

# Visualising VGI data with Leaflet

Dr. Peter Mooney



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

**There's no need to be afraid but there  
WILL be computer code in this lecture**



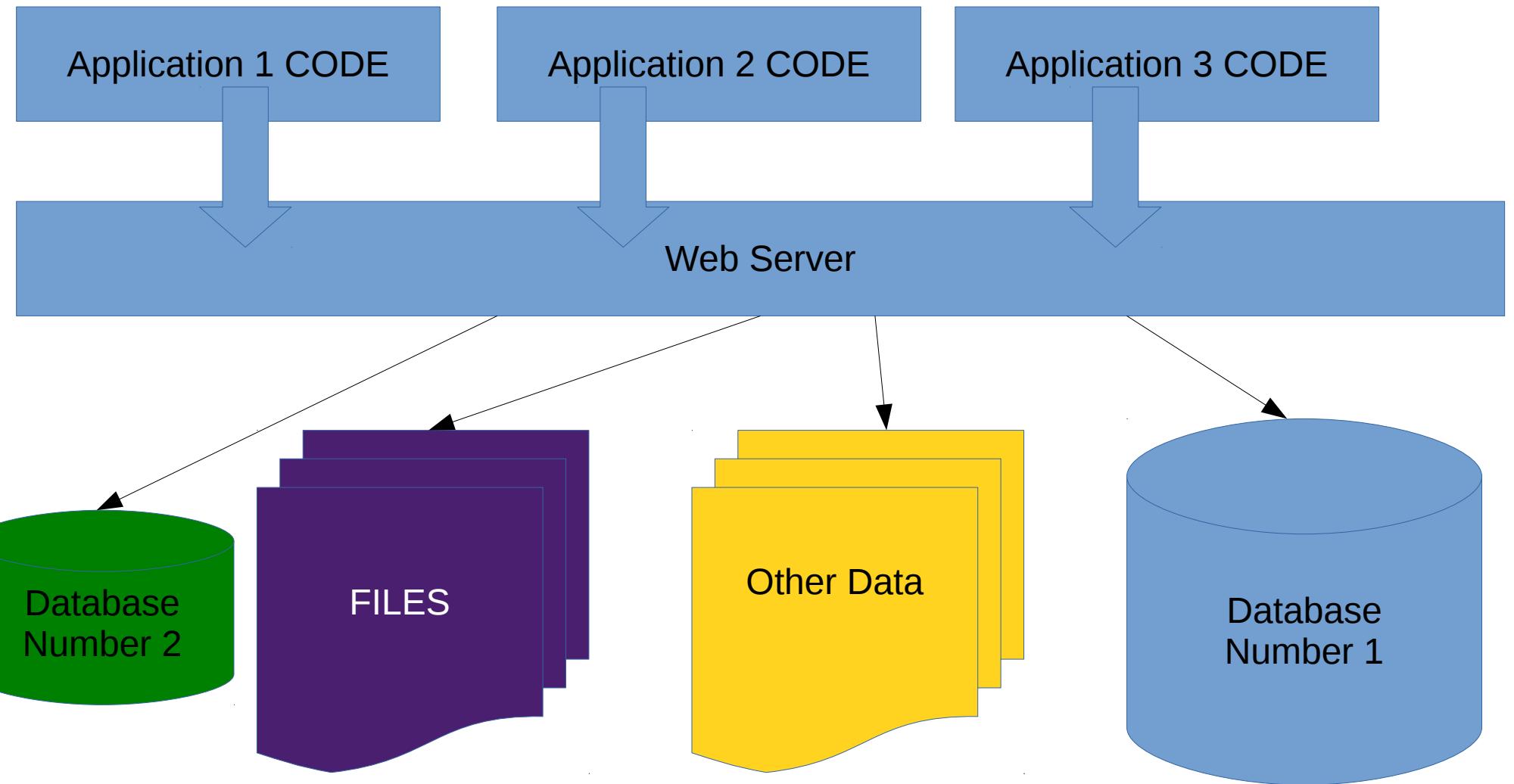
# Aims of this lecture today

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

# **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



**Installing a webserver  
for this lecture**

# 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

N4100 × M Inbox (1,8 × nō How did y × 47595359! × Deleting f × petermoo × Webtext - × Karlsruhe × overpass l × Cesanta S ×

https://cesanta.com/mongoose.shtml

129825 downloads since 2014-03-05

### Download Mongoose Free Edition

- Stable and mature: over 1 million downloads since 2004
- Cross-platform: works on Mac, Windows, UNIX/Linux
- No installation or configuration required: runs in one click
- IP-based ACL (deny access to certain IP addresses)
- File blacklist (hide certain files from serving)
- Resumed download support

-- download for free --

[Windows Executable](#)  
[Windows Mongoose+PHP Package](#)  
[Windows Mongoose+Lua Package](#)  
[MacOS Installer](#)  
[Linux x86\\_64 Executable](#)

-- download for free --

### Download Mongoose Pro Edition

- All features of the Free edition, plus:
- [CGI](#) (ability to run sites written in PHP, Ruby or any other scripting language)
- [Server Side Includes](#) (include one HTML page into another)
- [WebDAV](#) (attach shared directory as a remote drive)
- Pre-packaged, ready-to-go Mongoose + [PHP](#) bundle for Windows (start PHP project in seconds)

Life-time software updates

personal and Commercial use. Distribution not allowed.

[Accept Mongoose EULA](#)

### Download Mongoose Dev Edition

- All features of the Pro edition, plus:
- Ability to view and analyze all incoming and outgoing network packets



Experimental HTTPS security (TLS1.2) support

For personal and Commercial use. Distribution not allowed.

[I accept Mongoose EULA](#)

-- buy for \$8 --

[I accept Mongoose EULA](#)

-- buy for \$15 (with SSL) --

Documentation Licensing

# Mongoose – Windows Users

- Download the EXE file
- Move the EXE file to a new folder anywhere on your computer (*call the folder www or public\_html*)
- 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 – Linux Users (1)

- Download the Linux executable
- Move this file to a new folder within your workspace – call the folder www or public\_html
- You will need to make this file executable
- You will need to type the command  
**sudo chmod ugo+x mongoose-lua-sqlite-ssl-static-x86\_64-5.2** in a terminal window
- Click on the file to run it
- Check in your browser that the URL **http://127.0.0.1:8080** or **http://localhost:8080** work

# 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 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

# Which browser should you use for the lecture today?



Safari



# 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

**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

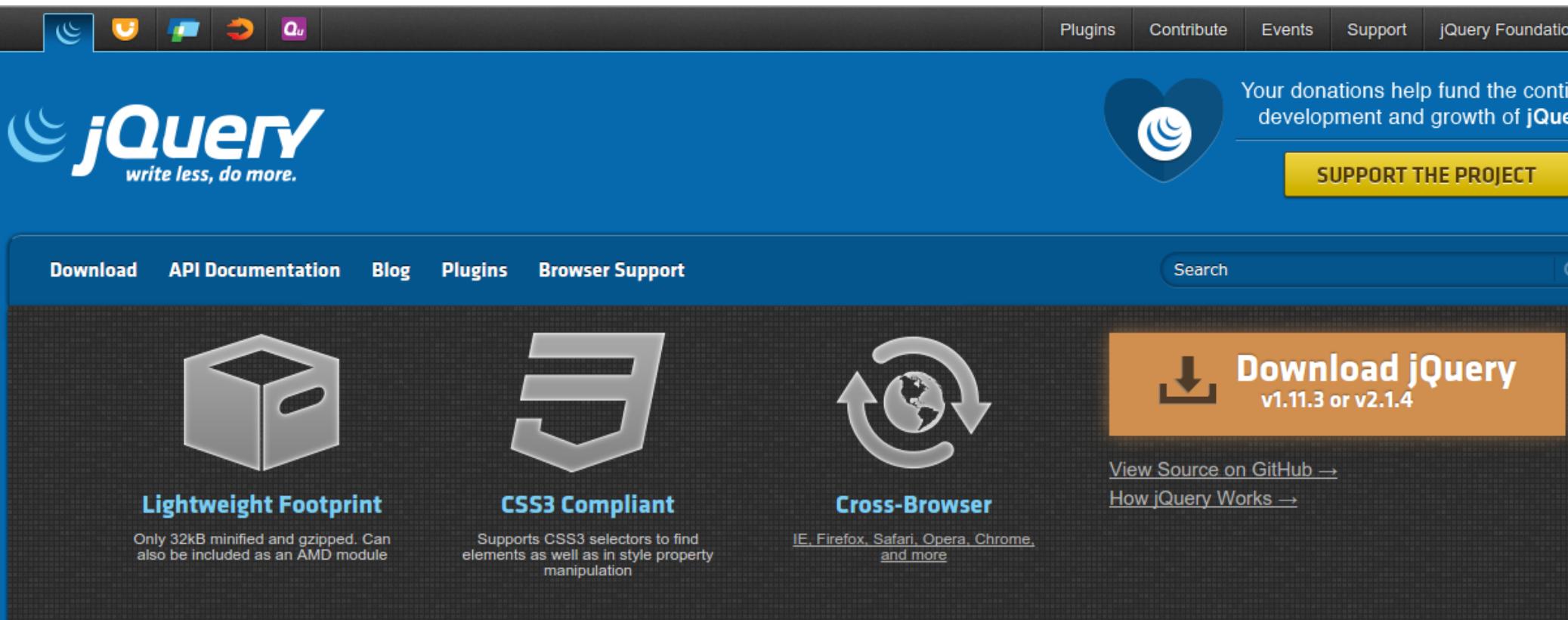
Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

