

Visualising Geographical Data with Leaflet

Dr. Peter Mooney



Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under CC BY SA.

Aims of this lecture today

- To introduce you to some basic concepts of Visualising Geographical data on a Web-based Map
- After the lecture you should be able to create simple visualisations of Geographical data on web-based maps using Open Source Components
- You will see that much of the work in Visualisation of Geographical comes from preparing the SPATIAL data for display on the web-based maps

TIMETABLE FOR TODAY

- Session 1 09:30 – 11:30 (no break)
- Session 2 14:00 – 16:00 (no break)
- The class today is a little shorter than normal as I am teaching in other lectures outside of the times mentioned above.

Which browser should you use for the lecture today?



Safari



GitHub Page for Today's Materials

<https://github.com/petermooney/SummerSchool2016>



Keep this page open in a browser today – as it will have lots of links to web applications which we will be using in the lecture

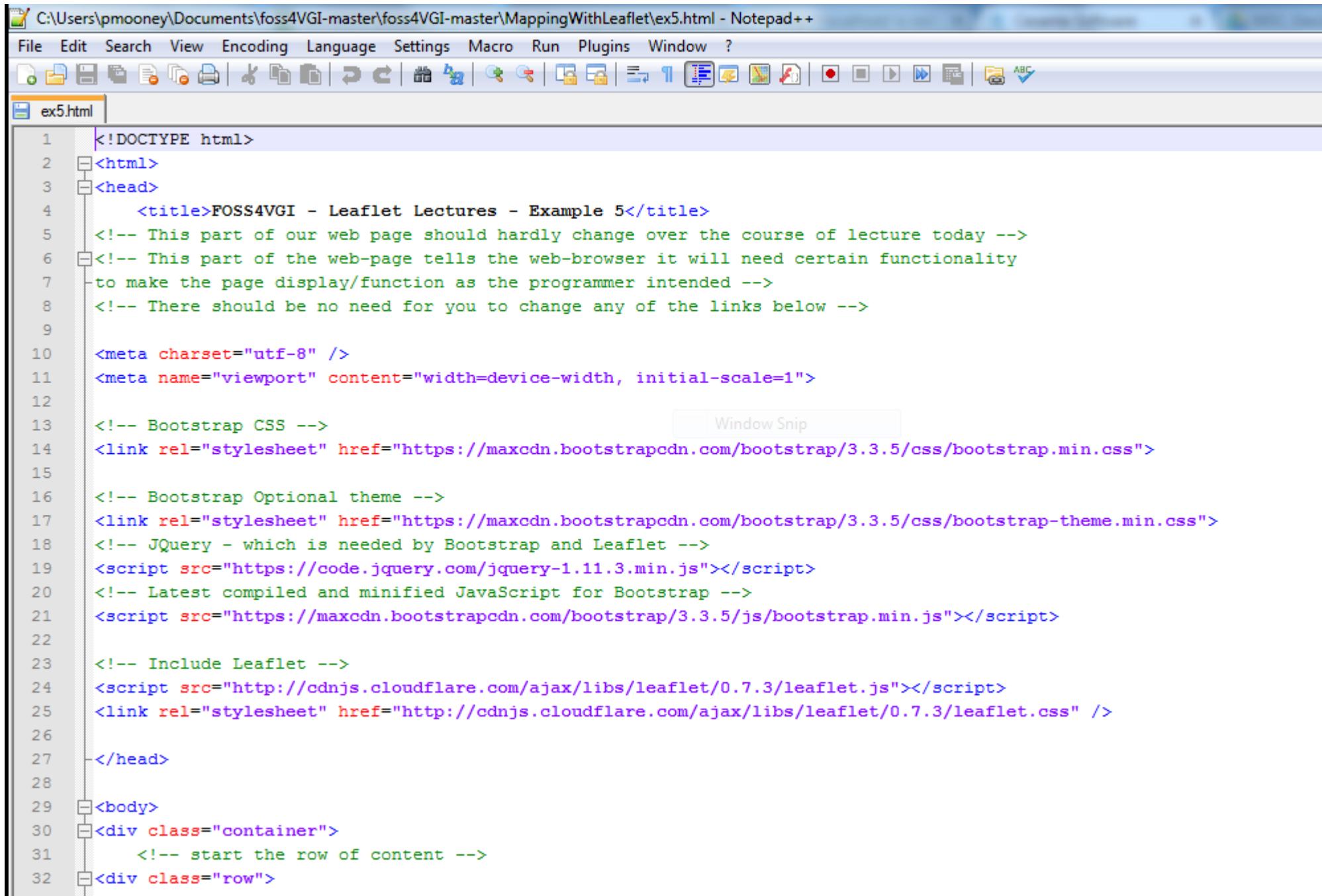
We are also going to need to have a nice text editor installed

- A text editor is one of the most important tools which a web developer uses
- Like web servers there are LOTS of options.
- We need a text editor which will give us some user friendly editing functionality for HTML and Javascript

Suggested Text Editors

- **ANY ONE OF**
- The Geany
- Bluefish
- Notepad++
- Sublime
- Perhaps Bracket or TextMate for MAC specifically
- The links are available on the GitHub page

