Web-based Mapping and Internet-GIS

A practical introduction

Who am I?

PhD Computer Science 2004

10 Years experience with Environmental Protection Agency (EPA) - web development, spatial databases, open access, informatics

10 Years research experience at NUIM - spatial data mining, algorithms, GIS, Volunteered Geographic Information, etc

Learning Outcomes

LOC 1: Use open source tools and free and open data to create simple web-based maps

LOC 2: Develop some web-based mapping code which you can use in other projects

LOC 3: Understand the concepts of how spatial data is delivered to the web using maps

What we are NOT going to do?

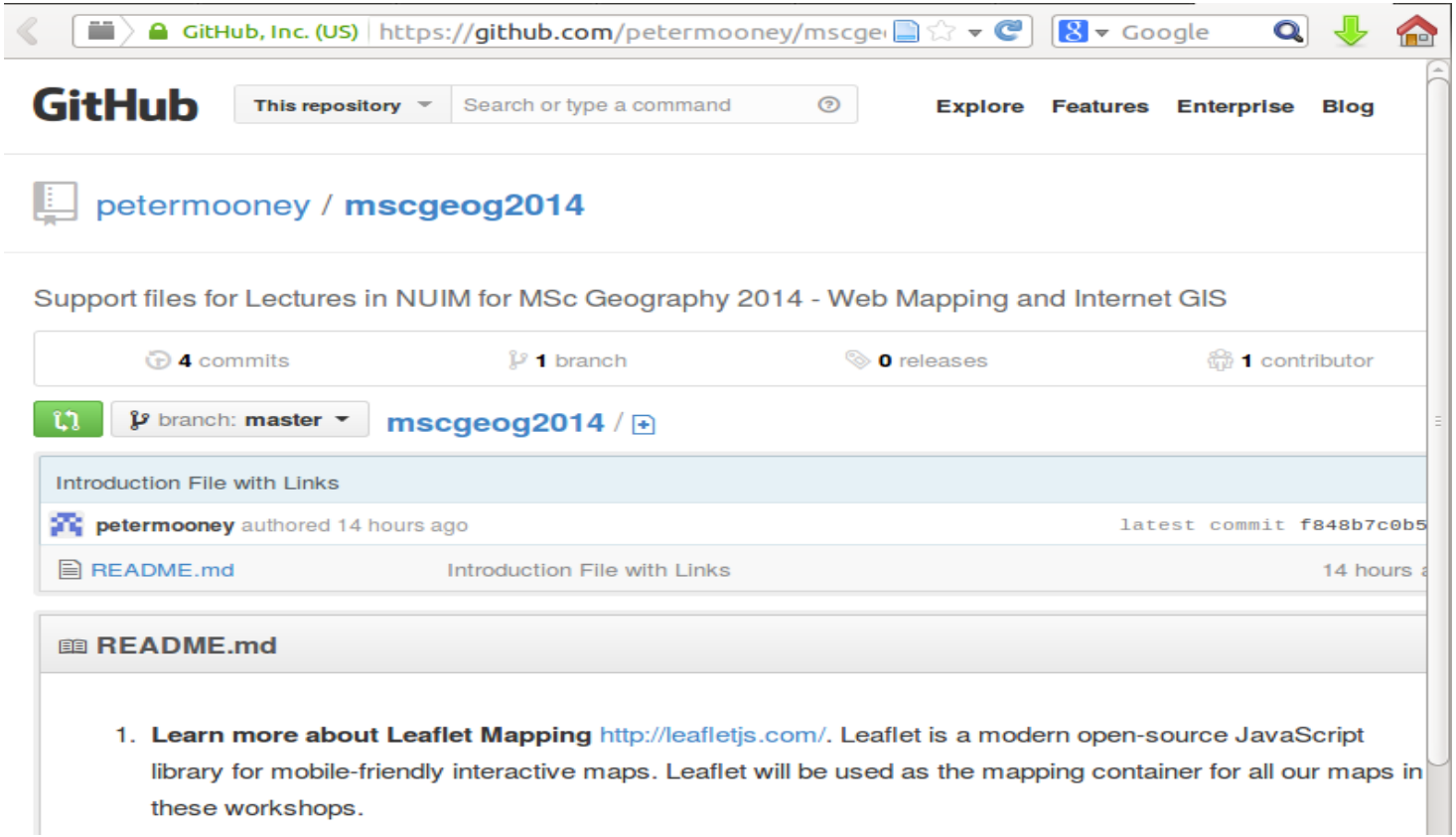
Complex programming: while programming is need to build full feature web-based GIS we won't be needing much programming here.

Web Development: There won't be much web-site user-interface development except for showing the map and the data

https://github.com/petermooney/mscgeog2014

Can I access these materials?

Yes!

