestion (an object with the "displayName" and "value" fields).

Methods

Name	Description
destroy()	Destroys the search suggestions panel.

Fields details

state

```
{data.Manager} state
```

State of the search suggestions panel. Names of fields that are available via the data.Manager.get method:

- request: String — Current active request.
- items: Object[] — Array of search suggestions (objects with the value and displayName fields).
- activeIndex: Number|null — Index of the currently active suggestion selected by the user with the mouse or keyboard, or null, if none of the suggestions is active.
- panelClosed: Boolean — Indicates whether the user closed the panel by pressing ESC or choosing one of the suggestions.

Events details

select

The user selected one of the search suggestions. Names of fields that are available via the Event.get method

- `item` – A search suggestion (an object with the "displayName" and "value" fields).

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

destroy

```
{} destroy()
```

Destroys the search suggestions panel.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

template

template.filtersStorage

Static object.

Instance of **util.Storage**

Storage for template filters. Filters from storage can be used in all layouts created using **templateLayoutFactory**. By default, the storage declares the following keys for filters:

- default — Allows setting default values. For example, like this: {{ ***properties.header***|**default:*"Title"*** }}

Methods

Examples:

1.

```
 // Writing a simple filter that will convert a date
 // from the format "dd.mm.yyyy" to the format "dd month yyyy".
// To do this, we need to create a filter function that will return the new
value.

 // When the filter is called, the following arguments are passed to the filter:
 // the template data manager data.Manager, the value, and the value set for the
filter.
 var dateFilter = function (data, dateString, filterValue) {
     var months = [
             'january', 'february', 'march', 'april',
             'may', 'june', 'july', 'august',
             'september', 'october', 'november', 'december'
         ],
        date = dateString.split('.');

     date[1] = months[parseInt(date[1], 10)];
     return date.join(' ');
};

ymaps.template.filtersStorage.add('date', dateFilter);

// Then we can use it in constructions like
// {{ "21.10.2014"|date }} the value will be "21 october 2014".
```

2.

```
 // Writing a filter that will find and replace substrings in text.
// The format for substitution values in the filter is "subString_newSubString".

 // When the filter is called, the following arguments are passed to the
function:
 // the template data manager data.Manager, the text, and the value set for the
filter.
 var replaceFilter = function (data, text, replace) {
     replace = replace.trim();
     // Removing quotation marks.
    replace = replace.slice(1, replace.length - 1);

     // Finding the part that comes before "_" in the text and replacing it with
what comes after it.
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
    var values = replace.split('_'),
        from = new RegExp(values[0], 'g'),
       to = values[1];

    return text.replace(from, to);
};

// Now we can use this in templates of constructions like
// {{"text test replace"|replace: "test_replaced test" }} the value will be "text
replaced test replace".
```

3.

```
// In this example, the value of the "colorClass" option and the value of the
"header" property are added to the layout.
// If the "header" property doesn't have a value, the string "Title" is
inserted.
var LayoutClass = ymaps.templateLayoutFactory.createClass(
    '&lt;h1 class="{{ options.colorClass }}"&gt;' +
        '{{ properties.header|default:"Title" }}' +
    '&lt;/h1&gt;'
);
```

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Template

Templating engine. The Yandex.Maps API supports the base syntax for the Twig/Django Templates languages. The following operations are supported:

- Substitution value — **{{ *field_name* }}**.
- If the requested data field is missing or has an empty value you can provide the default value — **{{ *field_name*|default:*default_value* }}** The default value can be a number, a string (in quotes) or a different data field.
- By default the value is processed by the escape function to prevent XSS vulnerabilities. To undo this behavior, add the **"raw"** filter — **{{ *field_name*|default: *default_value*|raw }}**.
- Using the **template.filtersStorage** you can create your own filters and use them similarly to the described above.
- To insert a sub layout, use a construction like **{% include *field_name_or_key* %}**. The templating engine when it founds such a construction, will try to use the value in the field as a key of the nested layout.
- The condition is written as: **{% if *condition* %} ... {% else %} ... {% endif %}** or you can ommit the **else** or **elseif** block. You can use any constructions of the template language inside **if**, **else** and **elseif** blocks.
- Use the **for** construction to iterate an array or an object. **{% for *value* in *array_or_hash* %} ... {% endfor %}**. You can use any constructions of the template language inside the **for** block.
- To obtain the iteration index in the array or the field name in the hash, use the following construction: **{% for key, value in *array_or_hash* %} ... {% endfor %}**.

Constructor | Methods

Constructor

```
Template(text)
```

Parameters:

Parameter	Default value	Description
text *	—	Type: String Template string

* Mandatory parameter/option.

Examples:

1.

```
// Getting the user name from the data manager data.Manager.
// If the name is not specified, the result string will be «Unregistered user».
var data = new ymaps.data.Manager({
         user: {
             name: "Viktor",
             age: 25
         },
         home: [55.72725771214265, 37.640390506634006]
     }),
    template = new ymaps.Template('{{ user.name |default: "Unregistered
user"}}');

var result = template.build(data);
console.log(result.text); // Outputs «Viktor» to the console.
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Let's assume we have 3 user groups and we need to print an individual greeting
for each group.
var data = new ymaps.data.Manager({
        groups: {
            administrator: {
                id: 1,
                name: "administrator"
            },
            moderator: {
                id: 2,
                name: "moderator"
            },
            user: {
                id: 3,
                name: "user"
            }
        },
        userGroupId: 2
    }),
    template = new ymaps.Template('Hi, \
        {% if (userGroupId == 1) %}{{ groups.administrator.name }}\
        {% elseif (userGroupId == 2) %}{{ groups.moderator.name }}\
        {% elseif (userGroupId == 3) %}{{ groups.user.name }}\
        {% else %}guest{% endif %}!'
    );

var result = template.build(data);
console.log(result.text); // Outputs «Hi, moderator!» to the console.
```

Methods

Name	Returns	Description
build(data)	Object	Returns object with following fields: • {String} text — the result of templating. • {Object[]} renderedValues — an array with the used data from the manager. .

Methods details

build

```
{Object} build(data)
```

Returns object with following fields:

- {String} text — the result of templating.
- {Object[]} renderedValues — an array with the used data from the manager.

.

Parameters:

Parameter	Default value	Description
data*	—	Type: **IDataManager** Data manager.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Example:

```
// Let's get the house address from the existing coordinates and output
(according to the template)
// all its inhabitants in the format: «name: age».
var data = new ymaps.data.Manager({
         users: [
             {name: "Vitaly", age: 40},
             {name: "George", age: 20}
         ],
         home: {
             coords: [55.736652, 37.620589],
             address: null
         }
     }),
     template = new ymaps.Template('{{home.address}}: &lt;ul&gt;{% for user in
users %}&lt;li&gt;{{user.name}}: {{user.age}}&lt;/li&gt;{% endfor %}&lt;/
ul&gt;');

// Let's do the reverse geocoding using geocode.
ymaps.geocode(data.get('home.coords')).then(function (res) {
     var address = res.geoObjects.get(0).properties.get('name');
     // Setting the obtained address to the manager.
     data.set('home.address', address);

     // Completing the template with the obtained data.
     var result = template.build(data);
     // Output the result to the console.
     console.log(result.text);
});
```

templateLayoutFactory

Static object.

Factory for creating a layout class from a text template. Allows creating classes that implement the interface **ILayout** using a template language. The Yandex.Maps API supports the base syntax for the Twig/Django Templates languages. For more information about the syntax, see the description of the **Template**.