Notepad++ is installed



The screenshot shows the Notepad++ interface with the file `ex5.html` open. The code is an HTML document with various parts commented out with `<!-- ... -->`. The code includes meta tags for charset and viewport, Bootstrap CSS imports, jQuery, and Leaflet scripts. A tooltip labeled "Window Snip" is visible over the code area.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>FOSS4VGI - Leaflet Lectures - Example 5</title>
5      <!-- This part of our web page should hardly change over the course of lecture today -->
6  <!-- This part of the web-page tells the web-browser it will need certain functionality
7  to make the page display/function as the programmer intended -->
8  <!-- There should be no need for you to change any of the links below -->
9
10 <meta charset="utf-8" />
11 <meta name="viewport" content="width=device-width, initial-scale=1">
12
13 <!-- Bootstrap CSS -->
14 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
15
16 <!-- Bootstrap Optional theme -->
17 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">
18 <!-- JQuery - which is needed by Bootstrap and Leaflet -->
19 <script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
20 <!-- Latest compiled and minified JavaScript for Bootstrap -->
21 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
22
23 <!-- Include Leaflet -->
24 <script src="http://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.js"></script>
25 <link rel="stylesheet" href="http://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.css" />
26
27 </head>
28
29 <body>
30 <div class="container">
31     <!-- start the row of content -->
32 <div class="row">
```

A web map is a
kind of digital
map.

In 2005 Google Maps was launched. It was a game changer

Google
Maps™ BETA

Maps Local Search Directions

Search Help

Maps

Map Satellite Hybrid

Print Email Link to this page

Update: See [satellite imagery of New Orleans](#) from Wednesday, August 31st at 10 am.

Drag the map with your mouse, or double-click to center.

Example searches:

Go to a location
"kansas city" [try it](#)
"10 market st, san francisco" [try it](#)

Find a business
"hotels near lax" [try it](#)
"pizza" [try it](#)

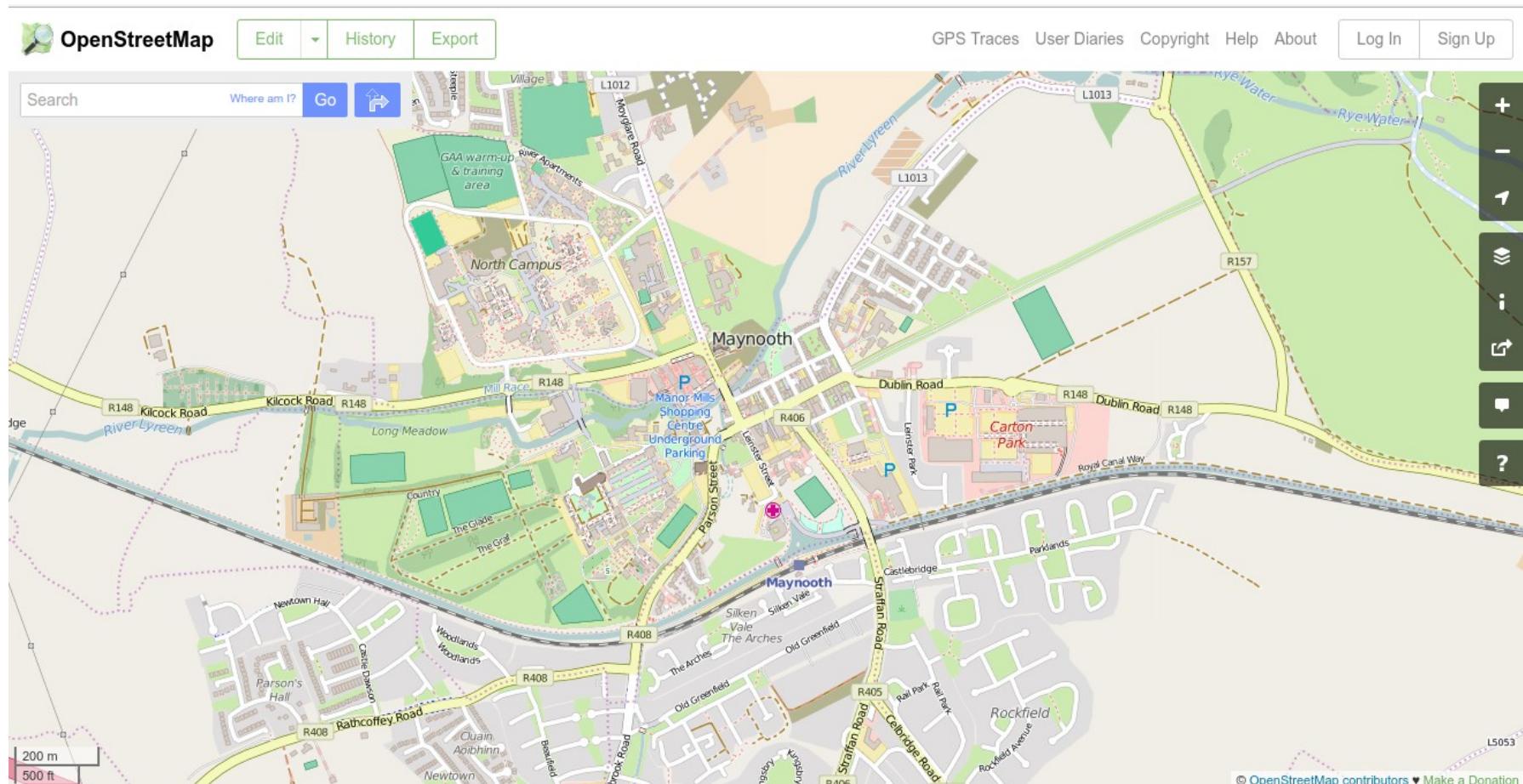
Get directions
"jfk to 350 5th ave, new york" [try it](#)
"seattle to 98109" [try it](#)

[Take a tour »](#)

1000 mi
1000 km

©2005 Google - Map data ©2005 NAVTEQ™ - [Terms of Service](#)

This is a web map



Here is four different web maps on the same page

Map Compare

Maynooth search Help GEOFABRIK tools Switch tool... ▾

Choose map type: OSM Mapnik

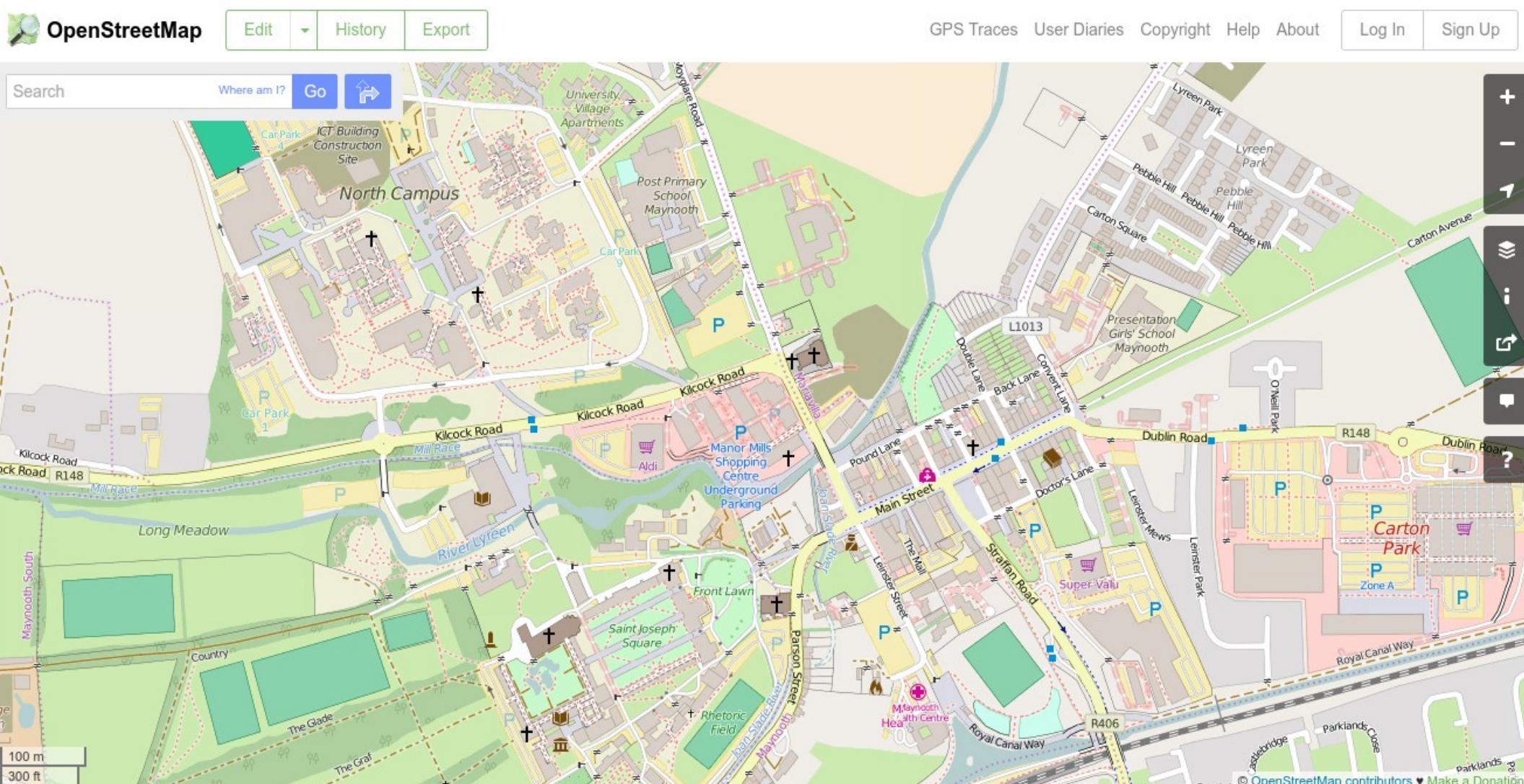
Choose map type: MapQuest Open

Choose map type: Google Map

Choose map type: HERE Map

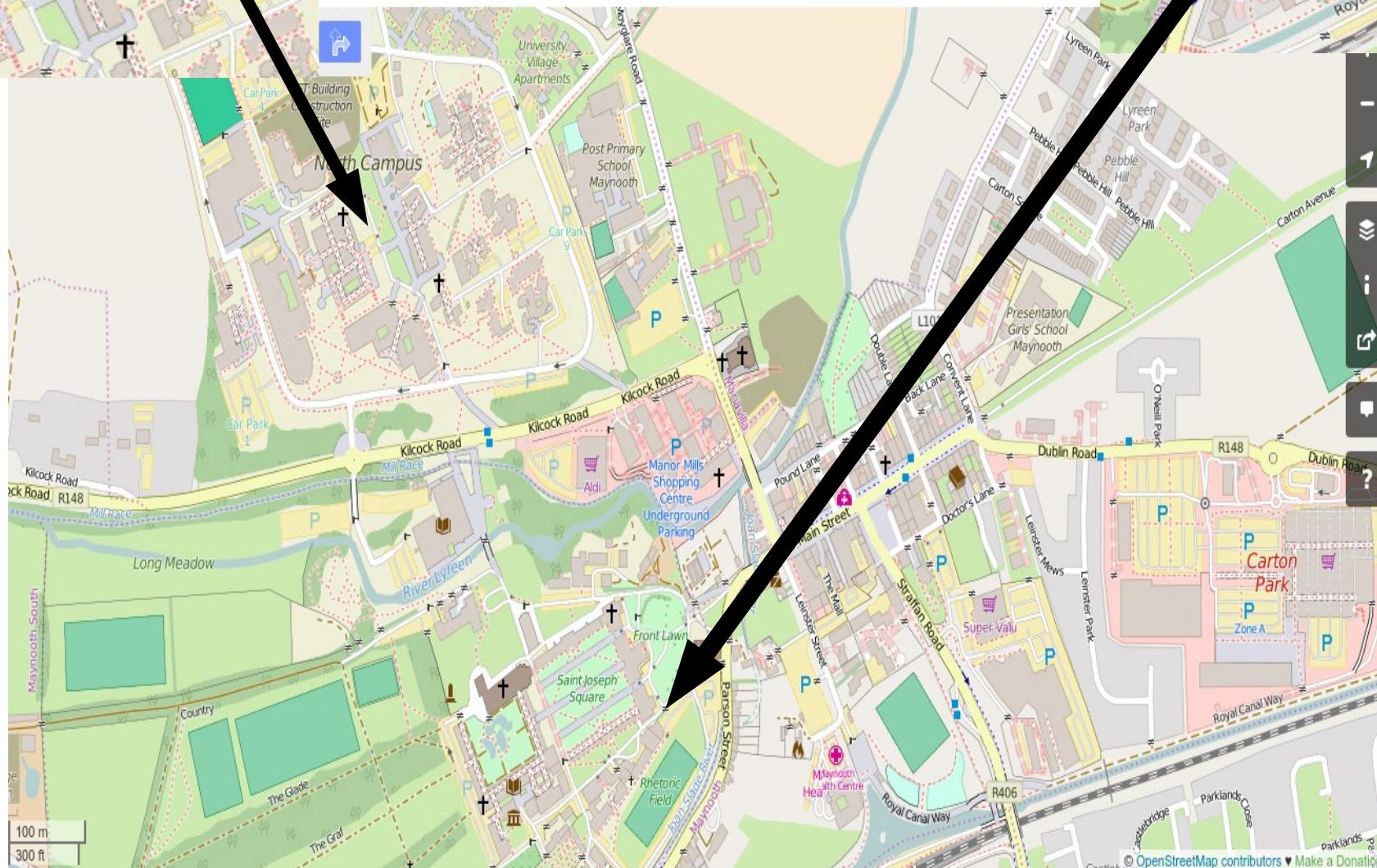
zoom=15 number of maps: 1 2 3 4 6 8

A web map is actually a collection of small map pictures called **TILES**





Web Map Tiles



**Luckily for us – there are lots of
FREE and OPEN web map tile
SERVICES online which we can
use to get web based maps**

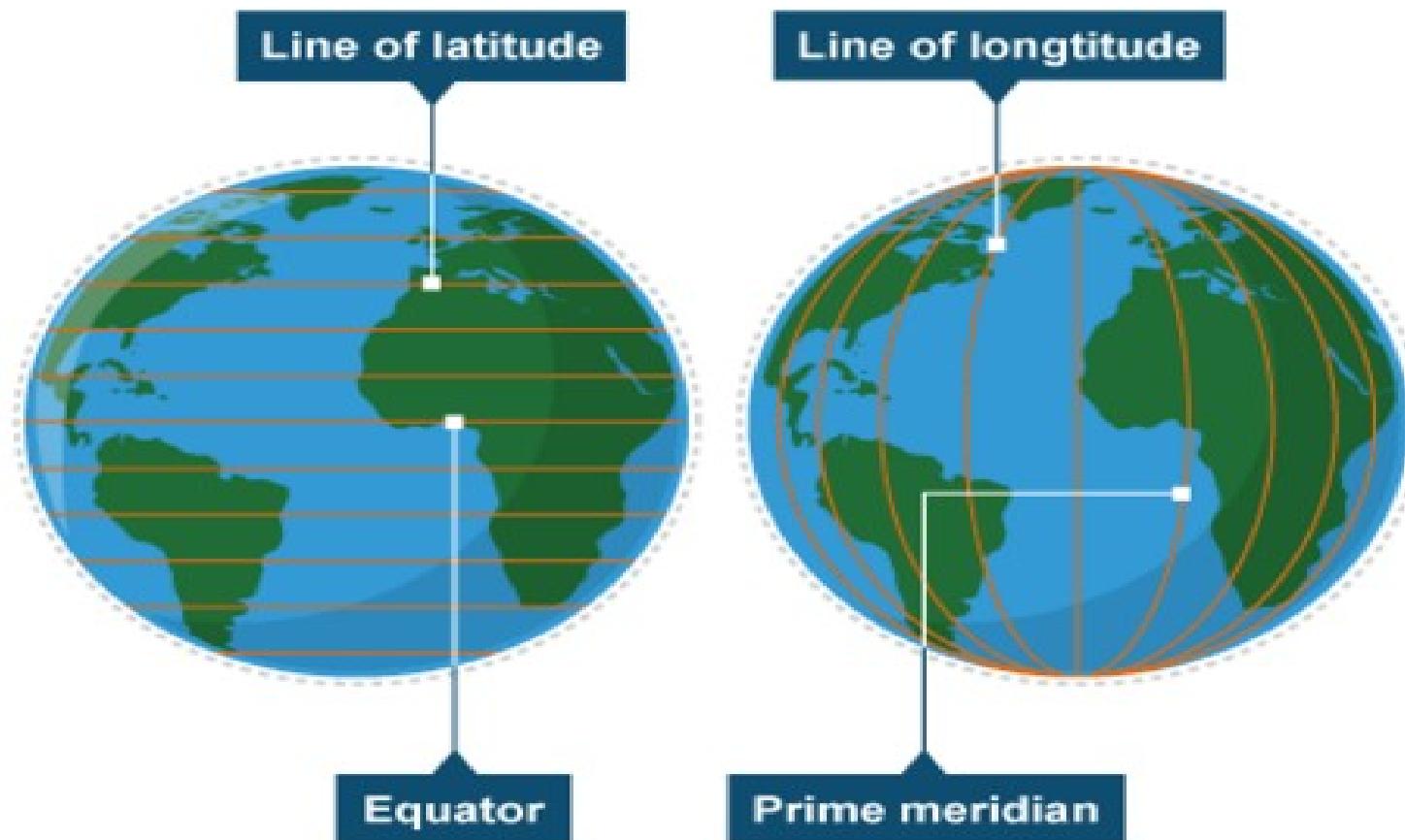
What we need to do is to write some
HTML and Javascript code to
display the web based maps inside
a web page

**Before we start we
need to do a little bit
of geography**

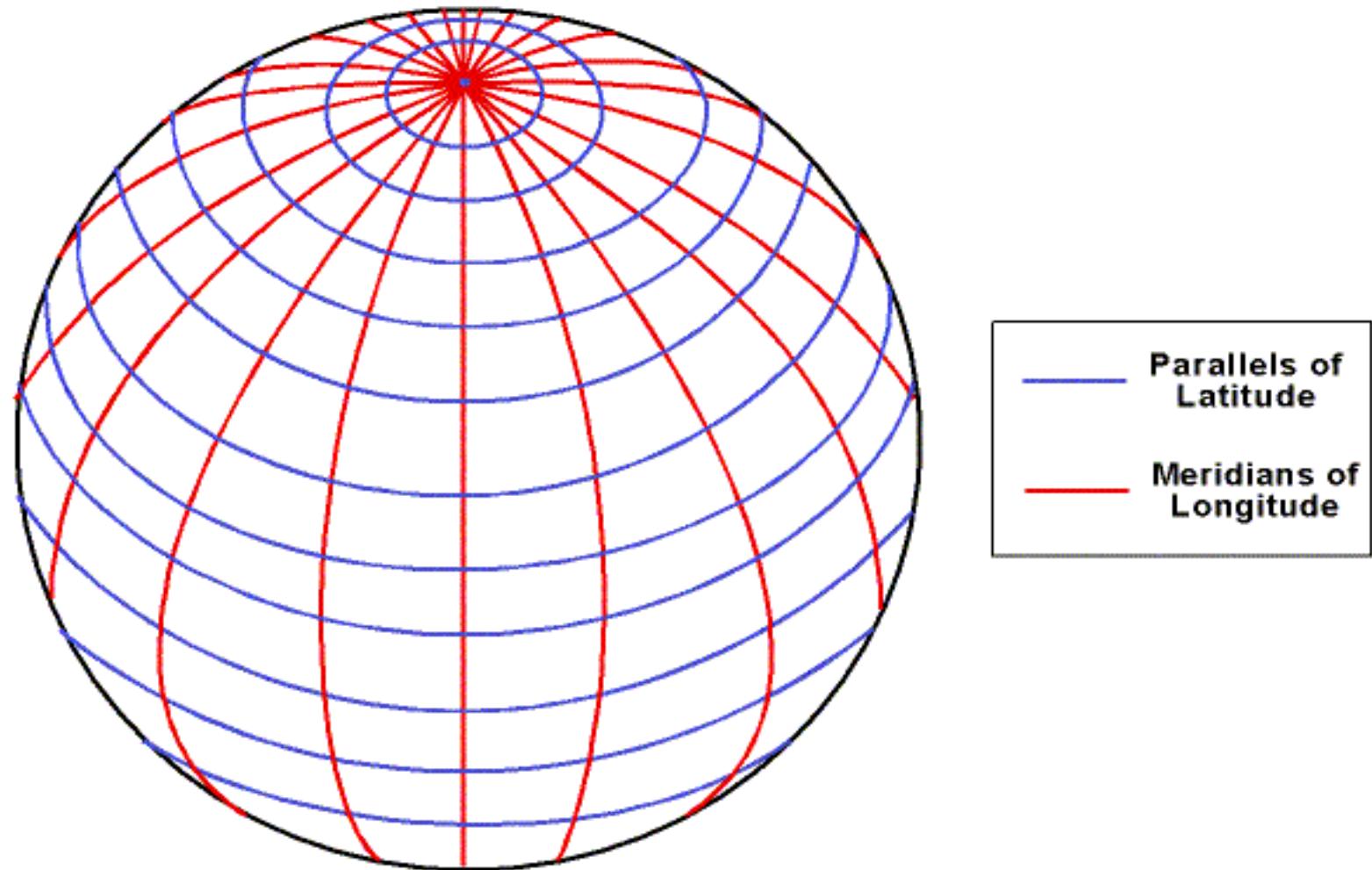
Latitude and Longitude

Latitude and longitude

Lines of *latitude* and *longitude* are used to locate places accurately on the Earth's surface.



Can you imagine our earth like this?



Tony Kirvan 11/8/97



IRELAND

LATITUDE & LONGITUDE MAP



The GPS on your smartphone uses Latitude and Longitude for positioning and navigation



The Sat Nav in your car uses latitude longitude



Two of the most important features of a web based map are (1) The Latitude Longitude of the Center of the Map and (2) The Zoom level that you display the map at

**By knowing the Latitude and
Longitude COORDINATES we
can give the location of
ANYWHERE in the world (on
the earth's surface)**

This is called GEOLOCATION

**How can we find out the
geolocation or the latitude
longitude coordinates of our
school, our house, our favourite
restaurant?**

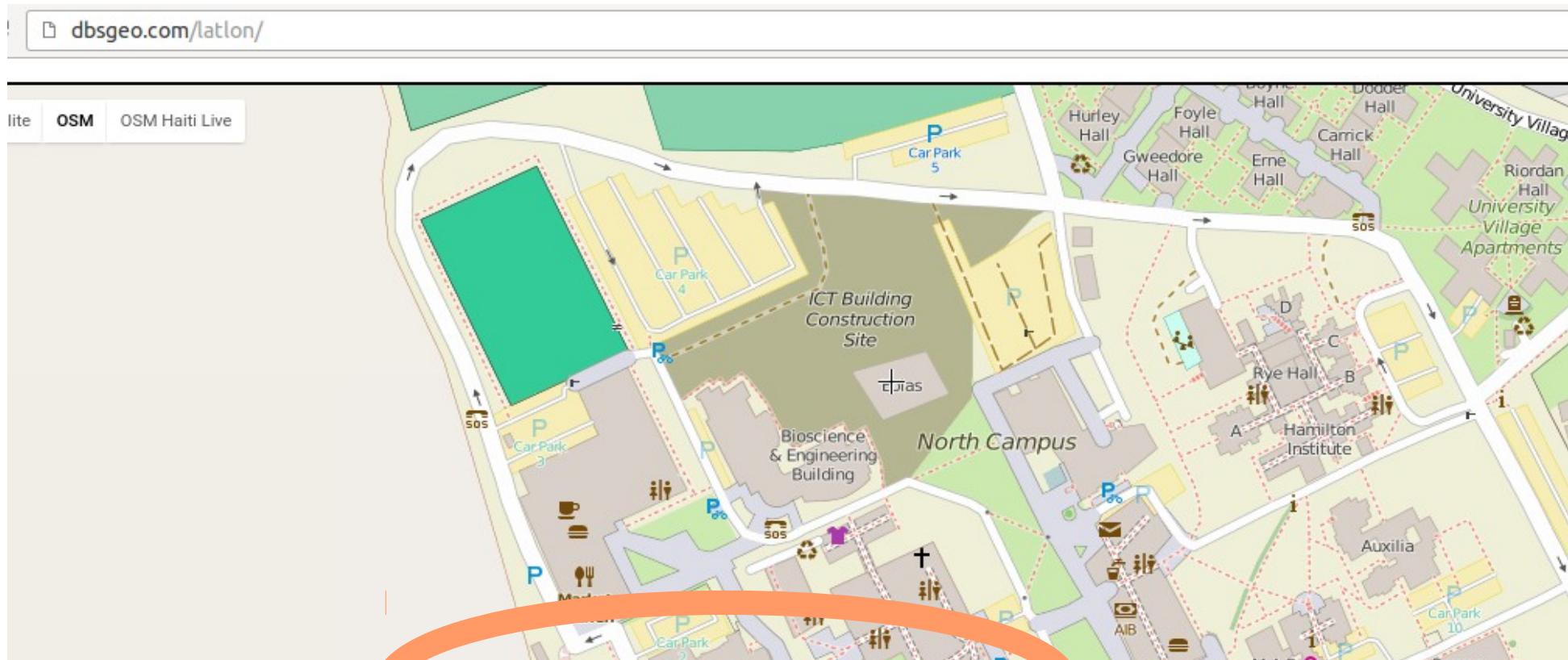


Go to this website

<http://dbsgeo.com/latlon/>

Use the map to zoom in and find the Latitude
Longitude of your school, or your football or
sports club or of your favourite shop

Here is the Latitude, Longitude of the Eolas Building (where we are)



Google Maps zoom level: 17

GeoJSON: { "type": "Point", "coordinates": [-6.60180,53.38494] }

Wikipedia: {{Coord|53.38494|-6.60180|display=title}}

Make note of this website

<http://dbsgeo.com/latlon/>

We will be using it a few times today

Zoom levels in web based maps

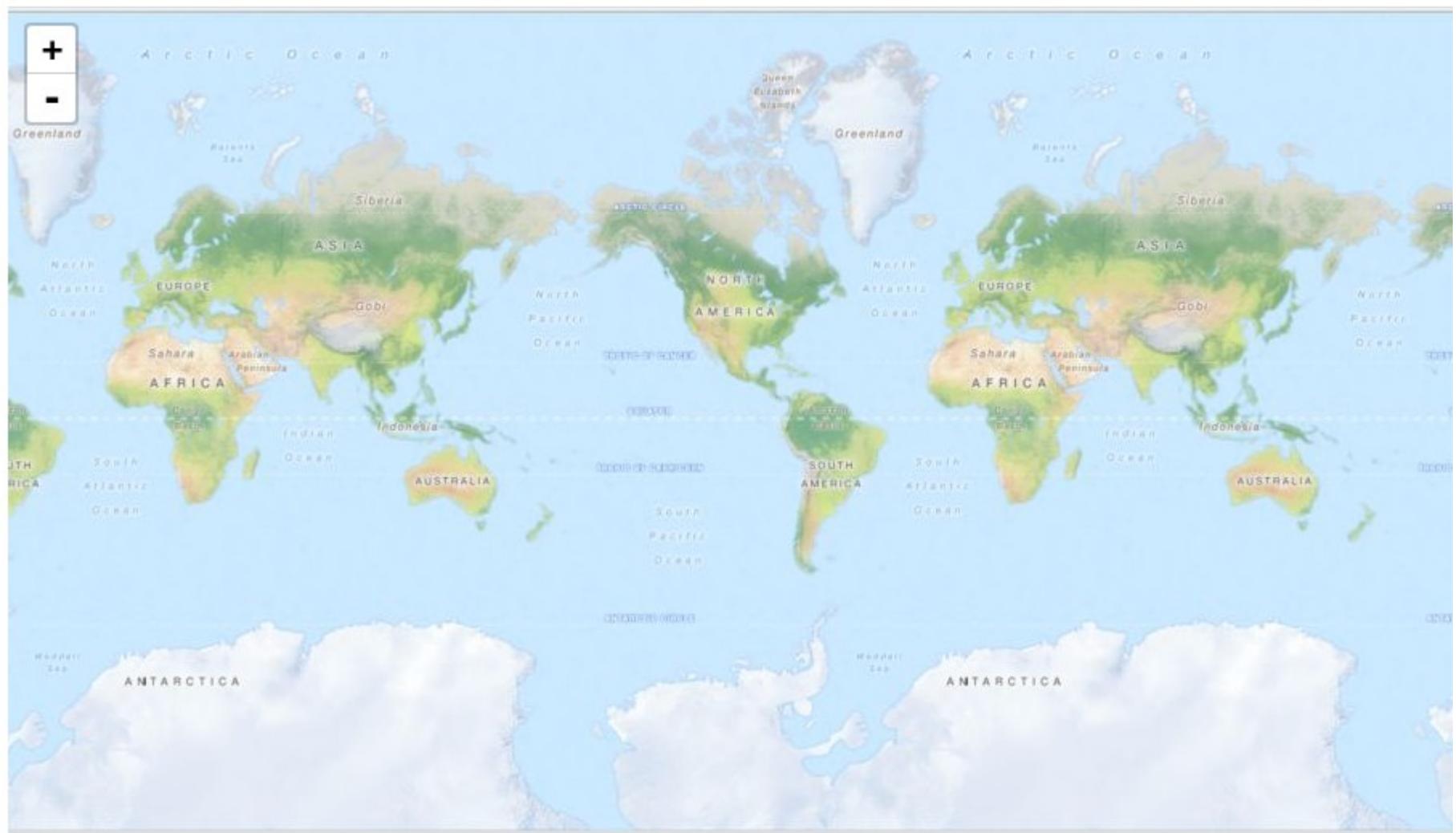


Zoom Levels in Web Maps

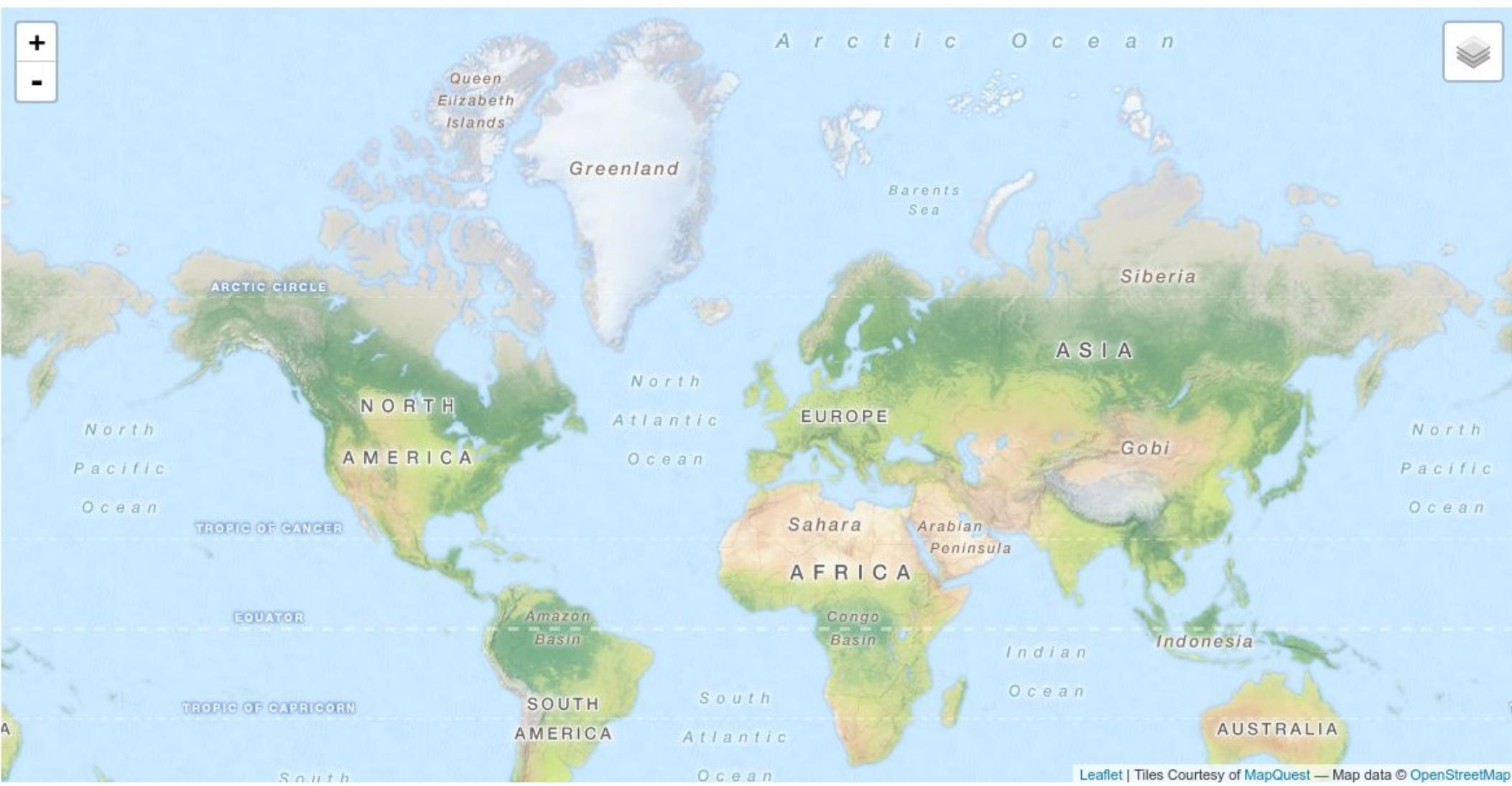
- Zoom Levels in web based maps are very simple to understand.
-
- Usually Zoom Levels go from Level = 1 (very very far away) to Level 18 (very very close to the ground)
- Let's have a look at an example

Zoom Level 1 (very very far away)

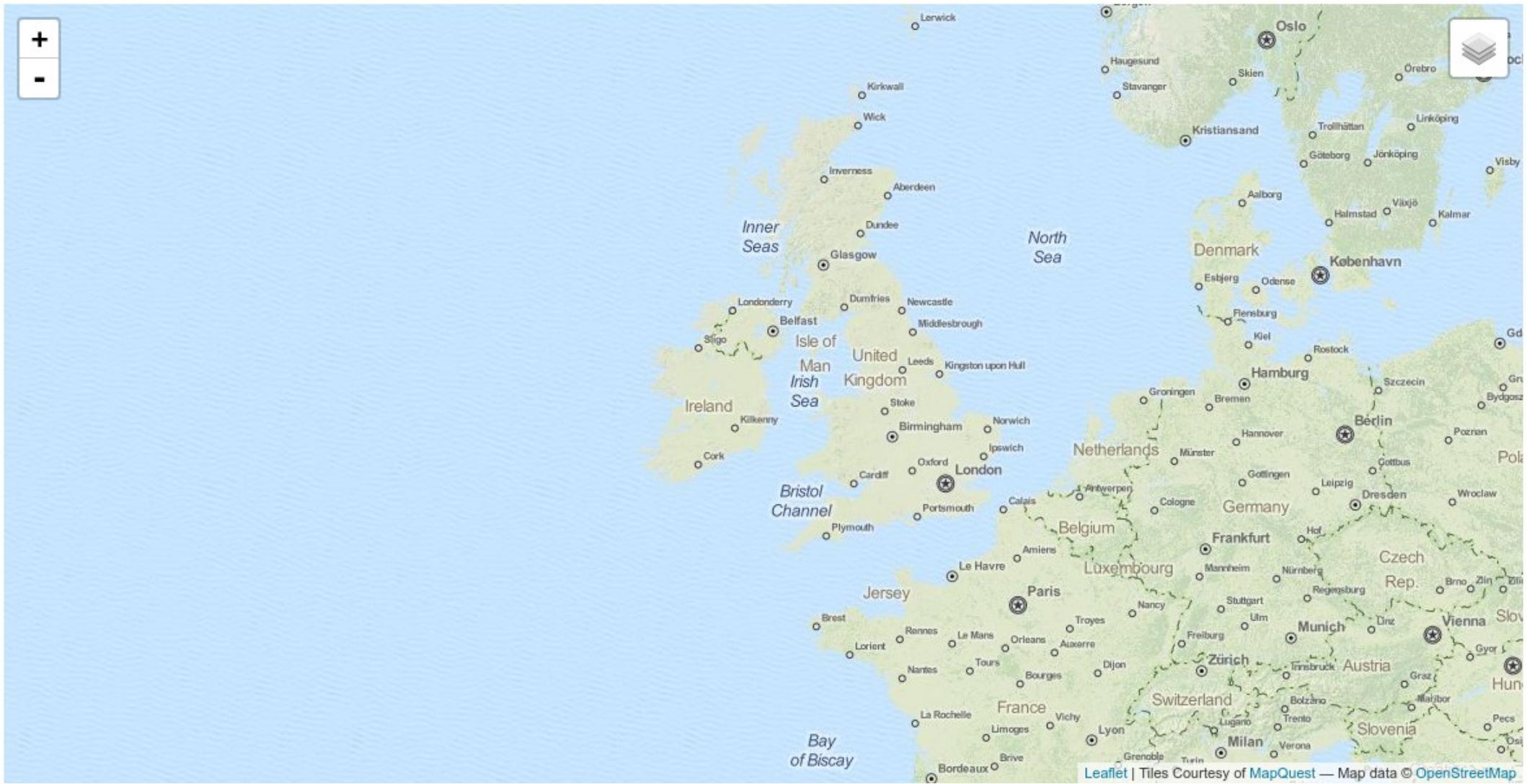
Example A: Getting Started with Web Mapping



Zoom Level 2 (still far far away)



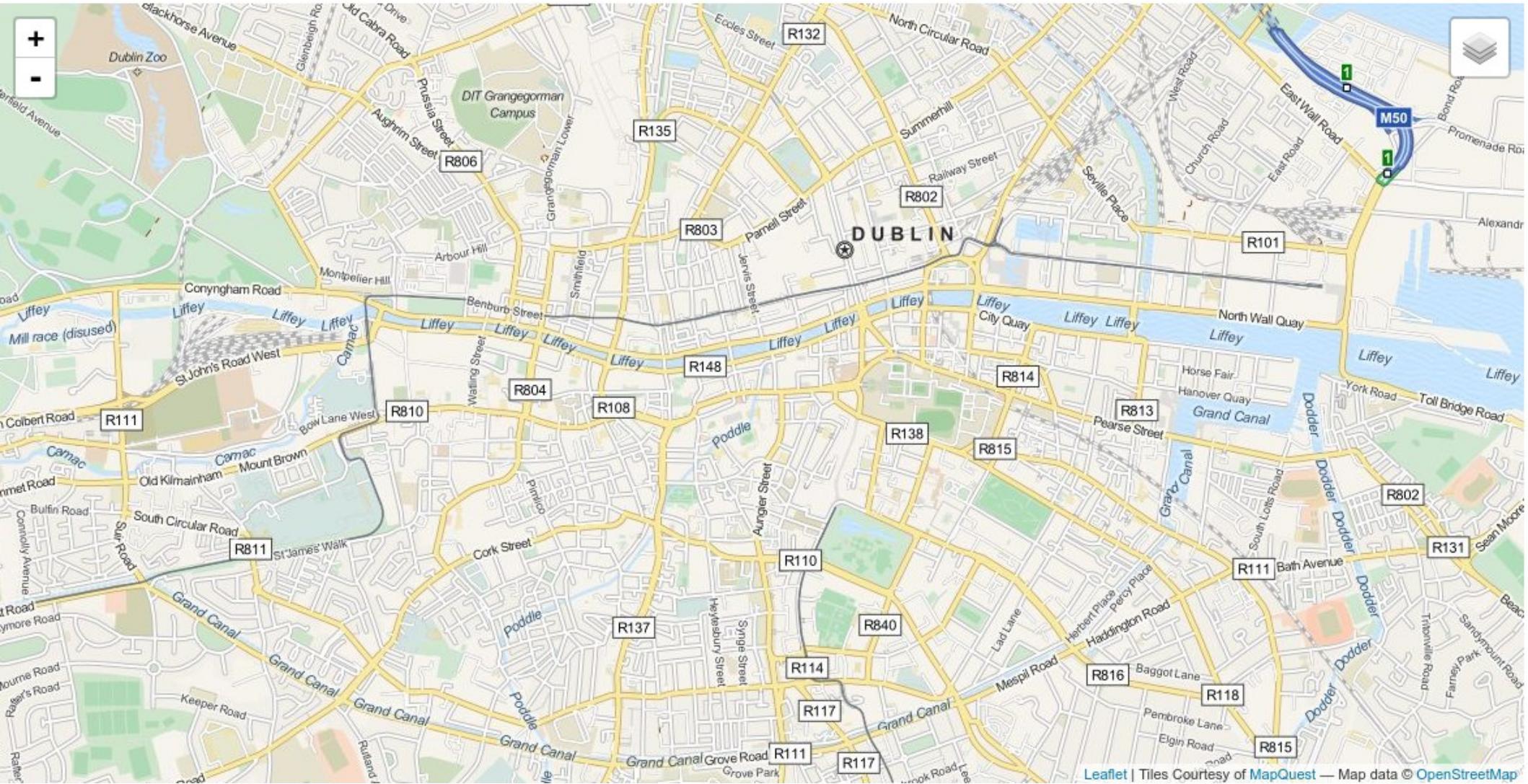
Zoom Level 5 (at continent level)



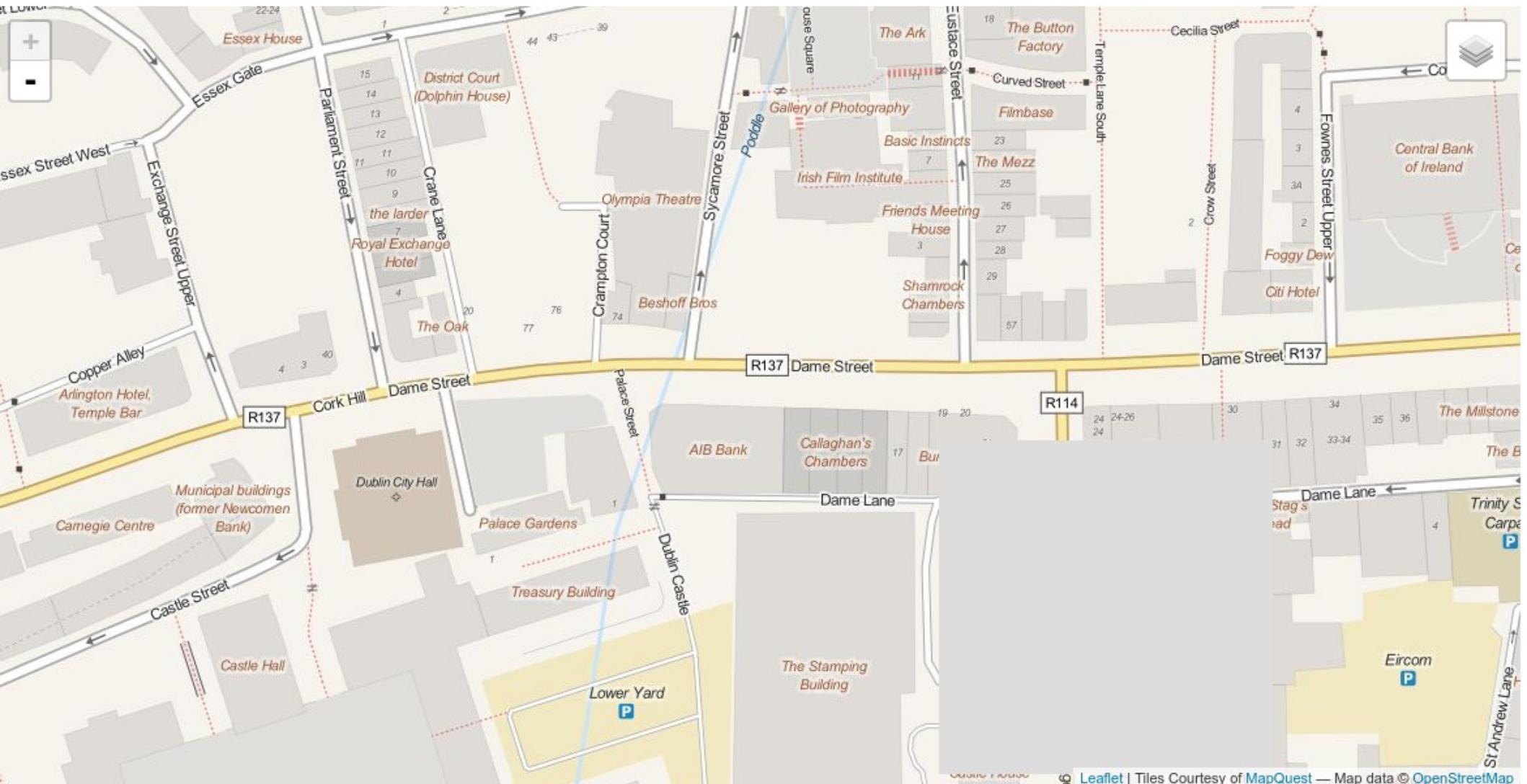
Zoom Level 8 (at country level)



Zoom Level 14 (at city level)



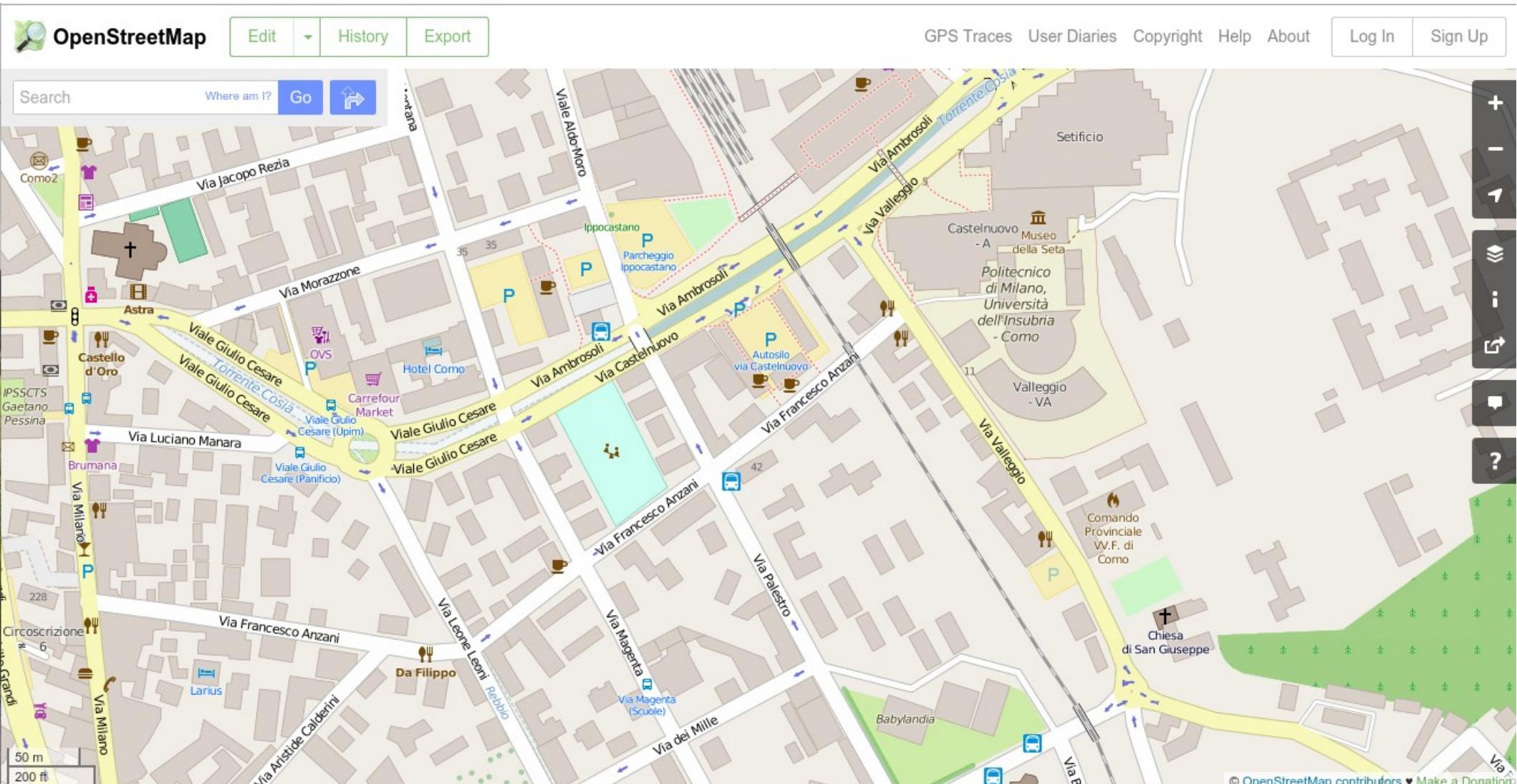
Zoom Level 17 (at ground level)



So ZOOM level goes from 1
(furthest away) to 17 (closest to the
ground)

A little bit about OpenStreetMap

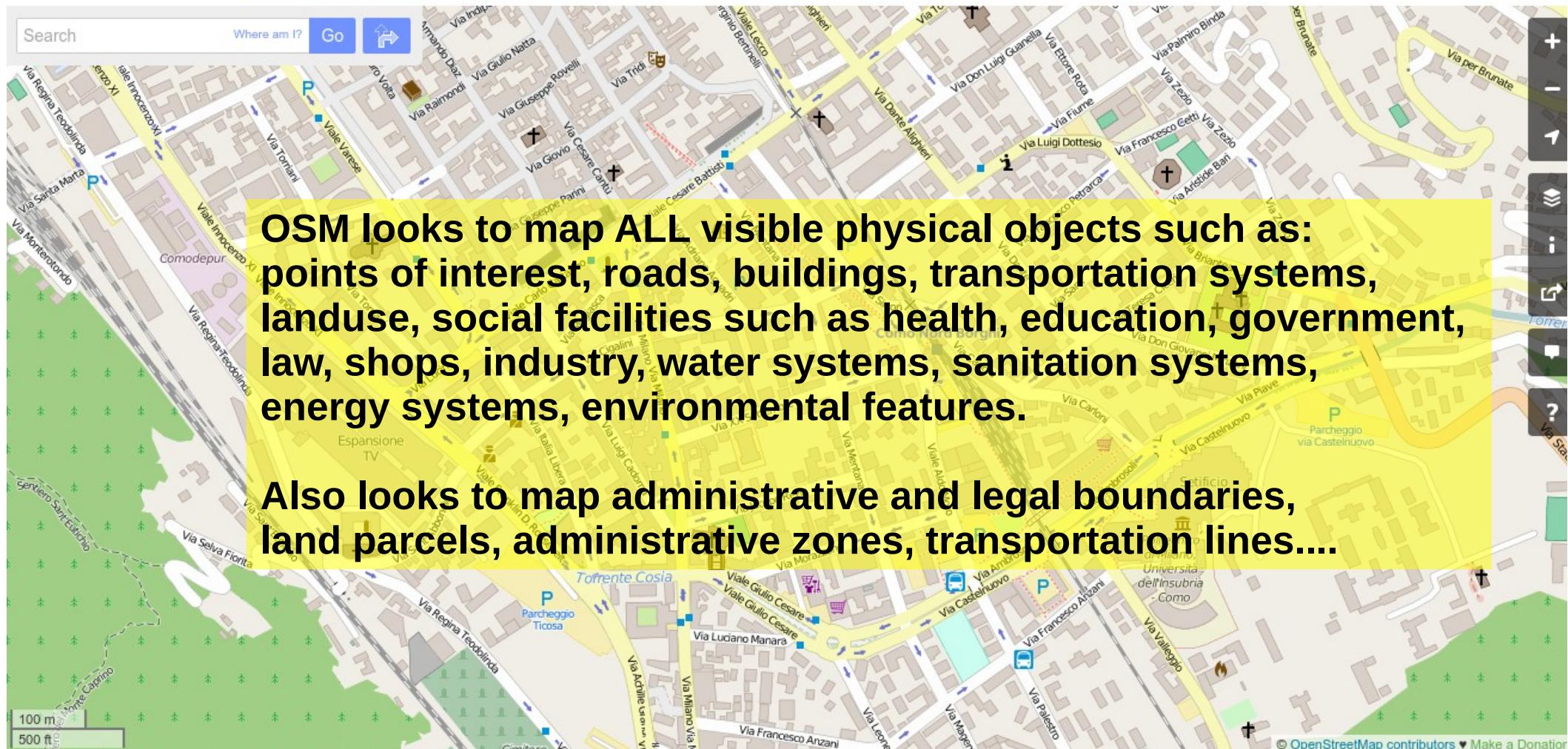
OpenStreetMap is more than just a map...



Search

Where am I?

Go



**OSM looks to map ALL visible physical objects such as:
points of interest, roads, buildings, transportation systems,
landuse, social facilities such as health, education, government,
law, shops, industry, water systems, sanitation systems,
energy systems, environmental features.**

