

AJAX AND WEB SERVICES

MAP / LOCATION BASED DATA

AIM

The aim of this lab is to introduce you to using AJAX with location based data and displaying this data on map

There are many web based mapping libraries that you can use. The most well known is Google Maps, however because its has multiple really well written tutorials I'm going to show you a more open source mapping library

GEOGRAPHIC DATA

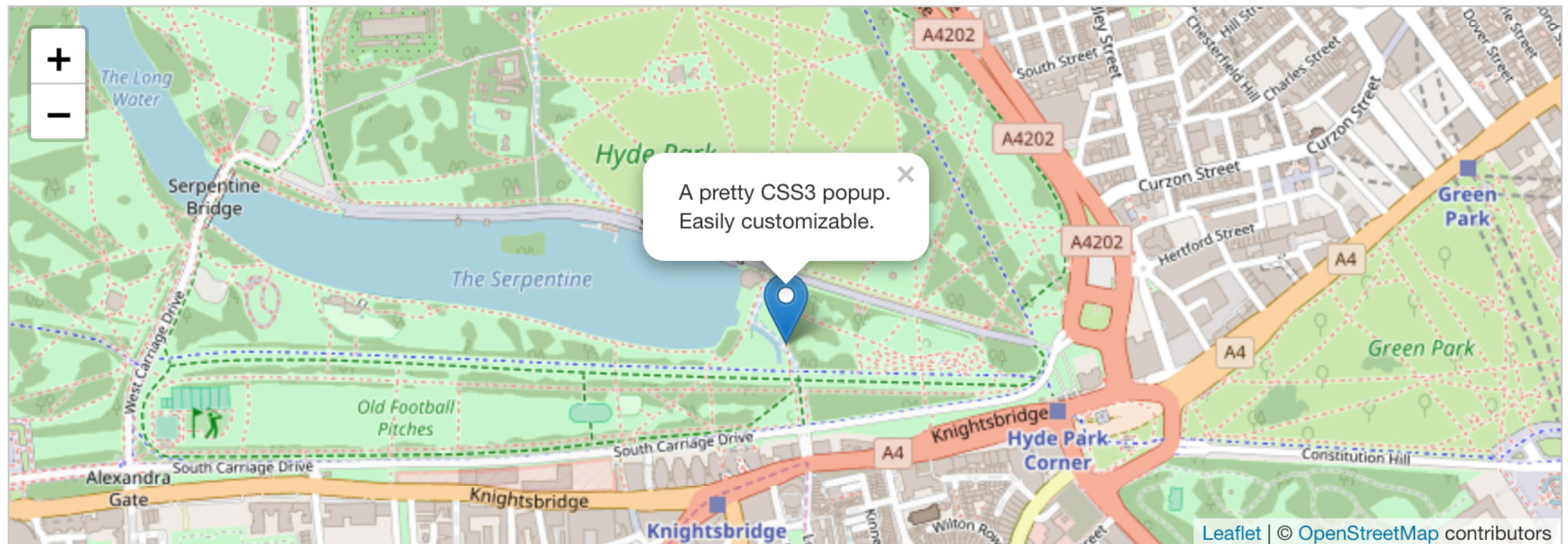
LEAFLET

LEAFLET

Leaflet is the leading Open Source (free) javascript mapping library.

Its pretty small and very customisable.

<http://leafletjs.com/>



SHAKEY - DROPEY

USING LEAFLET

EARTHQUAKES AND METEORS

- ▶ In this tutorial you are going go to build a map based web application that sources its data from external services
- ▶ We will be using the USGS earthquake list
- ▶ and the NASA meteor list
- ▶ We will access this data via AJAX and then display the content on the map.

STEP 1: SETTING UP OUR PROJECT

- ▶ Lets start with the HTML - create a new directory in your project and create a file called index.html
- ▶ The HTML itself isn't that complicated.
- ▶ We have a head section that imports our stylesheet
- ▶ and a body section that has a main content containing 2 sub sections one for the map and one for the controls
- ▶ The main points to note are jquery imported at the bottom with

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
```

- ▶ our two buttons **shakey** and **dropey**
- ▶ and the div **mapid** where our map will eventually go

```
<!doctype html>
<html>
<head>
  <!-- default stuff -->
  <meta charset="utf-8">
  <title>Leaflet Earthquake</title>
  <!-- my style sheets -->
  <link rel="stylesheet" href="main.css">
</head>
<body>
  <section class="main-container">
    <article>
      <!-- my map -->
      <div id="mapid"></div>
    </article>

