



Flash Under Javascript and XML Nuances

Version 1.0

October 25, 2007

Introduction

This document provides a brief overview of using the Flash Map under Javascript wrapper. For brevity, Flash under Javascript will be abbreviated to FuJAX in the remainder of this document.

Note

FuJAX contains advanced functions not found in the pure Javascript Toolkit. These include:

- Globe View: Show a revolvable globe at the top zoom level
 - Smooth Zoom: Animate zooming in and out
 - Rollover and InfoWindow colors and transparency
-

See the following sections for more information.

Getting Started

Before trying to use FuJAX, you must complete these steps:

1. When utilizing FuJAX, a **crossdomain.xml** file must be located in the web server's webroot to allow the **TilemapJS.swf** to communicate with the FuJAX page. If no **crossdomain.xml** file is present, custom POI icons will not resize and image overlays will not function.
2. When utilizing overlays with the FuJAX it is not necessary to call `MQInitOverlays()`, as is required in the JS Toolkit.
3. 5. When utilizing the JSToolkit, it is possible to position map controls outside of the `div` containing the map. This is not possible in FuJAX because the controls must appear within the Flash Player(`div`).
4. Set the constructor for `MQTileMap` to:

```
myMap=new MQTileMap(<div>,<zoomlevel>,<centerLatLng><maptype>  
<mapinit><callback>);
```

Using Internet Explorer Version 7 (IE7)

IE7 cannot utilize `mapDiv.style.visible = hidden/visible` using the `TilemapJS.swf` file. An alternative is to set the div size to 0, 0 to hide the div and adjust it appropriately when the div should be exposed.

To Create the Initial Div:

```
<div id="mapDiv" style="width:0px; height:0px;"></div>
```

To Display the Div:

```
myMap.setSize(new MQSize(520,900));  
document.getElementById("mapDiv").style.height = "520";  
document.getElementById("mapDiv").style.width = "900";
```

To Hide the Div:

```
myMap.setSize(new MQSize(0,0));  
document.getElementById("mapDiv").style.height = "0";  
document.getElementById("mapDiv").style.width = "0";
```

Using Colors

- All FuJAX references to colors, whether for use in overlays or as part of an HTML string for an InfoWindow, must be made in hexadecimal format. The use of RGBA arrays or HTML color names is not supported in the FuJAX.

InfoWindows and Rollovers

It is possible to set rollover and infowindow background colors and alphas in the FuJAX, as shown below:

```
myMap.getRolloverWindow().setBackgroundColor("#80CC8C");  
myMap.getRolloverWindow().setBackgroundAlpha(.80);  
myMap.getInfoWindow().setBackgroundColor("#D0D0B0");  
myMap.getInfoWindow().setBackgroundAlpha(.80);
```

Globe View

The Globe View feature enables you to set the map to show the globe at the maximum zoom out level to show the globe. The following code sample shows the Globe View Settings.

```
//Globe View settings  
var gs = new globeSettings();  
gs.showGlobe = true; //turns on Globe at highest out zoom level  
gs.showShading = true; //turns on shading  
myMap.setGlobeSettings(gs);
```

Smooth Zoom

It is possible to utilize Smooth Zoom functionality in the FuJAX. With the Smooth Zoom feature, you have some properties available to you that you can set:

- Animate
- Smoothing
- Easing
- Time
- Interval
- Fade Time

The following steps and code samples show the aspects of an implementation of the Smooth Zoom functionality.

1. Initialize the map using the `initMap()` function.

```
function initMap() {  
    myMap = new MQTileMap(document.getElementById('mapDiv'),  
        1, new MQLatLng(38.895, -77.036697), "map", null,  
        "onMapCreation");  
}
```

2. Enable the map smoothing. (Turns on Pixel smoothing - heavy on CPU usage.)

```
myMap.getZoomSettings().setAnimate(true);
```

3. Enable the Smooth Zoom feature.

```
myMap.getZoomSettings().setSmoothing(true);
```

4. Set the animation type. Options here include:

- `Linear.easeOut`
Starts a linear motion fast and then decelerates motion to a zero velocity as it executes.
- `Linear.easeIn`
Starts a linear motion from zero velocity and then accelerates motion as it executes.
- `Linear.easeNone`
Defines a constant motion with no acceleration.
- `Linear.easeInOut`
Combines the motion of the `easeIn()` and `easeOut()` methods to start the motion from zero velocity, accelerate motion, then decelerate back to zero velocity.
- `Elastic.easeOut`
Starts the linear motion slowly, accelerates the motion, and then decelerates the motion.
- `Quadratic.easeOut`
Starts a quadratic motion fast and then decelerates motion to a zero velocity as it executes.

-
- `Bounce.easeIn`

Starts a bounce motion fast and then decelerates motion to a zero velocity as it executes.

```
myMap.getZoomSettings().setEaseFunction(Animation.Type);
```

5. Set the time of the animation type. (Milliseconds the Zoom time is animated.)

```
myMap.getZoomSettings().setTime(parseInt  
    (document.getElementById('txtTime').value));
```

6. Set the fade time (milliseconds it takes to fade from 1 tileset to the next *after* Zoom).

```
myMap.getZoomSettings().setFadeTime(parseInt  
    (document.getElementById('txtFade').value));
```

Throwable Maps

Throwable maps enables you to allow the user to drag and throw the globe view and have the globe spin at the rate specified by the parameters set in the following code:

```
myMap.setMapFriction(1.3);
```

When a user releases the map the continued velocity is determined by the previous velocity divided by the friction.

- 2 means velocity is decreased by half every frame draw
- 1 is not allowed, since it would never stop moving
- -1 turns it off again