**Also looks to map administrative and legal boundaries,
land parcels, administrative zones, transportation lines....**

OpenStreetMap is actually a massive spatial database

PostgreSQL



Map Compare

tools.geofabrik.de/mc/#17/45.8023/9.0951&num=4&mt0=mapnik&mt1=google-map&mt2=bing-map&mt3=nokia-map

Como search Help Switch tool... ▾

Map Compare

Choose map type: OSM Mapnik

Choose map type: Google Map

Choose map type: Bing Map

Choose map type: HERE Map

zoom=17 number of maps: 1 2 3 4 6 8

All maps except Bing/Google/HERE based on OSM data © [OpenStreetMap](#) (License: ODbL 1.0), OSM Tiles licensed CC-BY-SA 2.0 - [help](#) - [contact](#) - [fullscreen](#)

<http://tools.geofabrik.de/mc/>

Exercise with Map Compare

- Let's look at Map Compare for an area which you are very familiar with – your home, your university, your favourite town, etc
- **Exercise and Discussion:** Use Map Compare to make a visual comparison of OSM in this area with Google Maps, Bing etc.
- Can you make some statements about: coverage, missing data, overall quality of the map data?

**What web technologies are we
using today for the lecture?**

We are using Leaflet for our map container Javascript



an open-source JavaScript library
for mobile-friendly interactive maps

[Overview](#) [Tutorials](#) [Docs](#) [Download](#) [Plugins](#) [Blog](#)

Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 33 KB of JS, it has all the mapping [features](#) most developers ever need.

We are using Bootstrap to make our pages display in a responsive way

Bootstrap

Getting started

CSS

Components

JavaScript

Customize

Expo

Blog



B

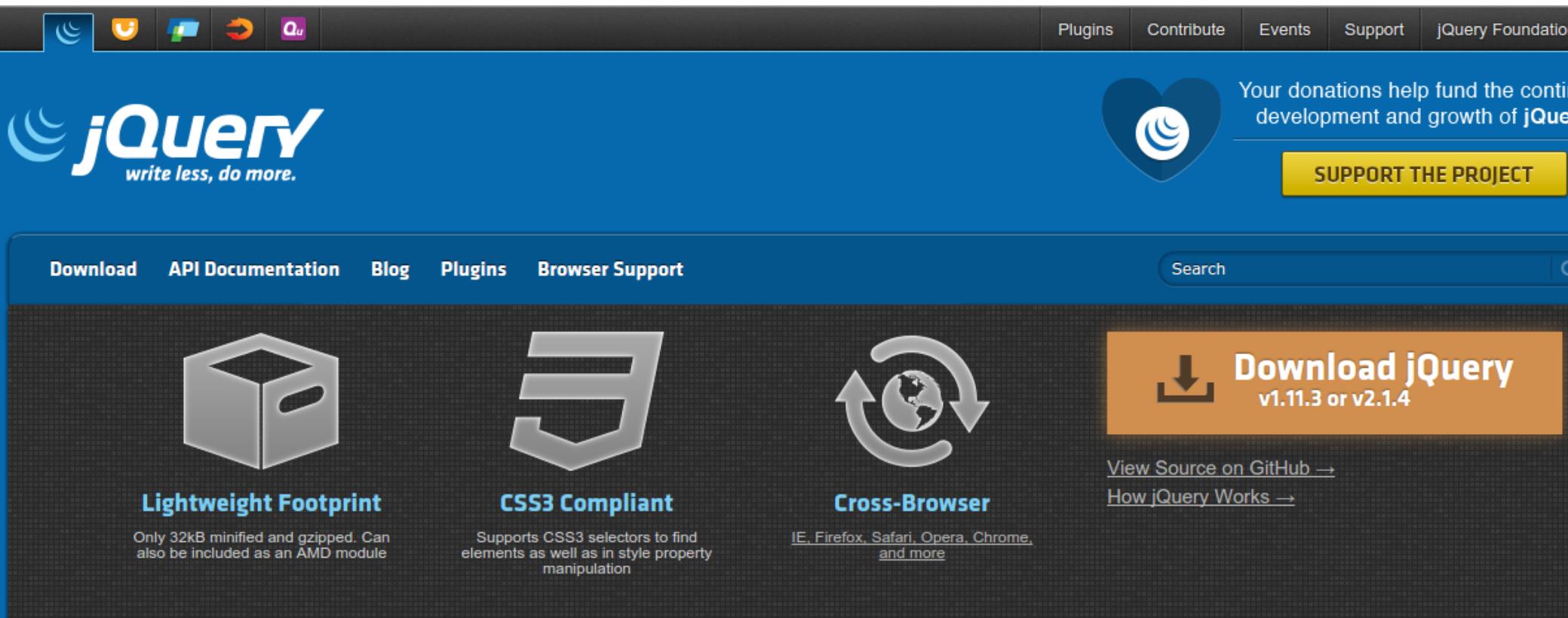
Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

Download Bootstrap

Currently v3.3.5



Jquery provides LOTS of very useful Javascript functionality



The screenshot shows the official jQuery website. At the top, there's a dark header bar with social media icons (Facebook, Twitter, LinkedIn, YouTube, GitHub) and navigation links for Plugins, Contribute, Events, Support, and jQuery Foundation. Below the header is a large blue banner featuring the jQuery logo and the tagline "write less, do more." On the right side of the banner is a heart-shaped donation button with the text "Your donations help fund the continued development and growth of jQuery". A yellow "SUPPORT THE PROJECT" button is also visible. The main content area has a dark background with four key features highlighted: "Lightweight Footprint" (with an icon of a small cube), "CSS3 Compliant" (with an icon of a stylized '3'), "Cross-Browser" (with an icon of a globe with arrows), and a prominent orange "Download jQuery" button for versions v1.11.3 or v2.1.4. Below these features, there are links to "View Source on GitHub" and "How jQuery Works". The bottom section contains two main sections: "What is jQuery?" (describing it as a fast, small, and feature-rich library) and "Resources" (listing links to the Core API Documentation, Learning Center, Blog, and Contribute pages).

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Resources

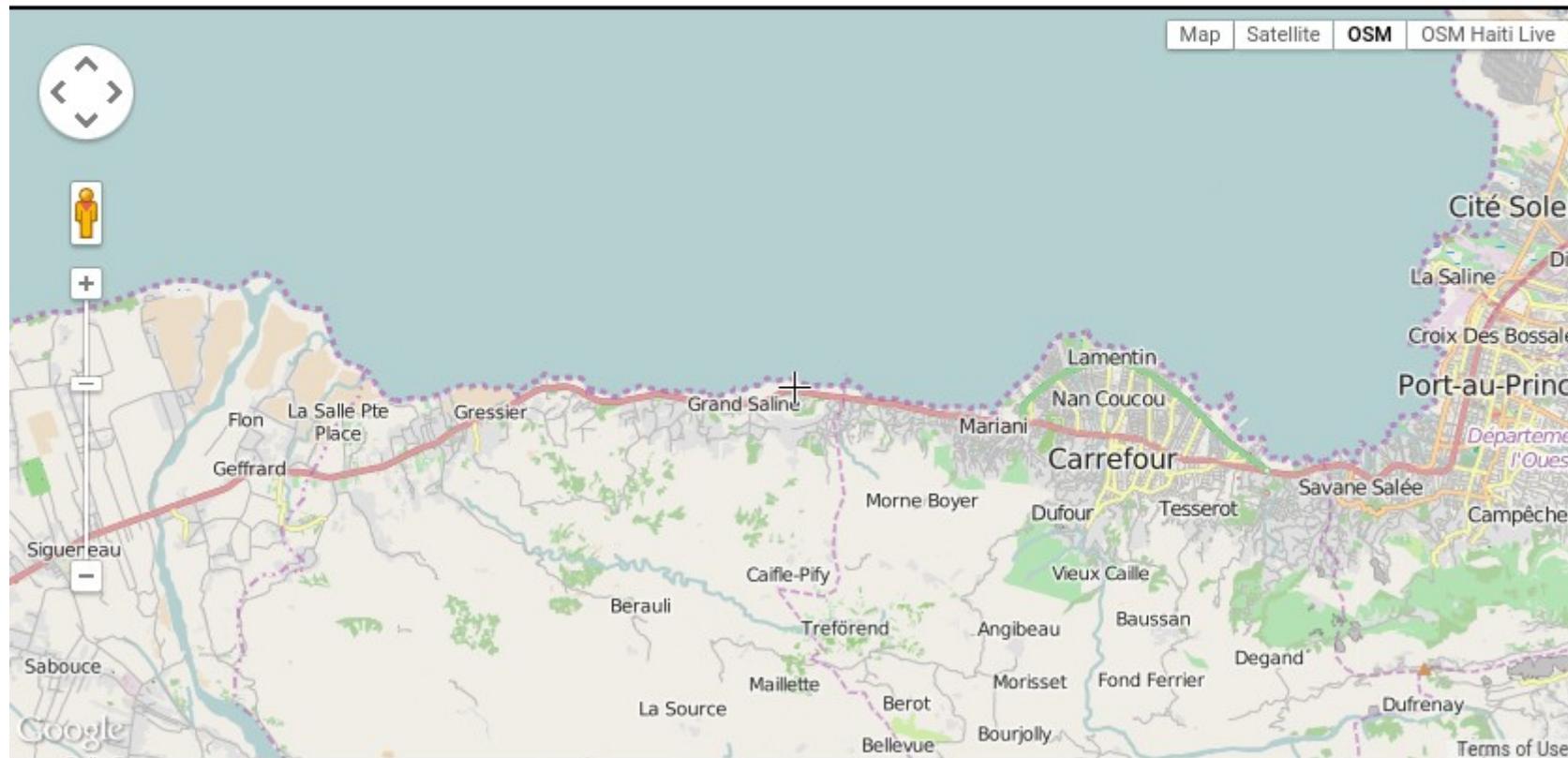
- [jQuery Core API Documentation](#)
- [jQuery Learning Center](#)
- [jQuery Blog](#)
- [Contribute to jQuery](#)

Finding the Latitude Longitude of any point will be very useful today

Get Lat Lon

Find the latitude and longitude of a point on a map.

Place name: [Zoom to place](#)



Latitude, Longitude: 18.54732, -72.46788

Getting Started

Get the Latitude Longitude of ANY point on the earth

<http://dbsgeo.com/latlon/>

Get Lat Lon

Find the latitude and longitude of a point on a map.

Place name: Zoom to place [Zoom to my location \(by IP\)](#)

Pan and Zoom into your place of interest – the Lat/Long will change accordingly

Latitude, Longitude: 45.81291, 9.07210

Getting our material together

- We will need to download all of the materials from GitHUB
- You will unzip these materials into a folder on your computer.
- You then need to copy the Mongoose executable into this folder. **[LATER ON]**
- **DO NOT CHANGE the names of any files or folders within this new folder.**

Download from GitHub

GitHub, Inc. [US] <https://github.com/petermooney/SummerSchool2016>

This repository Search Pull requests Issues Gist

petermooney / SummerSchool2016

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

Unwatch 1 Star 0 Fork 0

Repository for material for Maynooth University Chinese Summer School Lectures on Web Mapping — Edit

3 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

petermooney Changes to Readme and Setup Git

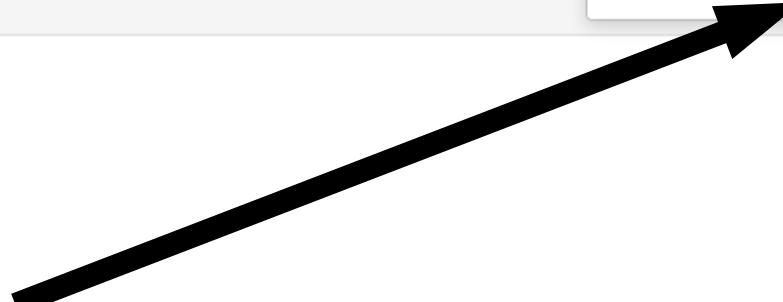
LICENSE Initial commit

README.md Changes to Readme and Setup Git

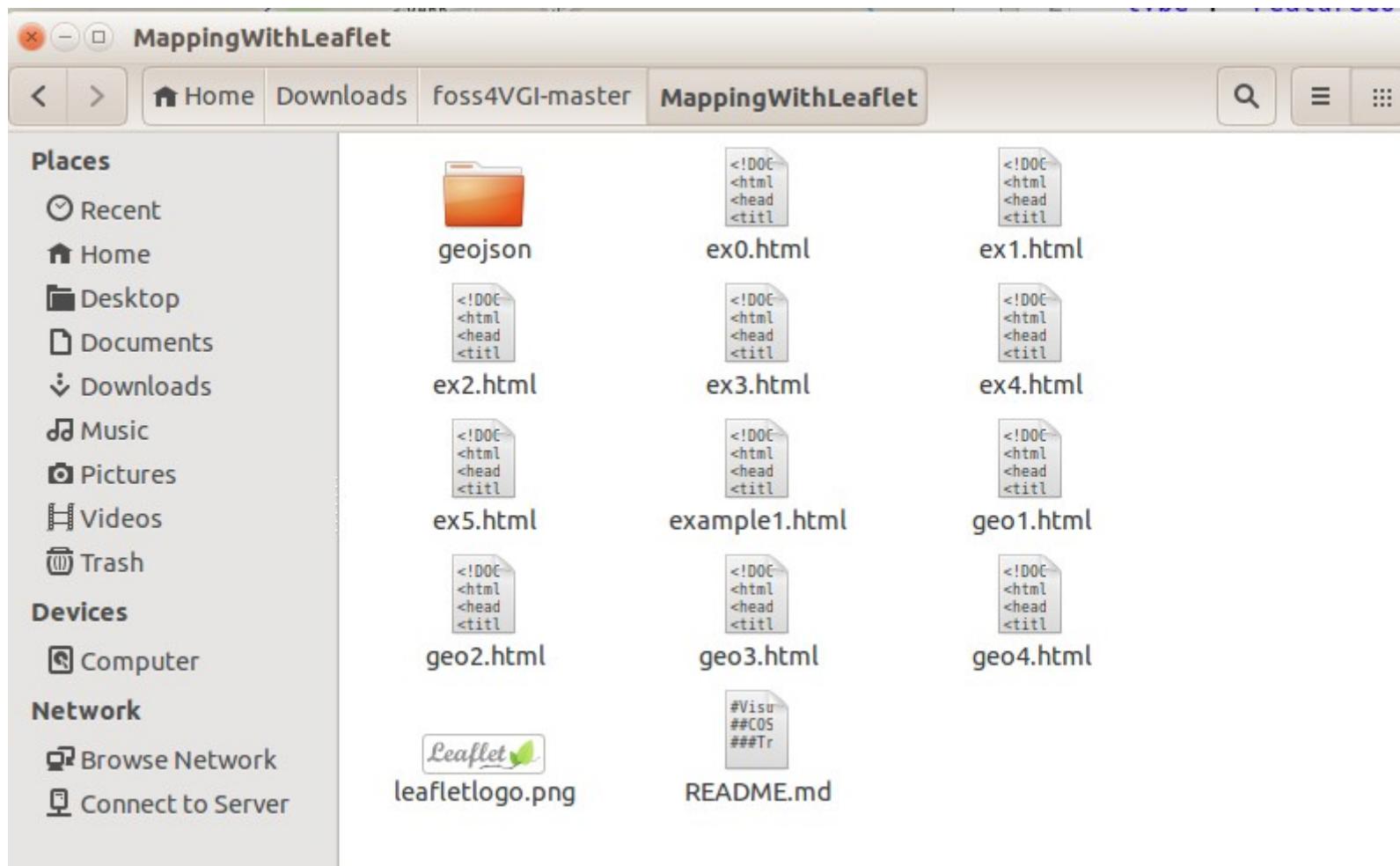
README.md

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/petermooney/SummerSchool2016>

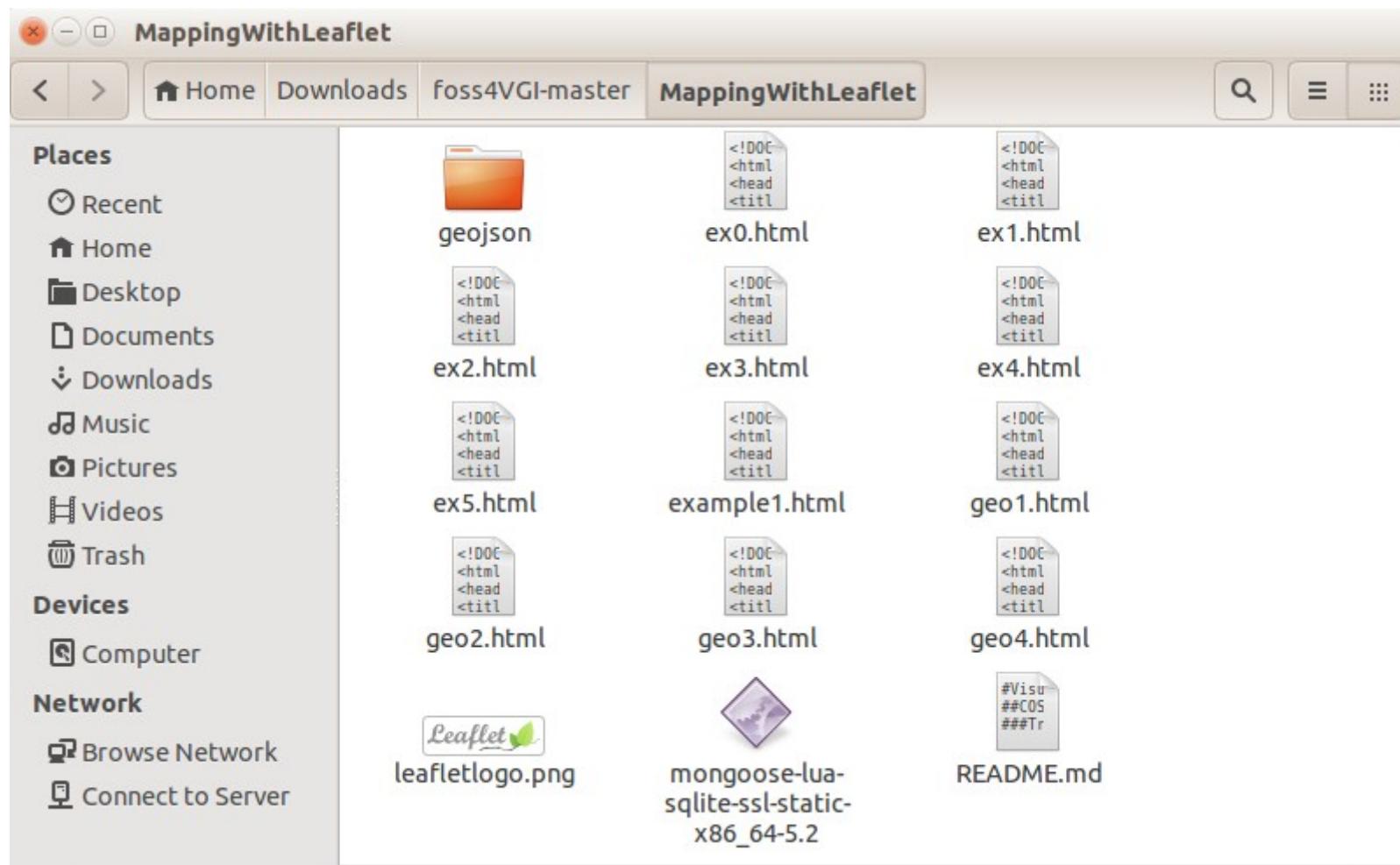
Download ZIP



The Unzipped Files in a Folder



And look – there is the Mongoose Executable Included now!



Every webpage in the world has the same structure regardless of what the webpage actually does



A webpage at its most fundamental has a `<head>` and has a `<body>`

The `<body>` is what you see in your web-browser. The `<head>` helps to support the technical details of how the `<body>` is managed and displayed.

Unfortunately – we do not have enough time to go into the deeper details of web page development

- So we are going to take a few things for granted and make some assumptions.
- In our examples – the <HEAD> of our pages WILL NOT CHANGE except for the <TITLE>
- In our examples – the <BODY> of our pages will change but we are not trying to design very fancy beautiful webpages.
- **The focus today is on DISPLAYING MAPS WITH GEODATA in a webpage and NOT webpage development.**

Let's open ex0.html in your text editor which you installed

- We are going to step through this very carefully so we can see the <HEAD> and the <TITLE> and the <BODY>

The <HEAD> of our Web Pages