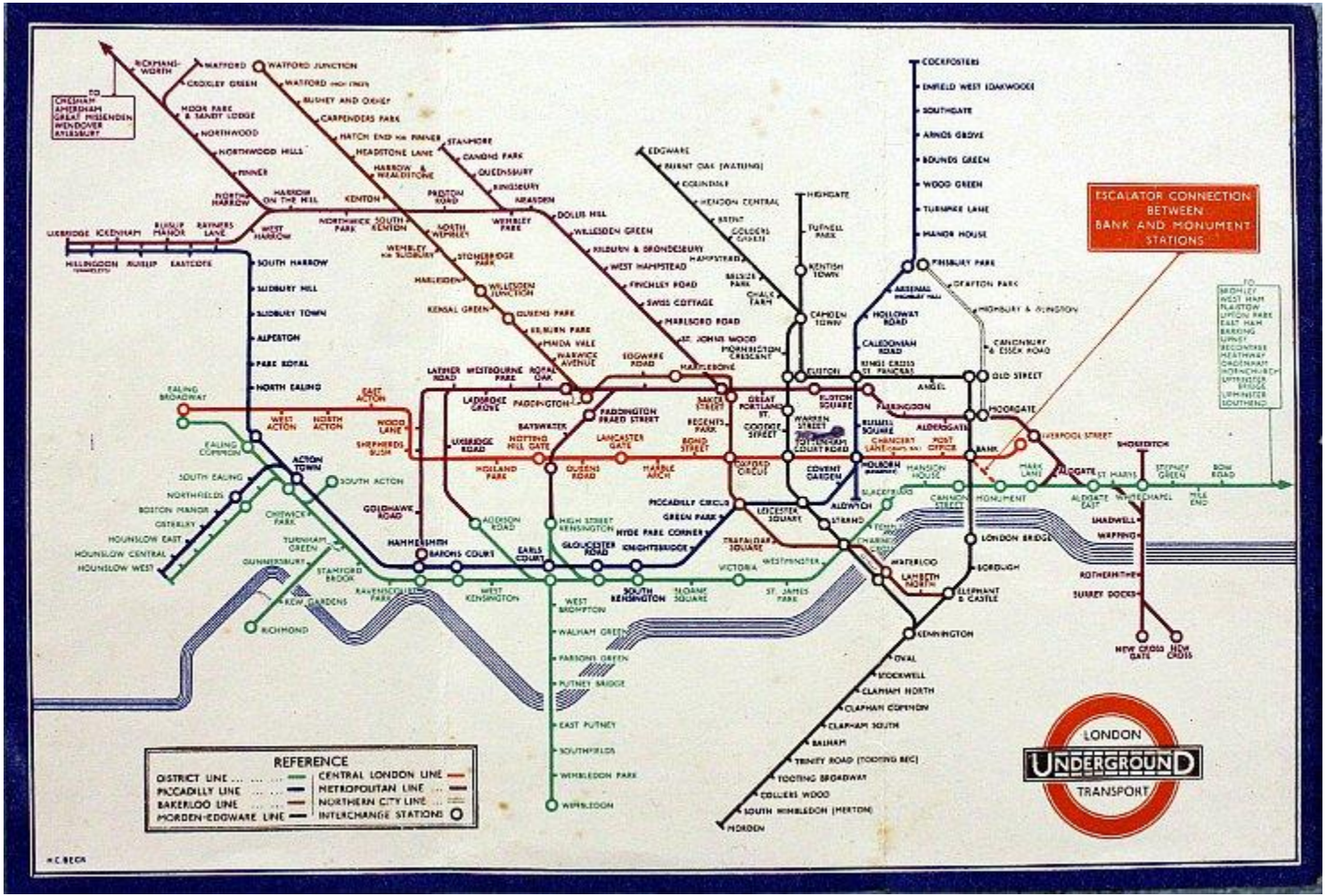


Motivation

Maps have always been important

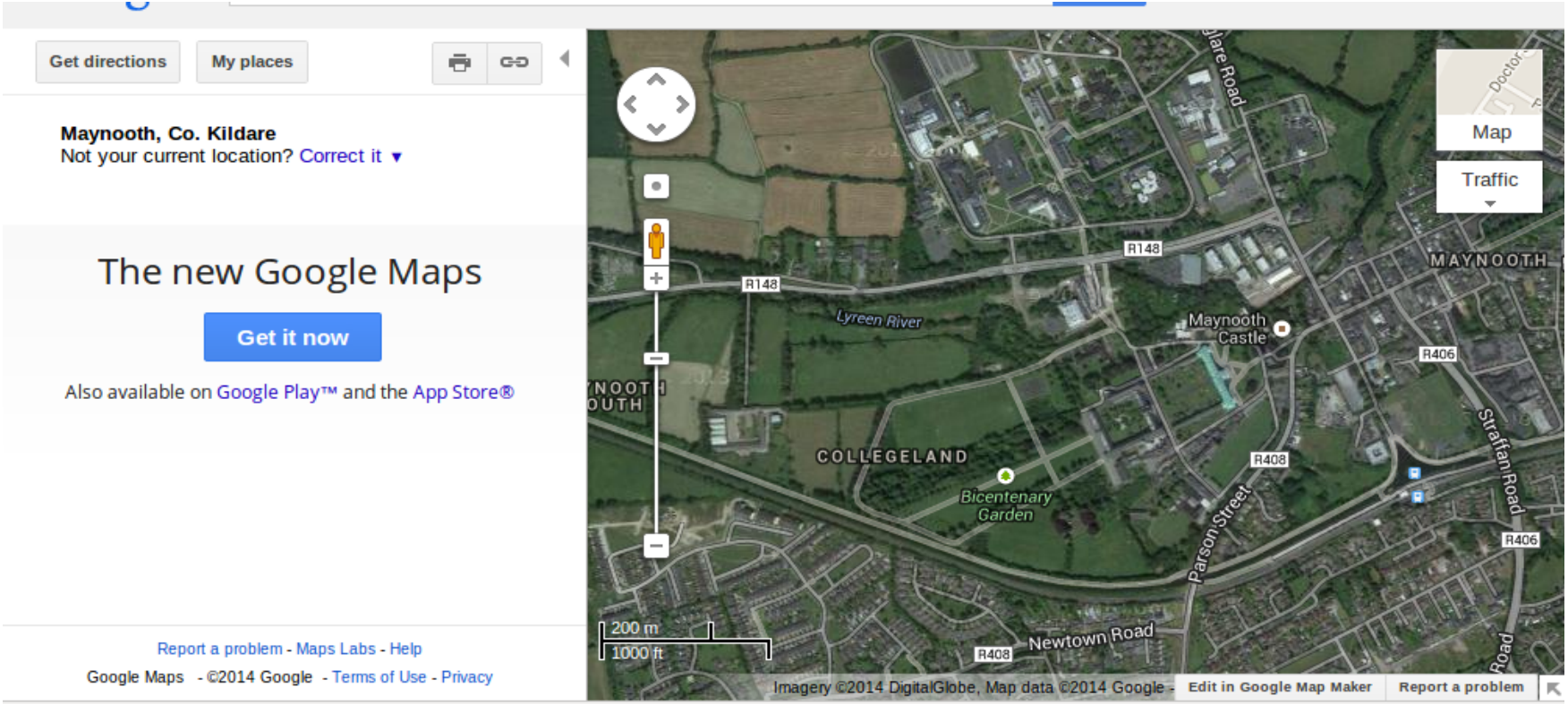
http://www.smh.com.au/ffximage/2006/09/20/knMAPS_main,0.jpg

