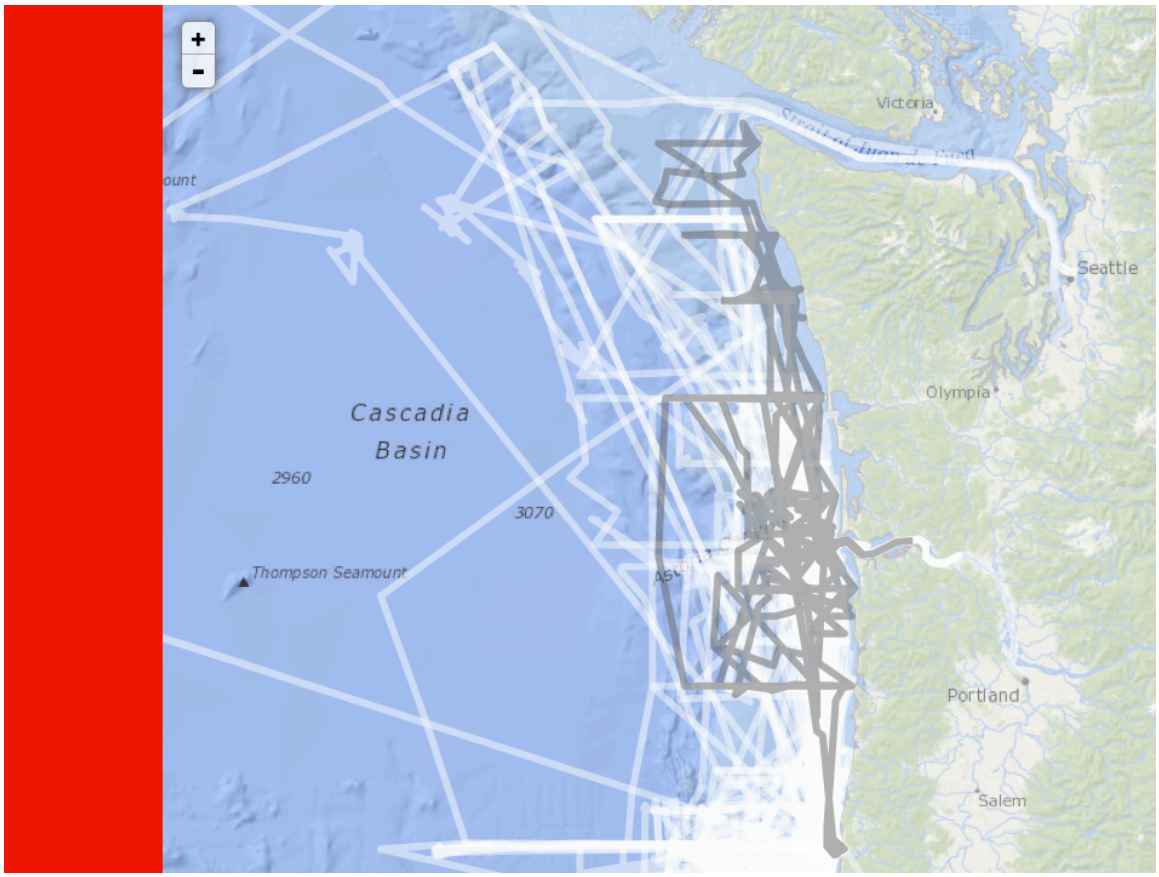
Lab 3: Interactive ShipTracks (50 points)

Due: Thursday, April 25, 2013 at noon on Blackboard

What to submit

Submit to Blackboard a zipped archive containing a working index.html, index.css and map.js. If you got stuck in the middle of a problem, submit an additional index.html and map.js with a short note where you got stuck. For extra credit, leave a note for the TA describing what extra functionality you added.

Introduction



Today, you will work with Leaflet's geoJSON layer and add interactivity to a bunch of ship tracks. When you roll with the mouse over a track, it will be highlighted. When you click on the map, attribute data of the selected ship track will be displayed outside of the map.

Note: The style and layout on the screen shot above are not necessarily good choices, just a quick way to demonstrate today's goal. Map controls as well as menu and text field outside of the map are missing. Feel free to give your page a better look.

Preparations

In a new web project in Aptana, create the following files:

- index.html – copy from http://jsbin.com/enuwap/1/edit to set up links to JQuery, JQueryUI and Leaflet.
- index.css - will contain your page styling
- map.js – will contain your JavaScript
- wecomatracks.js – contains ship track data in geoJSON format

1. Map and Background Layer (5 points)

Edit index.html and index.css, to reserve room for a Leaflet map. In init(), create a map and make map globally accessible. The following functions will be called by init. Create a function loadBaseLayer(), that adds an ocean floor background layer to the map. Add a global layer control variable and within loadBaseLayer(), add the base layer to the layer control (L.control.layers).
Also add a global variable data and assign null.

