Google maps panning example

Posted At : 11 January 2011 23:46 | Posted By : Shaun McCran
Related Categories: Javascript, Google

I've been tinkering with the Google Maps Api recently, in an effort to replicate a flash project I've seen. It was a Google maps interface that used Adobe livecycle (**http://www.adobe.com/products/livecycle/**) feeds to poll IP addresss and display them within the map interface.

I liked the look of it, but wanted to create it in JQuery as it required a lot of custom server side code, and several expensive server software installations. Also I'm migrating away from the flash / flex arena into a more purist JQuery / AJAX development mindset.

There is a full example of **Google Maps panning here**

First off we need to setup our page, Google maps requires an API key, so I'm setting that using ColdFusion, and passing it on the URL to the Google JavaScript API call. This validates against your URL, and is free. If you don't have one, you can get one here: **http://code.google.com/apis/maps/signup.html**.

After this I'm loading JQuery and the maps JS.

```
<cfset request.apiKey = "your key here">

<s/cript type="text/javascript"src="http://www.google.com/jsapi?key=<cfoutput>#request.apiKey#</cfoutput>"></script>

<s/cript type="text/javascript">
  google.load("jquery", '1.3');
   google.load("maps", "2.x");
          </script>
```

The next block of code is the html content of the page. It is really small as most of our content will be dynamically inserted from the Google Maps functions.

The important bits are the 'map' div, which will hold our map graphics, and the ul 'list' container. This will be populated with a series of random Geo points from Google maps.

```
<div id="wrapper">

<div id="map"></div>
   <ul id="list">
<div id="header">Random points</div><br id="clear-all">
          </ul>

<div id="message" style="display:none;">
          Label
         </div>

        </div>
```

Now we can create our map and assign it to our container ('map'). Next I have created a variable called 'cheltenham' , which has the lat long of (yes you guessed it!) the town of Cheltenham. This is the initial starting point for the map. Lastly we use the setCenter method to position the map, and set the starting zoom level

```
<s/cript type="text/javascript">
```

```
$(document).ready(function(){

// set the element 'map' as container
var map = new GMap2($("#map").get(0));

// set the lat long for the center point
var cheltenham = new GLatLng(51.89487062921521,-2.084484100341797);

// value 1 is the center, value 2 is the zoom level
map.setCenter(cheltenham, 9);
```

Now we will create ten random points on the map. The idea is to display the panning function, so I don't really care where they are. We get the boundaries of the map, then we convert them to lat long coordinates, then we loop over them ten times invoking the map.addOverlay to create new markers each time.

```
// setup 10 random points
var bounds = map.getBounds();
var southWest = bounds.getSouthWest();
var northEast = bounds.getNorthEast();
var lngSpan = northEast.lng() - southWest.lng();
var latSpan = northEast.lat() - southWest.lat();
var markers = [];

for (var i = 0; i < 10; i++) {
var point = new GLatLng(southWest.lat() + latSpan * Math.random(),
southWest.lng() + lngSpan * Math.random());
marker = new GMarker(point);
map.addOverlay(marker);
markers[i] = marker;
}
```

Next we will create a function to handle each of the markers. This function populates the ul list created above with a list of the markers. It also attached a click event to each marker that uses the function 'displayPoint', passing in the marker reference and index.

```
// markers click event
$(markers).each(function(i,marker){
$("<li />")
.html("Point "+i)
.click(function(){
displayPoint(marker, i);
})
.appendTo("#list");

GEvent.addListener(marker, "click", function(){
displayPoint(marker, i);
});
});
```

Now we create the displayPoint function referenced above. This has an event listener to watch for the end of a move event. It also calculates to distance from the current position to the clicked marker. It then uses the panTo method to pan the map display to the selected marker. (marker.getLatLng).

```
function displayPoint(marker, index){
        $("#message").hide();

  var moveEnd = GEvent.addListener(map, "moveend", function(){
var markerOffset = map.fromLatLngToDivPixel(marker.getLatLng());
                $("#message")
                   .fadeIn()
       .css({ top:markerOffset.y, left:markerOffset.x });

                GEvent.removeListener(moveEnd);
                    });
                map.panTo(marker.getLatLng());
                    }
```

It looks like a wedge of code, but if you view the source of the demo below it is pretty small.

In the next post I will add the AJAX polling to dynamically get the markers from a datasource.

There is a full example of **Google Maps panning here**