http://www.dartmoornationalpark.co.uk/images/maps/dartmoor_map.gif



http://www.lib.cam.ac.uk/deptserv/maps/exhibitionimages/Beck2.JPG

Around 2005 - Google Maps started the web mapping era

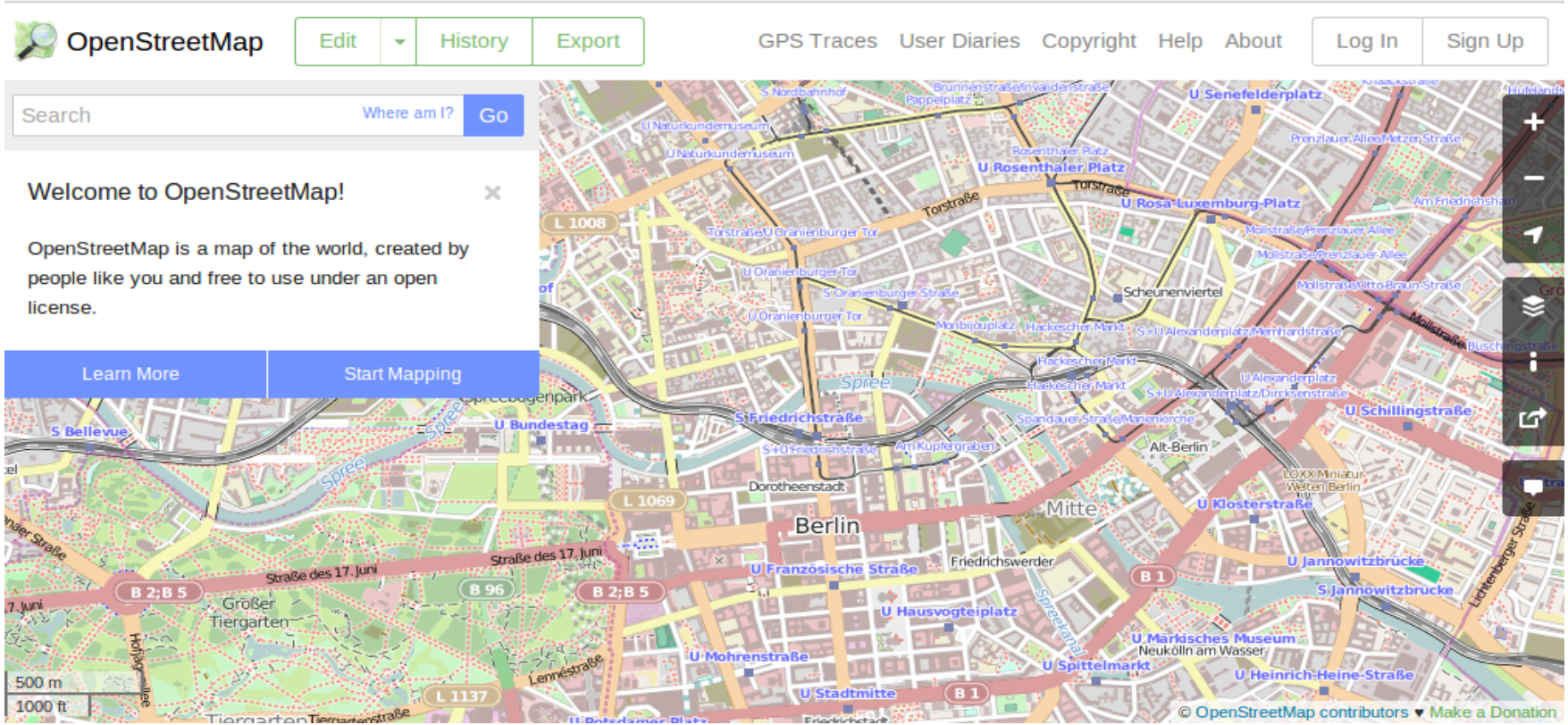


“The ancient art of cartography is now cutting edge”

Tocherman and Scharle (2008)

User Generated Content started appearing quickly

OpenStreetMap became wildly popular



OpenLayers

OpenLayers: Free Maps for the Web

Get OpenLayers Now!

- 2.13.1 (Stable): .tar.gz | .zip
- 2.13.1 Release Notes
- API Documentation, User documentation
- See examples of OpenLayers Usage: Release Examples (2.13.1), Development Examples
- Fork us on GitHub

Put an open map widget in any web page!

About OpenLayers

OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles and markers loaded from any source. OpenLayers has been developed to further the use of geographic

Books about OpenLayers

OpenLayers Cookbook

Google Maps Javascript API v3

Get Started ›

Demos: Base Maps Satellite Street View Places Routing Data Visualization


android


Plugins
Contribute
Events
Support
jQuery Foundation
jQuery
write less, do more.
.appendTo()
Sign Up
Virtual Training
April 16 - 17, 2014
Online
Naming Your Plugin
Publishing Your Plugin
Package Manifest
Search
The jQuery Plugin Registry
Search
Popular Tags
New Plugins
Recent Updates

GeoHack - Killary Harbour

Views

- Template
- Talk
- View source
- History

Languages

- Afrikaans
- Alemannisch
- العربية
- Aragonés
- Asturianu
- বাংলা
- Беларуская
- Беларуская (тарашкевіца)
- Български
- Bosanski
- Brezhoneg
- Català

WGS84	53° 37′ 0″ N, 9° 48′ 0″ W 53.616667, -9.8		
Geo URI	geo:53.616667,-9.8		
UTM	29U 447080 5941171		
Zoom	6	**Scale**	± 1:100000
Region	IE	**Type**	waterbody
Title	Killary Harbour (edit | report inaccuracies)		

Contents Global and Local services · Wikipedia articles · Photos · Other · Export

*View this location in **Google Maps**, in **OpenStreetMap**, or select one of the services listed below:*

Global services

Service	Map	Satellite	Topo	Ot
ACME Mapper	Map	Satellite	Topo	Terrain,
Apple Maps	Map			
Arctic.io				Daily Sat

Ireland

Indirect

- Ordnance Survey Ireland

View all regional services


16:02
Item List
Product
Map
Bowen Island
Description: Unlimited access to detailed topo maps of Canada from the Department of Natural Resources Canada.
The maps can be easily stored for offline usage on your mobile devices.
15:37
Connaught Range
Shirleys Bay
15:35