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

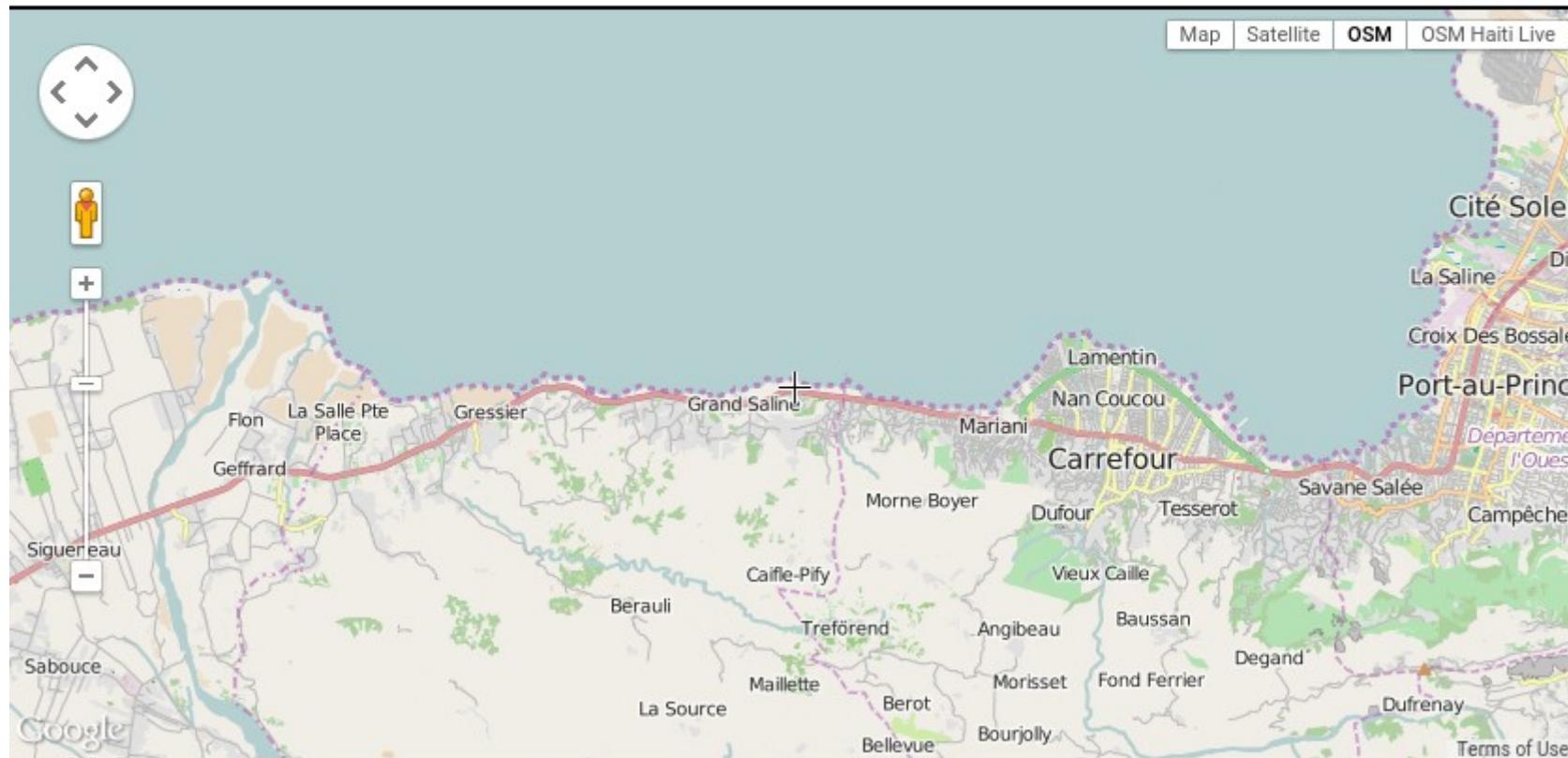
- [jQuery Core API Documentation](#)
- [jQuery Learning Center](#)
- [jQuery Blog](#)
- [Contribute to jQuery](#)

# 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

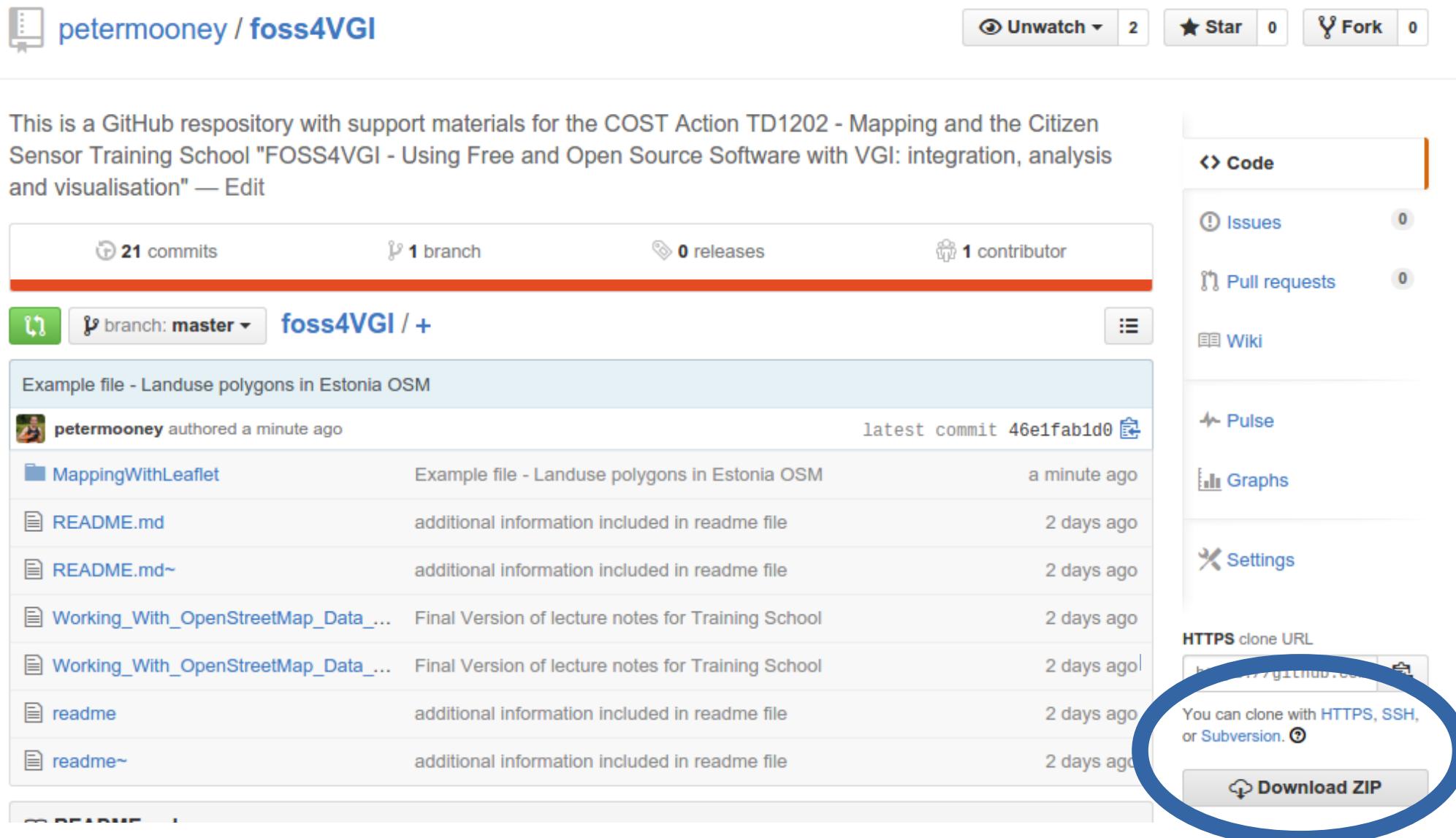
Latitude, Longitude: 45.81291, 9.07210

# 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.
- DO NOT CHANGE the names of any files or folders within this new folder.

# Downloading from GITHUB

<https://github.com/petermooney/foss4VGI>

A screenshot of a GitHub repository page for 'petermooney / foss4VGI'. The page shows basic repository statistics: 21 commits, 1 branch, 0 releases, and 1 contributor. The 'Code' tab is selected. The repository description states: "This is a GitHub respository with support materials for the COST Action TD1202 - Mapping and the Citizen Sensor Training School "FOSS4VGI - Using Free and Open Source Software with VGI: integration, analysis and visualisation" — Edit". A list of files includes 'MappingWithLeaflet', 'README.md', 'Working\_With\_OpenStreetMap\_Data...', 'readme', and 'readme~'. The 'readme' file was updated a minute ago. A blue circle highlights the 'Download ZIP' button at the bottom right of the page.

This is a GitHub respository with support materials for the COST Action TD1202 - Mapping and the Citizen Sensor Training School "FOSS4VGI - Using Free and Open Source Software with VGI: integration, analysis and visualisation" — Edit

21 commits 1 branch 0 releases 1 contributor

branch: master [foss4VGI](#) +

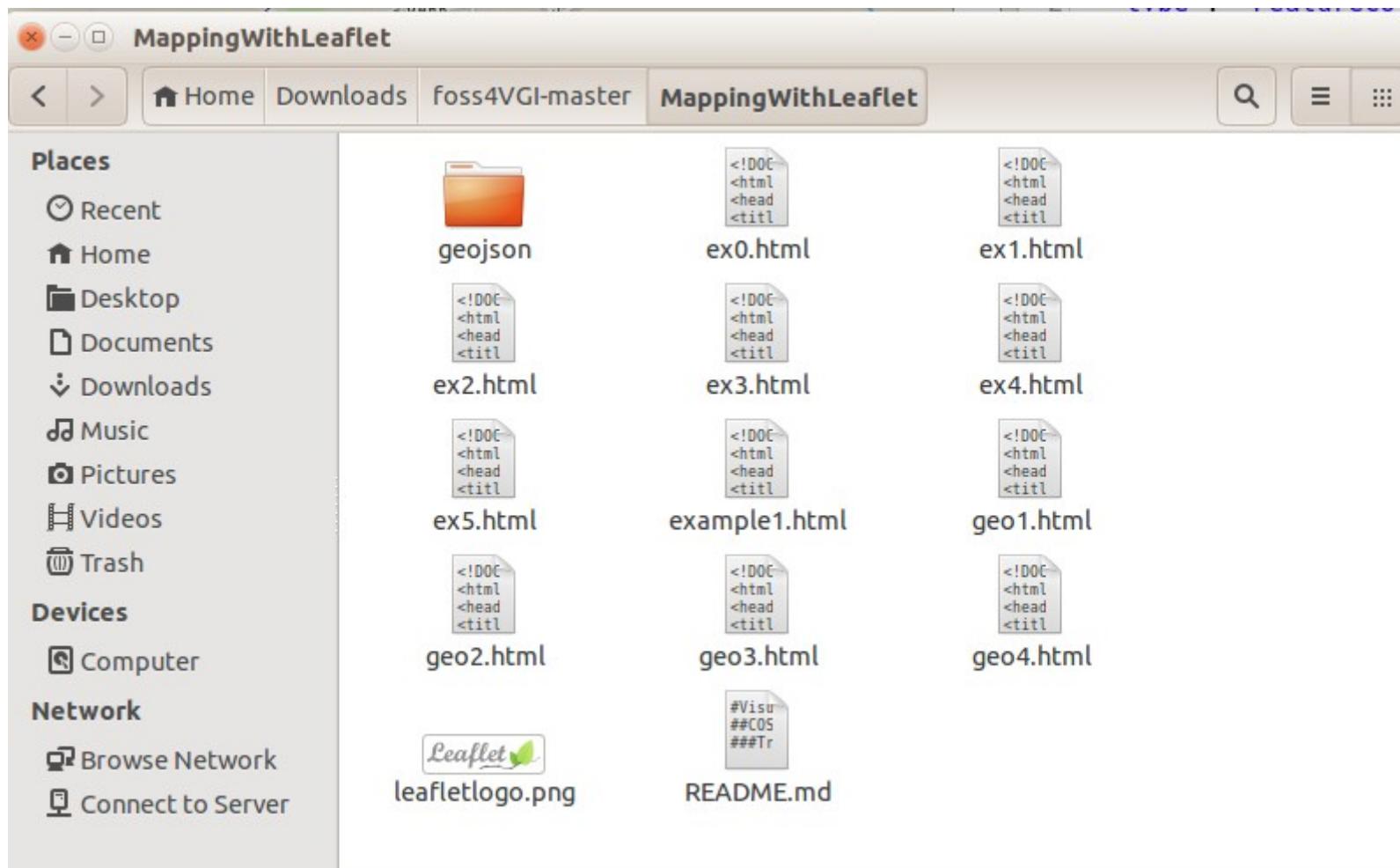
Example file - Landuse polygons in Estonia OSM