    <aside>
      <p>Red circles are earthquakes</p>
      <p>Blue circles are meteor strikes</p>
      <button id="shakey">Load Earthquakes</button>
      <button id="dropey">Load Meteors</button>
    </aside>

  </section>
  <!-- #main-container -->
  <!-- jquery and main scripts-->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
  <script src="main.js"></script>
</body>
</html>
```

STEP 2 INCLUDING LEAFLET

```
<head>
  <!-- default stuff -->
  <meta charset="utf-8">
  <title>Leaflet Earthquake</title>
  <!-- my style sheets -->
  <link rel="stylesheet" href="main.css">
  <!-- Leaflet Stylesheet -->
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.0.3/dist/leaflet.css" />

  <!-- Leaflet Script -->
  <script src="https://unpkg.com/leaflet@1.0.3/dist/leaflet.js"></script>
  <!-- more scripts at the end! -->
</head>
```

- ▶ all we need to do to add leaflet to our page is to include the **JS** and **CSS** leaflet libraries

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.0.3/dist/leaflet.css" />
```

```
<script src="https://unpkg.com/leaflet@1.0.3/dist/leaflet.js"></script>
```


STEP 3: CSS

- ▶ There isn't much in our css.
- ▶ The important bit is the map height!

```
html {  
    color: #222;  
    font-size: 1em;  
    line-height: 1.4;  
}  
#mapid { height: 400px; }
```

STEP 4 : MAIN.JS

- ▶ For our web application to work we need to now add some javascript.
- ▶ You will see in the HTML at the bottom of the page we have imported JQuery and a file called main.js
- ▶ main.js will contain all the functionality for our app.
- ▶ Create this file now in your project

STEP 5: INITIALISING THE MAP

- ▶ open main.js
- ▶ the first thing we need to do is initialise the map and tell it what DIV to use to display in.
- ▶ the following code initialises the map to our 'mapid' div and then sets the initial coordinates and zoom level

```
//set the map and initial coordinates  
var mymap = L.map('mapid').setView([0, 0], 1);
```

- ▶ if you view the page now you wont see much as no tiles have been added

STEP 6: ADDING THE TILES

- ▶ There are lots of different map tiles you can use for Leaflet this is what sets it apart from google maps (you can see them and how to add them here <https://leaflet-extras.github.io/leaflet-providers/preview/>)
- ▶ The code below sets up some basic grey types, the types are all streamed from the providers when the map is drawn, hence the need to specify where the tiles come from. Copy this into your main.js, just below where you initialised the map

```
var Esri_WorldGrayCanvas = L.tileLayer('http://  
server.arcgisonline.com/ArcGIS/rest/services/Canvas/  
World_Light_Gray_Base/MapServer/tile/{z}/{y}/{x}', {attribution:  
'Tiles &copy; Esri &mdash; Esri, DeLorme, NAVTEQ', maxZoom: 16 });
```

- ▶ Finally we need add the tiles to the map

```
Esri_WorldGrayCanvas.addTo(mymap);
```

- ▶ Save and view your pages now, you should be able to see a basic map

OPTIONAL EXTRA STEP: CHANGING MAPS

- ▶ if you don't like the grey map you can add your just choose an example from <https://leaflet-extras.github.io/leaflet-providers/preview/> and add it to the map
- ▶ Add the code below to create a topographical layer to you main.js

```
var OpenTopoMap = L.tileLayer('http://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {  
    maxZoom: 17,  
    attribution: 'Map data: &copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>, <a href="http://viewfinderpanoramas.org">SRTM</a> | Map style: &copy; <a href="https://opentopomap.org">OpenTopoMap</a> (<a href="https://creativecommons.org/licenses/by-sa/3.0/">CC-BY-SA</a>) '  
});
```

- ▶ then to change to this map type just add it as we did before

```
OpenTopoMap.addTo(mymap);
```

STEP 7: GETTING JSON

- ▶ We want to get the earthquakes when the **shakey** button is clicked.
- ▶ So with in this function we add the code to pull the earthquake data
- ▶ Add this code below where you have added your tiles in main.js
- ▶ Run the page and click the shakey button. You wont see much happen yet but if you open the console you will see that the JSON response has downloaded