```
2 <html>
3   <head>
4     <title>FOSS4VGI - Leaflet Lectures - Example 0</title>
5     <!-- This part of our web page should hardly change over the course of lecture today -->
6     <!-- This part of the web-page tells the web-browser it will need certain functionality
7       to make the page display/function as the programmer intended -->
8     <!-- There should be no need for you to change any of the links below -->
9
10    <meta charset="utf-8" />
11    <meta name="viewport" content="width=device-width, initial-scale=1">
12
13    <!-- Bootstrap CSS -->
14    <link rel="stylesheet" href=
15      "https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
16
17    <!-- Bootstrap Optional theme -->
18    <link rel="stylesheet" href=
19      "https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">
20
21    <!-- JQuery - which is needed by Bootstrap and Leaflet -->
22    <script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
23
24    <!-- Latest compiled and minified JavaScript for Bootstrap -->
25    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
26
27    <!-- Include Leaflet -->
28    <script src="http://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.js"></script>
29    <link rel="stylesheet" href=
30      "http://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.css" />
31
32  </head>
```

The <BODY> of our Web Pages

```
<body>
<div class="container">
    <!-- start the row of content -->
<div class="row">
    <h1>This is Example 0</h1> <!-- We can change this piece of text each time -->
    <div id="map" style="width: 100%; height: 600px"></div>
</div><!-- close the row of content-->

<script><!-- This is where our mapping code starts -->
// This is our base layer - you should not need to change this in many examples
var OpenStreetMap_Mapnik = L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
{maxZoom: 19, attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'});

// Notice how the MAP is centered at a particular point - notice the ZOOM level
var map = L.map('map', {center: [45.81288, 9.07454], zoom: 19, layers: [OpenStreetMap_Mapnik]});

var baseMaps = {"OpenStreetMap_Mapnik": OpenStreetMap_Mapnik};

var overlayMaps = { };

L.control.layers(baseMaps, overlayMaps).addTo(map);

// This is a marker pin on a particular spot. It does not have to be the center
L.marker([45.81288, 9.07454]).addTo(map).bindPopup("<b>Hello Everyone</b><br/>This is the
Cube Cafe in Como.");
bindPopup();

</script><!-- This is where our mapping code ends -->
</div><!-- close the container -->
</body>
</html>
```



Exercise A: Let's put some marker pins on our favourite places

- Copy the file ex0.html and rename the file exerciseA.html (*stay in the same folder*)
- Open exerciseA.html in your text editor.
- Using the 'get lat long' web page – find the lat/longitude of your favourite places – change the code in your file so that the marker pins are changed.
- CREATE THREE MARKERS
- Change also the <TITLE> and the <h1> text
- SAVE the file and then look at **exerciseA.html**

Exercise B: Let's put multiple marker pins on a map using our own data

- Copy the file ex1.html and rename the file exerciseB.html (*stay in the same folder*)
- Open exerciseB.html in your text editor
- Use Get Lat Long to find the locations of FOUR TRAIN STATIONS close to MAYNOOTH
- You will need to also test and find an appropriate center for your map.
- You will need to fina an appropriate zoom
- **Result at exerciseB.html**

The code for adding more markers to your map is very simple

```
42 // Notice how the MAP is centered at a particular point - notice the ZOOM level
43 var map = L.map('map', {center: [45.80880, 9.08367], zoom: 15, layers: [OpenStreetMap_Mapnik]});
44
45 var baseMaps = {"OpenStreetMap_Mapnik": OpenStreetMap_Mapnik};
46
47 var overlayMaps = { };
48
49 L.control.layers(baseMaps, overlayMaps).addTo(map);
50
51
52 // This is a marker pin on a particular spot. It does not have to be the center
53 L.marker([45.81288, 9.07454]).addTo(map).bindPopup("<b>Hello Everyone</b><br/>This is the Cube Cafe in Como.");
54
55 L.marker([45.81413, 9.08407]).addTo(map).bindPopup("<b>Hello Train Users</b><br/>This is the Como Lago Nord train
station.");
56
57 L.marker([45.80168, 9.08913]).addTo(map).bindPopup("<b>Hello Shoppers</b><br/>This is the Carrefour Supermarket in
the south of Como city.");
58
59
```

NOTICE that the CENTER of the map is different to the locations of the 3 markers which we have added

Let's look at example0.html in your text editor – multiple markers

- You can have multiple markers on your map – it's no problem. You can actually have hundreds of marker icons on your map if you need.
- **Decision 1:** When you have more than one marker – you will need to decide where to center the map
- **Decision 2:** You will also need to decide upon the appropriate zoom level
- **Decision 3:** You might need to use a different LAT/LONG for your center

Exercise B: Let's put multiple marker pins on a map using our own data

- Copy the file exampleA.html and rename the file exerciseB.html (*stay in the same folder*)
- Open exerciseB.html in your text editor
- Use Get Lat Long to find the locations of FOUR TRAIN STATIONS close to MAYNOOTH
- You will need to also test and find an appropriate center for your map.
- You will need to find a an appropriate zoom and center
- **Result at exerciseB.html**

But figuring out the best center and zoom is tricky – what happens if there are lots of points or markers?

Let's look at ex1.html in the text editor window

```
// Notice here now that we do not specify the zoom or the center. |
var mymap = L.map('map', {
  layers: listOfLayers
});

L.control.layers(baseMaps).addTo(mymap);

// We can add actually make Leaflet handle the centering of the map.

var marker1Coords = L.latLng(53.38389, -6.57733);
var marker1 = L.marker(marker1Coords).addTo(mymap);      Creating multiple markers
marker1.bindPopup("This is Marker 1");

var marker2Coords = L.latLng(53.3865, -6.56723);
var marker2 = L.marker(marker2Coords).addTo(mymap);
marker2.bindPopup("This is Marker 2");

var marker3Coords = L.latLng(53.2765, -6.58700);
var marker3 = L.marker(marker3Coords).addTo(mymap);
marker3.bindPopup("This is Marker 3");

var marker4Coords = L.latLng(53.1865, -6.72710);
var marker4 = L.marker(marker4Coords).addTo(mymap);      Creating an array of coordinates
marker4.bindPopup("This is Marker 4");

// We can use the latLngBounds method (which takes an array of coordinates)
var mapBounds = L.latLngBounds([marker1Coords, marker2Coords, marker3Coords, marker4Coords]);

// The map will be fitted to these boundaries.      Then use the Leaflet function to
mymap.fitBounds(mapBounds);                      fitBounds()
```

Exercise C: Using Leaflet MapBounds function

- Copy **ex1.html** and rename it as **exerciseC.html**
- Pick 5 cities in the UK and Ireland – find a train station in each city.
- Create LatLong coordinates for each and put some marker text in each popup.
- Use the **ex1.html** code to get Leaflet to calculate the correct bounds

Changing the Background Layers

Let's look at e2.html – where we have TWO layers

```
49
50 var OpenStreetMap_Mapnik = L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
51   maxZoom: 19,
52   attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
53 });
54 var Esri_WorldImagery = L.tileLayer('http://server.arcgisonline.
55 com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', {
56   attribution: 'Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping,
57   Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community'
58 });
59
60 var listOfLayers = [Esri_WorldImagery,OpenStreetMap_Mapnik];
61
62 var baseMaps = {
63   "ESRI Aerial": Esri_WorldImagery,
64   "OpenStreetMap Streets": OpenStreetMap_Mapnik
65 };
66
67 // Notice here now that we do not specify the zoom or the center.
68 var mymap = L.map('map', {
69   layers: listOfLayers
});
```

If we wanted to change an existing layer – or add new ones – where would we need to make changes?

We would make changes in THREE places in our code