File	Description	Last Commit
MappingWithLeaflet	Example file - Landuse polygons in Estonia OSM	a minute ago
README.md	additional information included in readme file	2 days ago
README.md~	additional information included in readme file	2 days ago
Working_With_OpenStreetMap_Data...	Final Version of lecture notes for Training School	2 days ago
Working_With_OpenStreetMap_Data...	Final Version of lecture notes for Training School	2 days ago
readme	additional information included in readme file	2 days ago
readme~	additional information included in readme file	2 days ago

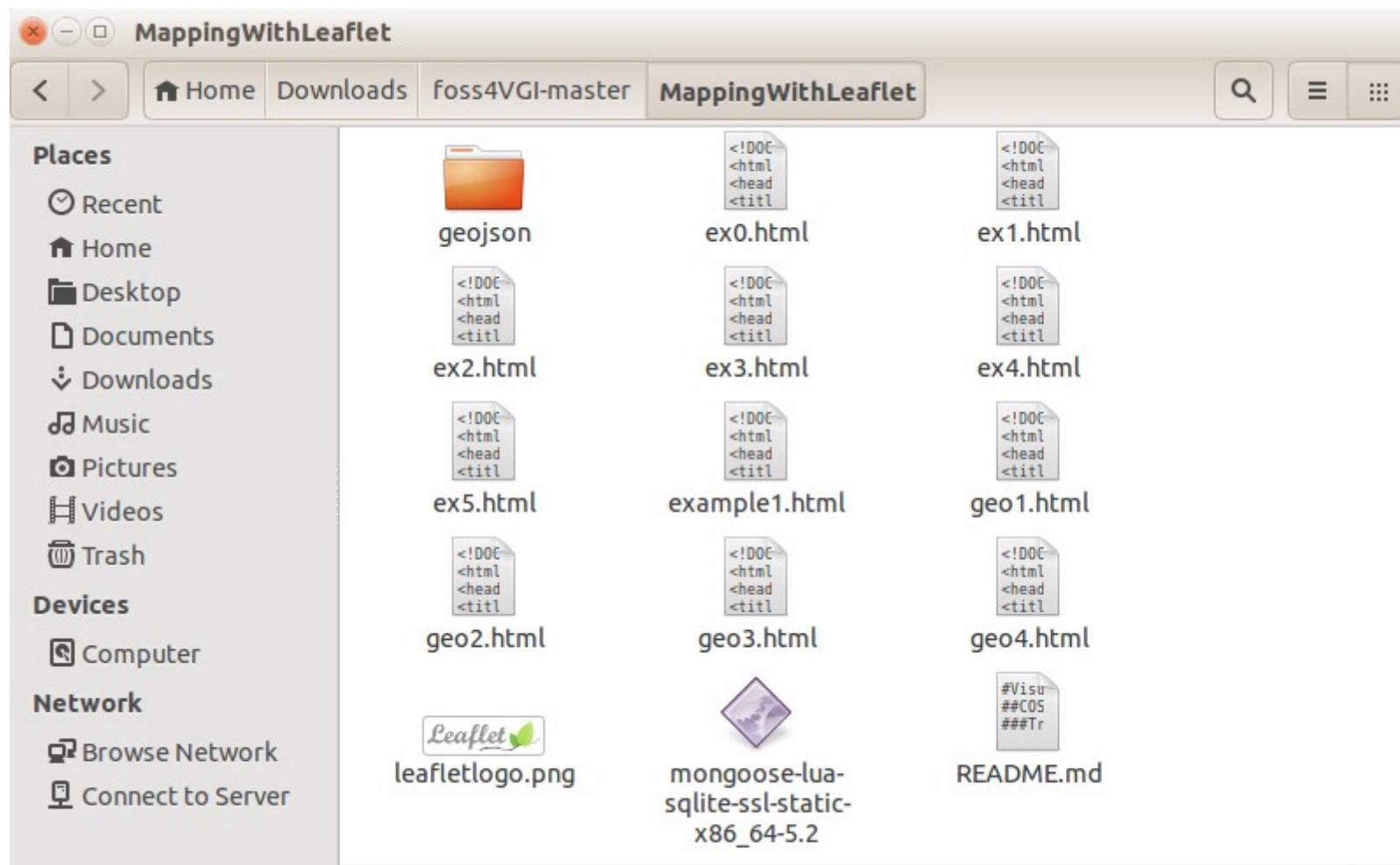
HTTPS clone URL  
You can clone with HTTPS, SSH, or Subversion.

[Download ZIP](#)

# The Unzipped Files in a Folder

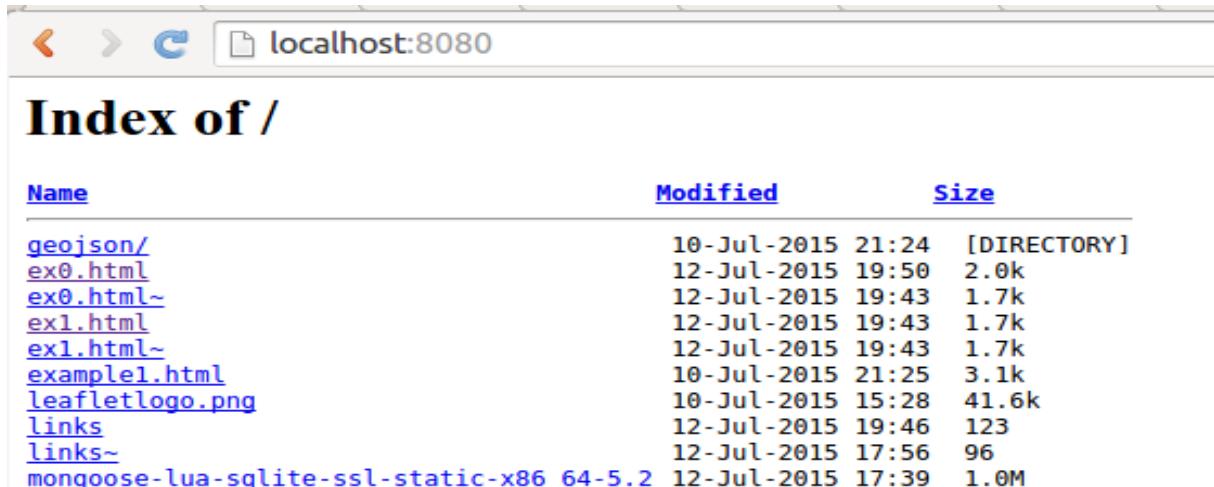


# And look – there is the Mongoose Executable Included now!



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

# 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]});

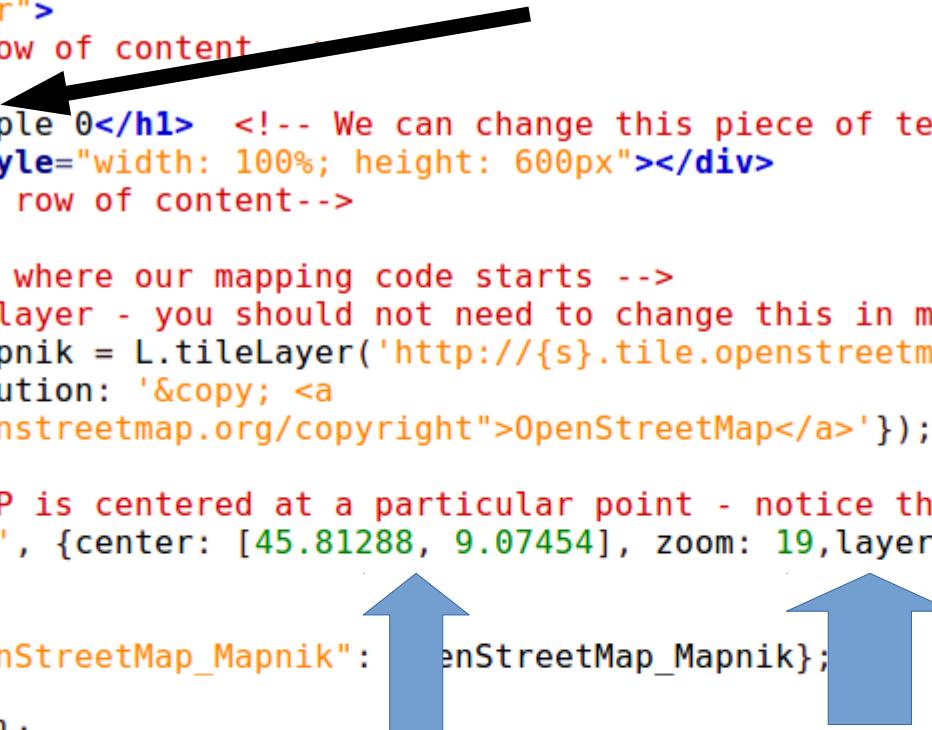
var baseMaps = {"OpenStreetMap_Mapnik": OpenStreetMap_Mapnik};

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 a marker pin on our own house/appartment

- 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 home – change the code in your file so that the marker pin is changed.
- Change also the <TITLE> and the <h1> text
- SAVE the file and then look at  
<http://localhost:8080/exerciseA.html>

# Let's look at Ex1.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

# 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**

# 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 YOUR HOME
- You will need to also test and find an appropriate center for your map.
- You will need to fina an appropriate zoom
- Result at <http://localhost:8080/exerciseB.html>

# Changing the Background Layer

- In the previous examples we have used the default OpenStreetMap MAPNIK layer
- BUT the great thing about Leaflet is that we can easily change the background layer