See layout.templateBased.Base

Methods

Examples:

1.

```
// In this example, the value of the "colorClass" option and the value of the
"header" property are added to the layout.
// If the "header" property doesn't have a value, the string "Title" is
inserted.
var LayoutClass = ymaps.templateLayoutFactory.createClass(
     '&lt;h1 class="{{ options.colorClass }}"&gt;' +
         '{{ properties.header|default:"Title" }}' +
     '&lt;/h1&gt;'
);
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// One of the layouts is enabled, depending on the value of the "width" option.
var LayoutClass = ymaps.templateLayoutFactory.createClass(
     '{% if options.width > 200 %}' +
         // The appropriate layout will be found in the options.
         '{% include options.wideLayout %}' +
     '{% else %}' +
         // Writing the key explicitly.
         '{% include "cluster#balloonCarousel" %}' +
     '{% endif %}'
);
```

3.

```
// Outputting an array of names to the balloon layout.
var CustomLayoutClass = ymaps.templateLayoutFactory.createClass(
     '&lt;ul&gt;' +
     '{% for name in properties.names %}' +
         // The "name" variable is only visible in the for ... endfor block
         '&lt;li&gt;{{ name }}&lt;/li&gt;' +
     '{% endfor %}' +
     '&lt;/ul&gt;'
 );

var placemark = new ymaps.Placemark([54.83, 37.11], {
     names: ['Logan', 'Sofia', 'Mason', 'Layla']
}, {
     balloonContentLayout: CustomLayoutClass
});
```

4.

```
// Getting the names of fields.
var CustomLayoutClass = ymaps.templateLayoutFactory.createClass(
 '<ul>' +
     '{% for key, value in properties.hash %}' +
         '<li>{{ key }} {{ value }}</li>' +
     '{% endfor %}' +
 '</ul>'
);

var placemark = new ymaps.Placemark([54.83, 37.11], {
     hash: { key1: "value1", key2: "value2", key3: "value3" }
}, {
     balloonContentLayout: CustomLayoutClass
});
```

Methods

Name	Static	Returns	Description
templateLayoutFactory.**createClass**(template[, overrides[, staticMethods]])	✔	Function	Returns layout constructor. The created class inherits from the class **layout.templateBased.Base** with a redefined list of methods specified in overrides.

Methods details

createClass

```
{Function}   <static> templateLayoutFactory.createClass(template[, overrides[, staticMethods]])
```

Returns layout constructor. The created class inherits from the class **layout.templateBased.Base** with a redefined list of methods specified in overrides.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
template*	—	Type: String Template for HTML content for layouts.
overrides	—	Type: Object Redefining parent methods. The build, clear and rebuild methods can be redefined or expanded.
staticMethods	—	Type: Object Setting static methods for classes.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

traffic

traffic.provider

traffic.provider.Actual

Extends **ITrafficProvider**.

Provider of real-time traffic data. Accessible in the provider storage by the key 'traffic#actual'.

Constructor | Fields | Events | Methods

Constructor

```
traffic.provider.Actual([options[, state]])
```

Creates a provider of real-time traffic data.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Provider options. Options for provider layers are set via the options for the global collection of layers, **Map.layers**. • Options for the image layer **Layer** are defined with the 'trafficImage' prefix. • Options for the hotspot layer **hotspot.Layer** are defined with the 'trafficJam' prefix. • Options for the infopoint layer are defined with the 'trafficInfo' prefix. The infopoint layer is an instance of the **hotspot.Layer** class.
options.autoUpdate	true	Type: Boolean Flag that enables automatically updating traffic data. Automatic updates occur only when the "mousemove" event occurs every 4 minutes on the map. If this event does not occur, traffic stops being updated until there is a new event.
state	—	Type: Object Provider state.
state.infoLayerShown	false	Type: Boolean Flag that enables displaying the traffic events layer.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Creating a provider for current traffic with the traffic events layer enabled
// and putting it on the map.
var actualProvider = new ymaps.traffic.provider.Actual({}, {infoLayerShown:
true});
actualProvider.setMap(myMap);

// Forbidding showing balloons for clicks on the infopoint layer.
myMap.layers.options.set({
    // The option name is formed by adding the 'trafficInfo' prefix
    // to the hotspot layer option 'openBalloonOnClick'.
    trafficInfoOpenBalloonOnClick: false
});

// ...
// Deleting the provider from the map.
actualProvider.setMap(null);
```

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**data.Manager**	Provider state. Names of fields that are available via the data.Manager.get method: • isInited — Flag for whether the provider is ready to provide data. • infoLayerShown — Flag for whether the traffic events layer is shown. • timestamp — Current time in the UTC+0 time zone, in seconds. • localtime — Local time that the server is currently sending data for, in the format HH:MM. • level — Traffic level in points from 0 to 10. • isotime — String containing the current date in the format "YYYY-MM-DDThh:mm:ss±hhmm".

Events

Name	Description
optionschange	Change to the object options. Inherited from **ICustomizable**.

Methods

Name	Returns	Description
getMap()	**Map**|null	Returns reference to the map. Inherited from **ITrafficProvider**.
setMap(Reference)		
update()		Sends a request to update traffic.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Fields details

state

```
{data.Manager} state
```

Provider state. Names of fields that are available via the data.Manager.get method:

- isInited — Flag for whether the provider is ready to provide data.
- infoLayerShown — Flag for whether the traffic events layer is shown.
- timestamp — Current time in the UTC+0 time zone, in seconds.
- localtime — Local time that the server is currently sending data for, in the format HH:MM.
- level — Traffic level in points from 0 to 10.
- isotime — String containing the current date in the format "YYYY-MM-DDThh:mm:ss±hhmm".

Example:

```
var actualProvider = new ymaps.traffic.provider.Actual();
actualProvider.setMap(myMap);
actualProvider.state.events.add('change', function () {
    if (actualProvider.state.get('isInited')) {
        alert('The provider is ready to provide data.');
    }
});
```

Methods details

update

```
{} update()
```

Sends a request to update traffic.

Example:

```
var trafficControl = new ymaps.control.TrafficControl({shown: true});
map.controls.add(trafficControl);
function updateProvider () {
     trafficControl.getProvider('traffic#actual').update();
}
// Sending a request to update data every 4 minutes.
window.setInterval(updateProvider, 4 * 60 * 1000);
```

traffic.provider.Archive

Extends **ITrafficProvider**.

Provider for the traffic archive. This lets us show the normal state of traffic for a given region on a particular day of the week and at a particular time.

Constructor | Fields | Events | Methods

Constructor

```
traffic.provider.Archive([options[, state]])
```

Creates an instance of the traffic archive provider.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
options	—	Type: Object Provider options. Options for provider layers are set via the options for the global collection of layers, **Map.layers**. • Options for the image layer **Layer** are defined with the 'trafficImage' prefix. • Options for the hotspot layer **hotspot.Layer** are defined with the 'trafficJam' prefix.
options.showCurrentTimeFirst	true	Type: Boolean When archived data is first shown, set a time near the current time.
state	—	Type: Object Provider state.
state.timestamp	—	Type: Number The time that "normal" traffic is being shown for. This is the time from Monday at 00:00 to the desired time, in seconds. It must be a multiple of 60 * 15 = 900, since data on the server is available for times with a difference of 15 minutes. The time is set for the universal time zone (UTC+0).

Example:

```
// Creating a provider for "normal" traffic and giving it a timestamp of 17:47 on
Wednesday
// in the universal time zone. Note that the local time depends on
// the location of the map center.
// For example, 17:47 in the universal time zone is 21:47 in Moscow.

// Calculating the value of the timestamp parameter for the desired time.
var timestamp = 2 * 24 * 60 * 60 + // twice every 24 hours - this is the time for
Monday and Tuesday
        17 * 60 * 60 + // 17 hours have passed since 00:00 Wednesday
       45 * 60, // since the time must be in 15-minute increments, use 45 instead
of 47.
    archiveProvider = new ymaps.traffic.provider.Archive({
        // Don't display a time near the current time on the first opening
        showCurrentTimeFirst: false
    }, {
        // Setting the starting time independently.
        timestamp: timestamp
    });
archiveProvider.setMap(map);
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Don't show popup hints for the traffic layer.
myMap.layers.options.set({
    // The option name is formed by adding the 'trafficJam' prefix
    // to the hotspot layer option 'openHintOnHover'.
    trafficJamOpenHintOnHover: false
});

// ...
// Removing the provider from the map.
archiveProvider.setMap(null);
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.
options	IOptionManager	Options manager. Inherited from ICustomizable.
state	data.Manager	Provider state. Names of fields that are available via the data.Manager.get method: • isInited — Flag for whether the provider is ready to provide data. • timeZone — Time offset for the current time zone relative to UTC +0. Measured in seconds. • dst — Flag for switching to winter/summer time (daylight saving time). When dst='dst' it is summer time. • timestamp — Current time in the UTC+0 time zone, in seconds. • localtime — The local time that the server returns in the response. • level — Traffic level in points from 0 to 10.

Events

Name	Description
optionschange	Change to the object options. Inherited from ICustomizable.