4
London
9:33pm
50mph
0.35
miles
10:19pm
0:45hrs
TOMTOM

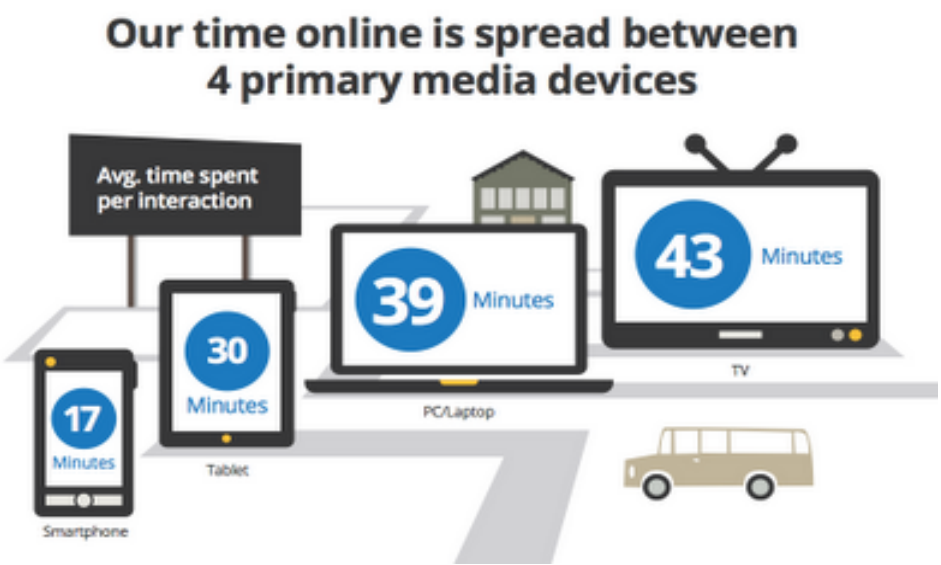

SAMSUNG
12:45 PM
DIRECTIONS LIST

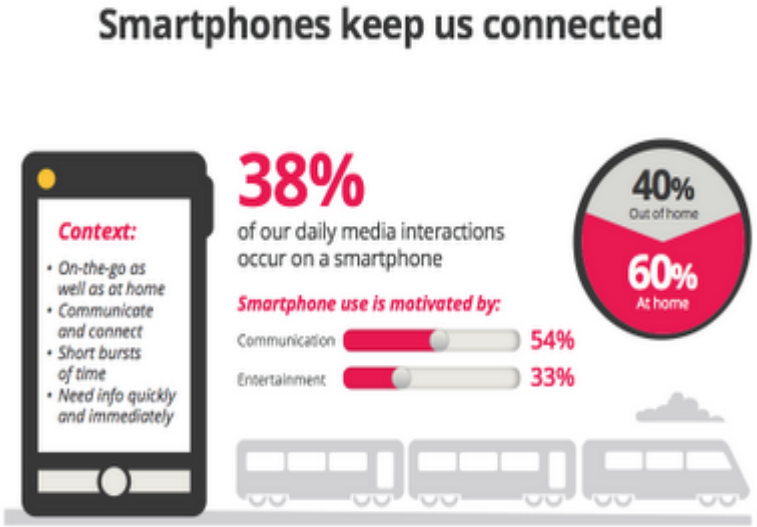
So essentially - everyone wants maps

They want them:

- On their smartphones
- On their web-pages
- On their web-applications
- Linked to their content on the Internet