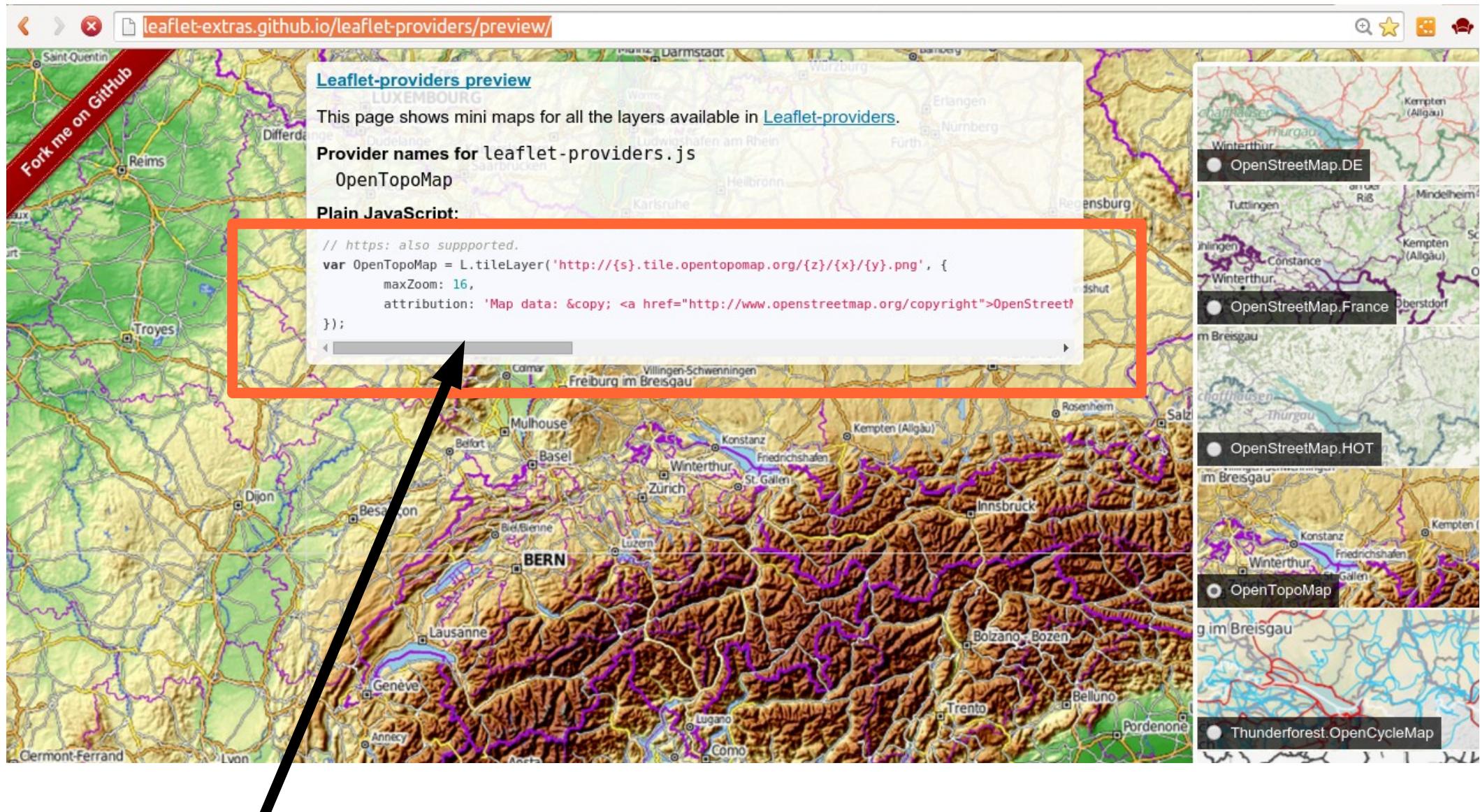
You will need to change 4 pieces of information

```
1 <script><!-- This is where our mapping code starts -->  
2 // This is our base layer - you should not need to change this in many examples  
3 var OpenStreetMap_Mapnik = L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {maxZoom:  
4 19, attribution: '&copy; OpenStreetMap'});  
  
// Notice how the MAP is centered at a particular point - notice the ZOOM level!  
var map = L.map('map', {center: [45.80888, 9.08367], zoom: 15, layers: [OpenStreetMap_Mapnik]});  
  
var baseMaps = {"OpenStreetMa2_Mapnik": OpenStreetMap_Mapnik};  
var overlayMaps = { };  
  
L.control.layers(baseMaps, overlayMaps).addTo(map);
```

# 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  
<http://leaflet-extras.github.io/leaflet-providers/preview/>

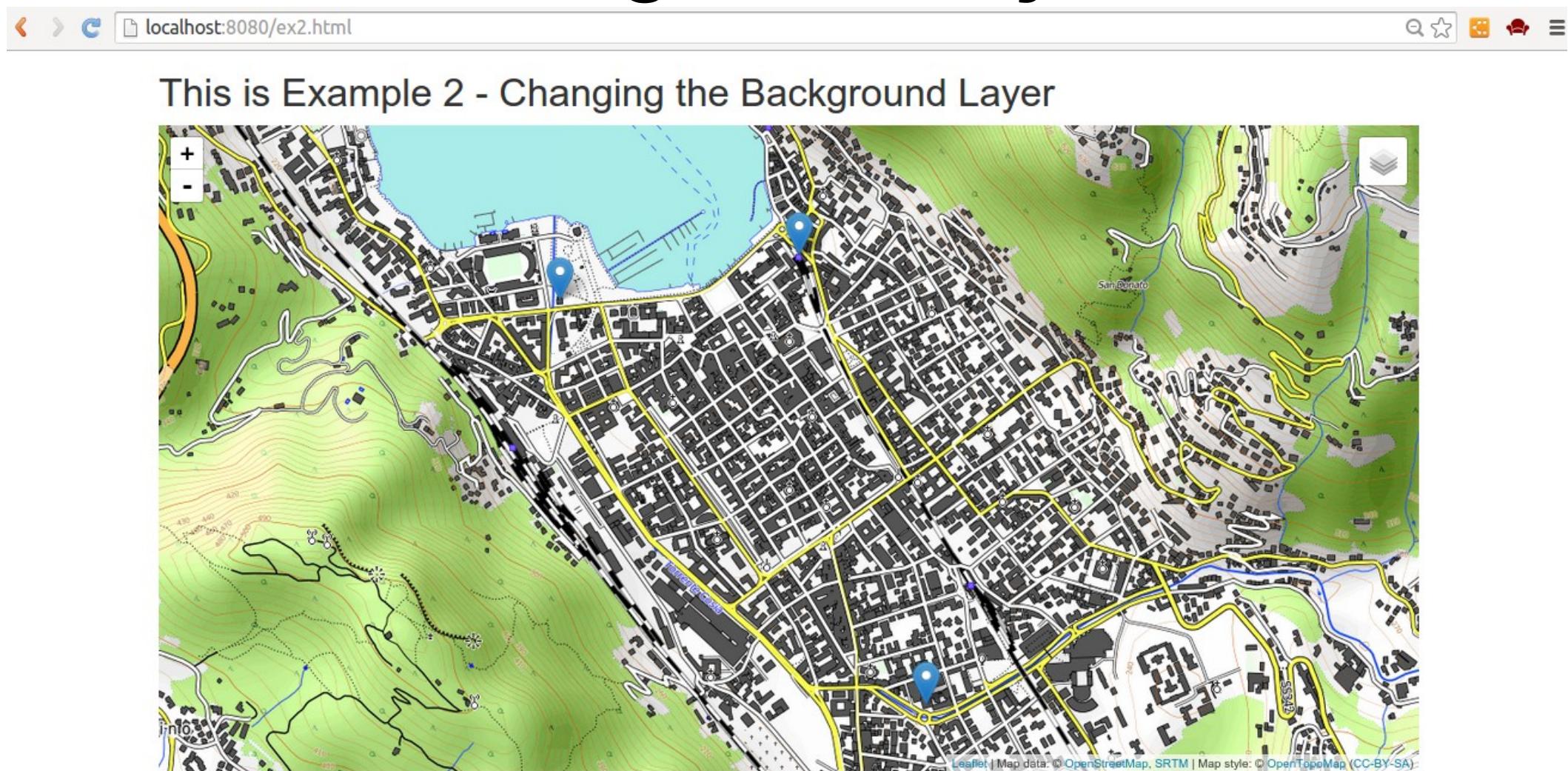
# Leaflet Layer Providers



# Exercise C: 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'

# This is one example of a different background layer



# Exercise D – Having Multiple Background Layers

- It's possible to have several different background layers which the user can easily switch between.
- Let's look at ex3.html in your text editor
- We just use Leaflet Layer Providers to get the code for our new layer
- Let's look and see where the new code fits in and then we'll test it.

# Exercise D: Having Multiple Background Layers

```
<script><!-- This is where our mapping code starts --&gt;
// This is our base layer - you should not need to change this in many examples
var OpenTopoMap = L.tileLayer('http://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {
  maxZoom: 16,
  attribution: 'Map data © &lt;a href="http://www.openstreetmap.org/copyright"&gt;OpenStreetMap&lt;/a&gt;, &lt;a href="http://viewfinderpanoramas.org"&gt;SRTM&lt;/a&gt; | Map style: © &lt;a href="https://opentopomap.org"&gt;OpenTopoMap&lt;/a&gt; (&lt;a href="https://creativecommons.org/licenses/by-sa/3.0/"&gt;CC-BY-SA&lt;/a&gt;)'
});

var Stamen_Toner = L.tileLayer('http://stamen-tiles-{s}.a.ssl.fastly.net/toner/{z}/{x}/{y}.png', {
  attribution: 'Map tiles by &lt;a href="http://stamen.com"&gt;Stamen Design&lt;/a&gt;, &lt;a href="https://creativecommons.org/licenses/by/3.0"&gt;CC-BY 3.0&lt;/a&gt; — Map data © &lt;a href="http://www.openstreetmap.org/copyright"&gt;OpenStreetMap&lt;/a&gt;',
  subdomains: 'abcd',
  minZoom: 0,
  maxZoom: 20,
  ext: 'png'
});

// Notice how the MAP is centered at a particular point - notice the Z0 Level
var map = L.map('map', {center: [5.80880, 9.08367], zoom: 15, layers: [OpenTopoMap,Stamen_Toner]});

var baseMaps = {"OpenTopoMap": OpenTopoMap,"Nice Stamen Toner Map": Stamen_Toner};</pre>
```

# Exercise D: Making a map with THREE background layers

- Copy the file ex3.html and rename the file exerciseD.html (*stay in the same folder*)
- Open exerciseD.html in your text editor
- Use Leaflet Map Providers page to choose THREE different background layers.
- BE CAREFUL – You will need to make the changes to your code as shown
- Look at the result in your browser at <http://localhost:8080/exerciseD.html>