Methods

Name	Returns	Description
getMap()	Map|null	Returns reference to the map. Inherited from ITrafficProvider.
getTime()	Object|null	Returns the day of the week, hour, and minutes for the provider's status with consideration of the time zone and adjustments for daylight saving time. In other words, this is the time the user sees in the traffic control.
setMap(Reference)		

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
setTime(time[, callback])		Allows to set the time for an archive provider in minutes, hours and days. Sets the local time only after the provider initializes the "timeZone" and "dst" fields. • timeZone — Field that displays which time zone the map center is currently located in. When the map center is moved from one time zone to another, the local time may change. • dst — Flag for switching to summer/winter time (daylight saving time). When dst='dst' it indicates summer time. The "timestamp" field serves as a permanent part of the time for providers of "normal" traffic, and reflects the current time in the zero time zone (UTC+0). When changing from one time zone to another, "timestamp" does not change. You can get the values of the "timestamp", "dst", and "timeZone" fields from the **traffic.provider.Archive.state** field.

Fields details

state

```
{data.Manager} state
```

Provider state. Names of fields that are available via the data.Manager.get method:

- isInited — Flag for whether the provider is ready to provide data.
- timeZone — Time offset for the current time zone relative to UTC+0. Measured in seconds.
- dst — Flag for switching to winter/summer time (daylight saving time). When dst='dst' it is summer time.
- timestamp — Current time in the UTC+0 time zone, in seconds.
- localtime — The local time that the server returns in the response.
- level — Traffic level in points from 0 to 10.

Example:

```
var archiveProvider = new ymaps.traffic.provider.Archive();
archiveProvider.setMap(myMap);
archiveProvider.state.events.add('change', function () {
    if (archiveProvider.state.get('isInited')) {
        alert('Provider is ready to provide data.');
    }
});
```

Methods details

getTime

```
{Object|null} getTime()
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns the day of the week, hour, and minutes for the provider's status with consideration of the time zone and adjustments for daylight saving time. In other words, this is the time the user sees in the traffic control.

Returns object with fields

- dayOfWeek — Abbreviations of days of the week. 'mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun'.
- hours — Hours.
- minutes — Minutes.

If the map center is located at a point that we cannot determine the time zone for, the function returns null; if we don't know which time zone we are in, we can't find out the local time.

setTime

```
{} setTime(time[, callback])
```

Allows to set the time for an archive provider in minutes, hours and days. Sets the local time only after the provider initializes the "timeZone" and "dst" fields.

- timeZone — Field that displays which time zone the map center is currently located in. When the map center is moved from one time zone to another, the local time may change.
- dst — Flag for switching to summer/winter time (daylight saving time). When dst='dst' it indicates summer time.

The "timestamp" field serves as a permanent part of the time for providers of "normal" traffic, and reflects the current time in the zero time zone (UTC+0). When changing from one time zone to another, "timestamp" does not change. You can get the values of the "timestamp", "dst", and "timeZone" fields from the **traffic.provider.Archive.state** field.

Parameters:

Parameter	Default value	Description
time*	—	Type: Object Object with the set parameters.
time.dayOfWeek	—	Type: String Abbreviated name of a day of the week. 'mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun'.
time.hours	—	Type: Number Hour.
time.minutes	—	Type: Number Minutes.
callback	—	Type: Function A function that is called after the time was set. Accepts a hash with the set data as input.

* Mandatory parameter/option.

Example:

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Creating a control that immediately shows the
// provider of "normal" traffic on the map.
var trafficControl = new ymaps.control.TrafficControl({
        shown: true,
        providerKey: 'traffic#archive'
    });
map.controls.add(trafficControl);
// The local time will be set as soon as the provider
// gets data on the current time zone.
trafficControl.getProvider('traffic#archive').setTime({
    dayOfWeek: 'fri',
    hours: 9,
    minutes: 15
}, function (time) {
    alert('Local time ' + time.hours + ':' + time.minutes + ' set!');
});
```

traffic.provider.Forecast

Extends **ITrafficProvider**.

Provider for the traffic forecast. Accessible in the provider storage by the key 'traffic#forecast'.

Constructor | Fields | Events | Methods

Constructor

```
traffic.provider.Forecast([options[, state]])
```

Creates a provider for the traffic forecast.

Parameters:

Parameter	Default value	Description
options	—	Type: Object Provider options. Options for provider layers are set via the options for the global collection of layers, **Map.layers**. • Options for the image layer **Layer** are defined with the 'trafficImage' prefix. • Options for the hotspot layer **hotspot.Layer** are defined with the 'trafficJam' prefix.
options.autoUpdate	true	Type: Boolean Flag that enables automatically updating traffic data. Automatic updates occur only when the "mousemove" event occurs every 4 minutes on the map. If this event does not occur, traffic stops being updated until there is a new event.
state	—	Type: Object Provider state.
state.timeOffset	900	Type: Number

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description

		time interval between the current time and forecast time. Measured in seconds, in 15-minute intervals (900 seconds).
		`// Creating a traffic provider that predicts traffic in half an hour,`
		`// and putting it on the map.`
		`var forecastProvider = new ymaps.traffic.provider.Forecast({}, {timeOffset: 30 * 60});`
		`forecastProvider.setMap(myMap);`
		
		`// Forbidding displaying popup hints for traffic.`
		`myMap.layers.options.set({`
		`// The option name is formed by adding the 'trafficJam' prefix`
		`// to the hotspot layer option 'openHintOnHover'.`
		
		`trafficJamOpenHintOnHover: false`
		`});`
		`// ...`
		`// Removing the provider from the map.`
		`forecastProvider.setMap(null);`

Fields

Name	Type	Description
events	**IEventManager**	Event manager. Inherited from **IEventEmitter**.
options	**IOptionManager**	Options manager. Inherited from **ICustomizable**.
state	**data.Manager**	Provider state. Names of fields that are available via the data.Manager.get method: • isInited — Flag for whether the provider is ready to provide data. • timestamp — Current time in Unix timestamp format, in seconds. • localtime — Local time that the server is currently sending data for, in the format HH:MM. • level — Traffic level in points from 0 to 10. • timeOffset — Time interval between the current time and forecast time. Measured in seconds, in 15-minute intervals (900 seconds) • isotime — String containing the current date in the format "YYYY-MM-DDThh:mm:ss±hhmm".

Events

Name	Description
optionschange	Change to the object options. Inherited from **ICustomizable**.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
getMap()	Map|null	Returns reference to the map. Inherited from ITrafficProvider.
getTime()	Object|null	Returns the time the user sees in the traffic control. The object has the following fields: • dayOfWeek — Abbreviations of days of the week. 'mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun'. • hours — Hours. • minutes — Minutes. If the map center is located at a point that we cannot determine the time zone for, the function returns null; if we don't know which time zone we are in, we can't find out the local time.
setMap(Reference)		

Fields details

state

```
{data.Manager} state
```

Provider state. Names of fields that are available via the data.Manager.get method:

- isInited — Flag for whether the provider is ready to provide data.
- timestamp — Current time in Unix timestamp format, in seconds.
- localtime — Local time that the server is currently sending data for, in the format HH:MM.
- level — Traffic level in points from 0 to 10.
- timeOffset — Time interval between the current time and forecast time. Measured in seconds, in 15-minute intervals (900 seconds)
- isotime — String containing the current date in the format "YYYY-MM-DDThh:mm:ss±hhmm".

Methods details

getTime

```
{Object|null} getTime()
```

Returns the time the user sees in the traffic control. The object has the following fields:

- dayOfWeek — Abbreviations of days of the week. 'mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun'.
- hours — Hours.
- minutes — Minutes.

If the map center is located at a point that we cannot determine the time zone for, the function returns null; if we don't know which time zone we are in, we can't find out the local time.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

traffic.provider.storage

Static object.

Instance of **util.Storage**

Storage for providers.

Methods

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

util

util.AsyncStorage

Extends **util.Storage**.

Storage that provides asynchronous access to key values.

Constructor | Methods

Constructor

```
util.AsyncStorage()
```

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage. Inherited from **util.Storage**.
define(key[, depends, resolveCallback[, context]])	**util.AsyncStorage**	Defines an asynchronous value in storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string. Inherited from **util.Storage**.
isDefined(key)	Boolean	Checking if the key can be accessed in the storage.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage. Inherited from **util.Storage**.
require(keys[, successCallback[, errorCallback[, context]]])	**vow.Promise**	Async request to get values from the storage.