All web-development must consider 3 or 4 primary display devices

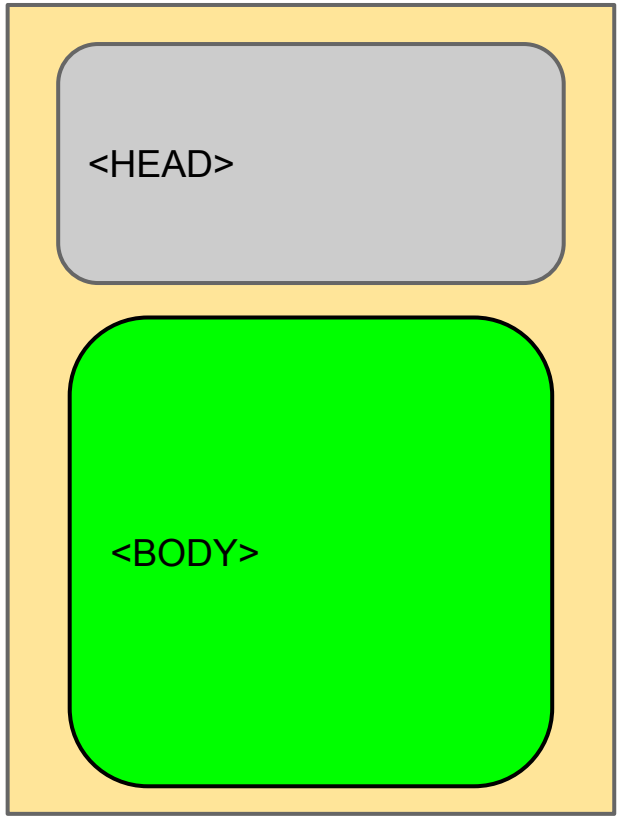




http://steverayson.kineo.com/2012/09/google-survey-on-cross-platform.html

Getting maps into web-pages

Anatomy of a web-page

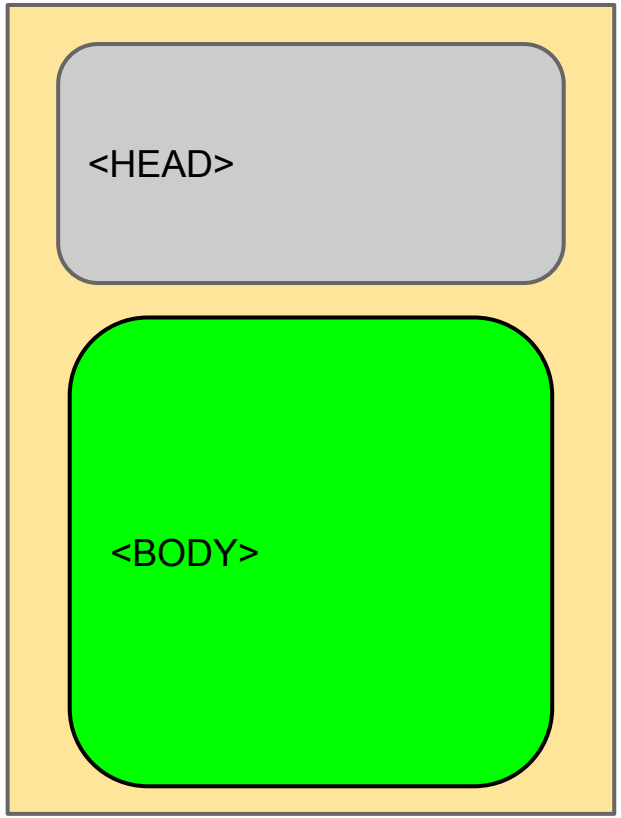


This is a very simplistic way of looking at how a web-page is constructed

In the <HEAD> - we place web programming code (Javascript) which is needed to provide our functionality in the <BODY>. We can provide links to other people's code (we try to re-use where possible)

In the <BODY> - This is where what we as human see in a webpage - the maps, the text, the images, the buttons, the search box, etc ….

Where do we put our maps?



To get a map displayed in a web-page we need to do two things

1: In the <HEAD> we need to supply the links to the Javascript code which will provide the functionality for the map

2. In the <BODY> we need to have a container (called a DIV) which will hold our map - so we drop the map into a container. In this way the web-browser understands how to display it and allow you to interact with it.

This is what the <head> of our pages will look like for this workshop

```
<head>
    <title>This is the title of your webpage</title>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <!-- this line makes sure that the scale of your device is understood
    by the browser -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- This is our leaflet mapping imports -->
    <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.css" />
    <script src="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.js"></script>

    <!-- BOOTSTRAP  -->
    <!-- This helps us create a device responsive webpage -->
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css">
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap-theme.min.css">
    <script src="//netdna.bootstrapcdn.com/bootstrap/3.1.1/js/bootstrap.min.js"></script>

    <!-- JQuery is a very popular library used to provide functionality for many other
    Javascript libraries -->
    <script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
    <script src="http://code.jquery.com/jquery-migrate-1.2.1.min.js"></script>

    <!-- Normalize is used to further ensure your pages look the same on all browsers and devices -->
    <link rel="stylesheet" src="//normalize-css.googlecode.com/svn/trunk/normalize.css" />

</head>
```

We won't really need to change any of this - but we will need to include it in every page we create

Web Servers: To provide a web-page for access we must ‘serve it out’

Web Servers are the cornerstone of the Internet

Luckily there are many free web-servers out there.

APACHE Server is probably the world’s greatest web-server - and it’s free

When we install web-server software on our computer - your computer becomes a web-server

Now - don’t worry - people on the Internet cannot access your computer.

Crucially - **your computer now has a special web address or URL** through which we will test and run all of our web mapping code

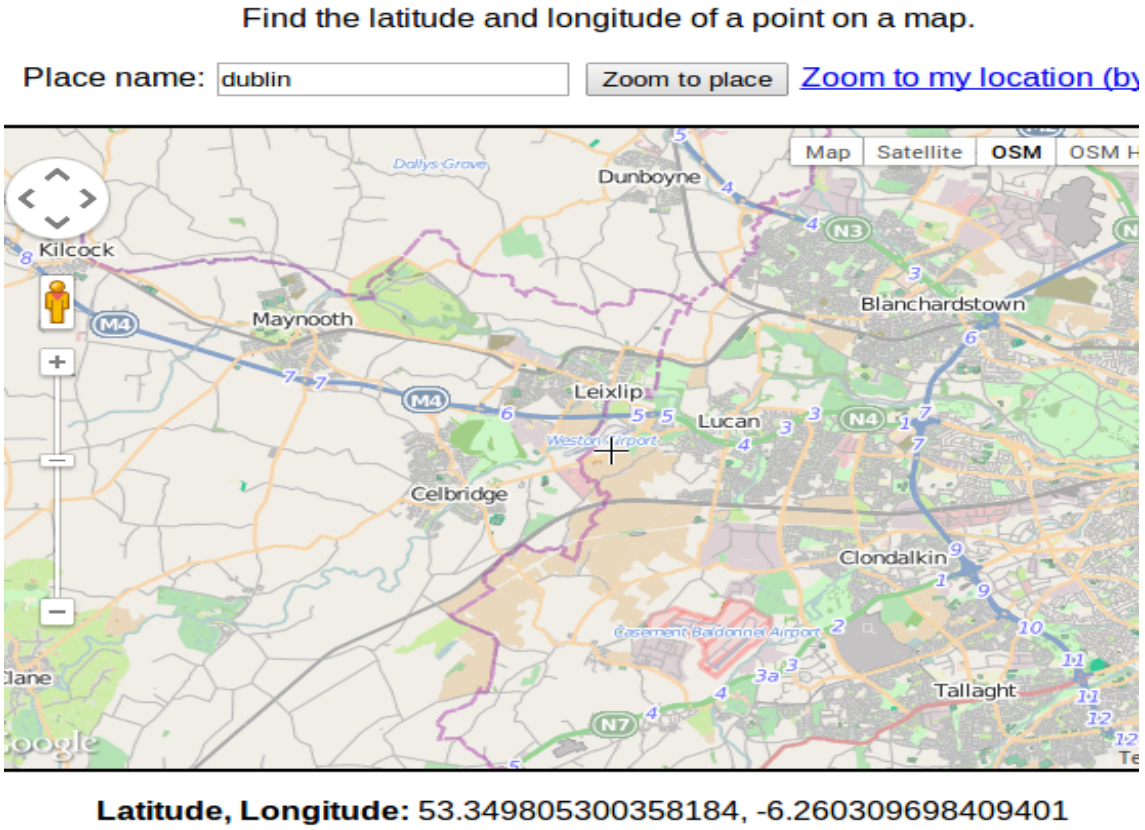
The localhost or 127.0.0.1

On your computers in the lab there is a web-server installed (APACHE XAMPP from http://www.apachefriends.org/)