# 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(map);
```

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

# Ex4: 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 ex4.html in your text editor

```
// Notice how the MAP is centered at a particular point - notice the ZOOM level
var map = L.map('map', {center: [45.80880, 9.08367], zoom: 15, layers: [OpenStreetMap_BlackAndWhite,MapQuestOpen_OSM]}); 

var baseMaps = {"OSM BW": OpenStreetMap_BlackAndWhite, "MapQuest OSM": 
var overlayMaps = { };

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

var polygon1 = L.polygon([
[45.81424, 9.07097],
[45.81460, 9.07350],
[45.81339, 9.07386],
[45.81307, 9.07144],
[45.81374, 9.07068],
[45.81424, 9.07097]
]).addTo(map);
polygon1.bindPopup("I am a polygon showing the outline polygon for the Stadio Giuseppe Sinigaglia in Como");
```



# Exercise E: Adding more Polygons

- Copy ex4.html and rename it as exerciseE.html
- **AIM:**
- Use 'Get Lat Long' to help create TWO NEW POLYGONS (you can fully delete the polygon which is already in ex4.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**

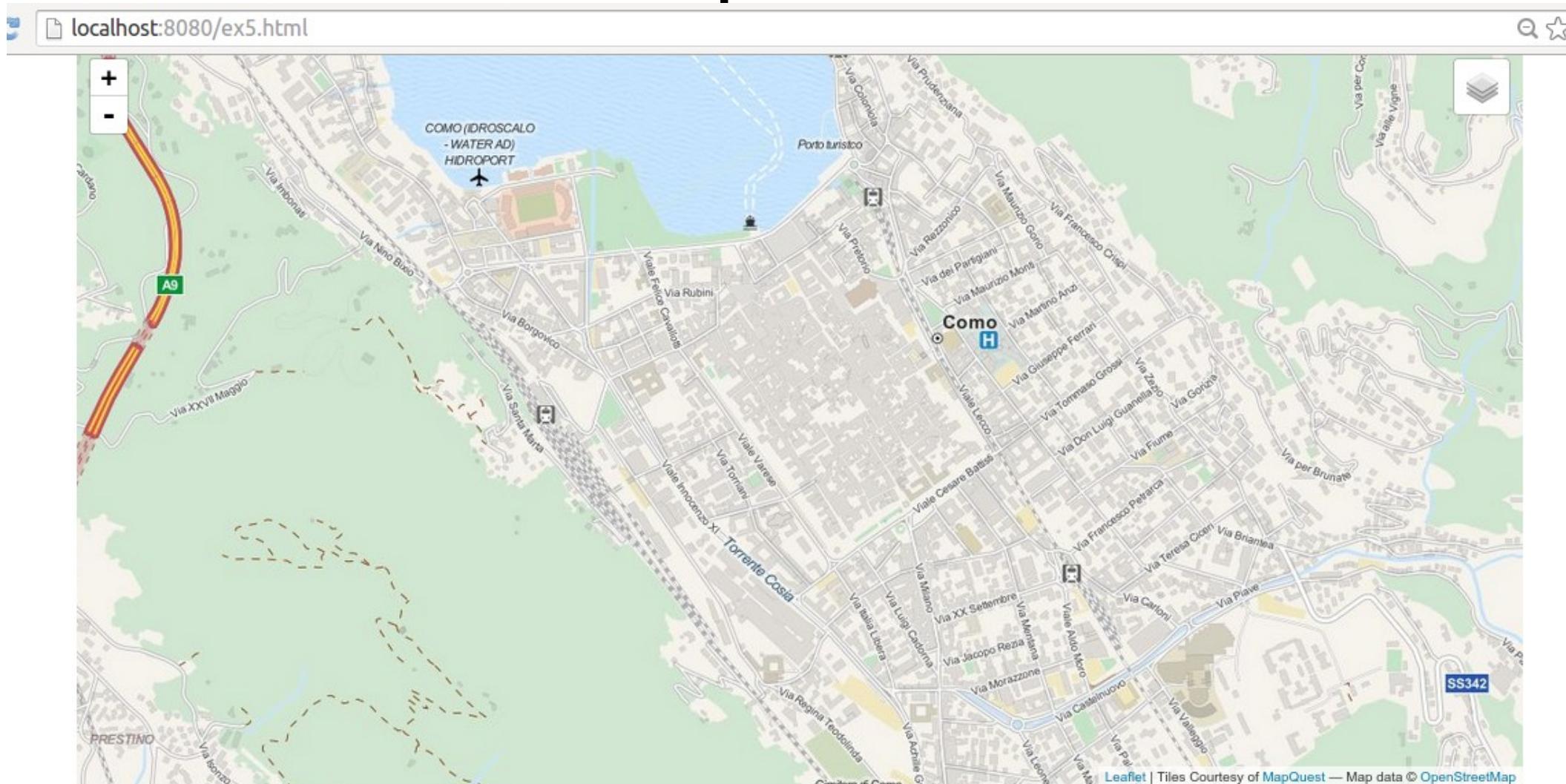
**Let's create our very own simple  
version of Get Lat Long for  
ourselves**

# Ex5.html – Open this file in your text editor and view it in the browser

- Open Ex5.html in your text editor
- Check out <http://localhost:8080/ex5.html> in the browser
- **We have added some code – but it is a really useful piece of code**