Methods details

define

```
{util.AsyncStorage} define(key[, depends, resolveCallback[, context]])
```

Defines an asynchronous value in storage.

Returns self-reference.

Parameters:

Parameter	Default value	Description
key*	—	Type: String The key to use for making an async call.
depends	—	Type: String[]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Array of keys from the current storage that must be initialized before this key. This argument can be omitted.
resolveCallback*	—	Type: Function Function that defines the value accessed by the key. The first argument in resolveCallback is the provide function to pass the value to. Invocation of the provide function can be deferred. The following arguments are values from storage that are specified in dependencies. The order of modules follows their order in the depends array.
context	—	Type: Object Execution context for the function.

* Mandatory parameter/option.

Examples:

1.

```
asyncStorage
    .define('red', function (provide) {
        provide('#FF0000');
     });
```

2.

```
asyncStorage
    .define('green', function (provide) {
        // The provide function can be called asynchronously.
        setTimeout(function () {
            provide('#008000');
        }, 400);
    });
```

3.

```
asyncStorage
    .define('yellow', function (provide) {
         provide('#FFFF00');
    })
    // To define the 'violet' key value, the 'yellow' key value is required.
    .define('violet', ['yellow'], function (provide, yellow) {
        console.log(yellow); // #FFFF00
        setTimeout(function () {
            provide('#9B30FF');
        }, 400);
    });
```

4.

```
var asyncStorage = new ymaps.util.AsyncStorage();
asyncStorage
    .define('red', function (provide) {
        provide('#FF0000');
     })
    .define('green', function (provide) {
        setTimeout(function () {
            provide('#008000');
        }, 400);
    })
    .define('yellow', function (provide) {
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
            provide('#FFFF00');
        })
        .define('violet', ['yellow'], function (provide, yellow) {
            setTimeout(function () {
                provide('#9B30FF');
            }, 400);
        });
// Requesting
asyncStorage.require(['red', 'green', 'violet'])
        .spread(function (red, green, violet) {
            console.log(red, green, violet); // #FF0000 #008000 #9B30FF
            // After the first async access, values can be accessed from the
synchronous interface.
            console.log(asyncStorage.get('red'), asyncStorage.get('green'),
asyncStorage.get('violet')); // #FF0000 #008000
            // The value for the 'yellow' key is also in the storage now, because it
was required in order to define 'violet'.
            console.log(asyncStorage.get('yellow')); // #FFFF00
        });
```

isDefined

```
{Boolean} isDefined(key)
```

Checking if the key can be accessed in the storage.

Returns true — defined, false — not.

Parameters:

Parameter	Default value	Description
key*	—	Type: String Key for the value.

* Mandatory parameter/option.

Example:

```
if (asyncStorage.isDefined('red')) {
    asyncStorage.require('red')
        .spread(function (red) {
            // ...
        });
}
```

require

```
{vow.Promise} require(keys[, successCallback[, errorCallback[, context]]])
```

Async request to get values from the storage.

Returns Promise that represents async access to the value.

Parameters:

Parameter	Default value	Description
keys*	—	Type: String|String[] Key or array of keys.
successCallback	—	Type: Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Callback to invoke after getting all the values. Values from storage are passed to the function as arguments. The arguments are ordered according to their order in the keys array.
errorCallback	—	Type: Function Callback to use if an error occurs. The error object is passed to the function.
context	—	Type: Object Callback execution context.

* Mandatory parameter/option.

Examples:

1.

```
asyncStorage.require(['green'])
    .spread(function (green) {
        // ...
    });
```

2.

```
var asyncStorage = new ymaps.util.AsyncStorage();
asyncStorage
    .define('red', function (provide) {
        provide('#FF0000');
     })
    .define('green', function (provide) {
        setTimeout(function () {
            provide('#008000');
        }, 400);
    });
// Requesting
asyncStorage.require(['red', 'green'])
    .spread(function (red, green) {
         console.log(red, green); // #FF0000 #008000
    });
```

util.augment

Static function.

Base function implementing inheritance in JavaScript. Implements prototype inheritance without executing the parent constructor. The "superclass" field is appended to the child class, specifying the prototype of the parent class and the 'constructor' field, specifying the constructor of the class. Use the 'constructor' field of the 'superclass' object to refer to the constructor of the parent class.

Returns a prototype of the child class.

```
{ Object } util.augment(ChildClass, ParentClass, override)
```

Parameters:

Parameter	Default value	Description
ChildClass*	—	Type: Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Child class.
ParentClass*	—	Type: Function Parent class.
override*	—	Type: Object Set of additional fields and functions that will be appended to the prototype of the child class.

* Mandatory parameter/option.

Example:

```
// Parent class
var ParentClass = function (param1, param2) {
    this.param1 = param1;
    this.param2 = param2;
};

ParentClass.prototype = {
    foo: function () {
        alert('Parent!');
    }
};
// Child class
var ChildClass = function (param1, param2, param3) {
    // Calling the parent's constructor
    ChildClass.superclass.constructor.call(this, param1, param2);
    this._param3 = param3;
}

// inheriting ChildClass from ParentClass
ymaps.util.augment(ChildClass, ParentClass, {
    // redefining the "foo" method in the descendant
    foo: function () {
        // Calling the parent class method
        ChildClass.superclass.foo.call(this);
        alert('Child!');
    }
});
```

util.bind

Static function.

Binds the passed function to the passed context.

Returns a copy of the given function with the specified `this` value.

```
{ Function } util.bind(callback, context)
```

Parameters:

Parameter	Default value	Description
callback*	—	Type: Function Function.
context*	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Execution context.

* Mandatory parameter/option.

Example:

```
var myObject = {
        name: 'test!'
    };
ymaps.geocode.load('Moscow')
    then(ymaps.util.bind(function (res) {
        alert(this.name); // test!
    }, myObject));
```

util.bounds

Static object.

Set of statistical methods for working with rectangular areas, represented as two opposite points in the coordinate system of the projection.

Methods

Methods

Name	Returns	Description
areIntersecting(bounds1, bounds2[, projection])	Boolean	Determines whether two rectangular areas intersect.
containsBounds(outer, inner[, projection])	Boolean	Determines whether a rectangular area completely contains another rectangular area.
containsPoint(bounds, point[, projection])	Boolean	Determines whether a rectangular area contains a point.
fromBounds(sourceBounds[, projection])	Number[][]	Calculates the rectangular area that everything passed falls inside of.
fromGlobalPixelBounds(pixelBounds, zoom[, projection])	Number[][]	Converts a rectangular area from pixel coordinates to geo coordinates with the zoom factor taken into account.
fromPoints(points[, projection])	Number[][]	Calculates the minimal rectangular area that contains all the passed points.
getCenter(bounds[, projection])	Number[]	Calculates the center of a rectangular area in the coordinate system of the projection.
getCenterAndZoom(bounds, containerSize[, projection[, params]])	Object	Calculates the center and zoom that should be set for the map in order to display the passed area in its entirety.
getIntersections(bounds1, bounds2[, projection])	Number[][][]	Returns all intersections of two rectangular areas. If data is passed in geo coordinates, there may be more than one intersection. If the areas do not intersect, an empty array is returned.
getSize(bounds[, projection])	Number[]	Calculates the dimensions of a rectangular area in the coordinate system of the projection.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Name	Returns	Description
toGlobalPixelBounds(geoBounds, zoom[, projection])	Number[][]	Converts boundaries from geo coordinates to global pixels, accounting for the zoom level.

Methods details

areIntersecting

```
{Boolean} areIntersecting(bounds1, bounds2[, projection])
```

Determines whether two rectangular areas intersect.

Returns intersection attribute.

Parameters:

Parameter	Default value	Description
bounds1*	—	Type: Number[][] First area.
bounds2*	—	Type: Number[][] Second area.
projection	**projection.wgs84Mercator**	Type: **IProjection** Projection.

* Mandatory parameter/option.

containsBounds

```
{Boolean} containsBounds(outer, inner[, projection])
```

Determines whether a rectangular area completely contains another rectangular area.

Returns inclusion attribute.

Parameters:

Parameter	Default value	Description
outer*	—	Type: Number[][] External area
inner*	—	Type: Number[][] The area being checked.
projection	**projection.wgs84Mercator**	Type: **IProjection** Projection.

* Mandatory parameter/option.

containsPoint

```
{Boolean} containsPoint(bounds, point[, projection])
```

Determines whether a rectangular area contains a point.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns inclusion attribute.