```
49
50 var OpenStreetMap_Mapnik = L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', { 1
51   maxZoom: 19,
52   attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
53 });
54 var Esri_WorldImagery = L.tileLayer('http://server.arcgisonline.
55 com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', {
56   attribution: 'Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping,
      Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community'
57 });
58 var listOfLayers = [Esri_WorldImagery,OpenStreetMap_Mapnik]; 2
59
60 var baseMaps = {
61   "ESRI Aerial": Esri_WorldImagery, 3
62   "OpenStreetMap Streets": OpenStreetMap_Mapnik
63 };
64
65 // Notice here now that we do not specify the zoom or the center.
66 var mymap = L.map('map', {
67   layers: listOfLayers
68 });
69
```

Choosing a background layer

- We are going to look at some choices of background layers
- REMEMBER
- You will choose a layer which is appropriate to your data – you need to make sure that your users can easily understand the data and understand the significance of the background layer
- GO TO (this link is on the GitHub page)
<http://leaflet-extras.github.io/leaflet-providers/preview/>

Leaflet Layer Providers

Fork me on GitHub

[leaflet-extras.github.io/leaflet-providers/preview/](#)

Leaflet-providers preview

This page shows mini maps for all the layers available in [Leaflet-providers](#).

Provider names for leaflet-providers.js

OpenTopoMap

Plain JavaScript:

```
// https: also supported.  
var OpenTopoMap = L.tileLayer('http://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {  
    maxZoom: 16,  
    attribution: 'Map data: © <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'  
});
```

YOU WILL NEED TO COPY THIS EXACTLY

OpenStreetMap.DE

OpenStreetMap.France

OpenStreetMap.HOT

OpenTopoMap

Thunderforest.OpenCycleMap

Exercise D: Changing the background layer in Leaflet

- Copy the file **exerciseB.html** and rename the file **exerciseC.html** (*stay in the same folder*)
- Open **exerciseC.html** in your text editor
- Pick your favourite layer from the Leaflet Layer Providers page
- Copy this code and REPLACE the Mapnik Definition. You will also need to change the line containing 'var baseMaps'

If you want you can ADD a THIRD or FOURTH layer....

This is one example of a different background layer



This is Example 2 - Changing the Background Layer



Adding Polygons to Leaflet

It is easy to add polygons to Leaflet using similar code to that of points

```
var polygon1 = L.polygon([
  [51.509, -0.08],
  [51.503, -0.06],
  [51.502, -0.05],
  [51.51, -0.047],
  [51.509, -0.08]
]).addTo(mymap);
```

REMEMBER: A polygon is a closed object – so the start and end nodes must be the same!

Example E: Adding a polygon

- We can use a tool such as 'Get Lat Long' to help us get the points we need to make up a polygon. Look at **e3.html** in your text editor

```
65  <!-- Here is the CENTER coordinates of our map -->
66
67  var LatitudeCenter = 51.509;
68
69  var LongitudeCenter = -0.05;
70
71  <!-- here is the starting zoom level of our map -->
72
73  var ZoomLevel = 14;
74
75  var mymap = L.map('map', {
76    center: [LatitudeCenter, LongitudeCenter],
77    zoom: ZoomLevel,
78    layers: list0fLayers
79  });
80
81  L.control.layers(baseMaps).addTo(mymap);
82
83 // Here we can create a Polygon Object and add it to the map
84  var polygon1 = L.polygon([
85    [51.509, -0.08],
86    [51.503, -0.06],
87    [51.502, -0.05],
88    [51.51, -0.047],
89    [51.509, -0.08]
90  ]).addTo(mymap);
91
```

Exercise E: Adding more Polygons

- Copy **e3.html** and rename it as **exerciseD.html**
- **AIM:**
- Use 'Get Lat Long' to help create TWO NEW POLYGONS (you can fully delete the polygon which is already in **e3.html**)
- The TWO NEW POLYGONS will represent TWO Educational facilities in your town, city or Region. You will need to find an appropriate center for the map. You will need to provide text for the popup windows.

**OK! Let's take a break from
Polygons for a moment**

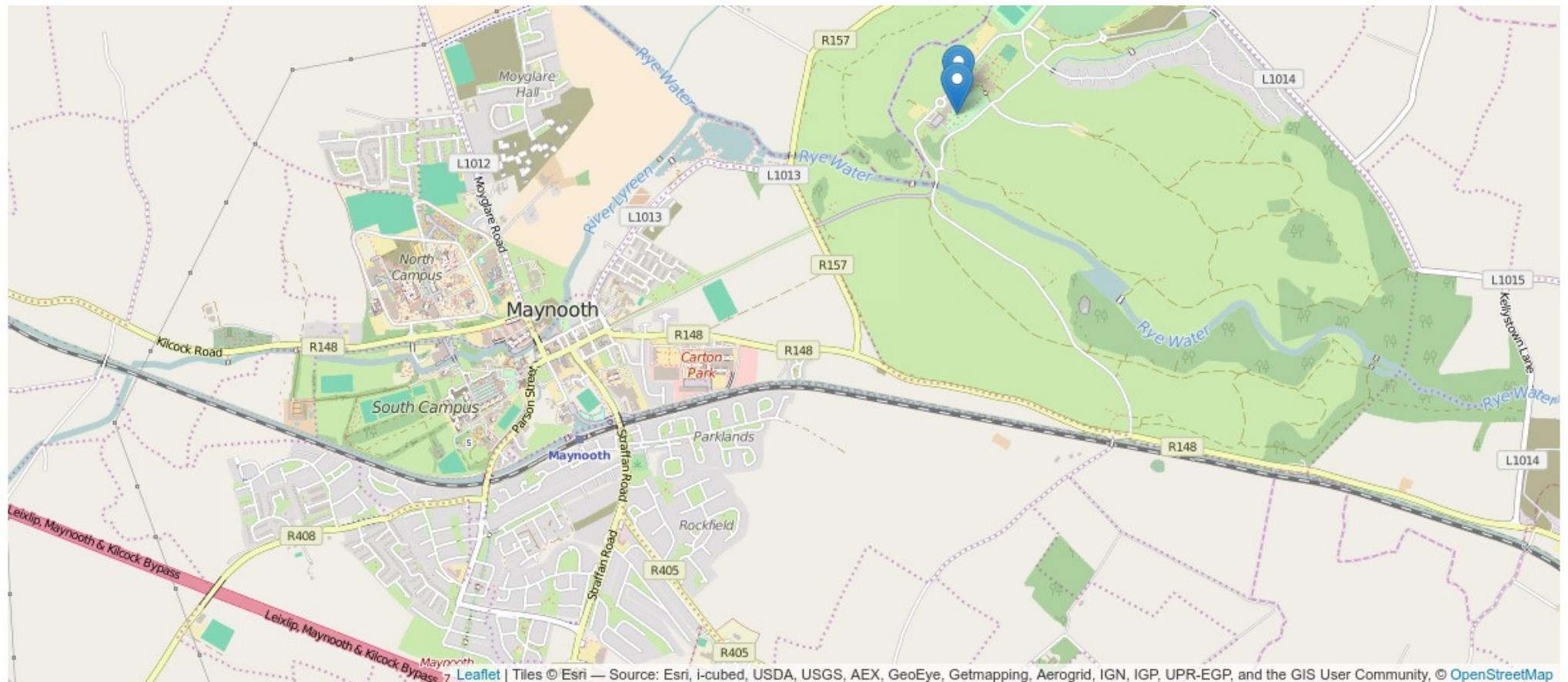
**Let's create our very own simple
version of Get Lat Long for
ourselves**

E4.html – Open this file in your text editor and view it in the browser

- Open **e4.html** in your text editor
- Check out **e4.html** in the browser
- **We have added some code – but it is a really useful piece of code**

```
93
94 // This is the where Javascript is connected to the map when you 'click' on the map
95 mymap.on('click', onMapClick);
96
97 // THIS IS NEW. This is a special FUNCTION from Javascript.
98 // Leaflet has lots of functionality that we can use. This is very useful to display information about the map
99 function onMapClick(e) {
100     var theDateAndTime = new Date();
101     var mapCenter = map.getCenter();
102     $("#mycoordinates").html("<p class = 'lead'>The Maynooth MyGetLatLong Says:<br/> At " + theDateAndTime + "<br/>You clicked the map at Coordinates " + "(Lat, Lon) = " + e.latlng.lat + "," + e.latlng.lng + "<br/>Also the map is at Zoom level " + map.getZoom() + "<br/>The CENTER of the map is currently at Coordinates " + "(Lat,Lon) = " + mapCenter.lat + "," + mapCenter.lng + "</p>");
103 }
104 }
```

This is the output from E4.html



The Maynooth MyGetLatLong Says:

At Wed Aug 03 2016 14:46:14 GMT+0100 (IST)

You clicked the map at Coordinates (Lat, Lon) = 53.37411237125759,-6.545276641845702

Also the map is at Zoom level 14

The CENTER of the map is currently at Coordinates (Lat,Lon) = 53.38389,-6.57733

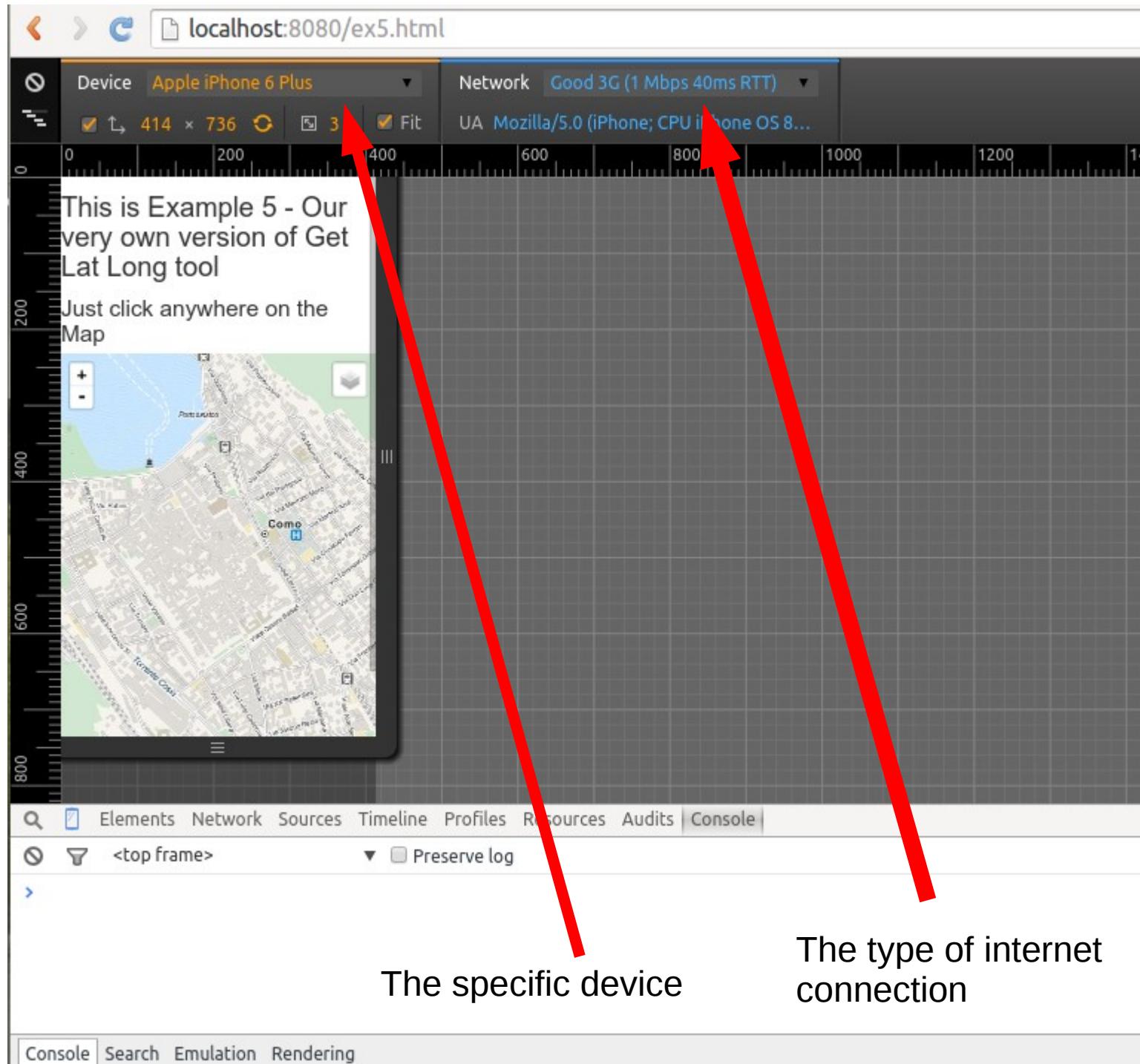
How does all this look on your mobile phone?

Can we test how this will all look on your smartphone or tablet?



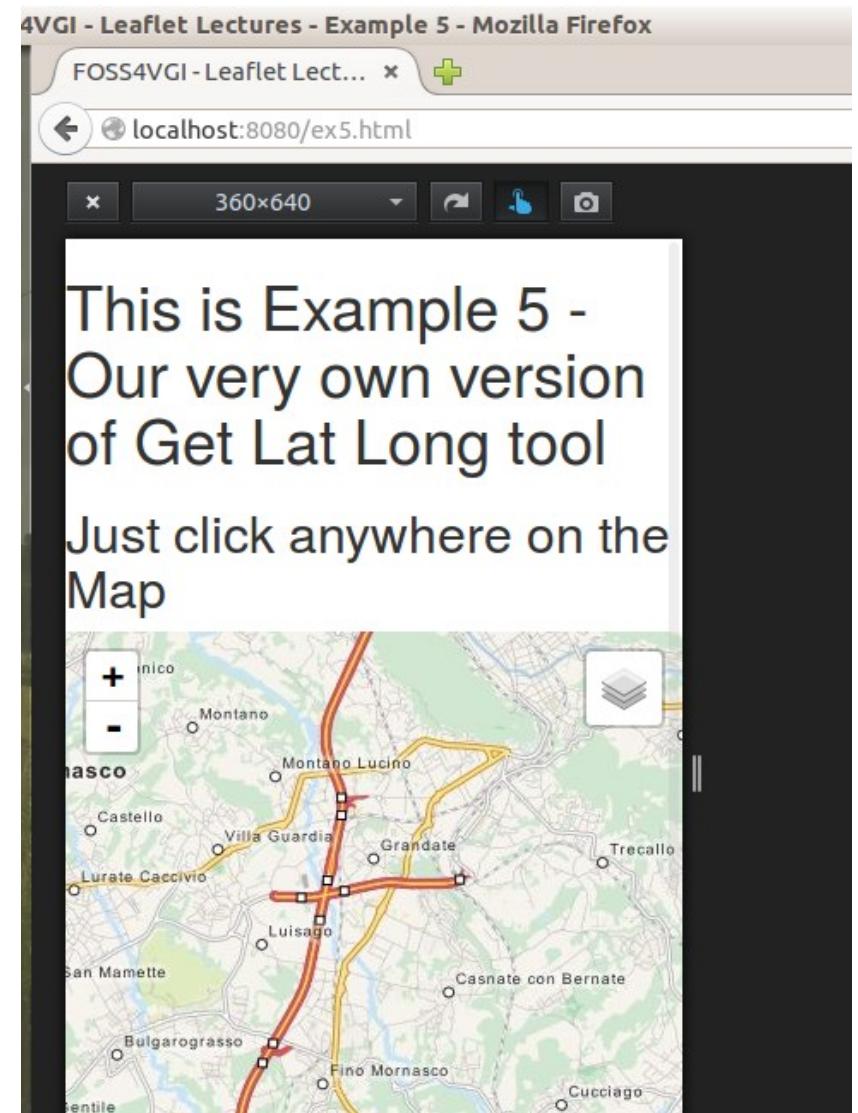
If you are using Google Chrome

- Open up **e4.html** in your Google Chrome
- Go to the options menu
- Then go to More Tools
- Then go to Developer Tools
- Click on the little smartphone icon in the left hand corner.



If you are using Firefox

- Go to options
- Go to Developer
- Go to 'Responsive Design View"

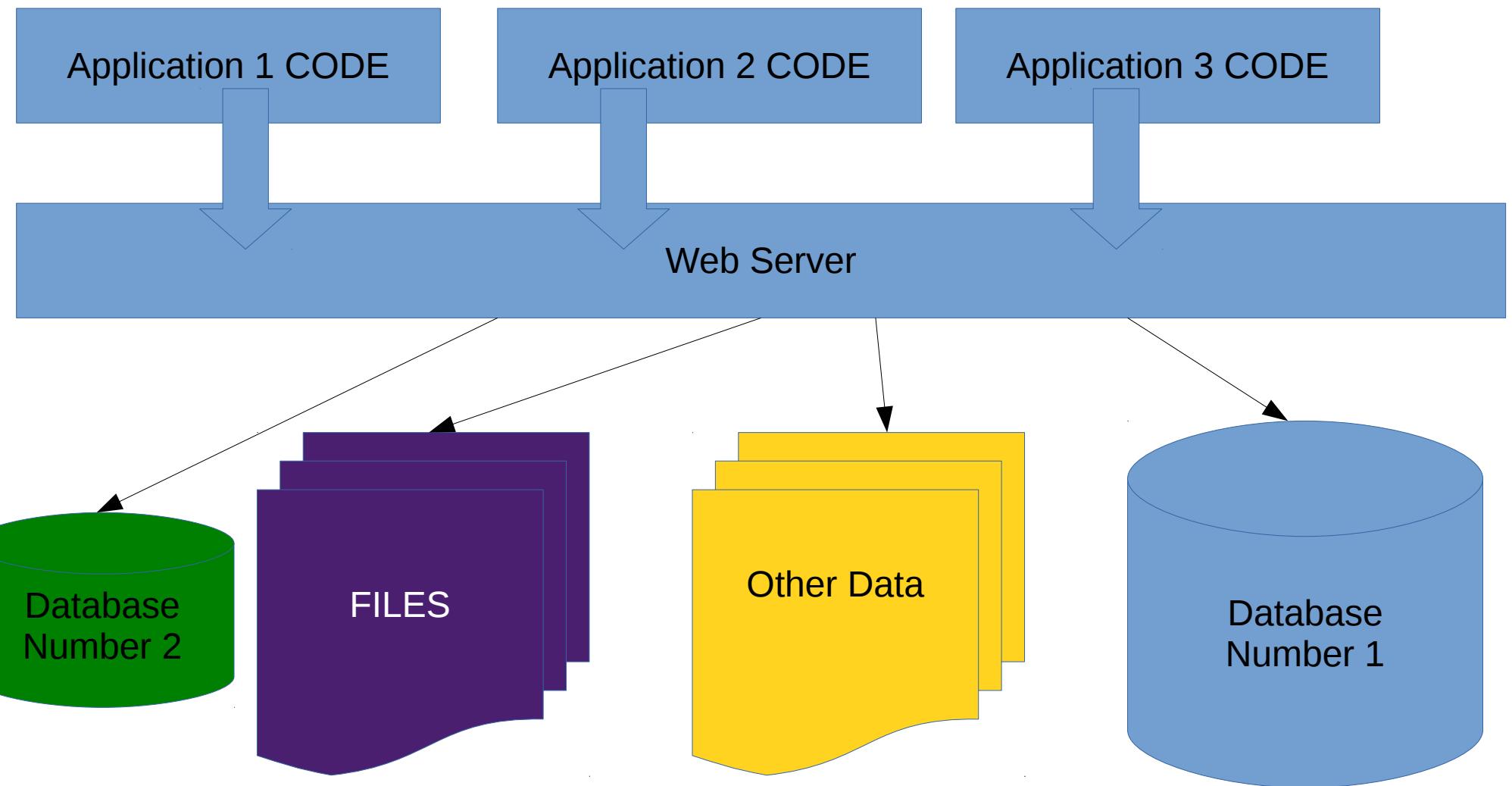


WEB SERVERS

We are going to need to have a web-server to run some of the examples today

- A web server is a special piece of software which provides the software technology to make data and content available on the Internet
- There are millions of web servers running around the world but very often we don't even think about them or what they do!

A very simplistic view of what a web-server actually does



We will need a web server for some of the more advanced examples

- There are LOTS of choices for this.
- The BEST solution (in my opinion) would be to use Apache Server
- It is probably the best web-server available (and it is Open Source)
- But it can be tricky to install when we are dealing with lots of different machines and operating systems.

Download Mongoose Free



Products Use Cases About Developers Blog Contact



Mongoose Binary

CONTACT US FOR
MULTIPLE BINARY SALES

One license per unique device.
Commercial use and distribution is
highly restricted. [Click here for
more information.](#)

Looking for Mongoose Embedded
Web Server & Multiprotocol
Networking Library? [Click here.](#)

FREE 6.5

\$ 0.00

Stable version with all
features listed below as well
as a free 1-week trial of all
Pro and PHP Dev edition
features.

I accept Mongoose End-User
License Agreement ([EULA](#))

Download for Mac

Download for Windows

PRO 6.5

\$ 10.00

Mongoose for Windows and
MacOS including life-time
software update.

I accept Mongoose End-User
License Agreement ([EULA](#))

Download for Mac

Download for Windows

PHP DEV 6.5

\$ 30.00

Professional grade, ready-to-go
Mongoose + PHP + SQLite
bundle for Windows.

I accept Mongoose End-User
License Agreement ([EULA](#))

Download for
Windows

Chat with us

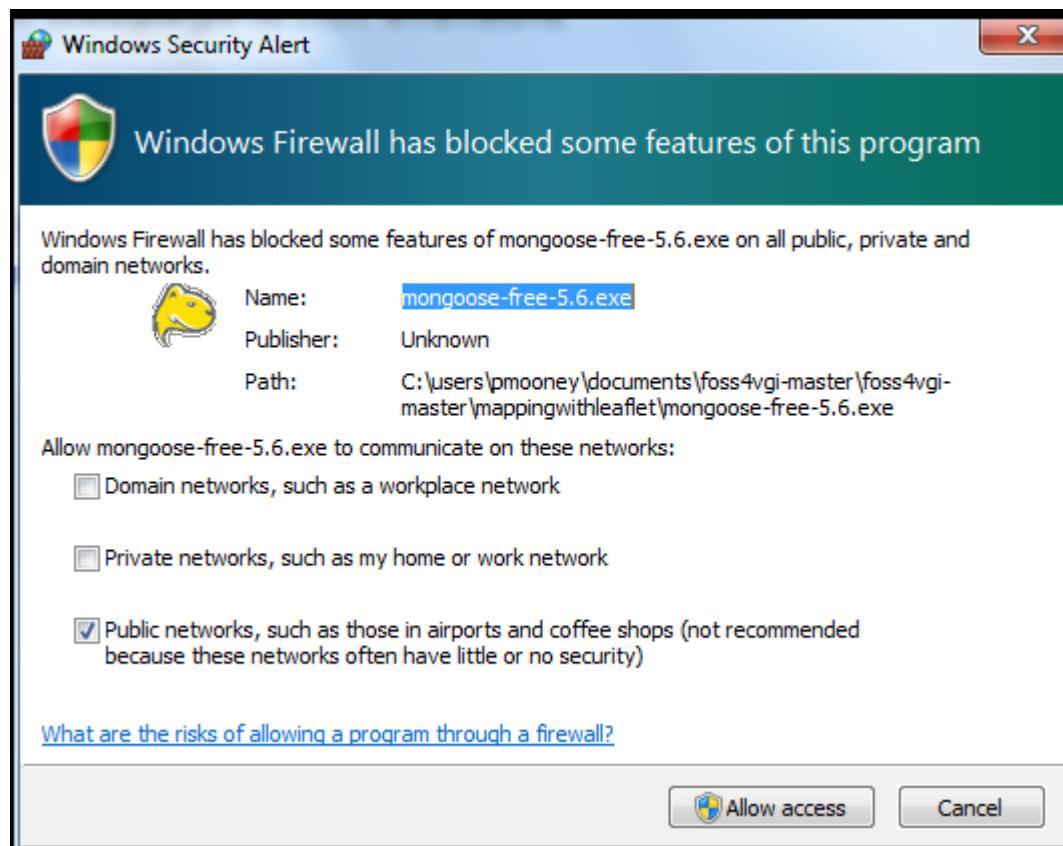


We've made the Windows
executable available for
download in the GitHub
download

Mongoose – Windows Users

- Double click on the EXE file
- Ensure that when you run the file that you do so as administrator (check that Windows has not prevented you from running it)
- Check in your browser that the URL
<http://127.0.0.1:8080> or **<http://localhost:8080>** work

Mongoose Windows Users



When Mongoose starts up you should see something like this



The screenshot shows a web browser window with the URL `149.157.243.132:8080` in the address bar. The page title is "Index of /". Below the title is a table listing files and their details.

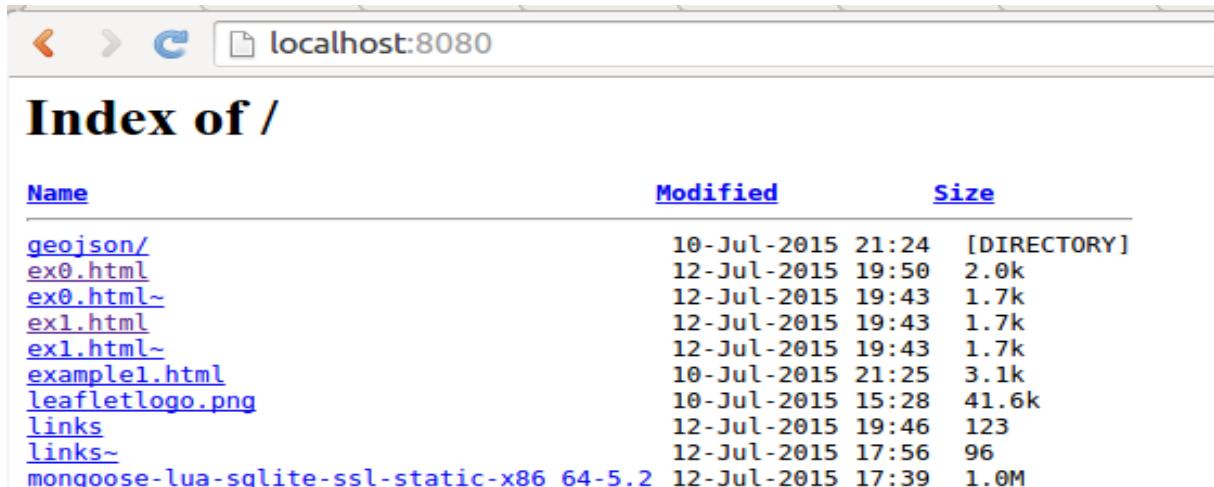
Name	Modified	Size
geojson/	09-Nov-2015 12:33	[DIRECTORY]
README.md	09-Nov-2015 12:29	1.8k
Visualizing_VGI_With_Leaflet_PeterMooney.pdf	09-Nov-2015 12:29	5.1M
access_not_allowed.PNG	09-Nov-2015 12:42	50.1k
ex0.html	09-Nov-2015 12:29	2.5k
ex1.html	09-Nov-2015 12:29	2.7k
ex2.html	09-Nov-2015 12:29	2.9k
ex3.html	09-Nov-2015 12:29	3.4k
ex4.html	09-Nov-2015 12:29	2.9k
ex5.html	09-Nov-2015 12:29	3.2k
example1.html	09-Nov-2015 12:29	3.1k
example1.html~	09-Nov-2015 12:29	3.1k
geo1.html	09-Nov-2015 12:29	3.5k
geo2.html	09-Nov-2015 12:29	4.8k
geo3.html	09-Nov-2015 12:29	4.8k
geo4.html	09-Nov-2015 12:29	5.0k
leafletlogo.png	09-Nov-2015 12:29	41.6k
mongoose-free-5.6.exe	09-Nov-2015 12:39	220.1k
mongoose_in_windows_explorer.PNG	09-Nov-2015 12:43	182.7k

Windows and Linux Users

- DO NOT DELETE the Mongoose file
- As this is just a TEST ENVIRONMENT it is OK here – but this technique should not be used in a production environment
- However that's a different set of lecture notes!

We have a special web address now that Mongoose is operational

- We can now type in the address
<http://127.0.0.1:8080> or <http://localhost:8080>
into your browser
- You should be given a list of the files in the folder.
- Just click on “ex0.html” and wait for the lecture slides to catch up



A screenshot of a web browser window showing a file listing. The address bar at the top displays "localhost:8080". Below the address bar, the text "Index of /" is centered. A table lists various files and folders:

Name	Modified	Size
geojson/	10-Jul-2015 21:24	[DIRECTORY]
ex0.html	12-Jul-2015 19:50	2.0k
ex0.html~	12-Jul-2015 19:43	1.7k
ex1.html	12-Jul-2015 19:43	1.7k
ex1.html~	12-Jul-2015 19:43	1.7k
example1.html	10-Jul-2015 21:25	3.1k
leafletlogo.png	10-Jul-2015 15:28	41.6k
links	12-Jul-2015 19:46	123
links~	12-Jul-2015 17:56	96
mongoose-lua-sqlite-ssl-static-x86_64-5.2	12-Jul-2015 17:39	1.0M

We cannot proceed beyond this point unless you have Mongoose running successfully!



Download from
Dreamstime.com

This watermarked comp image is for previewing purposes only.



ID 33352709

© Mila Gligoric | Dreamstime.com

Leave Mongoose Running!

Starting with GeoJSON and Leaflet

GeoJSON and Leaflet are made for each other :-)



- **Leaflet is able to process, manipulate and display GeoJSON data**
- So it is in our best interests (because it is very efficient) to try to format our spatial data in GeoJSON format so that Leaflet can display it for us on a web map

Remember: Every GeoJSON file is a dataset in its own right

- This is an important consideration.
- We cannot just “throw” the GeoJSON file at Leaflet and expect Leaflet to display the file.
- **We will have to understand what the data contents of our GeoJSON file is in order to generate the best way to display our data**
- We shall start off with a GeoJSON file from the web and then proceed to OpenStreetMap data

geo1.html – US Geological Survey EarthQuake Data

- <http://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php>

The screenshot shows a web browser displaying the USGS Earthquake Hazards Program website. The URL in the address bar is <http://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php>. The page title is "GeoJSON Summary Format". On the left, there's a sidebar with "Real-time Feeds" (ATOM, KML, Spreadsheet, QuakeML) and "Real-time Notifications" (Earthquake Notification Service, Tweet Earthquake). The main content area has sections for "Description" (explaining GeoJSON), "Usage" (GeoJSON as a programmatic interface), and "Output" (GeoJSON format). On the right, there are "Feeds" for "Past Hour" (updated every 5 minutes) and "Past Day" (updated every 5 minutes), each with a list of links: "Significant Earthquakes", "M4.5+ Earthquakes", "M2.5+ Earthquakes", "M1.0+ Earthquakes", and "All Earthquakes".

earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php

USGS science for a changing world

Earthquake Hazards Program

Feeds & Notifications

Real-time Feeds

- ATOM
- KML
- Spreadsheet
- QuakeML

Real-time Notifications

- Earthquake Notification Service
- Tweet Earthquake

GeoJSON Summary Format

Description

GeoJSON is a format for encoding a variety of geographic data structures. A GeoJSON object may represent a geometry, a feature, or a collection of features. GeoJSON uses the [JSON standard](#). The GeoJSONP feed uses the same JSON response, but the GeoJSONP response is wrapped inside the function call, eqfeed_callback. See the [GeoJSON site](#) for more information.

This feed adheres to the USGS Earthquakes [Feed Life Cycle Policy](#).

Usage

GeoJSON is intended to be used as a programmatic interface for applications.

Output

Feeds

Past Hour

Updated every 5 minutes.

- [Significant Earthquakes](#)
- [M4.5+ Earthquakes](#)
- [M2.5+ Earthquakes](#)
- [M1.0+ Earthquakes](#)
- [All Earthquakes](#)

Past Day

Updated every 5 minutes.

geo1.html – please scroll to the bottom of this page and download the Past 7 Days 'All Earthquakes'

For Developers

- API Documentation - EQ Catalog
- GeoJSON Summary
- GeoJSON Detail
- Developers Corner
- Glossary
- Change Log
- Feed Lifecycle Policy
- Mailing List-Announcements
- Mailing List-Forum/Questions

Earthquakes

Hazards

Data

earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php