If you have a file called “1.html” inside the special folder “c:\xampp\htdocs” then the web link http://localhost/1.html is the web address

Finding Lat/Long of any location

http://dbsgeo.com/latlon/



Spatial Coordinates in Leaflet

You always specify coordinates in Leaflet Javascript as

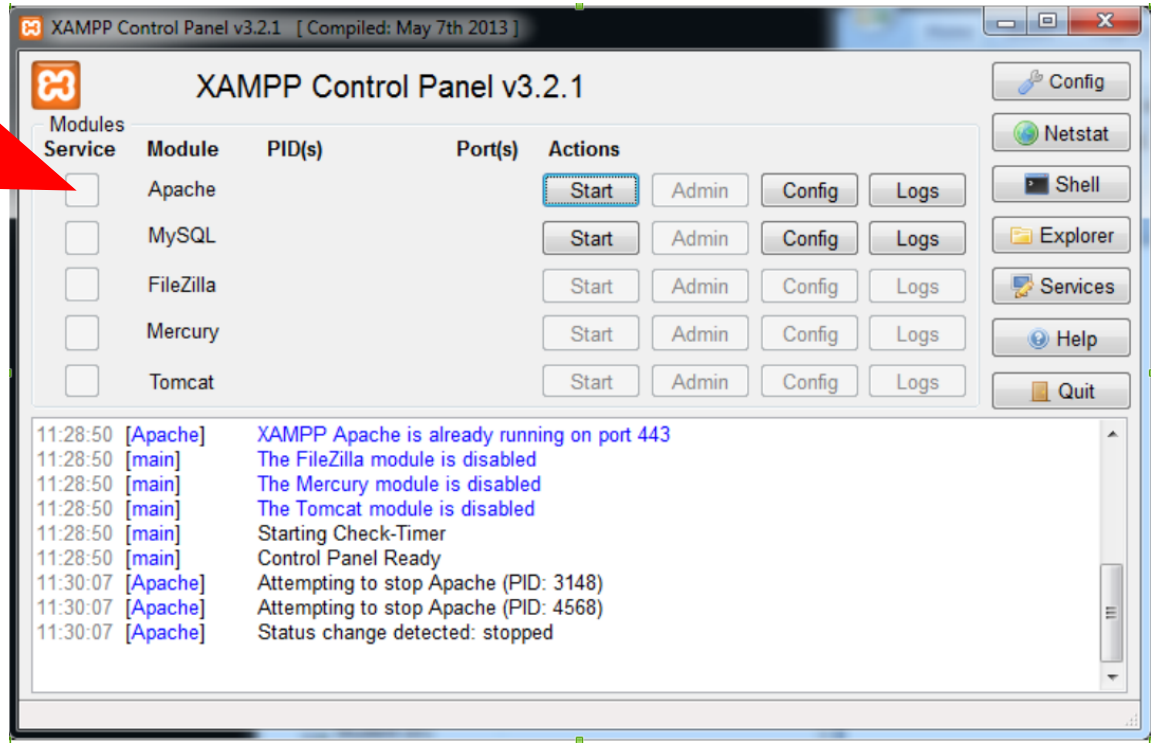
Latitude, Longitude

Otherwise - your spatial data will be rendered on the wrong part of the map!

Getting started - getting the materials required for today…

Let’s start our server

Check this box and click START



Go to the Windows START menu

Go to XAMPP - and open the XAMPP Control Panel

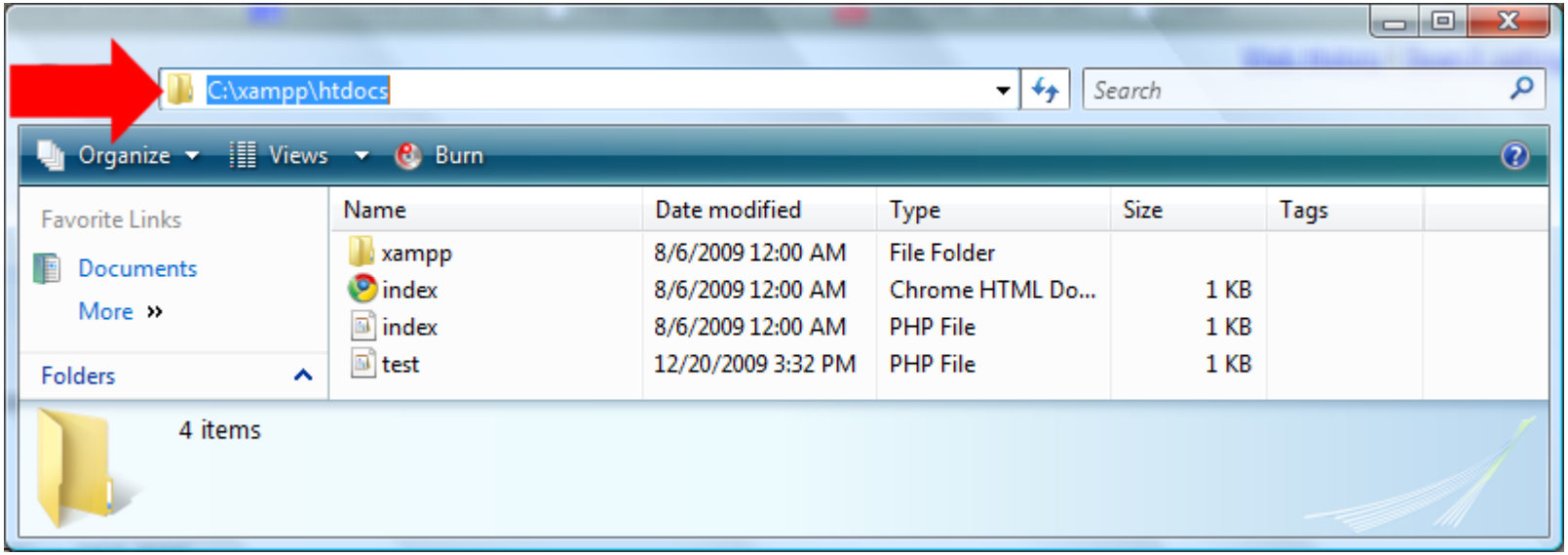
Open up the web browser (Firefox)

Go to

http://localhost

If everything worked you should see this!