//when the button on is clicked

```
$('#shakey').click(function() {
```

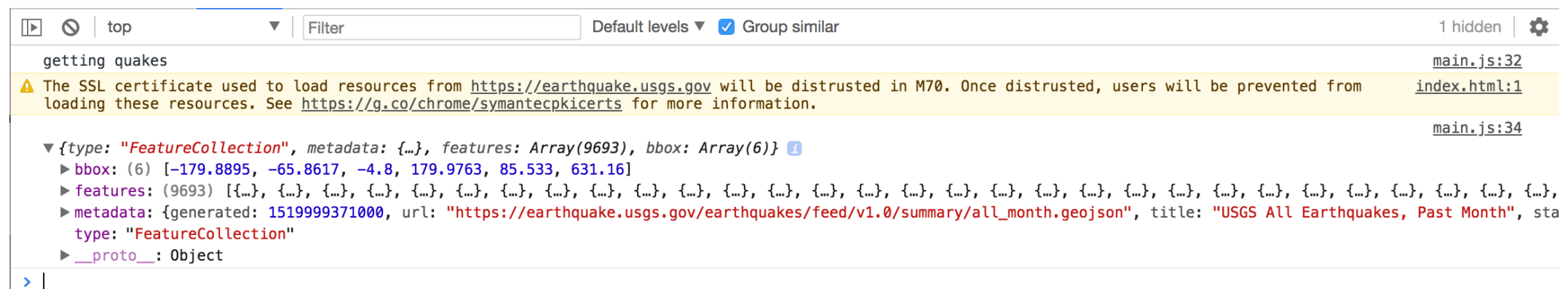
```
    console.log("getting quakes");
```

```
    $.getJSON("https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_month.geojson", function(result) {
```

```
        console.log(result)
```

```
    });
```

```
});
```



STEP8: DRAWING DATA ON THE MAP

- ▶ If you look at the data in the console that appears when the api data is downloaded you will see that there are a number of different parts to the data.
- ▶ If we went and looked at the USGS api we would know that the data we are interested in is in the features array
- ▶ the code opposite uses the features array in a **ForEach** loop to run through all the items in this array. Just like we say in the film example last lab.
- ▶ for each item found we create a **quake** object
- ▶ again from studying the API documentation I know that the location of the quake is stored in the **geometry.coordinates** array.
- ▶ For each quake found we uses the leaflet library to create a circle at that location with a size and a style.
- ▶ Write this code into your main.js **directly below the console.log(result) line**
- ▶ **Run your page and see what happens when you press the button - it may take a few seconds to show.**

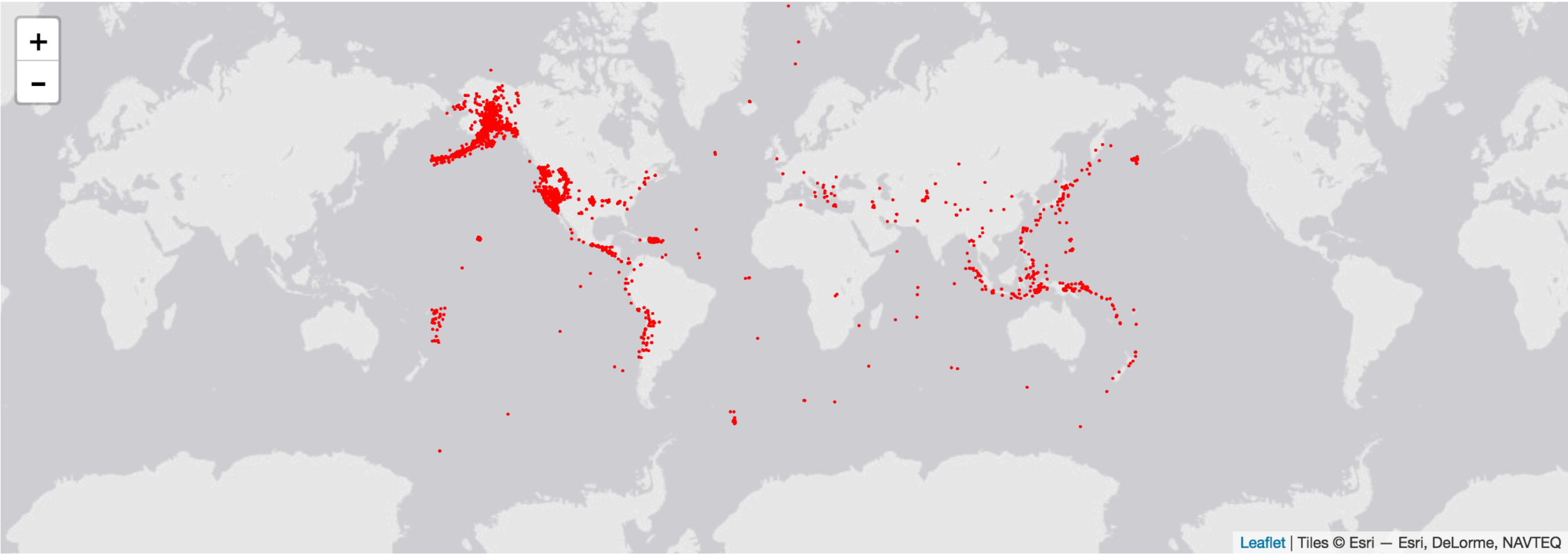
```
get the coordinates
result.features.forEach(function(quake) {
  //for each earthquake
  //get its coordinates
  var lng = quake.geometry.coordinates[0];
  var lat = quake.geometry.coordinates[1];

  var circle = L.circle([lat, lng], 1, {
    color: 'red',
    opacity: 0,
    fillColor: 'red',
    fillOpacity: 0.8
  })
  //and add it to the map
  circle.addTo(mymap);
});

create a circle at that point
style the circle
add it to the map
```