```
type: "FeatureCollection",
metadata: {
    generated: Long Integer,
    url: String,
    title: String,
    api: String,
    count: Integer,
    status: Integer
},
bbox: [
    minimum_longitude,
    minimum_latitude,
    minimum_depth,
    maximum_longitude,
    maximum_latitude,
    maximum_depth
],
features: [
{
    type: "Feature",
    properties: {
        mag: Decimal,
        place: String,
        time: Long Integer,
        updated: Long Integer,
        title: String
    }
}
]
```

Download this geojson file into the folder/directory you are using for all of these examples. Do not change the filename or extension

Past 7 Days
Updated every 5 minutes.

- [M2.5+ Earthquakes](#)
- [M1.0+ Earthquakes](#)
- [All Earthquakes](#)

Past 30 Days
Updated every 15 minutes.

- [Significant Earthquakes](#)
- [M4.5+ Earthquakes](#)
- [M2.5+ Earthquakes](#)
- [M1.0+ Earthquakes](#)
- [All Earthquakes](#)

[All Earthquakes](#)

If you downloaded the data correctly – the link <http://localhost:8080/geo1.html> should look like this

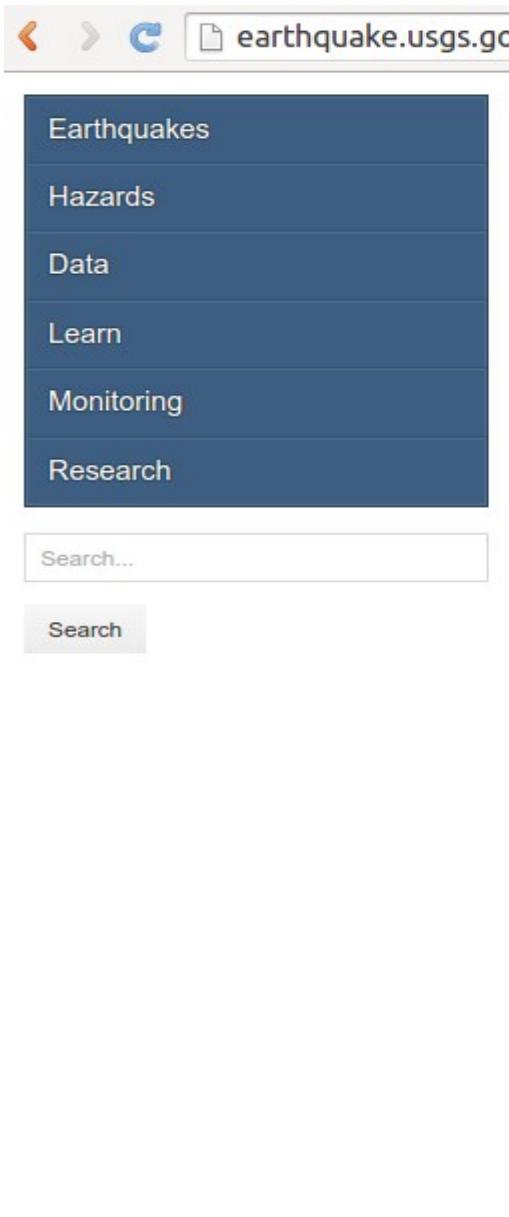
Connecting Leaflet to a Local GeoJSON file for Spatial Data



Unfortunately there is one major problem with the map we have created

- If you look at the map – it is just a bunch of pins or markers
- YES – We state on the page that it is a USGS dataset
- We need to make those pins or markers 'clickable'
- We need to display some information about each pin.
- **We need to revisit the specification for the GeoJSON file on the USGS Website first.**

Here is the specification of the PROPERTIES of the GeoJSON data



```
features: [
  {
    type: "Feature",
    properties: {
      mag: Decimal,
      place: String,
      time: Long Integer,
      updated: Long Integer,
      tz: Integer,
      url: String,
      detail: String,
      felt: Integer,
      cdi: Decimal,
      mmi: Decimal,
      alert: String,
      status: String,
      tsunami: Integer,
      sig: Integer,
      net: String,
      code: String,
      ids: String,
      sources: String,
      types: String,
      nst: Integer,
      dmin: Decimal,
      rms: Decimal,
      gap: Decimal,
      magType: String,
      type: String
    },
  }
],
```

There are three very useful properties

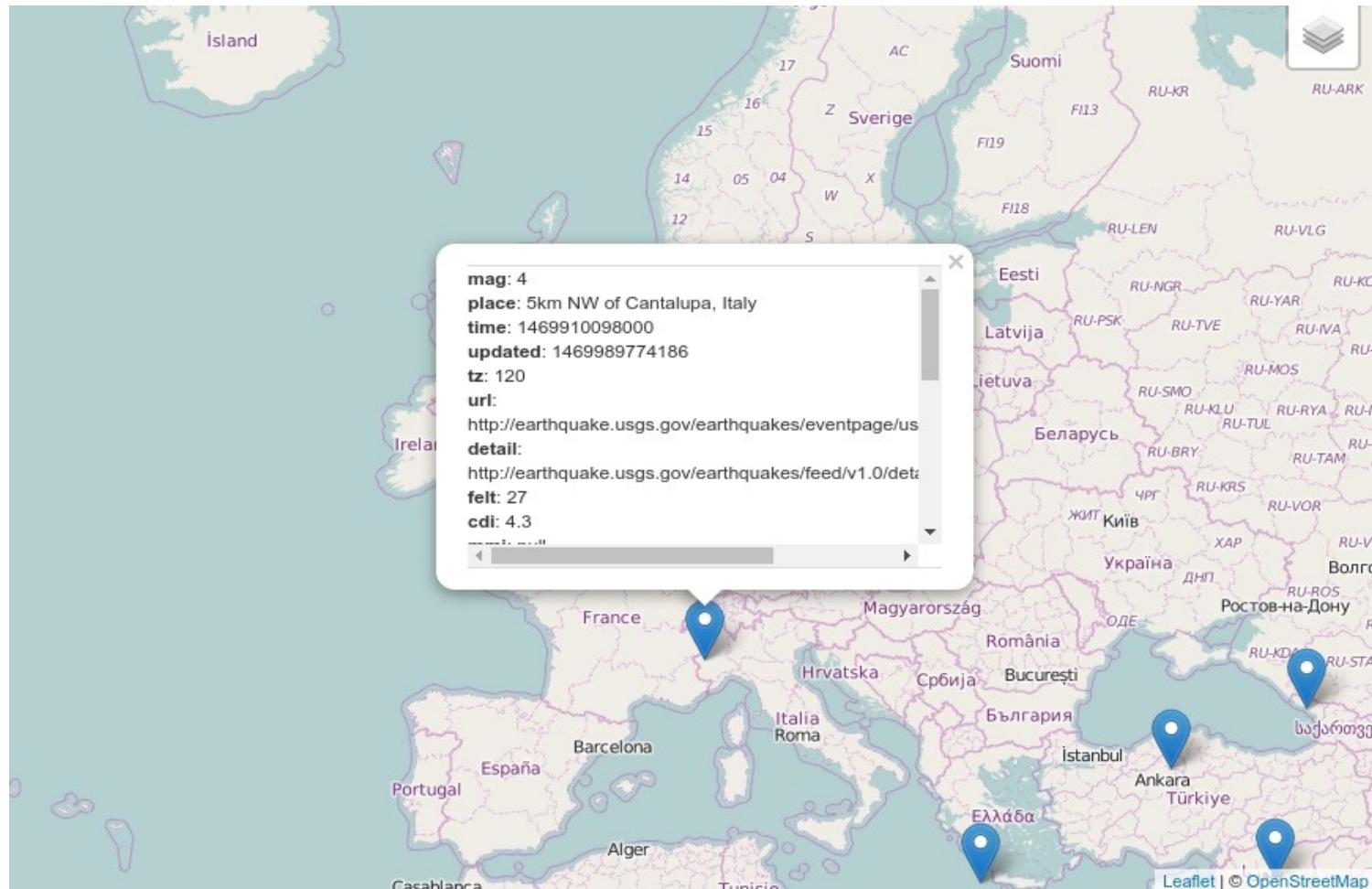
They are:

- = place
- = type
- = mag

Let's try to display these as a popup with each pin or marker on our map

Let's open up geo2.html in your text editor and look at it in the browser

- REMEMBER to view it at
<http://localhost:8080/geo2.html>



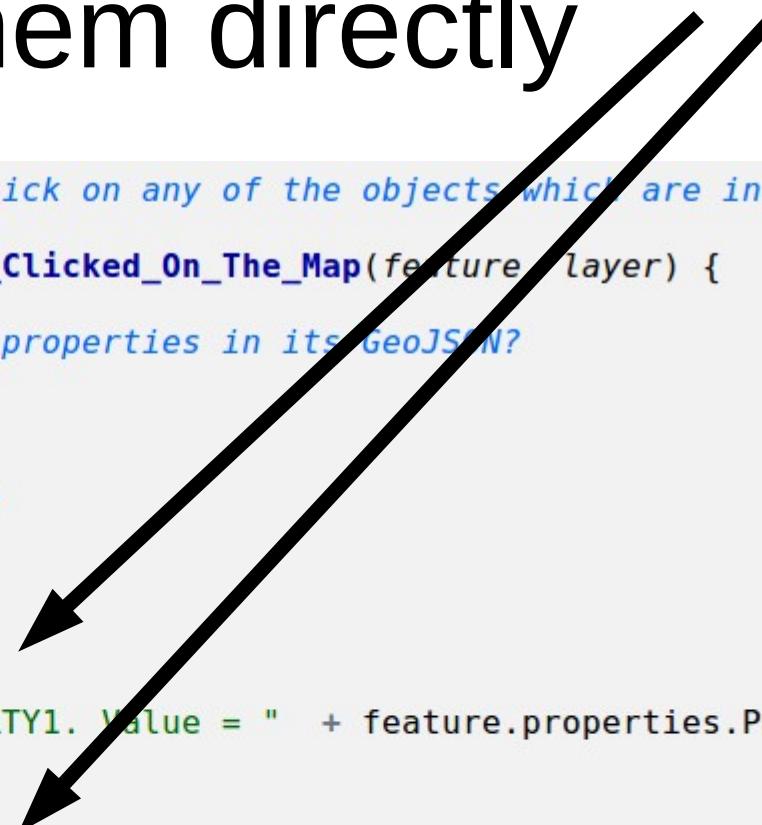
It's very important for you to understand the link between the Javascript in Leaflet and the PROPERTIES in the GeoJSON

- There is a VERY IMPORTANT new piece of code in the Javascript. We will REUSE this for all of our GeoJSON