Open up the Windows File Manager or Windows Explorer



Go to c:\xampp\htdocs - so it will look something like this here…

c:\xampp\htdocs is our local working directory for web development

We will place all of web pages and data into this folder (or sub folders).

The web server (XAMPP) then makes these files accessible to others via your web-browser.

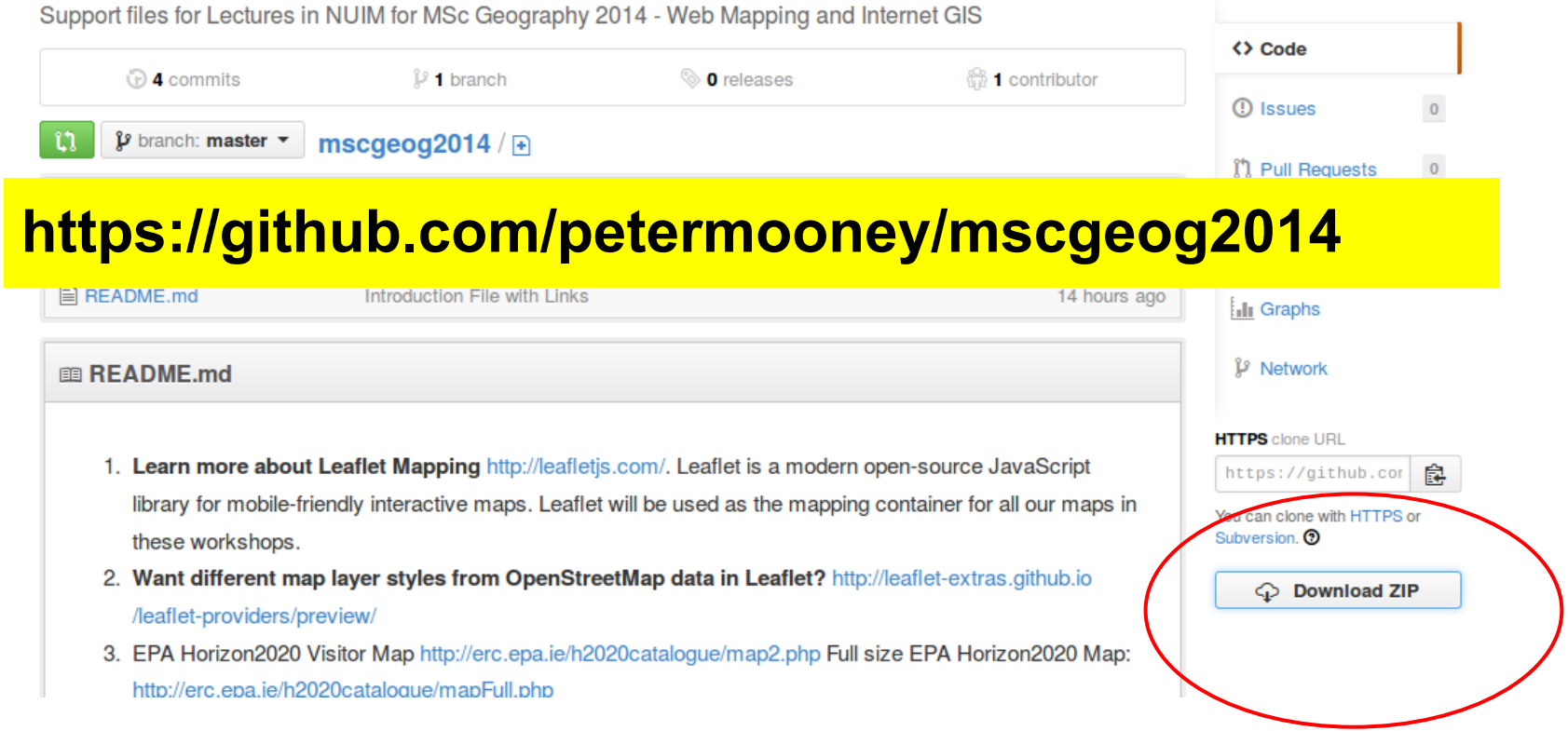
Things to know about the working web directory

It’s a very important folder (and sub folders) on your machine

Normally - you would have a completely separate working folder and copy everything to this (when it’s ready for the web)

There are normally access restrictions to this folder in real web development situations.

Download the workshop materials



Download the ZIP file to the c:\xampp\htdocs folder

Change the name of the ZIP file to '**webmapping**' (no quotes)

Extract the ZIP file to the 'htdocs' folder

Use Windows Explorer to verify you have done it correctly. You should have a folder with no subfolders called c:\xampp\htdocs\webmapping

Let’s start some web mapping

Example 1 (1.html)

Aim: Centering the map and working with the zoom levels

Try http://localhost/webmapping/1.html

TASK: Edit this by changing the map center and map zoom - open in the Geany editor

Zoom levels

Web-based maps zoom from 1 (the whole world) down to 17 or 18 (on street level)

You can easily change this:

```
<!-- this is the center of the map and the zoom level -->
var map = L.map('map').setView([53.34530871461457, -6.2721650622552269], 13);
```

In this example - zoom is 13 - we can change from 1 to 17 or 18 - try it out!

Example 2: (2.html)

Aim: To put pins onto the map and center it accordingly.

Try: http://localhost/webmapping/2.html

Open “2.html” in the Geany and look at the source code of the two pins.

Example 2: (2.html)

The map center - you will need to decide on this yourself.