```
56  
57 // This is the where Javascript is connected to the map when you 'click' on the map  
58 map.on('click', onMapClick);  
59  
60 // THIS IS NEW. This is a special FUNCTION from Javascript.  
61 // Leaflet has lots of functionality that we can use. This is very useful to display information about  
the map  
62 function onMapClick(e) {  
 63   var theDateAndTime = new Date();  
 64   var mapCenter = map.getCenter();  
 65   $("#mycoordinates").html("<p class = 'lead'>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>");  
66 }  
67 }
```

# This is the output from Ex5.html



MyGetLatLong Says:

At Mon Jul 13 2015 17:10:54 GMT+0100 (IST)

You clicked the map at Coordinates (Lat, Lon) = 45.808823378560895,9.085154533313471

Also the map is at Zoom level 15

The CENTER of the map is currently at Coordinates (Lat,Lon) = 45.808803435510455,9.081478118932864

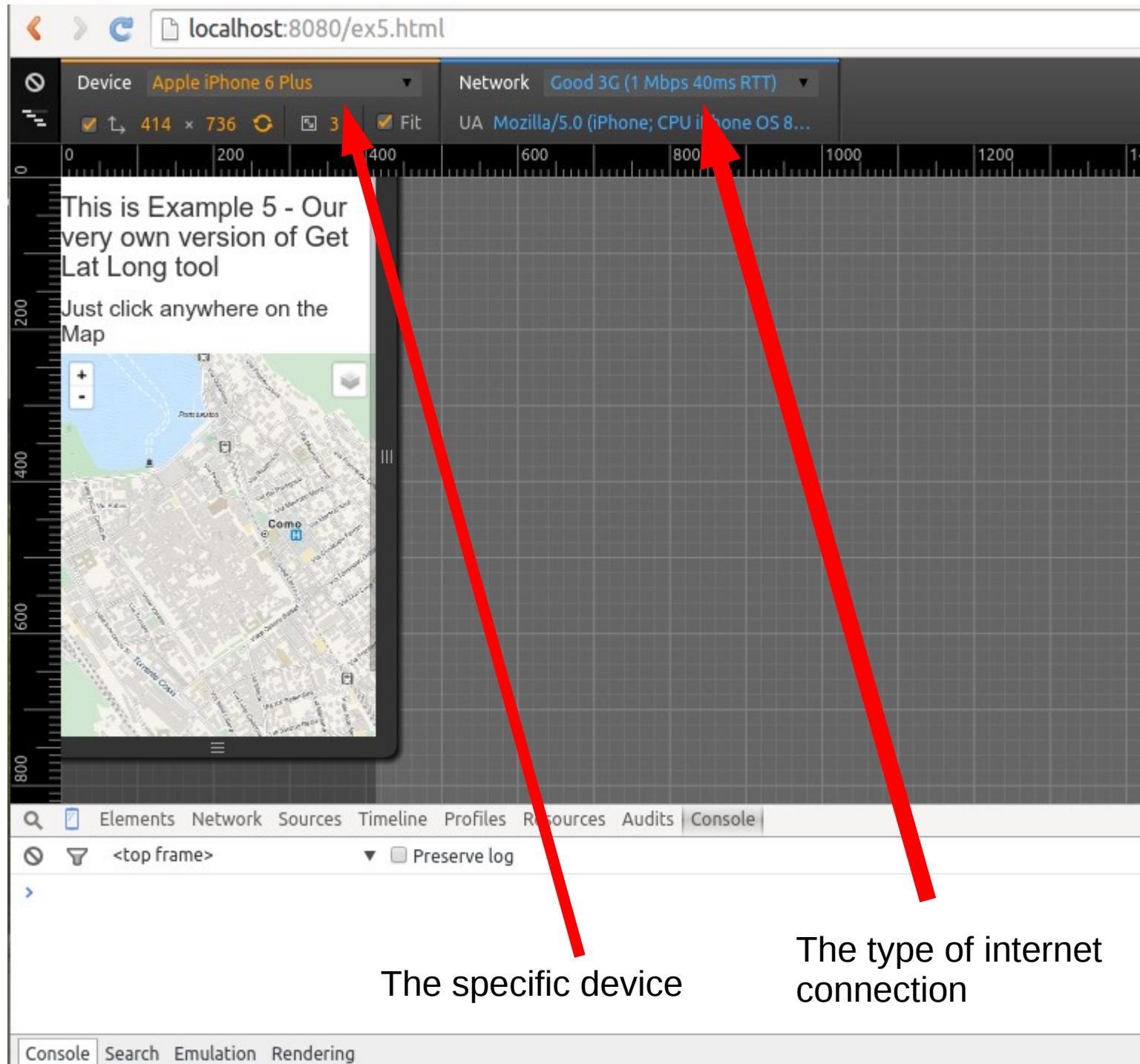
**How does all this look on your mobile phone?**

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



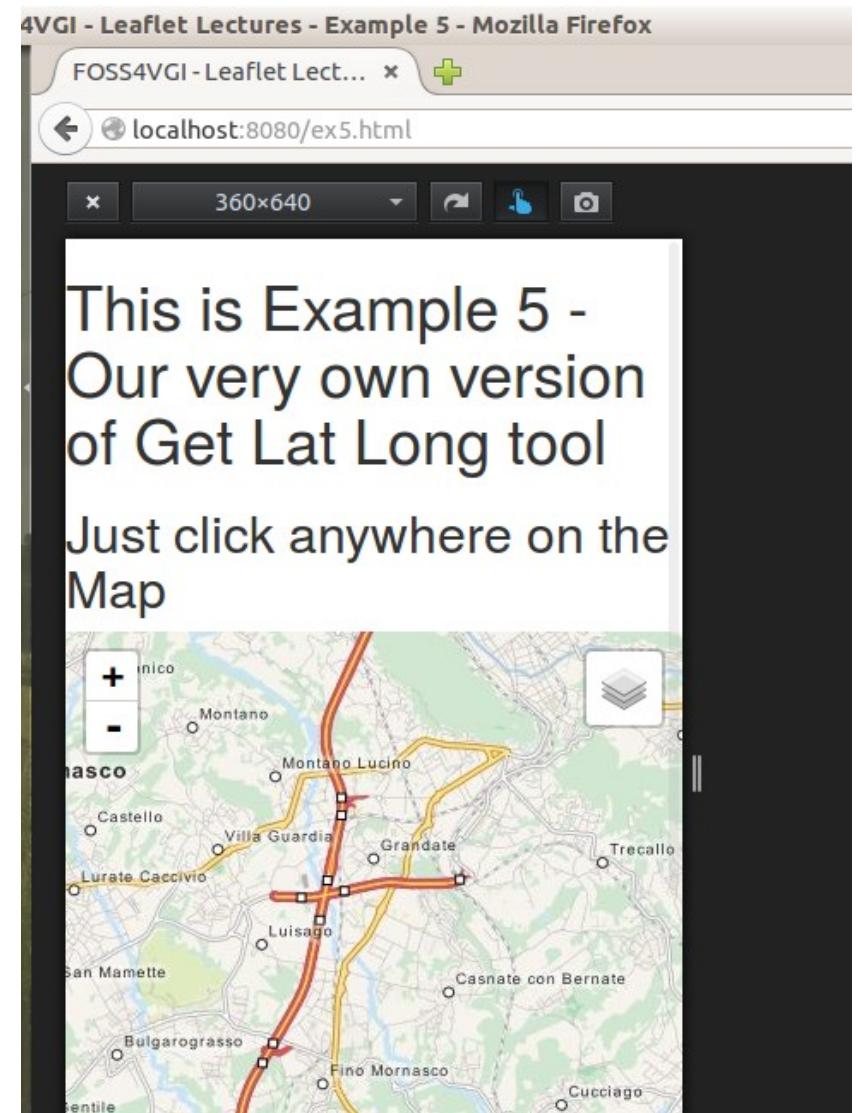
# If you are using Google Chrome

- Open up ex5.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"



# **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

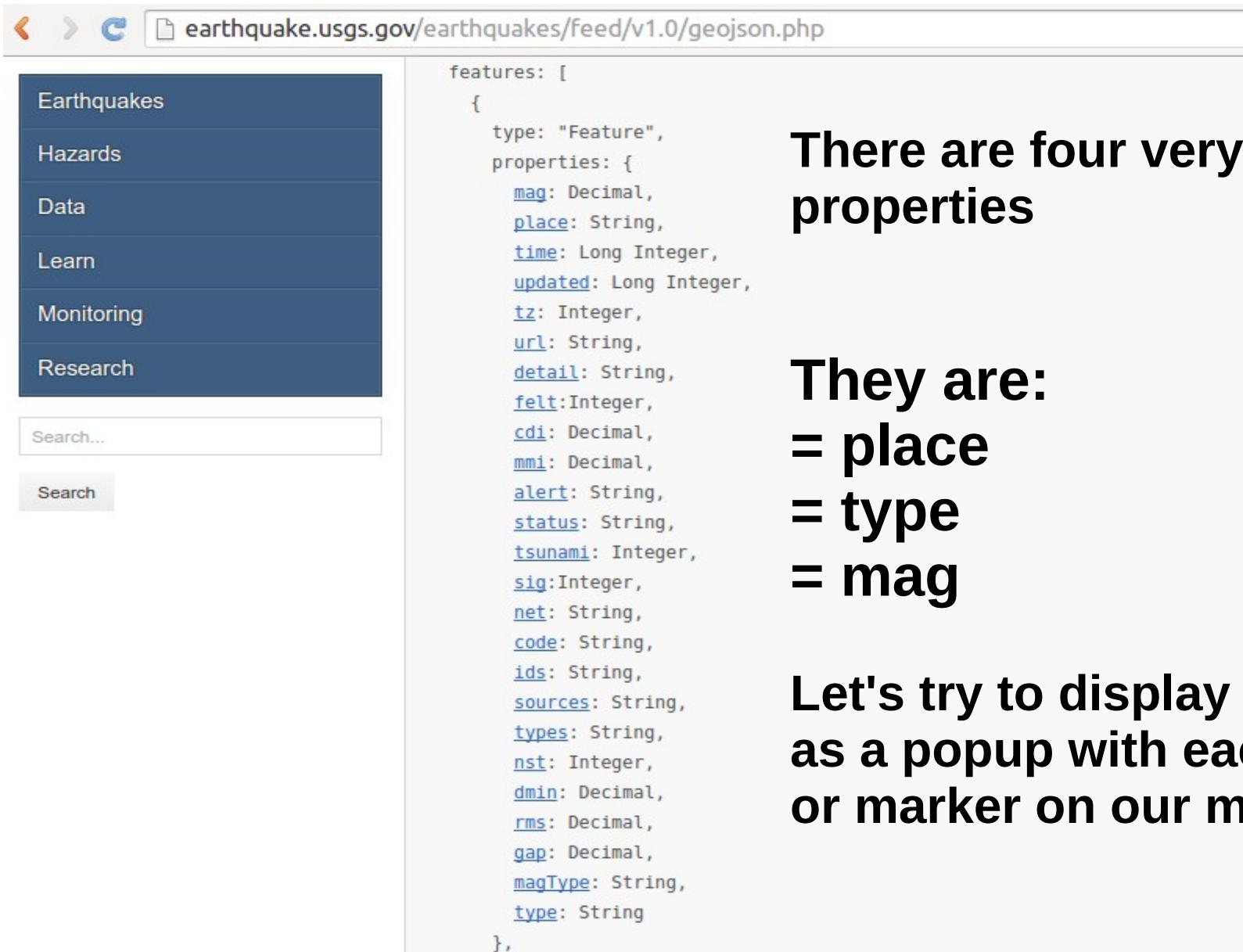
US Geological Survey - Past 7 Days Earthquakes



# **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



The screenshot shows a browser window with the URL `earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php`. On the left, there is a sidebar with links: Earthquakes, Hazards, Data, Learn, Monitoring, and Research. Below these is a search bar labeled "Search..." and a "Search" button. The main content area displays a JSON object:

```
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 four very useful properties**

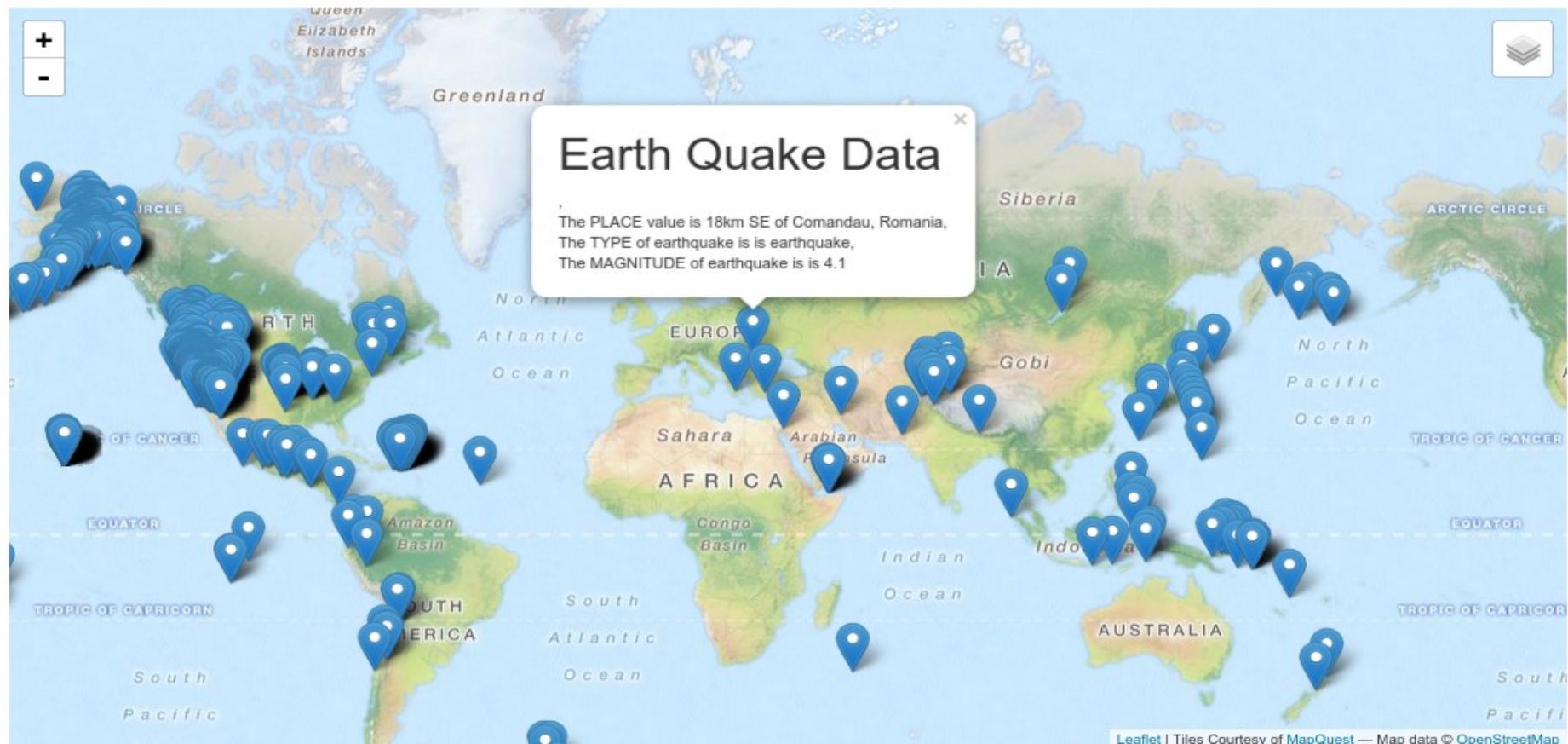
**They are:**  
**= place**  
**= type**  
**= mag**

**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>

US Geological Survey - Past 7 Days Earthquakes - This time with clickable icons with information!