```
114 // This is where Leaflet will go if you click on any of the objects which are in the
115 // GeoJSON file
116 function action_To_Perform_When_Marker_Is_Clicked_On_The_Map(feature, layer) {
117
118     // does this clicked feature have any properties in its GeoJSON?
119     if (feature.properties)
120     {
121         // the text for the Popup
122         // Each feature in the GeoJSON dataset has a Feature Object.
123         // This object has a list of properties (an array). We get
124         // Javascript to map these to a string. We put some HTML formatting in the string
125         // This is then bound to the popup.
126         // There is the option to access specific attributes (properties) by name
127         // For example: feature.properties.place for the place attribute
128
129         layer.bindPopup(
130             Object.keys(feature.properties).map(
131                 function(k) {
132                     return "<b>" + k + "</b>" + ":" + feature.properties[k];
133                 }
134             ).join("<br />"),
135
136             {
137                 maxHeight: 200
138             });
139
140     } // end if
141 } // end of function
142
```

If we know the PROPERTIES in our GeoJSON very well then we can access them directly

```
114 // This is where Leaflet will go if you click on any of the objects which are in the  
115 // GeoJSON file  
116 function action_To_Perform_When_Marker_Is_Clicked_On_The_Map(feature, layer) {  
117  
118     // does this clicked feature have any properties in its GeoJSON?  
119     if (feature.properties)  
120     {  
121         // the text for the Popup  
122         // We can pick specific properties  
123  
124         var PopUpText=[];  
125  
126         if (feature.properties.PROPERTY1)  
127         {  
128             PopUpText.push("We have PROPERTY1. Value = " + feature.properties.PROPERTY1);  
129         }  
130  
131         if (feature.properties.PROPERTYX)  
132         {  
133             PopUpText.push("We have PROPERTYX. Value = " + feature.properties.PROPERTYX);  
134         }  
135  
136         layer.bindPopup(PopUpText.join());  
137  
138     } // end if  
139 } // end of function  
140
```



Attribute information on Geographical Objects in OpenStreetMap

Tagging in OpenStreetMap

- Tagging is one of the most important aspects of OpenStreetMap
- It also provokes the most debate, conversation, chat, discussion, disagreement, etc
- Tagging in OpenStreetMap is very flexible
- It is very different to the type of tagging (or ontology) that one would find in databases such as those used by National Mapping Agencies

The “Map Features” Page – one of the cornerstones of OSM

http://wiki.openstreetmap.org/wiki/Map_Features



Main Page
The map
Map Features
Contributors
Help
Blogs
Shop
Donations
Recent changes

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information
Cite this page

A あ English Create account Log in

Page Discussion Read View source View history Search

We hit our fundraising target. Thank you! (You can still donate)

Map Features

Available languages — Map Features

- azərbaycanca • Bahasa Indonesia • bosanski • català • čeština • dansk • Deutsch • eesti • English • español • français • hrvatski • íslenska • italiano • latviešu • lietuvių • magyar • Nederlands • norsk bokmål • polski • português • português do Brasil • română • shqip • slovenčina • slovenščina • suomi • svenska • Tiếng Việt • Türkçe • српски / srpski • български • македонски • русский • українська • ქართული • தமிழ் • 한국어 • 日本語 • 中文 (简体) • 中文 (繁體) • فارسی • العربية • עברית

Other languages — Help us translate this wiki

- Afrikaans • Alemannisch • asturianu • Bahasa Melayu • Bân-lâm-gú • Basa Jawa • Baso Minangkabau • brezhoneg • corsu • Esperanto • euskara • Frysk • Gaeilge • Gàidhlig • galego • Hausa • Igbo • interlingua • Interlingue • isiXhosa • isiZulu • Kiswahili • Kreyòl ayisyen • Kréyòl gwadloupéyen • Kurdî • Lëtzebuergesch • Malagasy • Malti • Nedersaksies • norsk nynorsk • occitan • Oromoo • o'zbekcha • Plattdüütsch • Soomaaliga • Vahcuengh • Wolof • Yorùbá • Zazaki • беларуская • қазақша • монгол • тоҷикӣ • Ελληνικά • Հայերեն • ନେପାଳୀ • ମରାଠୀ • ହିନ୍ଦୀ • ଅଞ୍ଚିତା • ଗାଁତା • ପଞ୍ଜାਬୀ • ଗୁଜରାତୀ • ଓଡ଼ିଆ • ଶଲଗୁ • କଣ୍ଠଜ୍ଞ • ମଧ୍ୟାଭ୍ରାତା • ପିହାଳ • ଔଯ୍ • ଭିନ୍ଦିଭାଷା • ଲାବ • ଗାନ୍ଧାରୀ • ଅମ୍ବାରୀ

OpenStreetMap represents physical **features** on the ground (e.g., roads or buildings) using **tags** attached to its basic data structures (its **nodes**, **ways**, and **relations**). Each tag describes a geographic attribute of the feature being shown by that specific node, way or relation.

OpenStreetMap's free tagging system allows the map to include an unlimited number of attributes describing each feature. The community agrees on certain key and value combinations for the most commonly used tags, which act as informal standards. However, users can create new tags to improve the style of the map or to support analyses that rely on previously unmapped attributes of the features. Short descriptions of tags that relate to particular topics or interests can be found using the [feature pages](#).

Most features can be described using only a small number of tags, such as a path with a classification tag such as `highway=footway`, and perhaps also a name using `name=*`. But, since this is a worldwide, inclusive map, there can be many different feature types in OpenStreetMap, almost all of them

Tags can be considered as the attributes of an object

- An object can have at minimum 1 attribute (tag)
– and there is no upper limit.
- There are no strict rules on how many attributes any object should have
- This flexibility means that you must use your own judgement in regards to how many attributes you provide

Let's look at the OSM tag for restaurants



Main Page
The map
Map Features
Contributors
Help
Blogs
Shop
Donations
Recent changes

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information
Cite this page

We hit our fundraising target. Thank you! (You can still donate [↗](#))

Tag:amenity=restaurant

Available languages — Tag:amenity=restaurant [Help](#) [show](#)

• Deutsch • English • español • français • italiano • português do Brasil • русский • 日本語
[Other languages — Help us translate this wiki](#)

amenity=restaurant is for a generally formal place with sit-down facilities selling full meals served by waiters and often licensed (where allowed) to sell alcoholic drinks.

Contents [hide]

- 1 How to Map
- 2 Examples
- 3 Rendering
- 4 See Also



Description
Is for a generally formal place with sit-down facilities selling full meals served by waiters and often licensed (where allowed) to sell alcoholic drinks.

How to Map

Add a node at the centre of the building and add **amenity=restaurant** to it. You can name it with **name=***. If the whole building is used for this feature and its footprint is present in OSM, you can apply the tags on the area if you prefer.

Optionally, you can further classify it using the **cuisine=*** tag with a value suitable for your nation's environment

We use the amenity = restaurant tag and others where applicable

Amenity = restaurant

Name = Peter's Italian Place

Cuisine = Italian

Smoking = no

internet_access = wlan

In OSM we try to supply an appropriate number of tags to an object/feature

- We try to use the tags to supply as much accurate and timely attribute information about the object/feature as we can
- **ALWAYS USE MAP FEATURES FOR GUIDANCE WHEN USING TAGS**
- In the case of many tags there are suggested accompanying tags you should use

Example: Amenity = School

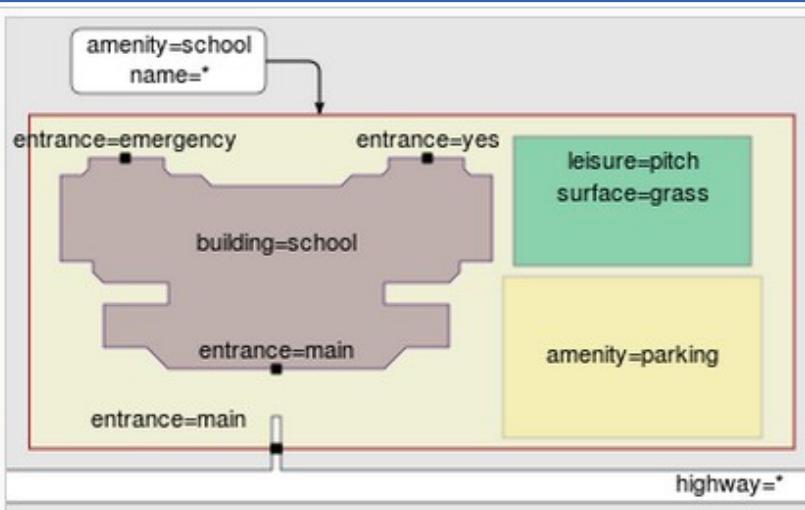
How to map

Mark the boundary of the school using an [Area](#) or place a [Node](#) in the middle of the site if you are in a hurry (or don't have access to information about the boundary and it is not obvious from aerial imagery).

This area should envelop the full grounds of the school including all the buildings, sports facilities and grounds. For schools with multiple sites the [multipolygon](#) relation can be used. Tag the element with [amenity=school](#) and

[name=*](#) for the name of school. Suggested additional tags include:

- [operator=*](#) - Name of operator, often the local education authority.
- [addr=*](#) - Address details
- [capacity=*](#), for the number of pupils taught at the school
- [isced:level=*](#), for the educational level (proposed tags)
- [fee=yes](#) if the school makes a direct charge for core services.
- [religion=*](#), if the school is associated with a particular religion (also [denomination=*](#))
- [wikipedia=*](#), for a link to a Wikipedia article about the school
- [ref=*](#), if schools are numbered
- [contact=*](#), for contact details including the schools website, email and phone number



School buildings should be tagged with [building=school](#) where they are used generally, and optional also with details of use with appropriate additional tags, for example [amenity=swimming_pool](#).

Consider adding [entrance=*](#) and if appropriate [barrier=gate](#) to the points where roads, paths or other access routes into the school grounds cross the boundary of the school (which is tagged with [amenity=school](#)).



Description

Institution designed for learning under the supervision of teachers.

Group: Education features

Used on these elements



Useful combination

- [name=*](#)
- [operator=*](#)
- [isced:level=*](#)

Status: Approved

[taginfo](#) [More...]

<input type="radio"/>	334 305	6.60 %
<input checked="" type="checkbox"/>	315 360	10.11 %
<input checked="" type="checkbox"/>	8 017	21.19 %

Tools for this tag

- [taginfo](#), [fr](#), [uk](#), [ie](#), [us](#)
- [overpass-turbo](#)

Choosing the correct tags is not an easy task

- Choosing the correct tags requires you to think about the object/feature you are mapping
- As you get more experienced with mapping you will become more familiar with tagging
- **Remember** – you should always use tags which you can **VERIFY**
- **Remember** – you should use the Map Features guidance on the popular combinations of tags

**Let's get some OSM GeoJSON
data from OVERPASS**

Suppose we are feeling crazy and decide to drive our car into central London

http://overpass-turbo.eu/

```
1 node  
2   [amenity=parking]  
3   (around:3000,51.507343,-0.127656);  
4 out body;
```

Run Share Export Wizard Save Load Settings Help overpass turbo Map Data

The screenshot shows the Overpass Turbo web application. At the top, there's a navigation bar with buttons for Run, Share, Export, Wizard, Save, Load, Settings, Help, and a Flattr this! button. Below the navigation bar is a search bar containing the query text. To the right of the search bar are tabs for Map and Data. The main area displays a map of central London, specifically the City of Westminster and parts of the City of London. Numerous parking locations are marked with blue circles of varying sizes. Labels for various neighborhoods like Mayfair, Covent Garden, St. Giles, Bloomsbury, Clerkenwell, Finsbury, Saint Luke's, and Islington are visible. A legend on the left side of the map provides icons for nodes, ways, and relations. A callout box in the bottom right corner contains the text: "Export and SAVE as GeoJSON in the folder called geojson – remember to give the file a sensible name – no spaces". At the bottom of the interface, there are status bars for loaded and displayed data counts.

Export and SAVE as GeoJSON in the folder called geojson – remember to give the file a sensible name – no spaces

Loaded – nodes: 84, ways: 0, relations: 0
Displayed – pois: 84, lines: 0, polygons: 0

When downloaded and in correct folder – let's open it in a text editor

```
},
{
  "type": "Feature",
  "id": "node/368044118",
  "properties": {
    "@id": "node/368044118",
    "amenity": "parking",
    "capacity": "233",
    "capacity:disabled": "5",
    "covered": "yes",
    "fee": "yes",
    "name": "Baynard House",
    "operator": "Corporation of London",
    "website":
    "http://www.cityoflondon.gov.uk/services/transport-and-streets/parking/where-to-park/car-parks/Pages/Baynar d-House-car-park.aspx"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      -0.1000953,
      51.5118468
    ]
  }
}
```

**Look CAREFULLY at the properties of each Feature/Object/Node
Notice – each node has a different number of tags**

Let's choose three: **name** **operator** **capacity**

Check Map Features if you need more information

Open the file geo3.html in your text editor AND your browser

- You'll need to check out one very important detail
- You will need to change the filename of the geojson file to the name you used
- You will then save the file and refresh the page in your browser
- Otherwise you won't see any car parks!!!

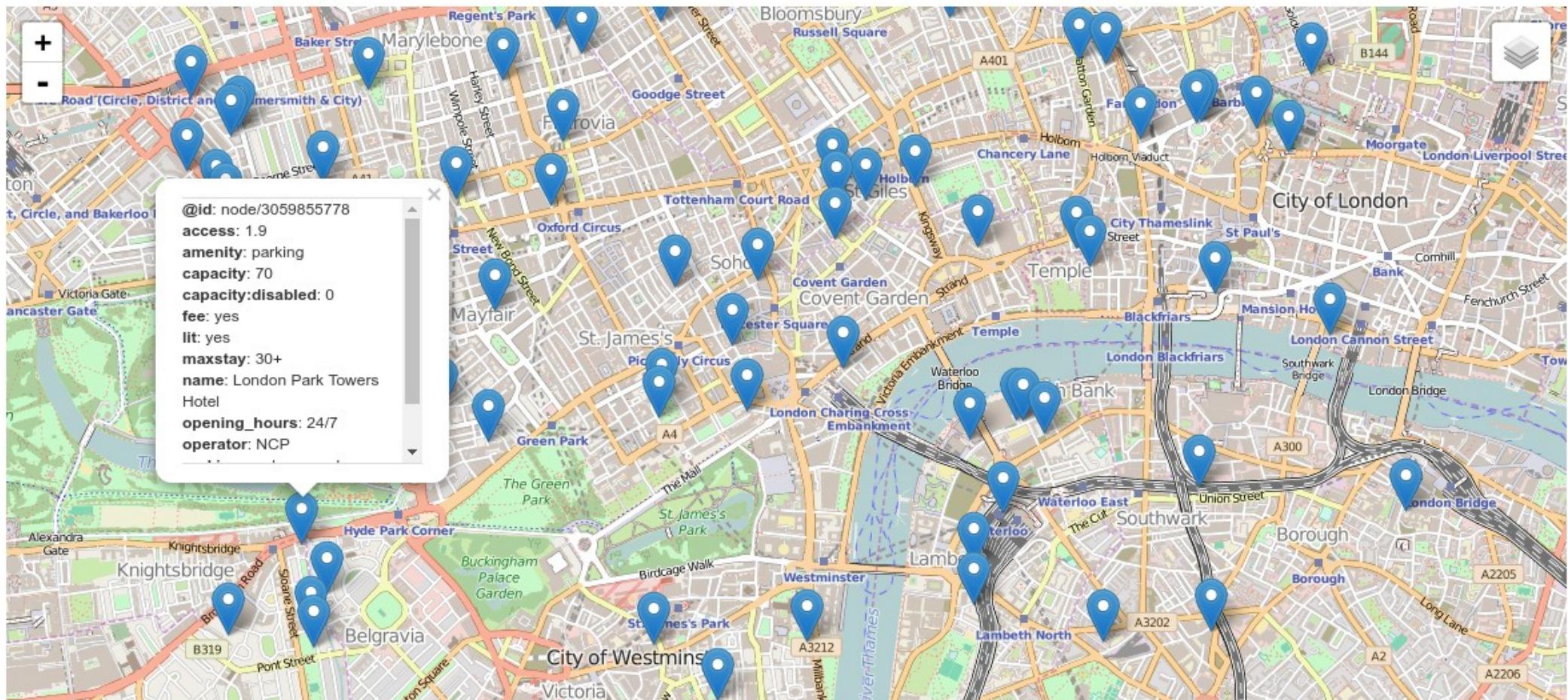
This is what <http://localhost:8080/geo3.html> should look like!

Connecting Leaflet to a Local GeoJSON file for Spatial Data - Using OpenStreetMap as a Data Source



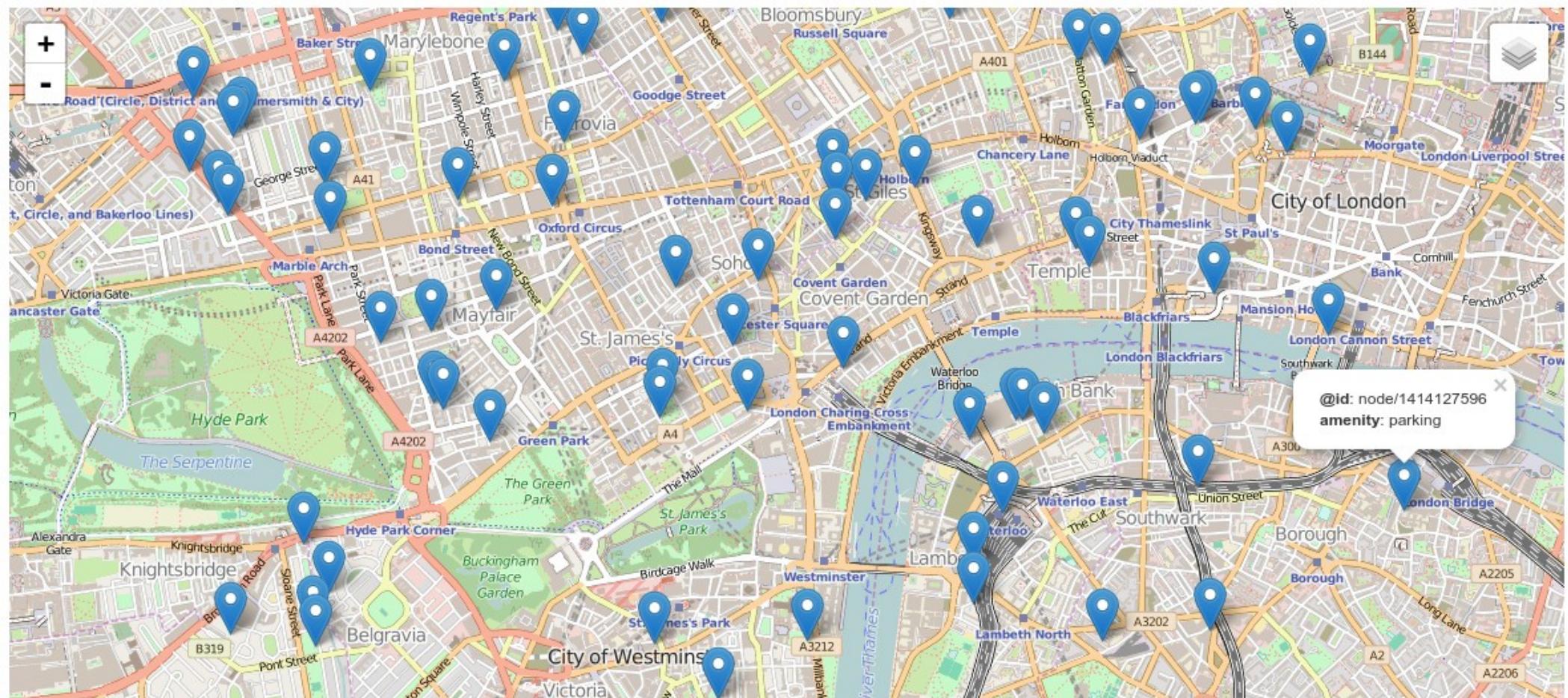
Notice – that this parking node has LOTS of attributes/properties

Connecting Leaflet to a Local GeoJSON file for Spatial Data - Using OpenStreetMap as a Data Source



Notice – that this parking node has only TWO attributes/properties

Connecting Leaflet to a Local GeoJSON file for Spatial Data - Using OpenStreetMap as a Data Source



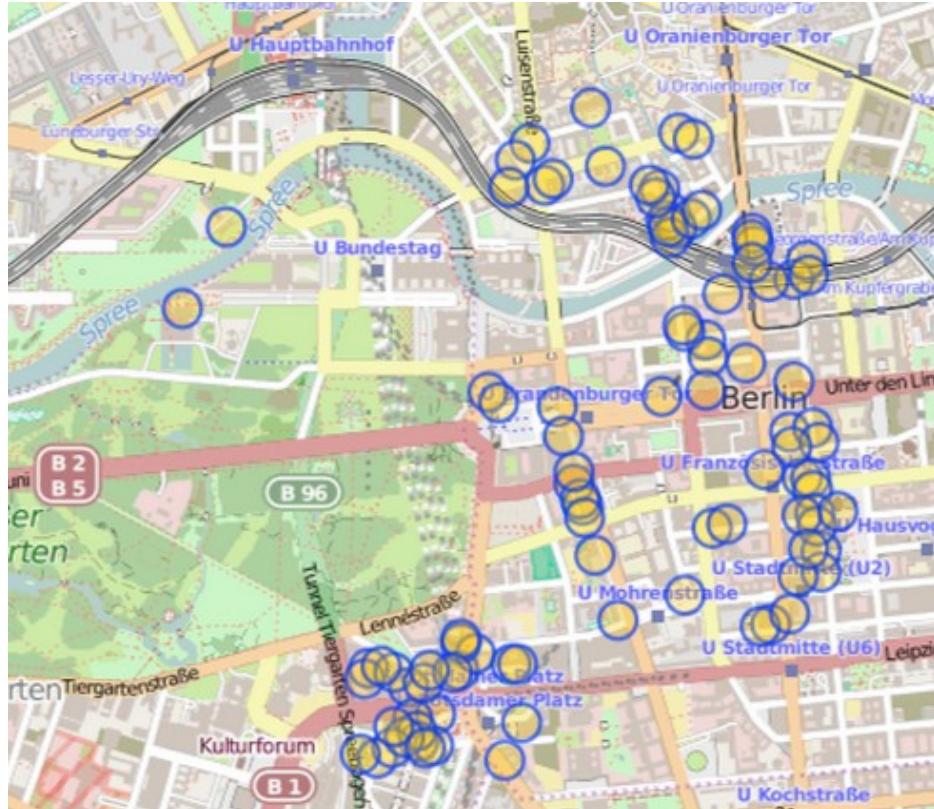
We can easily query NODES in Overpass Turbo