TEXT

THE LITTLE RED DOTS ARE EARTHQUAKES!



Red circles are earthquakes

Blue circles are meteor strikes


Load Earthquakes

Load Meteors

STEP9: BIGGER RED DOTS

- ▶ Ok so we have lots of little red dots.
- ▶ The USGS api actually gives us some more info we can use to make this look better.
- ▶ We can ask for the earthquake magnitude
quake.properties.mag
- ▶ we can then use this to change the size of the circles we add

```
result.features.forEach(function(quake) {  
    //for each earthquake  
    //get its coordinates  
    var lng = quake.geometry.coordinates[0];  
    var lat = quake.geometry.coordinates[1];  
    //and its magnitude  
    var mag = parseFloat(quake.properties.mag);  
    //for each earthquake create a circle  
  
    var circle = L.circle([lat, lng], mag * 10000, {  
        color: 'red',  
        opacity: 0,  
        fillColor: 'red',  
        fillOpacity: 0.8  
    })  
    //and add it to the map  
    circle.addTo(mymap);  
});
```



resize the circle
based on the
magnitude

TEXT

MUCH BETTER!



Red circles are earthquakes

Blue circles are meteor strikes

Load Earthquakes

Load Meteors

YOUR TURN

ADDING METEORS

METEORS

- ▶ Nasa provide a great api that details all the meteors that hit the planet.
- ▶ I have created a subset of this for you here <https://data.nasa.gov/resource/gh4g-9sfh.json>
- ▶ the data contains a list of meteor objects like the one opposite
- ▶ including an array **geolocation** that contains the lat and long where the meteor fell.

```
{
  "fall" : "Fell",
  "year" : "1951-01-01T00:00:00",
  "nametype" : "Valid",
  "mass" : "720",
  "name" : "Aarhus",
  "recclass" : "H6",
  "reclat" : "56.183330",
  "reclong" : "10.233330",
  "id" : "2",
  "geolocation" : {
    "latitude" : "56.18333",
    "needs_recoding" : false,
    "longitude" : "10.23333"
  }
}
```

YOUR TASK – ADDING METEORS

- ▶ Using the example of the earthquakes can you add the following functionality to your web app.
- ▶ When the user clicks the meteors button all the meteors should be displayed in Blue at the locations they fell
- ▶ The size of the meteor should reflect the mass of that meteor
- ▶ finally add popups to the data. using **`circle.addTo(mymap).bindPopup('+meteor.name+');`**

ALTERNATIVES

GOOGLE MAPS

GOOGLE MAPS

- ▶ Google have produces a number of great tutorials about how to use their mapping system.
- ▶ In many ways it is very similar to Leaflet, but is more restricted in the maps that it can use.
- ▶ full tutorials about google maps are available here
- ▶ <https://developers.google.com/maps/documentation/javascript/>

FINAL THOUGHTS

WHAT HAVE WE DONE?

WHAT HAVE WE DONE?

- ▶ In this lab we have seen how data with geographical information can be easily retrieved and drawn on a map
- ▶ What you should realise by now is that every single API is different and that you need to study the API documentation to understand how to use the data that it provides
- ▶ There is very little consistency across APIs so you really do need to read the docs.

WHAT NOW

FINISHED?

FURTHER WORK

Seriously, explore the APIs you want to use

You need to understand the APIs that you want to use for your project.

You need to know what data they provide and if you can access it.

If you haven't already you should at the very least try to get some json through a request as shown this week.

Just looking at the api website doesn't not count!