Parameters:

Parameter	Default value	Description
bounds*	—	Type: Number[][] External area
point*	—	Type: Number[] The point being checked.
projection	**projection.wgs84Mercator**	Type: **IProjection** Projection.

* Mandatory parameter/option.

fromBounds

```
{Number[][]} fromBounds(sourceBounds[, projection])
```

Calculates the rectangular area that everything passed falls inside of.

Returns the calculated area.

Parameters:

Parameter	Default value	Description
sourceBounds*	—	Type: Number[][][] Array of rectangular areas
projection	**projection.wgs84Mercator**	Type: **IProjection** Projection.

* Mandatory parameter/option.

fromGlobalPixelBounds

```
{Number[][]} fromGlobalPixelBounds(pixelBounds, zoom[, projection])
```

Converts a rectangular area from pixel coordinates to geo coordinates with the zoom factor taken into account.

Returns calculated boundaries in geo coordinates.

Parameters:

Parameter	Default value	Description
pixelBounds*	—	Type: Number[][] Original boundaries.
zoom*	—	Type: Number Scale.
projection	**projection.wgs84Mercator**	Type: **IProjection** The projection that will be used for claculating geo coordinates.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

* Mandatory parameter/option.

fromPoints

```
{Number[][]} fromPoints(points[, projection])
```

Calculates the minimal rectangular area that contains all the passed points.

Returns the calculated area.

Parameters:

Parameter	Default value	Description
points*	—	Type: Number[][] Array of points.
projection	**projection.wgs84Mercator**	Type: **IProjection** Projection.

* Mandatory parameter/option.

getCenter

```
{Number[]} getCenter(bounds[, projection])
```

Calculates the center of a rectangular area in the coordinate system of the projection.

Returns center point in the coordinate system of the input data.

Parameters:

Parameter	Default value	Description
bounds*	—	Type: Number[][] Area.
projection	**projection.wgs84Mercator**	Type: **IProjection** Projection.

* Mandatory parameter/option.

getCenterAndZoom

```
{Object} getCenterAndZoom(bounds, containerSize[, projection[, params]])
```

Calculates the center and zoom that should be set for the map in order to display the passed area in its entirety.

Returns object with the center (Number[]) and zoom (Number) fields.

Parameters:

Parameter	Default value	Description
bounds*	—	Type: Number[][] An area set in geographical coordinates. The first point contains the minimum values for latitude and longitude, and the second point contains the maximum values.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
containerSize*	—	Type: Number[] Size of the map container.
projection	projection.wgs84Mercator	Type: IProjection Projection.
params	—	Type: Boolean|Object Parameters or value of the preciseZoom option.
params.inscribe	true	Type: Boolean If true, fit the area into the map; if false, fit the map into the area.
params.margin	0	Type: Number|Number[] Offset from the borders of the visible area of the map. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins.
params.preciseZoom	false	Type: Boolean When the value is "false", the zoom level is rounded down.

* Mandatory parameter/option.

getIntersections

```
{Number[][][]} getIntersections(bounds1, bounds2[, projection])
```

Returns all intersections of two rectangular areas. If data is passed in geo coordinates, there may be more than one intersection. If the areas do not intersect, an empty array is returned.

Parameters:

Parameter	Default value	Description
bounds1*	—	Type: Number[][] First area.
bounds2*	—	Type: Number[][] Second area.
projection	projection.wgs84Mercator	Type: IProjection Projection.

* Mandatory parameter/option.

getSize

```
{Number[]} getSize(bounds[, projection])
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Calculates the dimensions of a rectangular area in the coordinate system of the projection.

Returns size of the area.

Parameters:

Parameter	Default value	Description
bounds*	—	Type: Number[][] Area.
projection	**projection.wgs84Mercator**	Type: **IProjection** Projection.

* Mandatory parameter/option.

toGlobalPixelBounds

```
{Number[][]} toGlobalPixelBounds(geoBounds, zoom[, projection])
```

Converts boundaries from geo coordinates to global pixels, accounting for the zoom level.

Returns resulting pixel boundaries.

Parameters:

Parameter	Default value	Description
geoBounds*	—	Type: Number[][] Original boundaries.
zoom*	—	Type: Number Scale.
projection	**projection.wgs84Mercator**	Type: **IProjection** The projection in the coordinate system that the geo coordinates are set in.

* Mandatory parameter/option.

util.cursor

util.cursor.Accessor

Object providing access to the cursor added to the map.

Constructor | Methods

Constructor

```
util.cursor.Accessor(key)
```

Parameters:

Parameter	Default value	Description
key *	—	Type: String

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Key that corresponds to the cursor in the cursor storage.

* Mandatory parameter/option.

Methods

Name	Returns	Description
getKey()	String	Returns the current key for access to the cursor in the cursor storage.
remove()		Removes the cursor from the map.
setKey()		Sets a new key for accessing the cursor. For this key, there should be a corresponding cursor in the cursor storage. If this cursor is active, it is immediately changed.

Methods details

getKey

```
{String} getKey()
```

Returns the current key for access to the cursor in the cursor storage.

remove

```
{} remove()
```

Removes the cursor from the map.

setKey

```
{} setKey()
```

Sets a new key for accessing the cursor. For this key, there should be a corresponding cursor in the cursor storage. If this cursor is active, it is immediately changed.

Parameters:

Parameter	Default value	Description
key*	—	Type: String

* Mandatory parameter/option.

util.cursor.Manager

Cursor manager.

Constructor | Fields | Events | Methods

Constructor

```
util.cursor.Manager(element)
```

Manager for cursors over a DOM element. Uses direct assignment via style.cursor.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
element *	—	Type: HTMLElement The DOM element that cursors are being set for.

* Mandatory parameter/option.

Example:

```
// putting the "arrow" cursor over the map
var cursor = map.cursors.push('arrow');
setTimeout(function () {
    // setting a different key for the cursor after 5 seconds
    cursor.setKey('grabbing');
    setTimeout(function () {
        // removing this cursor from the map after 5 more seconds
        cursor.remove();
    }, 5000);
}, 5000);
```

Fields

Name	Type	Description
events	event.Manager	Event manager for the cursor manager.

Events

Name	Description
change	Change to the cursor.

Methods

Name	Returns	Description
push(key)	util.cursor.Accessor	Sets a new cursor and writes it to the stack of cursors for the object.

Fields details

events

```
{event.Manager} events
```

Event manager for the cursor manager.

Events details

change

Change to the cursor.

Methods details

push

```
{util.cursor.Accessor} push(key)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Sets a new cursor and writes it to the stack of cursors for the object.

Returns object providing access to the cursor added to the map.

Parameters:

Parameter	Default value	Description
key*	—	Type: String Cursor. Acceptable values: • "arrow" — Arrow cursor. • "crosshair" — Crosshair cursor. • "grab" — Hand cursor. • "grabbing" — Closed hand. • "help" — Arrow cursor with a question mark. • "zoom" — Magnifying glass. • "move" — Cursor consisting of four directional arrows. • "pointer" — Pointing finger. • "inherit" — Inherit the cursor from the parent.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

util.defineClass

Static function.

Base function implementing class declaration in the Yandex.Maps API. Use this function to declare a new class, specify a set of methods for it and inherit from another class.

The "superclass" field is appended to the child class, specifying the prototype of the parent class. Use the 'constructor' field of the 'superclass' object to refer to the constructor of the parent class.

Returns class.

```
{ Object } util.defineClass(constructor[, parentClass[, override]])
```

Parameters:

Parameter	Default value	Description
constructor*	—	Type: Function Class constructor.
parentClass	—	Type: Function Parent class to inherit from. This argument may be omitted.
override	—	Type: Object Set of additional fields and functions that will be appended to the class prototype. There can be several sources (the function can have any number of parameters), and data is copied from right to left (the last argument has highest priority during copying).

* Mandatory parameter/option.