```
node
[amenity=restaurant]
(around:3000,51.507343,-0.127656);
out body;
```

```
node
[highway=bus_stop]
(around:2000,51.507343,-0.127656);
out body;
```

GeoJSON Exercise on your own!

Exercise: Map of places to eat in Berlin, Germany



- Using the Overpass API generate GeoJSON representing all amenity=restaurant around 1000m radius of the Brandenburg Gate in Berlin
- Use name, cuisine, opening_hours, wheelchair as properties
- Center and zoom the map appropriately
- You can use geo3.html as your starting point

You can easily make your own
GeoJSON

Go to GeoJSON.io

The screenshot shows a map of the Maynooth University campus area in Ireland. The map includes labels for 'North Campus', 'Arts Building', 'Science Building', 'Post Primary School Maynooth', 'South Campus', 'Maynooth Castle', 'Royal Canal Way', 'Carton Ave', 'DUBLIN RD', 'LEINSTER PK', 'MAIN ST', 'PARSON ST', 'KILCOCK RD', 'Laraghbryam', 'THE STEEPLE', 'MOYGLARE RD', 'LARE HALL', 'THE PARK', 'THE WALK', and 'Rivertyreen'. A large dark grey polygon highlights the North Campus area. On the right side of the interface, the JSON code for this polygon is displayed:

```
1 {  
2   "type": "FeatureCollection",  
3   "features": [  
4     {  
5       "type": "Feature",  
6       "properties": {"name": "Maynooth University"},  
7       "geometry": {  
8         "type": "Polygon",  
9         "coordinates": [  
10            [  
11              [  
12                [-6.605830192565918,  
13                  53.387346941404545  
14                ],  
15                [-6.6031694412231445,  
16                  53.387654077940816  
17                ],  
18                [-6.59879207611084,  
19                  53.38668147130748  
20                ],  
21                [-6.595573425292969,  
22                  53.385606459176216  
23                ],  
24                [-6.595573425292969,  
25                  53.385606459176216  
26                ]  
27              ]  
1]          ]  
1]        ]  
1]      ]  
1]    ]  
1]  ]  
1] }  
1] }
```

You can make really nice GeoJSON datasets here

- Remember PROPERTIES – the name of the property and the value MUST always have double quotes
- For example “population”: “200,000”
- You can have a mixture of polylines, polygons and points

You can easily SAVE the data you create as GeoJSON

The screenshot shows a map of Maynooth University campus and its surroundings. A polygon is drawn over the university campus buildings. The 'Save' menu is open, with 'GeoJSON' selected. The right panel shows the generated GeoJSON code.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {"name": "Maynooth University"},
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              [
                [
                  -6.605830192565918,
                  53.387346941404545
                ],
                [
                  -6.6031694412231445,
                  53.387654077940816
                ],
                [
                  -6.59879207611084,
                  53.38668147130748
                ],
                [
                  -6.595573425292969,
                  53.385606459176216
                ],
                [
                  -6.605830192565918,
                  53.387346941404545
                ]
              ]
            ]
          ]
        ]
      }
    }
  ]
}
```

Adding multiple overlay layers

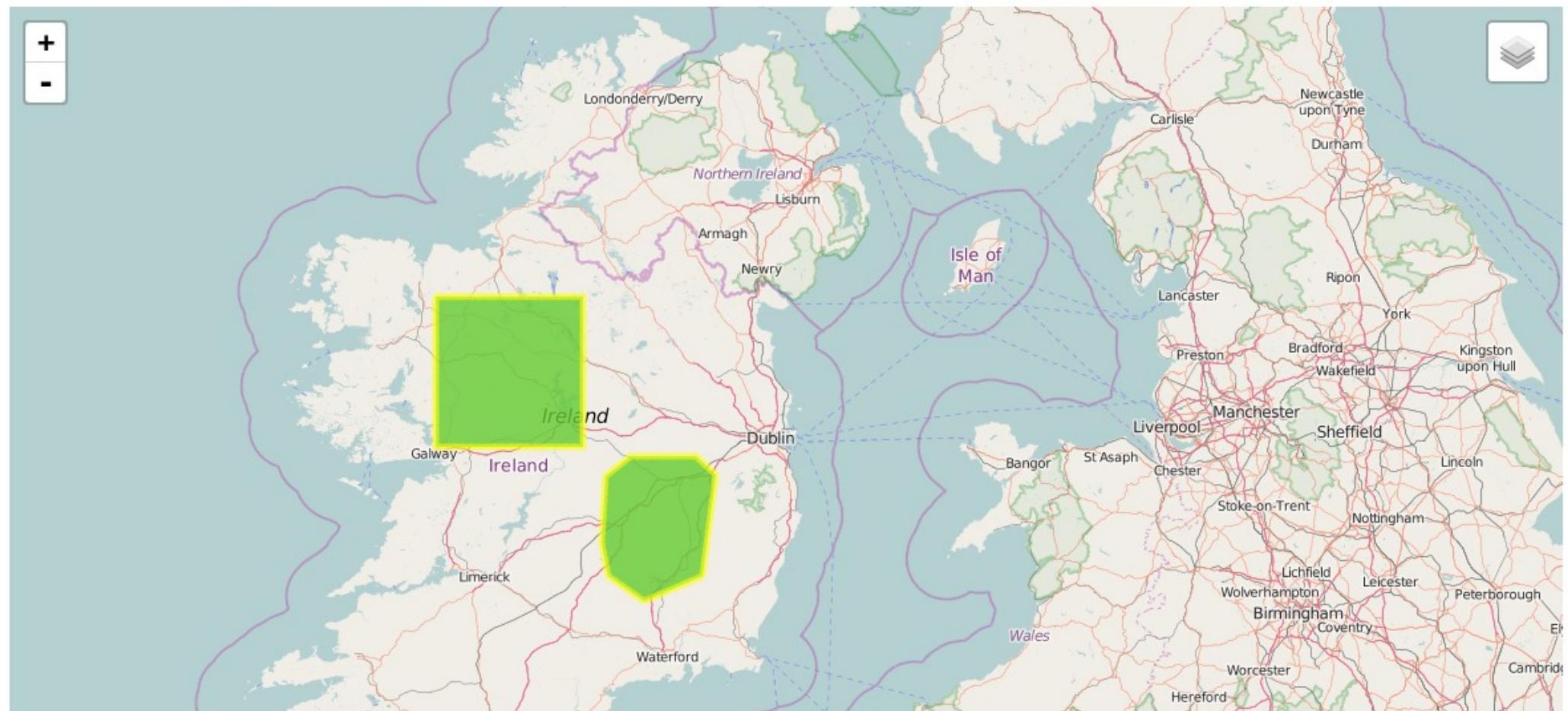
We need to create a new LayerGroup and add to the overlayMaps variable

```
83 <!-- This is where we are going to get Javascript to gather the GeoJSON and make it available-->
84 <!-- so that Leaflet can display it -->
85 <!-- We need to make a special LAYER for our GeoJson -->
86 <!-- We are going to make a group for our 'top layers' -->
87
88 var myGeoJSONLayers = L.layerGroup();
89
90 var layerVariableName = L.layerGroup();
91
```

```
102 // Notice that we have made a change here also.  
103 // We have added the myGeoJSONLayers variable so that Leaflet knows that you  
104 // want to include the GeoJSON as a layer on the map  
105 // Notice that we also make a place where we can switch on/off the layer  
106 var overlayMaps = {"London Parking": myGeoJSONLayers,  
107 | "Layer 2" : layerVariableName  
108 };
```

Can we add some STYLE to our GeoJSON layers?

Connecting Leaflet to a Local GeoJSON file for Spatial Data - Adding a colour or style to layers



Let's look at geo4.html

```
93
94 <!-- create a style for the geojson layer -->
95 var myGeoJSONLayerStyle = {
96   "color": "#FFFF00",
97   "weight": 5,
98   "opacity": 0.6,
99   "dashArray": 3,
100  "fillColor": "#66CC33",
101  "fillOpacity": 0.8
102}
103
```

```
// we include our style definition here as part of the Layer Definition

$.getJSON("./map.geojson",
  function(data) {
    // tell Leaflet to go to the function action_To_Perform_When_Marker_Is_Clicked_On_The_Map
    // when someone clicks on one of the objects in the GeoJSON layer
    var geojson = L.geoJson(data, {onEachFeature: action_To_Perform_When_Marker_Is_Clicked_On_The_Map,
      style:myGeoJSONLayerStyle
    });
    <!-- add our new layer to the group of top layers -->
    myGeoJSONLayers.addLayer(geojson);
  }
);
```

Number of data classes: 6 i

Nature of your data: i
 sequential diverging qualitative

Pick a color scheme:
 Multi-hue:
 Single hue:

Only show: i
 colorblind safe
 print friendly
 photocopy safe

Context: i
 roads
 cities
 borders

Background: i
 solid color
 terrain

color transparency

how to use | updates | downloads | credits

COLORBREWER 2.0

color advice for cartography

BuGn class 3
 RGB: 153,216,201
 CMYK: 40,0,15,0
 HEX: #99d8c9

6-class BuGn i

EXPORT i

HEX i

#edf8fb
#ccece6
#99d8c9
#66c2a4
#2ca25f
#006d2c

But can we apply different styles to features within a GeoJSON datasets depending on the value(s) of a given property or attribute?



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search

Choropleth map

From Wikipedia, the free encyclopedia

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. (March 2009) (Learn how and when to remove this template message)

A **choropleth map** (from Greek χώρο ("area/region") + πλήθος ("multitude")) is a **thematic map** in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map, such as population density or per-capita income.

The choropleth map provides an easy way to visualize how a measurement varies across a geographic area or it shows the level of variability within a region.

Contents [hide]

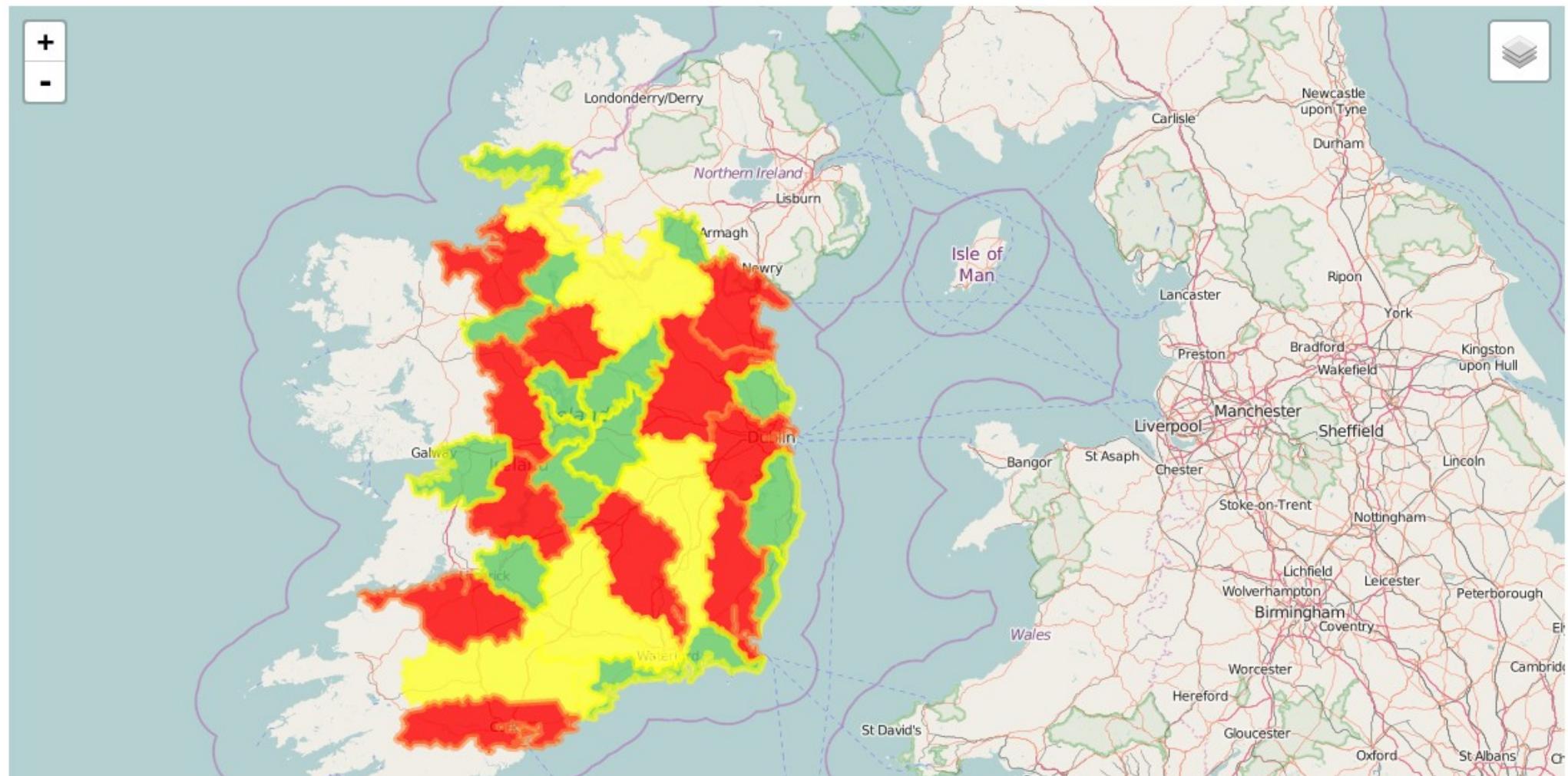
- 1 Overview
- 2 Classification
- 3 Color progression
 - 3.1 Usability
- 4 See also
- 5 References
- 6 External links

A choropleth map of Australia visualizing the fraction of Australians that identified as Anglican at the 2011 census. The map uses a color gradient from pink to dark green to represent different percentages. A legend on the left indicates four categories: >32.97% (dark green), 17.53% (medium green), <7.08% (light yellow-green), and ? (pink). The map shows higher concentrations of Anglicans in the southern and eastern parts of the country, with significant portions of the interior and northern regions colored pink. A scale bar indicates 500 km. An inset map shows the location of Australia within the world map.

A choropleth map that visualizes the fraction of Australians that identified as Anglican at the 2011 census

Let's look at geo5.html

Connecting Leaflet to a Local GeoJSON file for Spatial Data - Adding a colour or style to individual features in a layer depending on properties



We can setup a series of different layer styles for different features

```
95  <!-- create a set of styles for the geojson layer -->
96  var myGeoJSONLayerStyle1 = {
97    "color": "#ffff33",
98    "weight": 5,
99    "opacity": 0.6,
100   "dashArray": 3,
101   "fillColor": "#ffff33",
102   "fillOpacity": 0.8
103 };
104
105 var myGeoJSONLayerStyle2 = {
106   "color": "#fd8d3c",
107   "weight": 5,
108   "opacity": 0.6,
109   "dashArray": 3,
110   "fillColor": "#FF0000",
111   "fillOpacity": 0.8
112 };
113
114 var myGeoJSONLayerStyle3 = {
115   "color": "#FFFF00",
116   "weight": 5,
117   "opacity": 0.6,
118   "dashArray": 3,
119   "fillColor": "#66CC66",
120   "fillOpacity": 0.8
121 };
```

We add a function – to make the decision on which style to apply to a given feature based on the Area_km2 property

```
121
122 // This function checks every feature in the GeoJSON and looks at the Area_km2 property (careful - it is case
123 sensitive)
124 function specific_style_for_geojsonfeatures(feature) {
125     if (feature.properties.Area_km2) {
126         if (feature.properties.Area_km2 > 3000.00)
127             return myGeoJSONLayerStyle1;
128         else if ((feature.properties.Area_km2 > 1500.00) && (feature.properties.Area_km2 <= 3000.00))
129             return myGeoJSONLayerStyle2;
130         else
131             return myGeoJSONLayerStyle3; // this is when we don't know the property value
132         // or it is different to the ones above
133     }
134 }
```

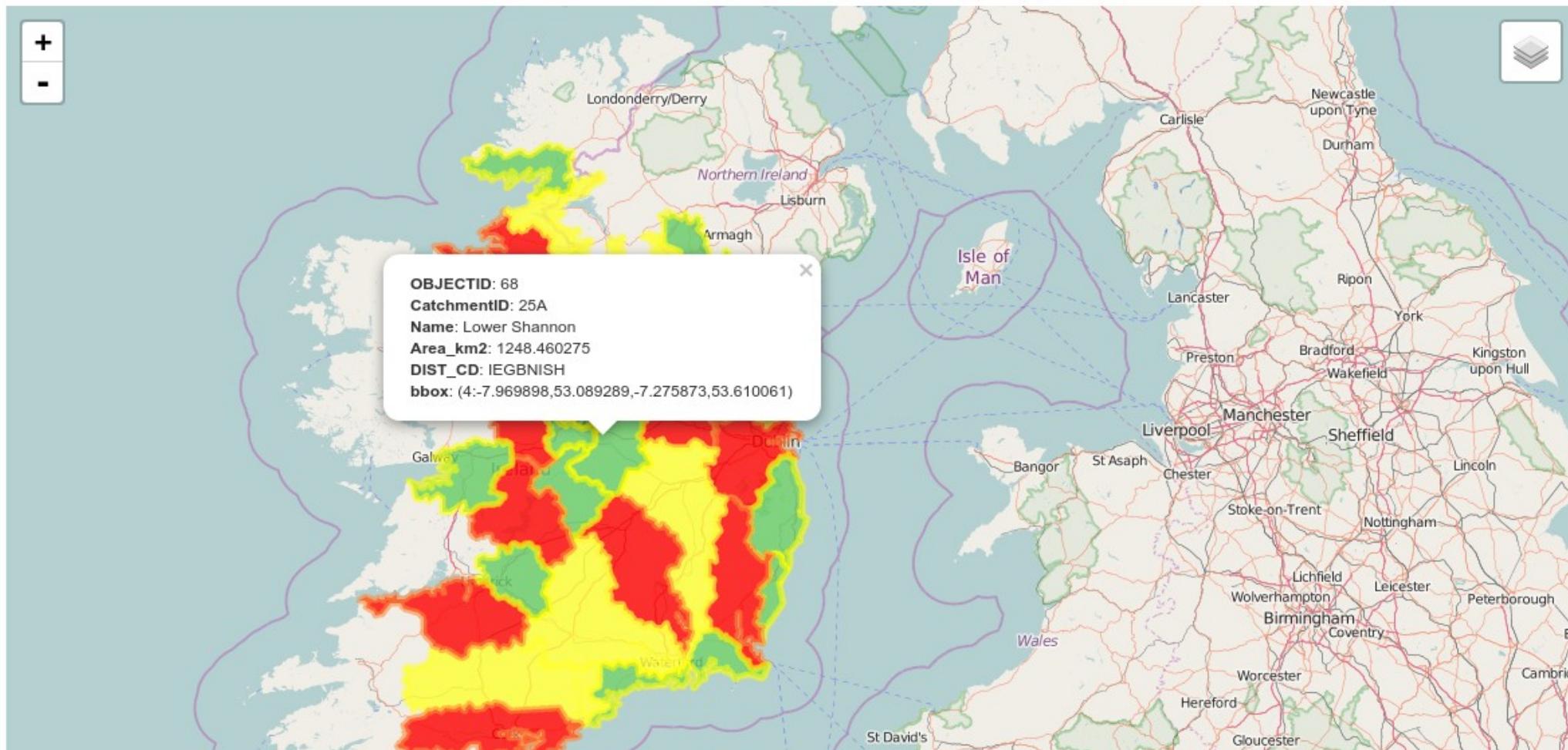
Then simply tell Leaflet that you want to apply this function to handle the style for the GeoJSON layer

```
135 // we include our style definition here as part of the Layer Definition
136
137 $.getJSON("./catchments2016.geojson",
138   function(data) {
139     // tell Leaflet to go to the function action_To_Perform_When_Marker_Is_Clicked_On_The_Map
140     // when someone clicks on one of the objects in the GeoJSON layer
141     var geojson = L.geoJson(data,{onEachFeature: action_To_Perform_When_Marker_Is_Clicked_On_The_Map,
142       style:specific_style_for_geojsonfeatures
143     });
144     // add our new layer to the group of top layers -->
145     myGeoJSONLayers.addLayer(geojson);
146
147   });

```

This type of visualisation is very useful for online mapping

Connecting Leaflet to a Local GeoJSON file for Spatial Data - Adding a colour or style to individual features in a layer depending on properties



I'm happy to help with questions at
peter.mooney@nuim.ie

