Geomajas JavaScript Documentation 1.0.0-SNAPSHOT

Geomajas Developers and Geosparc

Geomajas JavaScript Documentation 1.0.0-SNAPSHOT

by Geomajas Developers and Geosparc

Version 1.0.0-SNAPSHOT
Copyright © 2014-2015 Geosparc nv

Table of Contents

Chapter 1. Introduction

The Geomajas Javascript project provides the javascript based version of the Geomajas functions. Most of the functions of the Geomajas api are available in javascript code.

Chapter 2. Configuration

This section explains how to use this library in your project.

1. Sources

- Online
- ZIP file
- maven dependency
- JAR file

2. Online

The javascript files are available online: http://dev.geomajas.org/geomajas-project-javascript-gwt2-distribution-1.0.0-SNAPSHOT/ Note: to save bandwidth and guarantee availability you are encouraged to host the javascript yourself, especially for production environments.

3. Usage of the zip file

This package contains and index.html example with the needed js files.

4. Usage of the dependency

Create a Web project and add the following to your pom file. This will output the needed js file to the selected output directory.

```
<dependency>

<groupId>org.geomajas.project</groupId>

<artifactId>geomajas-project-javascript-gwt2-impl</artifactId>

<version>1.0.0-SNAPSHOT</version>

   </dependency>



<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-dependency-plugin</artifactId>

<executions>
```

```
<execution>

<id>unpack assembly</id>


<phase>generate-sources</phase>


<goals>

<goal>unpack</goal>

      </goals>


<configuration>

<artifactItems>

<artifactItem>

<groupId>org.geomajas.project</groupId>


<artifactId>geomajas-project-javascript-gwt2-impl</artifactId>


<version>1.0.0-SNAPSHOT</version>

         jar
        </artifactItem>

       </artifactItems>


<excludes>
        **/WEB-INF/**,
        **/META-INF/**,
        **/org/**,
        index.html
       </excludes>


<outputDirectory>${basedir}/src/main/webapp</outputDirectory>

      </configuration>

     </execution>

    </executions>

   </plugin>
```

5. Usage of the jar file

Copy the jar file to the WEB-INF/lib folder of your web project. You only need to include the 'gm/gm.js' file in your page to make use of the api.

Chapter 3. How-to Javascript Geomajas

After configuration, Geomajas javascript can create maps, layers, listens to map events, ... When the web application is started up, the geomajas code will look for and call function "onGeomajasLoad". This body of this function can be customized to perform certain actions. Often, it is useful to create and initialize the map(s) within this function.

```
function onGeomajasLoad() {
   // something can be done
}
```

1. Creating and initializing a map.

A map can be added to a div or span element on the page; this will be the parent element. The parent element will need to have a distinctive id, that is passed to the map at construction. This map can then be initialized, using predefined information. Currently, initialization can only use server side configured applications and mapConfigurations.

The following code will create a map under element with id "idOfParentElement", then initialize the map with the server side configurations "app" for application and "mapMain" for mapConfiguration.

```
var map = new gm.Map("idOfParentElement");
gm.ServerExtension.initializeMap(map, "app", "mapMain");
```

2. Events from the map

A number of types of events are fired when the map is constructed or changed. It is possible to register handlers to these events, to perform personalized code when they are fired.

The following code show registration for `MapInitialization` and `Layer events`.

```
map.getEventBus().addLayerAddedHandler(function(event) {
   // custom code; event contains the added layer
   alert('layer added: ' + event.getAddedLayer().getTitle());
});
map.getEventBus().addLayerRemovedHandler(function(event) {
   // custom code; event contains the removed layer
   alert('layer removed: ' + event.getRemovedLayer().getTitle());
});
map.getEventBus().addMapInitializationHandler(function(event) {
   // custom code
   alert('map fully initialized');
});
```

3. LayersModel: Change layers

The layers of the map can be looked up via the map's `LayersModel`. The display of the layers on the map can be changed, but they can also be added or removed form the map entirely.

The following code show some features of the `LayersModel`.

```
var layersModel = map.getLayersModel();
var layerCount = layersModel.getLayerCount();
```

```
// a layer can be found via index or by string id
var firstLayer = layersModel.getLayerAtIndex(0);
var backgroundLayer = layersModel.getLayer('background'); // background layer c
// change the layers appearance
firstLayer.setOpacity(0.5);
backgroundLayer.setMarkedAsVisible(false);
firstLayer.refresh();
// add or remove layers from the layersModel
layersModel.removeLayer(firstLayer);
layersModel.addLayer(firstLayer);
```

4. Map cursor

The map's cursor can be changed easily. Any css value of the cursor can be set. It is also possible to set a custom image.

```
map.setCursor('default'); // set to default
map.setCursor('crosshair'); // set cursor css value to crosshair
map.setCursor('url(images/customImage.cur), auto'); // custom cursor image
```

5. Map Controllers

The mouse events on a map can be processed by a `MapController`. Only one map controller can be active at the time. The default map controller is the `NavigationController`, that enables panning and zooming.

