



JavaScript-ActionScript Linkage for InstantAtlas Flash Templates

Theme 1 >> Indicator 1 >> 2005

© XYZ Mapping Company 2007

Data
Legend 1
Filter
Post Code Sectors

Quantile Legend

- 1.4 - 3.4
- 3.5 - 4.1
- 4.2 - 5.8
- 5.9 - 7.1
- 7.2 - 16.2

Map Layers

- post_code_districtsCopy
- Background Mapping

Help Clear

Name ▲	Map 1	Count	Map 1 ▲
EH1 1	8.41	No Data	2
EH1 2	7.01	No Data	16
EH1 3	7.12	No Data	19
EH10 4	3.68	No Data	9
EH10 5	3.58	No Data	5
EH10 6	2.18	No Data	3
EH10 7	1.73	No Data	3
EH11 1	4.94	No Data	8
EH11 2	6.16	No Data	11
EH11 3	8.8	No Data	17
EH11 4	10.19	No Data	11
EH12 0	3.92	No Data	7
EH12 1	No Data	No Data	No D
FH12 5	3.67	No Data	7

Theme 1 >> Indicator 1 >> 2004

© XYZ Mapping Company 2007

Data
Legend 2
Filter
Post Code Sectors

Quantile Legend

- 0.0 - 5.2
- 5.3 - 7.7
- 7.8 - 11.0
- 11.1 - 14.7
- 14.8 - 39.2

Correlation coefficient (r) = 0.9 >> r-squared = 0.8 >> Regression Equation: y = 2.0x + -0.3

Indicator 1 >> 2004

Indicator 1 >> 2005

instantatlas™
© Copyright Geowise Ltd.

Version 4.1



Contents

Introduction	1
Getting a Reference to the Movie	1
Functions	1



Introduction

These functions only become available to JavaScript if the parameter **enableJavaScript** is appended to the URL that calls the atlas e.g. **atlas.html?enableJavaScript=true** or **atlas.swf?enableJavaScript=true**. This can also be achieved by using code in the enclosing HTML page (see SWFObject documentation for the addVariable function).

Getting a Reference to the Movie

Use some JavaScript like this to create the movie:

```
<script type="text/javascript">
  // 
  var so = new SWFObject("atlas.swf", "instantAtlasReport", "88%", "88%", "8", "#ffffff", false);
  var qs = new Array('config', 'indicator', 'date', 'select', 'filter');
  for (var i = 0; i &lt; qs.length; i++) {
    if (getQueryParamValue(qs[i]) != "")
      so.addVariable(qs[i], getQueryParamValue(qs[i]));
  }
  so.addVariable('enableJavaScript', 'true');
  so.write("flashcontent");
// ]]&gt;&lt;/script&gt;</pre>
</div>
<div data-bbox="91 483 486 500" data-label="Text">
<p>And some like this to get a reference to the movie:</p>
</div>
<div data-bbox="91 509 554 627" data-label="Text">
<pre>function thisMovie(movieName) {
  if (navigator.appName.indexOf("Microsoft") != -1) {
    return window[movieName];
  }
  else {
    return document[movieName];
  }
}
var myMovie = thisMovie('instantAtlasReport');</pre>
</div>
<div data-bbox="91 652 225 671" data-label="Section-Header">
<h2>Functions</h2>
</div>
<div data-bbox="91 685 439 699" data-label="Section-Header">
<h3>addMapZoomListener(externalFunction)</h3>
</div>
<div data-bbox="91 697 901 738" data-label="Text">
<p>where <i>externalFunction</i> is a string that is the name of the JavaScript function in the enclosing page. Whenever the map is zoomed this function will be called with one argument – the “view box” of the map in pixels as a space-delimited string of form “x y width height” e.g. “0 0 400 400”</p>
</div>
<div data-bbox="91 750 433 764" data-label="Section-Header">
<h3>addMapZoneListener(externalFunction)</h3>
</div>
<div data-bbox="91 763 900 842" data-label="Text">
<p>where <i>externalFunction</i> is a string that is the name of the JavaScript function in the enclosing page. Whenever a zone in the map interacts with the mouse this function will be called with two arguments – the “mouse event” (one of “over”, “out” or “click”) and the ID of the zone that generated the event (e.g. “map1~EH111”). For the double map you will be allowed one more parameter – <i>mapId</i> (“map1” or “map2”) i.e. <b>addMapZoneListener(externalFunction, mapId)</b>. If the <i>mapId</i> is not specified, the default behaviour will be to add the listener to <i>both</i> maps.</p>
</div>
<div data-bbox="91 852 360 868" data-label="Section-Header">
<h3>zoomToMapFeature(featureId)</h3>
</div>
<div data-bbox="91 865 900 894" data-label="Text">
<p>where <i>featureId</i> is a string that uniquely identifies the feature to zoom to. Note that is NOT necessary to prepend “map1~” to the feature ID – “EH111” would work in this case...</p>
</div>
<div data-bbox="91 903 486 920" data-label="Section-Header">
<h3>mapZoomIn(factor) and mapZoomOut(factor)</h3>
</div>
<div data-bbox="91 941 282 956" data-label="Page-Footer">
<p>Date: 17/05/07, ©GeoWise Ltd.</p>
</div>
<div data-bbox="475 941 859 956" data-label="Page-Footer">
<p>JavaScript-ActionScript Linkage for Flash Templates - 1 -</p>
</div>
```