Examples:

1.

```
// Declaring a class with methods.
var MyClass = ymaps.util.defineClass(function () {
   this.field = 'fieldValue';
    }, {
   doSomethingAwesome: function () {
  return 'methodResult';
   },
   stop: function () {

   }
    }),
    object = new MyClass();
    console.log(object.field); // 'fieldValue'
   console.log(object.doSomethingAwesome()); // 'methodResult'
```

2.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
// Creating a custom class for a button, inherited from the base class of the
button control.
// Pressing the button will switch the type of map tiles.
var CustomControl = ymaps.util.defineClass(function () {
    // Defining a limited set of options without a possibility to change them from
outside.
     var data = {
    data: { content: 'Change the map type' },
    options: {
   selectOnClick: false,
   maxWidth: 150
    }
     };
     CustomControl.superclass.constructor.call(this, data);
}, ymaps.control.Button, {
     // Setting a list of class methods.

     // Overriding the button enable and disable methods.
     enable: function () {
    // You must call base class methods
    // to avoid breaking the button logic.
    CustomControl.superclass.enable.call(this);
    // Enabling and disabling button behavior.
    this.events.add('click', this.switchType, this);
     },

     disable: function () {
    this.events.remove('click', this.switchType, this);
    CustomControl.superclass.disable.call(this);
     },

     // Implementing our own methods.
     switchType: function () {
    var map = this.getMap();
    if (map) {
   if (map.getType() == 'yandex#map') {
       this.setSatelliteMapType();
   } else {
       this.setSchemeMapType();
   }
    }
     },

     setSchemeMapType: function () {
    var map = this.getMap();
    if (map) {
   map.setType('yandex#map');
    }
     },

     setSatelliteMapType: function () {
    var map = this.getMap();
    if (map) {
   map.setType('yandex#satellite');
    }
     }
});
// Instantiating a new class and adding it to the map.
var typeSwitcherButton = new CustomControl();
// The map creation code was omitted in this example.
myMap.controls.add(typeSwitcherButton);
// Calling the method of the class instance.
typeSwitcherButton.setSatelliteMapType();
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

util.Dragger

Extends **IEventEmitter**.

A special tool that provides a mechanism for dragging page elements. When using it, note that the mousemove and mouseup events will not register in the Yandex.Maps API events system when working with the dragger.

Constructor | Fields | Events | Methods

Constructor

```
util.Dragger([params])
```

Parameters:

Parameter	Default value	Description
params	—	Type: Object Dragger parameters.
params.autoStartElement	—	Type: HTMLElement|**IDomEventEmitter** DOM element or implementation of the IDomEventEmitter interface, which starts the dragger when clicked.
params.byRightButton	false	Type: Boolean The right mouse button is used for starting the dragger using the 'autoStartElement' parameter. Dragging with the right button won't work on devices without mouse support.
params.tremor	3	Type: Number The minimum distance in pixels from the starting point necessary for starting the dragger.

Example:

```
// Example of dragging a DOM element to the map.
// A placemark is created where the DOM element is released.
var myMap = new ymaps.Map('map', {center: [35.76, 37.67], zoom: 5}),
// The DOM element must have the CSS property position: absolute.
    element = document.getElementById('someId'),
    dragger = new ymaps.util.Dragger({
        autoStartElement: element
    }),
    draggerEventsGroup = dragger.events.group();

draggerEventsGroup
    .add('start', function (event) {
        var pos = event.get('position');
        positionElement(pos[0], pos[1]);
        console.log('start');
    })
    .add('move', function (event) {
        var pos = event.get('position');
        positionElement(pos[0], pos[1]);
        console.log('move');
    })
    .add('stop', function (event) {
        draggerEventsGroup.removeAll();
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
        element.parentElement.removeChild(element);
        // Getting geographical coordinates at the point where the dragger
stopped.
        var placemarkPosition =
myMap.options.get('projection').fromGlobalPixels(
            myMap.converter.pageToGlobal(event.get('position')),
            myMap.getZoom()
        );
        myMap.geoObjects.add(
            new ymaps.Placemark(placemarkPosition)
        );
        console.log('stop');
    });

function positionElement (x, y) {
    element.style.left = x + 'px';
    element.style.top = y + 'px';
}
```

Fields

Name	Type	Description
events	IEventManager	Event manager. Inherited from IEventEmitter.

Events

Name	Description
cancel	Cancels the dragger. This event occurs if the dragger stopped without sending the util.Dragger.start event. For example, if the click stopped without moving the mouse. Instance of the Event class. Names of fields that are available via the Event.get method:
move	Changed position. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • delta — The difference between the locations of the current and previous dragger events.
start	The dragger starts working. This event does not occur at the time of pressing, but at the first change to the position after pressing. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY].
stop	The dragger stops working. This event cannot occur without the util.Dragger.start event. Instance of the Event class. Names of fields that are available via the Event.get method: • position — Coordinates relative to the document. Array in the format [pageX, pageY]. • delta — The difference between the locations of the current and previous dragger events.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods

Name	Returns	Description
destroy()		Stops the dragger and deletes listening to the "mousedown" event for the "autoStartElement" event.
isDragging()	Boolean	Returns whether the dragger is working now.
start(event)		Launches the dragger. This method is automatically called on the "mousedown" event for autoStartElement, if set. This method can be used for on-demand initialization. For example, when getting a response from the server. For correct functioning during on-demand initialization, the passed event must support the "position" field in the "get" method.
stop()		Stops the dragger. This method can be used for stopping the dragger prematurely.

Events details

cancel

Cancels the dragger. This event occurs if the dragger stopped without sending the util.Dragger.start event. For example, if the click stopped without moving the mouse. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

move

Changed position. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- position — Coordinates relative to the document. Array in the format [pageX, pageY].
- delta — The difference between the locations of the current and previous dragger events.

start

The dragger starts working. This event does not occur at the time of pressing, but at the first change to the position after pressing. Instance of the Event class. Names of fields that are available via the **Event.get** method:

- position — Coordinates relative to the document. Array in the format [pageX, pageY].

stop

The dragger stops working. This event cannot occur without the util.Dragger.start event. Instance of the **Event** class. Names of fields that are available via the **Event.get** method:

- position — Coordinates relative to the document. Array in the format [pageX, pageY].
- delta — The difference between the locations of the current and previous dragger events.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

destroy

```
{} destroy()
```

Stops the dragger and deletes listening to the "mousedown" event for the "autoStartElement" event.

isDragging

```
{Boolean} isDragging()
```

Returns whether the dragger is working now.

start

```
{} start(event)
```

Launches the dragger. This method is automatically called on the "mousedown" event for autoStartElement, if set. This method can be used for on-demand initialization. For example, when getting a response from the server. For correct functioning during on-demand initialization, the passed event must support the "position" field in the "get" method.

Parameters:

Parameter	Default value	Description
event*	—	Type: **IDomEvent** The event being initialized.

* Mandatory parameter/option.

stop

```
{} stop()
```

Stops the dragger. This method can be used for stopping the dragger prematurely.

util.extend

Static function.

Function that copies properties from one or several JavaScript objects to another JavaScript object.

```
util.extend(target, ...source)
```

Parameters:

Parameter	Default value	Description
target*	—	Type: Object Target JavaScript object. It will be modified as a result of the function.
source*	—	Type: Object

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Source JavaScript object. All of its properties will be copied. There can be several sources (the function can have any number of parameters), and data is copied from right to left (the last argument has highest priority during copying).

* Mandatory parameter/option.

Example:

```
var options = ymaps.util.extend({
     prop1: 'a',
     prop2: 'b'
}, {
     prop2: 'c',
     prop3: 'd'
}, {
     prop3: 'e'
});
// We get the result: {
//     prop1: 'a',
//     prop2: 'c',
//     prop3: 'e'
// }
```

util.hd

Static object.

Allows working with HD screens on various devices.

Methods

Methods

Name	Returns	Description
getPixelRatio()	Number	Returns ratio of virtual and physical pixels on the screen.
selectRatio(hash)	Number	Returns result of selection — either the pixel ratio, or 1 if, for example, a string was passed.
selectValue(hash)	Object	Returns an object from the passed hash that is equal to or closest to the pixel ratio key. If something other than a hash was passed, such as a string or function, then it is returned.

Methods details

getPixelRatio

```
{Number} getPixelRatio()
```

Returns ratio of virtual and physical pixels on the screen.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

selectRatio

```
{Number} selectRatio(hash)
```

Returns result of selection — either the pixel ratio, or 1 if, for example, a string was passed.

Parameters:

Parameter	Default value	Description
hash*	—	Type: **IRatioMap** Object of type **IRatioMap** for screens with different pixel ratios.