# 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 examples in the future!

```
// This is where Leaflet will go if you click on any of the objects which are in the
// GeoJSON file
function action_To_Perform_When_Marker_Is_Clicked_On_The_Map(feature, layer) {

    // does this clicked feature have any properties in its GeoJSON?
    if (feature.properties)
    {
        // the text for the Popup
        var PopupText = [];

        PopupText.push("<h1>Earth Quake Data</h1>");
        |
        if (feature.properties.place) // if this is a property called place
            PopupText.push("<br/>The PLACE value is " + feature.properties.place)

        if (feature.properties.type) // if this is a property called type
            PopupText.push("<br/>The TYPE of earthquake is " + feature.properties.type)

        if (feature.properties.mag) // if this is a property called type
            PopupText.push("<br/>The MAGNITUDE of earthquake is " + feature.properties.mag)

        // Join all the text to make the Popup
        layer.bindPopup("<p>" + PopupText.join() + "</p>");

    } // end if
} // end of function
```

Look at the PROPERTIES here  
- these appear EXACTLY as they are typed in the GeoJSON file

= place  
= type  
= mag

**Let's go back to 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 areas around Regents Park, Somers Town, Bloomsbury, Fitzrovia, St. Giles, Covent Garden, and the City of London. Numerous parking locations are marked with blue circles of varying sizes. A legend on the left side of the map identifies these symbols. A callout box with a black border and white text is overlaid on the map, containing the following instructions: "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 map, there are status bars showing "Loaded – nodes: 84, ways: 0, relations: 0" and "Displayed – pois: 84, lines: 0, polygons: 0".

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**

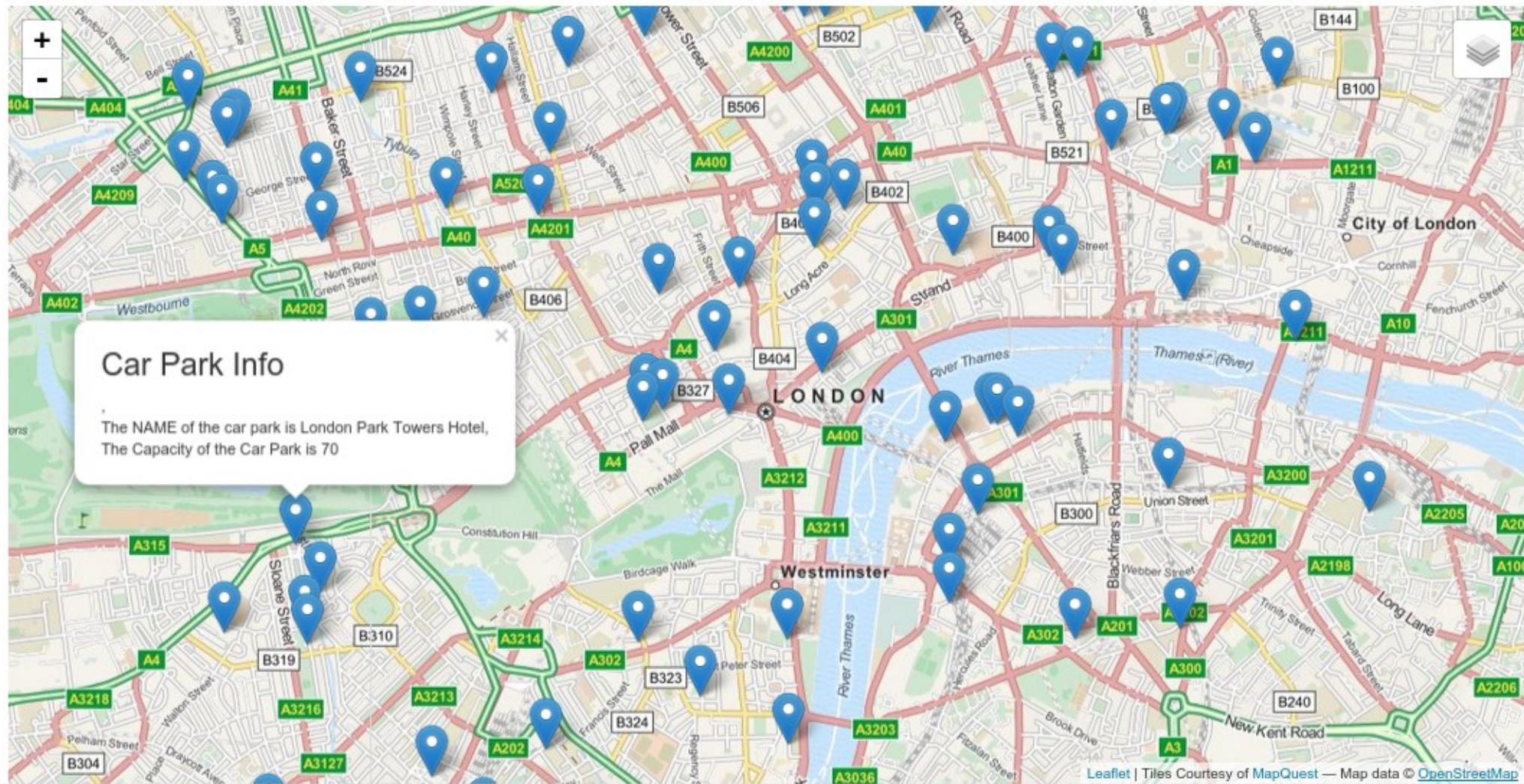
**Check Map Features if you need more information**

# 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!

PARKING PLACES in Central London, UK. GeoJSON data from the Overpass Turbo service



# Look VERY carefully at the action\_To\_Perform\_When\_Marker\_Is\_Clicked\_On\_The\_Map function

```
// This is where Leaflet will go if you click on any of the objects which are in the
// GeoJSON file
function action_To_Perform_When_Marker_Is_Clicked_On_The_Map(feature, layer) {

    // does this clicked feature have any properties in its GeoJSON?
    if (feature.properties)
    {
        // the text for the Popup
        var PopupText = [];

        PopupText.push("<h3>Car Park Info</h3>");

        if (feature.properties.name) // if this is a property called name
            PopupText.push("<br/>The NAME of the car park is " + feature.properties.name)

        if (feature.properties.type) // if this is a property called operator
            PopupText.push("<br/>The OPERATOR of the car park is " + feature.properties.operator)

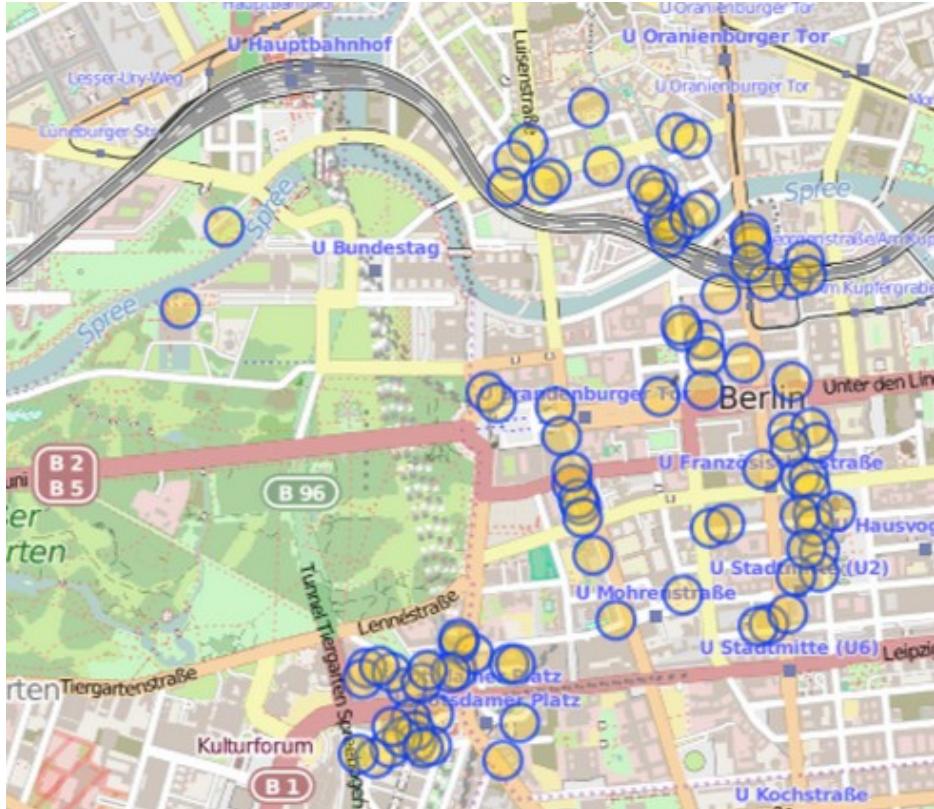
        if (feature.properties.capacity) // if this is a property called capacity
            PopupText.push("<br/>The Capacity of the Car Park is " + feature.properties.capacity)

        // Join all the text to make the Popup
        layer.bindPopup("<p>" + PopupText.join() + "</p>");
    } // end if
} // end of function
```

NOTICE – How we had to change very little – we just change the text a little and we put in our own properties which we picked out a few minutes ago

**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

# Colors for Polygons and other features in Javascript

# Visibone Color Lab is very useful

The screenshot shows a web browser window for the Visibone Color Lab at <https://www.visibone.com/colorlab/>. The main interface features a circular "The 216-Color Webmaster's Palette" with various colored hexagons. A specific color, "Light Dull Magenta" with the hex code CC66CC, is highlighted in purple. To the right of the color swatch, its CMYK values are listed: 204=R, 40=G, 0=Y, 20=K. Below this, a row of six color swatches shows the color in different color spaces: LSG (lightness), Y (yellow), R (red), RRO (cyan), LDM (lightness), and CC66CC (hex). At the bottom of the page, there is a multilingual instruction: "Click on another color..." followed by translations in French, German, Spanish, Norwegian, Dutch, Swedish, Italian, Czech, Danish, Indonesian, Portuguese, Polish, Japanese, and Russian.

CC66CC  
LDM  
Light Dull Magenta

204=R 40=G 0=Y  
102=G 0=Y 20=K  
204=B

99FF66 LSG	FFFF00 Y	FF0000 R	FF3300 RRO	CC66CC LDM
Y R RRO X LDM	LSG R RRO X LDM	LSG Y RRO X LDM	LSG Y R RRO X LDM	LSG Y R RRO X LDM

Click on another color...  
Cliquer une autre couleur...  
Klicken Sie auf eine andere Farbe...  
Haga clic en otro color...  
Klikk på en annen farge...  
Probeer nog een andere kleur...  
Klicka på en annan färg...  
Cliccare su un altro colore...  
Klikněte na jinou barvu...  
Klik på en anden farve...  
klik pada warna lainnya...  
Clique em outra cor...  
Kliknij inny kolor...  
再選一個顏色...  
他の色をクリックしてください...  
Нажмите на другой цвет...