2. Loading cruise track data from a geoJSON file (5 points)

In a function loadData():
Instead of creating a JavaScript file that contains the data as objects, use a JQuery method to load geoJSON from a file (local or server).

```
$.getJSON("myFile.json", function(data){
//your code goes here
});
```

When the function has finished loading, it will call the anonymous function and data will contain the geoJSON data. When the function has finished loading, it should save the data in a global variable.

3. Creating a geoJSON Layer (5 points)

In a function createGeoJSONLayer():
Check if the global data variable is not null, to make sure that the data has finished loading. Create a geoJSON layer using the global data object. Add a default style to the geoJSON Layer. Add the layer to map and layer control.

4. Highlighting ship tracks on mouseover/mouseout (10 points)

To the GeoJSONLayer() add a property onEachFeature. To this property assign the function onEachFeature. The function onEachFeature is a call back function that allows you to assign events to all features (tracks) in the layer. Add the function with the following skeleton:

```
function onEachFeature(feature, layer) {
        layer.on({
                mouseover : highlightFeature,
                mouseout : resetHighlight
```

```
        });
}
```

Define the functions:
Function highlightFeature(event){
//this function changes the style of the clicked feature
//event is the event Object that is passed to highlightFeature when a mouseover event occurs. It can tell you which path (=track) was rolled over. Change the style of the track using a method from L.Path.
}

Function resetHighlight(event){
Proceed as in highlightFeature, pass the color that was defined as default.
(Do NOT use layer.resetStyle, work with the event target).
}

4. Clicking on shipTracks (10 points)

The following are different ways of assigning a function to a property. You should choose the one that fits your need to work with local parameters:

```
function myEventHandler(event){
//your code goes here
// with the syntax below, you cannot pass additional parameters because //adding (param1) would call the function instead of assigning it.
}
…
click: myEventHandler;
```

or:

```
click: function (event){
//your code goes here
//this anonymous function has access to all local variables of the function that
//surrounds it (in this case of function onEachFeature(feature, layer). It is assigned and
//is executed on “click”
}
```

or:

```
function myEventHandler(event, param1){
//your code goes here
//this is an alternative to the anonymous function
//you can pass variables local to onEachFeature to myEventHandler as parameters
```

// myEventHandler(event, param1) is a function call. It is nested within an anonymous
//function definition that is executed on “click”, thus it will not be called before the click
//happens. Your code is better legible this way.
}

```
click: function(event){
        var param1 = feature;
        myEventHandler(event, param1);
}
```

You will now need to access properties of feature (feature.properties…). Use the anonymous function (middle syntax) or the last syntax.
Edit your css so that you have a panel outside of the map.
Feature.properties… gives access to the original geoJSON objects. When the user clicks on a feature, the names of the cruise participants are displayed in the panel. Use JQuery commands to add remove text to the panel.

5. Filtering Ship Tracks based on Data Attributes (10 points)

Leaflet does not yet offer a function to change the filter of a geoJSON layer after it has been created. Write a function addFilteredGeoJSONLayer() that creates a new geoJSON Layer based on the global data variable. Pass a filter function to it. The function should test if the attribute has a certain value, if so return true, otherwise return false. (true will be drawn, false will not be drawn). Add the layer to map and map control.
Pick a cruiseID out of the data file and use this as filter.

6. Allowing the User to Choose an Attribute (5 points)

Add a drop box to your panel based on the example http://jsbin.com/etihis/1/edit
Pick two or three cruise ID’s and hardcode them into your html list (menu):
CruiseID > VC…
WX…
RT…
ArrivePort > Honolulu
Newport

Adapt your filter function so that the filter is based on the selection of the user.

6. Adding additional Functionality (extra credit, points variable)

Add additional functionality of your choice, e.g.

- When a track is clicked, the browser zooms to the extend of the track (a Leaflet functions exists for that)
- When a track is clicked the layer with all tracks is removed, a filtered layer is added that contains only the clicked track (you can use cruiseID to remember

which track was clicked). Let the clicked track keep the highlight color, even when the mouse is not on it.

- Add a button that adds the all-tracks layer again and removes the single layer
- Location history: create a button that adds the current zoom and map center to an array. This way, you can save several locations. Add another button that goes backwards through the array and resets the map to the former location and zoom level
- add your own idea

Leave a note for the TA what extra credit functionality you added