* Mandatory parameter/option.

selectValue

```
{Object} selectValue(hash)
```

Returns an object from the passed hash that is equal to or closest to the pixel ratio key. If something other than a hash was passed, such as a string or function, then it is returned.

Parameters:

Parameter	Default value	Description
hash*	—	Type: Object|**IRatioMap** Object of type **IRatioMap** for screens with different pixel ratios.

* Mandatory parameter/option.

Example:

```
// We need to add a picture to the balloon so it looks good on HD screens.
var balloonImages = {
   // Normal picture.
    "100x100.png",
   // Picture for Retina.
    "2": "200x200.png"
},
 var img = ymaps.util.hd.selectValue(balloonImages);
// Adding the point to the map.
myMap.geoObjects.add(new ymaps.GeoObject({
    geometry: "Point",
    coordinates: [55, 37],
    properties: {
        balloonContent:
            '&lt;img src="' + img + '" width="100" alt="" &gt;'
}, {});
```

util.math

util.math.areEqual

Static function.

Compares two points with an error allowance.

Returns result of comparison.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
{ Boolean } util.math.areEqual(first, second[, diff])
```

Parameters:

Parameter	Default value	Description
first*	—	Type: Number[] Array for comparison.
second*	—	Type: Number[] Array to compare.
diff	—	Type: Number Precision of comparison.

* Mandatory parameter/option.

util.math.cycleRestrict

Static function.

Assigns a numeric value to the set range. The range of values is considered to be closed in a ring. If a value is outside one of the ends of the range, the extra is counted off around the circl from the other end.

Returns value can be limited.

```
{ Number } util.math.cycleRestrict(value, min, max)
```

Parameters:

Parameter	Default value	Description
value*	—	Type: Number Value can be limited.
min*	—	Type: Number Minimum limit.
max*	—	Type: Number Maximum limit.

* Mandatory parameter/option.

Example:

```
// Returns 110
ymaps.util.math.cycleRestrict(-250, -180, 180);
// Returns 60
ymaps.util.math.cycleRestrict(-300, -180, 180);
// Returns -170
ymaps.util.math.cycleRestrict(190, -180, 180);
```

util.math.restrict

Static function.

Restricts an input numeric value to the set minimum and maximum limits.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Returns value can be limited.

```
{ Number } util.math.restrict(value, min, max)
```

Parameters:

Parameter	Default value	Description
value*	—	Type: Number Value can be limited.
min*	—	Type: Number Minimum limit.
max*	—	Type: Number Maximum limit.

* Mandatory parameter/option.

Example:

```
ymaps.util.math.restrict(-250, -180, 180) = -180
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

util.pixelBounds

Static object.

Methods

Methods

Name	Returns	Description
areIntersecting(bounds1, bounds2)	Boolean	Determines whether two rectangular areas intersect.
containsBounds(outer, inner)	Boolean	Determines whether a rectangular area completely contains another rectangular area.
containsPoint(bounds, point)	Boolean	Determines whether a rectangular area contains a point.
fromBounds(sourceBounds)	Number[][]	Calculates the rectangular area that everything passed falls inside of.
fromPoints(points)	Number[][]	Calculates the rectangular area that the passed points fall inside of.
getCenter(bounds)	Number[]	Calculates the center of the rectangular area.
getIntersection(bounds1, bounds2)	Number[][]|null	Calculates the intersection of two rectangular areas.
getSize(bounds)	Number[]	Calculates the dimensions of a rectangular area.

Methods details

areIntersecting

```
{Boolean} areIntersecting(bounds1, bounds2)
```

Determines whether two rectangular areas intersect.

Returns intersection attribute.

Parameters:

Parameter	Default value	Description
bounds1*	—	Type: Number[][] First area.
bounds2*	—	Type: Number[][] Second area.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

containsBounds

```
{Boolean} containsBounds(outer, inner)
```

Determines whether a rectangular area completely contains another rectangular area.

Returns inclusion attribute.

Parameters:

Parameter	Default value	Description
outer*	—	Type: Number[][] External area
inner*	—	Type: Number[][] The area being checked.

* Mandatory parameter/option.

containsPoint

```
{Boolean} containsPoint(bounds, point)
```

Determines whether a rectangular area contains a point.

Returns inclusion attribute.

Parameters:

Parameter	Default value	Description
bounds*	—	Type: Number[][] External area
point*	—	Type: Number[] The point being checked.

* Mandatory parameter/option.

fromBounds

```
{Number[][]} fromBounds(sourceBounds)
```

Calculates the rectangular area that everything passed falls inside of.

Returns the calculated area.

Parameters:

Parameter	Default value	Description
sourceBounds*	—	Type: Number[][][] Array of rectangular areas

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

fromPoints

```
{Number[][]} fromPoints(points)
```

Calculates the rectangular area that the passed points fall inside of.

Returns the calculated area.

Parameters:

Parameter	Default value	Description
points*	—	Type: Number[][] Array of points.

* Mandatory parameter/option.

getCenter

```
{Number[]} getCenter(bounds)
```

Calculates the center of the rectangular area.

Returns center point in the coordinate system of the input data.

Parameters:

Parameter	Default value	Description
bounds*	—	Type: Number[][] Area.

* Mandatory parameter/option.

getIntersection

```
{Number[][]|null} getIntersection(bounds1, bounds2)
```

Calculates the intersection of two rectangular areas.

Returns the rectangular area that is formed by the intersection of the passed areas, or `null` if the areas do not intersect.

Parameters:

Parameter	Default value	Description
bounds1*	—	Type: Number[][] First area.
bounds2*	—	Type: Number[][] Second area.

* Mandatory parameter/option.

getSize

```
{Number[]} getSize(bounds)
```


Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Calculates the dimensions of a rectangular area.

Returns size of the area.

Parameters:

Parameter	Default value	Description
bounds*	—	Type: Number[][] Area.

* Mandatory parameter/option.

util.requireCenterAndZoom

Static function.

Calculates the optimal center and zoom level of the map to display the passed area on the specified type of map. The zoom level will be within the available zoom range.

Returns the promise object, which will be confirmed by an object with the center and zoom fields, or will be rejected with an error.

```
{ vow.Promise } util.requireCenterAndZoom(mapType, bounds, containerSize[, params])
```

Parameters:

Parameter	Default value	Description
mapType*	—	Type: String|**MapType**|**map.ZoomRange** Map type. Key string from **mapType.storage**, or an instance of the **MapType** class. Or the manager of map zoom coefficients for a specific map.
bounds*	—	Type: Number[][] An area set in geographical coordinates. The first point contains the minimum values for latitude and longitude, and the second point contains the maximum values.
containerSize*	—	Type: Number[] Size of the map container.
params	—	Type: Object Parameters.
params.inscribe	true	Type: Boolean If true, fit the area into the map; if false, fit the map into the area.
params.margin	0	Type: Number|Number[]

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		Offset from the borders of the visible area of the map. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins.
params.preciseZoom	false	Type: Boolean When the value is "false", the zoom level is rounded down.

* Mandatory parameter/option.

Example:

```
// Finding the optimal center and zoom level of the map..
ymaps.util.requireCenterAndZoom(
    myMap.getType(),
    [[50.531219,31.278264], [50.966841,31.964909]],
    myMap.container.getSize()
).then(function (result) {
    // Setting the optimal center and zoom level of the map.
    myMap.setCenter(result.center, result.zoom);
});
```

util.Storage

Object storage by keys.

Constructor | Methods

Constructor

```
util.Storage()
```

Methods

Name	Returns	Description
add(key, object)	**util.Storage**	Adds an object to storage.
get(key)	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
remove(key)	**util.Storage**	Deletes the "key: value" pair from storage.

Methods details

add

```
{util.Storage} add(key, object)
```

Adds an object to storage.

Returns self-reference.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameters:

Parameter	Default value	Description
key*	—	Type: String Key.
object*	—	Type: Object Stored object.

* Mandatory parameter/option.

get

```
{Object} get(key)
```

Returns object stored for the specified key, or the primary key, if this is not a string.

Parameters:

Parameter	Default value	Description
key*	—	Type: String|Object Key.

* Mandatory parameter/option.

remove

```
{util.Storage} remove(key)
```

Deletes the "key: value" pair from storage.

Returns self-reference.

Parameters:

Parameter	Default value	Description
key*	—	Type: String Key.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

vow

Static object.

Contains methods for creating and processing promise objects.

Note:
This class is a part of the Vow library.