should be self explanatory! *factor* is the zoom factor e.g. a factor of 2 is the standard, 3 is a "stronger" zoom, 1.5 is a "weaker" zoom... If factor is omitted, it defaults to 2.

getMapLayers()

gets an array (of strings) of the contextual and background layer IDs in the map

getMapLayerSummary(layerId)

gets an object that contains basic information about the given map layer, identified by *layerId* (string)

setMapLayerVisible(layerId, visible)

set the visibility of the given map layer, identified by *layerId* (string) where *visible* is Boolean true/false

showTooltip(text)

shows a tooltip in the Flash report at the current mouse position. **text** is the text to display. To clear the tooltip call **movie.setTooltip('')**

showDialog(title, message, messageType)

shows the built-in dialog in the report (e.g. the one you see when it is an eval report). *messageType* should be one of "information", "warning" or "error"

addDataListener(externalFunction)

where *externalFunction* is a string that is the name of the JavaScript function in the enclosing page. Whenever a data action occurs in the report this function will be called with one argument – a "data event" object which will have at least one property – "type" – which will be one of "dataLoad", "dataError", "themeLoad", "indicatorChange", "appLoaded". You should be able to work out what to do with them!

loadThemeFrom(url, indicatorId, dateId)

loads in (and replaces) a theme from a given URL. If *indicatorId* and *dateId* are specified, these will become the displayed indicator, otherwise the first indicator found in the theme file will become the current indicator. For the double map there will be a subtly different function: **loadThemeFrom(url, indicatorId, dateId, mapId)** where *mapId* will be "map1" or "map2"

setData(themeId, indicatorId, dateId)

should be easy enough to work out! For the double map there will be a subtly different function:

setData(themeId, indicatorId, dateId, mapId) where *mapId* will be "map1" or "map2"

getIndicatorSummary(indicatorId) and **getThemeSummary(themeId)** and **getDataSummary()**

should, equally, be easy enough! They all return simple objects that mirror *either* the current theme, indicator etc. or the theme or indicator specified by the *themeId* or *indicatorId* parameters (if supplied). So if the report is currently showing theme one, indicator one for 2005 and that indicator has a custom property "myProp", you would get something like this:

```
var io = thisMovie('instantAtlasReport').getIndicatorSummary();
var ioId = io.id; // value is 'i1'
var ioName = io.name; // value is 'Indicator One'
var ioDate = io.date; // value is '2005'
var ioDates = io.dates; // value is '2005,2006,2007' i.e comma-delimited string of dates
var ioNotes = io.href; // value is './notes.htm'
var ioCustom = io.myProp; // value is 'My custom property value'
```

Double map will be similar except all *getXXXSummary* functions will take one extra parameter – the map ID – "map1" or "map2"

getMapFeature(featureId)

get a simple object that mirrors the given feature, including the current indicator value etc. Double map will be similar except it will be **getMapFeature(featureId, includeDataForMapId)** – where *includeDataForMapId* is "map1" or "map2". The values the object has will then be the current values for that map (if the maps are showing different indicators)