The map controller can be set to one of the predefined controllers, via code below.

5.1. Predefined Map Controllers

```
map.setMapController(gm.MapControllerFactory.createMapController(controllerKey)
```

The string 'controllerKey' is a unique key for the predefined controller; supported keys can be found in `JsMapControllerFactoryImpl`. Currently 'controllerKey' can take following values:

- 'navigation' for `NavigationController`
- 'zoomToRectangle' for `ZoomToRectangleController`
- 'featureSelectionDrag' for `FeatureSelectionController` with `SelectionMethod.CLICK_AND_DRAG`
- 'featureSelectionSingle' for `FeatureSelectionController` with `SelectionMethod.SINGLE_SELECTION`

5.2. Custom Map Controller

The map's controller can also be set to a fully custom one. This controller can be created via the `MapControllerFactory`, now without providing a controllerKey. The custom controller's handlers can then be set to perform requested actions. Finally, set the custom controller to the map, in order to activate it.

```
// Create the custom MapController:
var customMapController = gm.MapControllerFactory.createMapController();

// Apply handlers for Mouse Events:
customMapController.setMouseMoveHandler(function(event) {
```

```
   // custom code; event contains the location of the mouse; this can be retriev
   var screenLocation = mapController.getLocation(event, "screen");
   var worldLocation = mapController.getLocation(event, "world");
   var screenLocationAsText = "Screen: " + screenLocation.getX() + ", " + screen
   var worldLocationAsText = "World: " + worldLocation.getX() + ", " + worldLoca
   // then show location ...
});
// other mouse controllers setters:
//setMouseOverHandler
//setMouseOutHandler
//setDownHandler
//setUpHandler
//setDragHandler
//setDoubleClickHandler

// Apply activation (for init) and deactivation (for cleanup) handlers:
customMapController.setActivationHandler(function() {
   // custom code
   alert('Custom controller activated!')
});
customMapController.setDeactivationHandler(function() {
   // custom code
   alert('Custom controller deactivated!')
});

// Apply the MapController on the map:
map.setMapController(customMapController);
```

6. Features

Some vector layers contain a number of distinguishable object or features. Their features can be selected individually. A number of methods are available to query/change the features. Only layers that implement the `JsFeaturesSupported` interface will respond to these methods.

6.1. Feature search

The map's `FeatureSearchService` offers some methods to retrieve features.

```
var service = map.getFeatureSearchService();
// search for a single feature of a layer
service.searchById(layer, [id], function(featureHolder){
   // custom code; the featureHolder contains the features
   var feature = featureHolder.getFeatures()[0];
   alert("Feature found: " + feature.getLabel());
});
// search for all features of a layer within some bounds
service.searchInBounds(layer, bounds, function(features) {
   // custom code; the featureHolder contains the features
   alert("Features found: " + featureHolder.getFeatures().size());
});
```

6.2. Feature selection

Features can be selected via the map by a user, when a `FeatureSelectionController` is set, see above.

Feature selection is possible in javascript code:

```
var layer = ...; // layer with feature
var feature = ...; // from the layer; retrieved from FeatureSearchService or fr
// is the feature selected or not?
var selected = eval(layer.isFeatureSelected(feature.getId()).toString()); // Bo
// select the specific feature
layer.selectFeature(feature);
// deselect the specific feature
layer.deselectFeature(feature);
```

When a feature is selected or deselected, events are fired on the map's event bus. It is possible to register custom handlers for the `FeatureSelected` and `FeatureDeselected` events:

```
// the addFeatureSelectionHandler takes two parameters
var registration = map.getEventBus().addFeatureSelectionHandler(
   // first parameter is a featureSelectedHandler
   function(event) {
      // custom code; event contains feature that was selected
      var selectedFeature = event.getFeature();
   },
   // second parameter is a featureDeselectedHandler
   function(event) {
      // custom code; event contains feature that was deselected
      var deselectedFeature = event.getFeature();
   }
);
```

At any time, a layer's selected features can be found. They can also be deselected by one call:

```
var features = layer.getSelectedFeatures(); // get selected features
layer.clearSelectedFeatures(); // deselect all features
```