Only some of the methods are described below. The complete list of methods is available here: http://dfilatov.github.io/vow/.

Copyright (c) 2012-2013 Filatov Dmitry (dfilatov@yandex-team.ru). Dual licensed under the MIT and GPL licenses.

Methods

Methods

Name	Returns	Description
all(iterable)	**vow.Promise**	Returns a promise object that will be either allowed or denied only if all the specified objects will be accepted or rejected, correspondingly.
defer()	**vow.Deferred**	Creates a new deferred object. The same as `new ymaps.vow.Deferred()`.
reject(reason)	**vow.Promise**	Returns a promise object rejected with the specified reason.
resolve(value)	**vow.Promise**	Returns the promise object which is accepted with the specified value.

Methods details

all

```
{vow.Promise} all(iterable)
```

Returns a promise object that will be either allowed or denied only if all the specified objects will be accepted or rejected, correspondingly.

Parameters:

Parameter	Default value	Description
iterable*	—	Type: Object|Object[] Set of promise objects and/or values.

* Mandatory parameter/option.

Examples:

1.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

```
var deferred1 = ymaps.vow.defer(),
    deferred2 = ymaps.vow.defer();

ymaps.vow.all([deferred1.promise(), deferred2.promise(), 3])
    .then(function(value) {
         // value =&gt; [1, 2, 3]
    });

deferred1.resolve(1);
deferred2.resolve(2);
```

2.

```
var deferred1 = ymaps.vow.defer(),
    deferred2 = ymaps.vow.defer();

ymaps.vow.all({ p1 : deferred1.promise(), p2 : deferred2.promise(), p3 : 3 })
    .then(function(value) {
         // value => { p1 : 1, p2 : 2, p3 : 3 }
    });

deferred1.resolve(1);
deferred2.resolve(2);
```

defer

```
{vow.Deferred} defer()
```

Creates a new deferred object. The same as `new ymaps.vow.Deferred()`.

See vow.Deferred

Returns a deferred object.

reject

```
{vow.Promise} reject(reason)
```

Returns a promise object rejected with the specified reason.

Parameters:

Parameter	Default value	Description
reason*	—	Type: Object The reason for rejection.

* Mandatory parameter/option.

resolve

```
{vow.Promise} resolve(value)
```

Returns the promise object which is accepted with the specified value.

Parameters:

Parameter	Default value	Description
value*	—	Type: Object Value.

* Mandatory parameter/option.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

vow.Deferred

A class which describes the deferred objects.

Note:
This class is a part of the Vow library. Only some of the methods are described below. The complete list of methods is available here: http://dfilatov.github.io/vow/. Copyright (c) 2012-2013 Filatov Dmitry (dfilatov@yandex-team.ru). Dual licensed under the MIT and GPL licenses.

Note:
It is not a stand-alone module: it is available only if the **vow** module is connected.

Constructor | Methods

Constructor

```
vow.Deferred()
```

Creates a deferred object.

Example:

```
function someAsyncMethod () {
    var deferred = new ymaps.vow.Deferred(); // or `var deferred =
ymaps.vow.defer();`

    doSomeAsyncStuff(function (err, value) {
        if (err) {
            deferred.reject(err);
            return;
        }

        deferred.resolve(value);
    });

    return deferred.promise();
}

someAsyncMethod().then(function (value) {
    console.log('The method result: ' + value);
}, function (err) {
    console.log('Error: ' + err);
});
```

Methods

Name	Returns	Description
promise()	**vow.Promise**	Returns the associated promise object.
reject(reason)		Rejects the associated promise object with the specified reason.
resolve(value)		Accepts the associated promise object with the specified value.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

promise

```
{vow.Promise} promise()
```

Returns the associated promise object.

reject

```
{} reject(reason)
```

Rejects the associated promise object with the specified reason.

Parameters:

Parameter	Default value	Description
reason*	—	Type: Object The reason for rejection.

* Mandatory parameter/option.

resolve

```
{} resolve(value)
```

Accepts the associated promise object with the specified value.

Parameters:

Parameter	Default value	Description
value*	—	Type: Object Value.

* Mandatory parameter/option.

vow.Promise

A class which describes the promise objects.

The Promise/A+ specification.

Note:
This class is a part of the Vow library. Only some of the methods are described below. The complete list of methods is available here: http://dfilatov.github.io/vow/. Copyright (c) 2012-2013 Filatov Dmitry (dfilatov@yandex-team.ru). Dual licensed under the MIT and GPL licenses.

Note:
It is not a stand-alone module: it is available only if the **vow** module is connected.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Constructor | Methods

Constructor

```
vow.Promise([resolver])
```

Creates a promise object.

Parameters:

Parameter	Default value	Description
resolver	—	Type: Function The function which takes the resolve and reject methods as parameters to set status and values for the created promise object.

Example:

```
function someAsyncMethod () {
    return new ymaps.vow.Promise(function (resolve, reject) {
        doSomeAsyncStuff(function (err, value) {
            if (err) {
                reject(err);
                return;
            }

            resolve(value);
        });
    });
}

someAsyncMethod().then(function (value) {
    console.log('The method result: ' + value);
}, function (err) {
    console.log('Error: ' + err);
});
```

Methods

Name	Returns	Description
done([onFulfilled[, onRejected[, onProgress[, ctx]]]])		Similar to the method **vow.Promise.then**, which closes the promise chain. Throws an exception if the promise object is rejected.
spread([onFulfilled[, onRejected[, ctx]]])	**vow.Promise**	Similar to the method **vow.Promise.then**, which calls the callback functions with a set of arguments (corresponding to an array) for which the promise object can be either allowed or rejected. Usually it is used in combination with methods like **vow.all**.
then([onFulfilled[, onRejected[, onProgress[, ctx]]]])	**vow.Promise**	Specifies a handler function for a promise object.
valueOf()	Object	Returns a value for an allowed promise object or a rejection reason for a rejected one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Methods details

done

```
{} done([onFulfilled[, onRejected[, onProgress[, ctx]]]])
```

Similar to the method **vow.Promise.then**, which closes the promise chain. Throws an exception if the promise object is rejected.

Parameters:

Parameter	Default value	Description
onFulfilled	—	Type: Function The callback function which will be called if the promise object is resolved.
onRejected	—	Type: Function The callback function which will be called if the promise object is rejected.
onProgress	—	Type: Function The callback function which will be called when "notifying" the promise object.
ctx	—	Type: Object The execution context of the callback functions.

Example:

```
var deferred = ymaps.vow.defer();
deferred.reject(Error('Internal error'));
deferred.promise().done(); // Throws an exception.
```

spread

```
{vow.Promise} spread([onFulfilled[, onRejected[, ctx]]])
```

Similar to the method **vow.Promise.then**, which calls the callback functions with a set of arguments (corresponding to an array) for which the promise object can be either allowed or rejected. Usually it is used in combination with methods like **vow.all**.

Returns a new promise object.

Parameters:

Parameter	Default value	Description
onFulfilled	—	Type: Function The callback function which will be called if the promise object is resolved.
onRejected	—	Type: Function

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Parameter	Default value	Description
		The callback function which will be called if the promise object is rejected.
ctx	—	Type: Object The execution context of the callback functions.

Example:

```
var deferred1 = ymaps.vow.defer(),
    deferedr2 = ymaps.vow.defer();

ymaps.vow.all([deferred1.promise(), deferred2.promise()]).spread(function(arg1,
arg2) {
    // arg1 =&gt; 1, arg2 =&gt; 'two'
});

deferred1.resolve(1);
deferred2.resolve('two');
```

then

```
{vow.Promise} then([onFulfilled[, onRejected[, onProgress[, ctx]]]])
```

Specifies a handler function for a promise object.

Returns a new promise object. See the specification.

Parameters:

Parameter	Default value	Description
onFulfilled	—	Type: Function The callback function which will be called if the promise object is resolved.
onRejected	—	Type: Function The callback function which will be called if the promise object is rejected.
onProgress	—	Type: Function The callback function which will be called when "notifying" the promise object.
ctx	—	Type: Object The execution context of the callback functions.

valueOf

```
{Object} valueOf()
```

Returns a value for an allowed promise object or a rejection reason for a rejected one.

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/

Yandex

Yandex.Maps API Reference

Version 2.1.63

13.04.2018

Antenna House XSL Formatter (Evaluation) http://www.antennahouse.com/