It is very good practice to always specify colors using their HEX value – all browsers and Javascript understand HEX

The screenshot shows a web browser window with the URL <https://www.visibone.com/colorlab/>. The page features a decorative background of stylized flowers and hexagonal color swatches. On the left, there's a large, colorful hexagonal color palette labeled "The 216-Color Webmaster's Palette". In the center, a large hexagon displays the color code CC66CC, labeled "LDM" (Light Dull Magenta). To the right of this main hexagon is a smaller one containing the color FF3300 (Red Orange). Below these are several smaller color swatches and text labels. A large, semi-transparent gray arrow points from the text "#CC66CC" at the bottom right towards the central color swatches. At the bottom of the page, there is a list of multilingual instructions: "Click on another color...", "Cliquer une autre couleur...", "Klicken Sie auf eine andere Farbe...", "Haga clic en otro color...", "Klikk på en annen farge...", "Probeer nog een andere kleur...", "Klicka på en annan färg...", "Cliccare su un altro colore...", "Klikněte na jinou barvu...", "Klik på en anden farve...", "klik pada warna lainnya...", "Clique em outra cor...", "Kliknij inny kolor...", "再選一個顏色...", "他の色をクリックしてください...", and "Нажмите на другой цвет...".

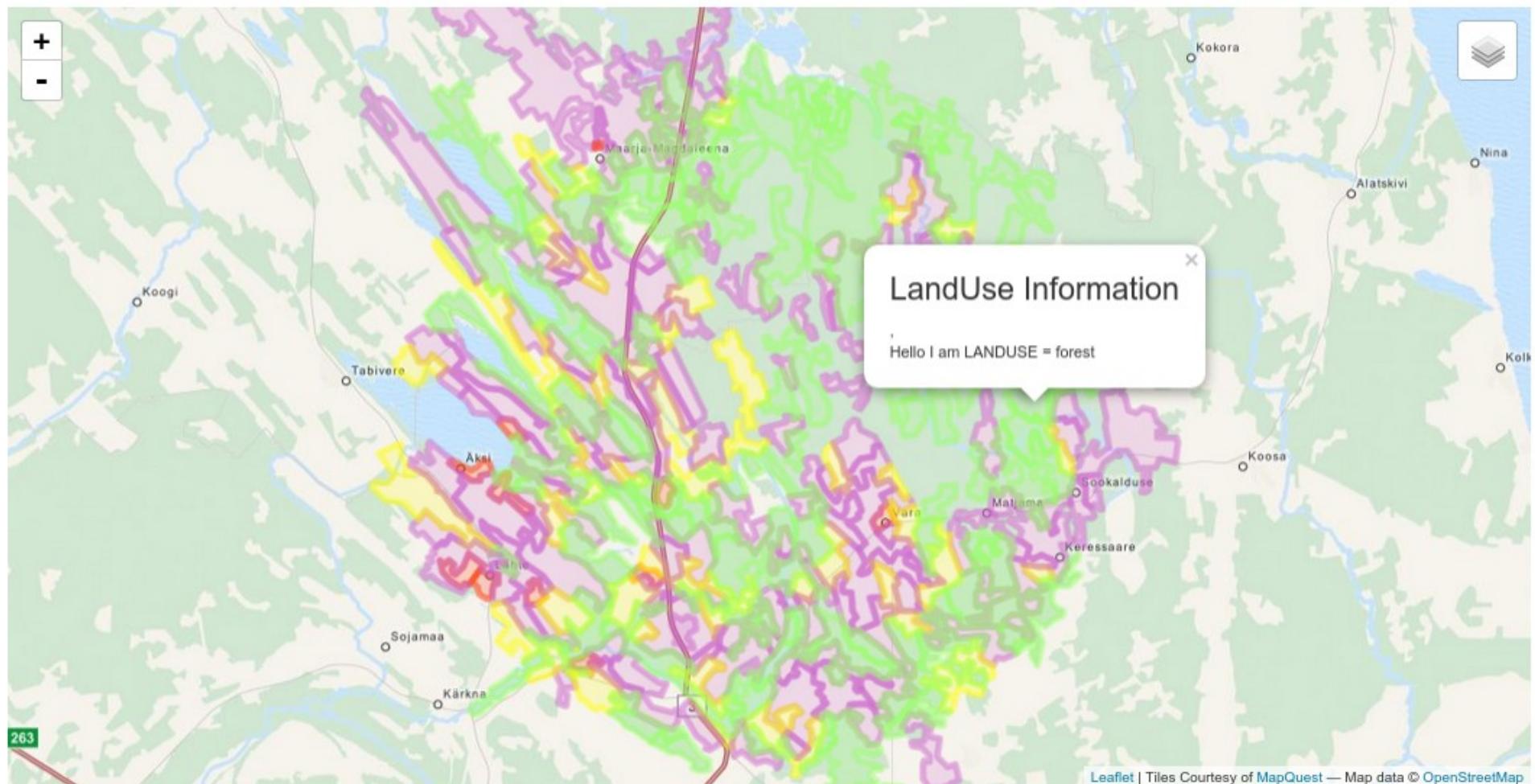
#CC66CC

# Colouring/Styling Landuse Polygons in Estonia

- Open up geo4.html in your text editor and in your browser at <http://localhost:8080/geo4.html>
- This is an example where we have POLYGONS and not points.
- To make a map a bit more interesting we have decided to give the polygons a different colour depending on the value of the tag
- We are looking at LANDUSE tags only

# The geo4.html web map should look something like this in your browser

Styling Land Cover Maps (Polygons) from Estonian OpenStreetMap

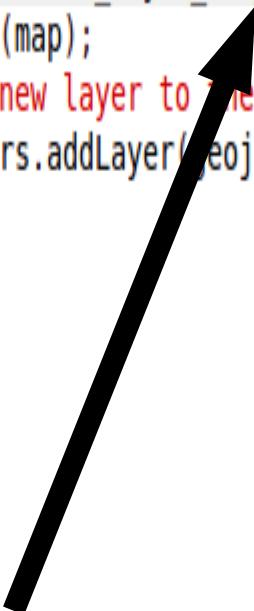


# There are a few VERY important changes to our Javascript

- 1 – We tell LEAFLET that we want to put a specific STYLE on the polygons
- 2 – We must have code to tell LEAFLET what the colours in the specific STYLE actually are
- 3 – We must also adapt our code for when someone clicks on one of the polygons – in this case we just show the landcover tag

# We tell LEAFLET that we want to put a specific STYLE on the polygons

```
$getJSON("./geojson/estoniaLanduse.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: specific_style_for_ways});
    geojson.addTo(map);
    <!-- add our new layer to the group of top layers -->
    myGeoJSONLayers.addLayer(geojson);
  }
);
```



# We must have code to tell LEAFLET what the colours in the specific STYLE actually are

```
// This is a VERY important function. This is where we say which colors will be
// displayed for specific values of the landuse tag
function specific_style_for_ways(feature) {
    switch (feature.properties.landuse) {
        case 'meadow': return {color: "#FFFF00"};
        case 'forest':  return {color: "#99FF66"};
        case 'farm':   return {color: "#CC66CC"};
        default: return {color: "#FF3300"}; // this is when we don't know the property value
                                         // or it is different to the ones above
    }
}
```

We must also adapt our code for when someone clicks on one of the polygons – in this case we just show the landcover tag

```
// This is where Leaflet will go if you click on any of the objects which are in the
// GeoJSON file
function action_To_Perform_When_Marker_Is_Clicked_On_The_Map(feature, layer) {

    // does this clicked feature have any properties in its GeoJSON?
    if (feature.properties)
    {
        // the text for the Popup
        var PopupText = [];

        PopupText.push("<h3>LandUse Information</h3>");

        if (feature.properties.landuse) // if this is a property called landuse
            PopupText.push("<br/>Hello I am LANDUSE = " + feature.properties.landuse)

        // Join all the text to make the Popup
        layer.bindPopup("<p>" + PopupText.join() + "</p>");
    } // end if
} // end of function
```

Just a very simple popup for this example

# Optional Exercise on Polygons

# Visualising Amenity polygons in Como

- Go to the GitHUB page for today's lecture
- Copy the final piece of OVERPASS code into overpass online
- This is for amenity objects in COMO
- Save the output as GeoJSON into your geoJSON folder

# Exercise – Colouring amenity objects in Como

- Copy the geo4.html file and rename as geo5.html
- Make the necessary changes to center, zoom and name of geoJSON file
- Color all Amenity = School (#009246)
- Color all Amenity = Parking (#CE2B37)
- Color everything else (White #FFFFFF)
- In the POPUP information just give the AMENITY type as text

So now you can develop your own  
web-based maps using Open  
Source Software AND Open Data



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. The right side of the interface displays the corresponding GeoJSON code:

```
1 {  
2   "type": "FeatureCollection",  
3   "features": [  
4     {  
5       "type": "Feature",  
6       "properties": {"name": "Maynooth University"},  
7       "geometry": {  
8         "type": "Polygon",  
9         "coordinates": [  
10            [  
11              [-6.605830192565918,  
12                53.387346941404545  
13              ],  
14              [-6.6031694412231445,  
15                53.387654077940816  
16              ],  
17              [-6.59879207611084,  
18                53.38668147130748  
19              ],  
20              [-6.595573425292969,  
21                53.385606459176216  
22              ],  
23              [-6.605830192565918,  
24                53.387346941404545  
25              ]  
26            ]  
27          ]  
28        }  
29      }  
30    ]  
31  }  
32 }  
33 }
```

# 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. The map includes labels for North Campus, Arts Building, Science Building, Post Primary School Maynooth, South Campus, Laraghbryam, Kilcock Rd, River-Lyreen, Main St, Parson St, Leinster Pk, Royal Canal Way, Carton Ave, and Maynooth Castle. A large dark green polygon highlights the North Campus area. The JSON code on the right defines this polygon as a FeatureCollection with one Feature containing a Polygon geometry.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {"name": "Maynooth University"},
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              [
                -6.605830192565918,
                53.387346941404545
              ],
              [
                -6.603169441223145,
                53.387654077940816
              ],
              [
                -6.59879207611084,
                53.38668147130748
              ],
              [
                -6.595573425292969,
                53.385606459176216
              ],
              [
                -6.605830192565918,
                53.387346941404545
              ]
            ]
          ]
        ]
      }
    }
  ]
}
```

# The final exercise.....



# Exercise – on your own

- Create a simple GeoJSON dataset on [GeoJSON.io](https://geojson.io)
- Give each feature one or two PROPERTIES (remember the double quotes)
- Use the code you have written for LEAFLET today to make a web-based map of the data you have created.
- In particular geo2.html, geo3.html and geo4.html will be of help! (if you are adventurous)

I'm happy to help with questions at  
[petermooney78@gmail.com](mailto:petermooney78@gmail.com)