The map zoom has been chosen appropriately.

```
    <!-- this is the center of the map and the zoom level -->
    var map = L.map('map').setView([53.34530871461457, -6.2721650622552269], 13);

    var MapQuestOpen_OSM = L.tileLayer('http://otile{s}.mqcdn.com/tiles/1.0.0/map/{z}/{x}/{y}.jpeg', {
attribution: 'Tiles Courtesy of <a href="http://www.mapquest.com/">MapQuest</a> &mdash; Map data &copy; <a
href="http://openstreetmap.org">OpenStreetMap</a> contributors, <a
href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>',
subdomains: '1234'
}).addTo(map);

var marker1 = L.marker([53.3527699073999, -6.2474350951379165]).addTo(map);
var marker2 = L.marker([ 53.34550726970357, -6.302763702697121]).addTo(map);
```

Here are the two markers. One for Heuston Station and one for Connolly Station in Dublin

Task - from example 2

Save the file ‘2.html’ as “2a.html”

TASK: Show a map - centered and zoomed appropriately - which shows pins at: The Aviva Stadium, Croke Park Stadium, and the O2 Concert venue.

You can use ‘GetLatLon’ to get coordinates

Example 3: Making popup pins (3.html)

So far the pins on the map don’t really show us much information. In this example we have a popup with the pins.

Try: http://localhost/webmapping/3.html

Look at the source code in Geany

Example 3: Popup Pins Source Code

We've changed the zoom level

```
<script>
    <!-- this is the center of the map and the zoom level -->
    var map = L.map('map').setView([53.34530871461457, -6.2721650622552269], 15);

    var MapQuestOpen_OSM = L.tileLayer('http://otile{s}.mqcdn.com/tiles/1.0.0/map/{z}/{x}/{y}.jpeg', {
attribution: 'Tiles Courtesy of <a href="http://www.mapquest.com/">MapQuest</a> &mdash; Map data &copy; <a
href="http://openstreetmap.org">OpenStreetMap</a> contributors, <a
href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>',
subdomains: '1234'
}).addTo(map);

var museum = L.marker([53.34792828641171, -6.287013771361671]).addTo(map);
var smithfield = L.marker([ 53.34715972457336, -6.2779479048913345]).addTo(map);
var fourcourts = L.marker([ 53.34690353421498, -6.273699285811745]).addTo(map);
var jervis = L.marker([53.34772333794288, -6.265341522521339]).addTo(map);

museum.bindPopup("<b>This is the Museum Luas Stop</b>");
smithfield.bindPopup("<b>This is the Smithfield Luas Stop</b>");
fourcourts.bindPopup("<b>This is the Four Courts Luas Stop</b>");
jervis.bindPopup("<b>This is the Luas stop at Jervis</b>: Get off here for shopping");

</script>
```

Notice the FOUR variables - variable names must not contain spaces - variable names don't have to be real names!

Here is where we have the popup information in each pin

Task - from example 3

Save “3.html” as “3a.html”

TASK - show a map - centered and zoomed appropriately - which shows popup pins with information on your three favourite holiday destinations anywhere in the world. Change the headline text on the web-page also.

Example 4: Polygons

Up to this example - we have concentrated on just having POINTS on the map.

What about if we want to show a polygon?

Let’s look at example “4.html” - how can you display this in the web browser?

Specifying Polygons in Leaflet

```
var polyName = L.polygon([
[x1,y1],
[x2,y2],
[x3,y3],
[x1,y1]
]).addTo(map);
```

x = Latitude, y = Longitude

You must enclose individual points in SQUARE BRACKETS

So you are building a list of points

So you must start with a [and end with a]

You can have as many points as you like

Your first and last point must be the same - otherwise it will be open

The order you list the points in matters!!!

Example 4: The INTEL polygon

```
<script>
    <!-- this is the center of the map and the zoom level -->
    var map = L.map('map').setView([53.37244944078806, -6.518617155379616], 15);

    var MapQuestOpen_OSM = L.tileLayer('http://otile{s}.mqcdn.com/tiles/1.0.0/map/{z}/{x}/{y}.jpeg', {
attribution: 'Mapquest and OSM',    subdomains: '1234'
}).addTo(map);

var INTEL = L.polygon([
    [53.37471631332996, -6.529539110488258],
    [53.37268087414204, -6.519110681838356],
    [53.37009482997908, -6.5080599806969985],
    [53.37287290086236, -6.506171705550514],
    [53.37603481636035, -6.517115118331276],
    [53.3774812824388, -6.516407015151344],
    [53.378543699679156, -6.520548345870338],
    [53.378850899724235, -6.529024126357399],
    [53.37471631332996, -6.529539110488258],
]).addTo(map);

INTEL.bindPopup("<h1>This is the world famous INTEL Campus at Leixlip</h1>. This polygon shows an outline of
the campus area");

</script>
```

Task - from Example 4

Save “4.html” as “4a.html”

TASK - show a map - centered and zoomed appropriately - which shows TWO polygons representing where you would build your two dream homes. Anywhere in the world. Your popup on the polygon should say why! Use GetLatLon to find coordinates

Managing your spatial data

As we have seen now from the previous tasks - it’s easy to place points and polygons in your Javascript code and display a map.

However - this gets a bit messy if you have lots of different objects (points and polygons).

Managing your spatial data (2)

Solution 1:

We are going to look at using an external file.
We will put the objects into this external file.
We will link to it from our web-page.
This makes it easier to manage.

Managing your spatial data (3)

Let's look at "5.html" - let's also look at "mydata.js" - open both of these in Geany

```
<script>
    <!-- this is the center of the map and the zoom level -->
    var map = L.map('map').setView([53.37244944078806, -6.518617155379616], 4);

    var MapQuestOpen_OSM = L.tileLayer('http://otile{s}.mqcdn.com/tiles/1.0.0/map/{z}/{x}/{y}.jpeg', {
attribution: 'Mapquest and OSM',    subdomains: '1234'
}).addTo(map);


</script>
<!-- Here is where we have linked to our external javascript file -->
<!-- in this javascript file we have a few examples of objects -->
<!-- This must appear after the map declaration above -->
<script src = "mydata.js"></script>
```

Here is where we link to the data

Managing your spatial data (4)

What's the advantage of the external file for storage of our spatial objects?

- It makes our web-programming code much easier to read.
- It also means that we can easily add, delete, or change the objects in the file.

Managing your spatial data (5)

Task: Save “5.html” as “5a.html”.

Add three more points (pins with popups) to the “mydata.js” file - these points can come from anywhere in the world.

You can add another polygon if you like - don’t delete any objects that are in the file.

GeoJSON

GeoJSON is a format for encoding a variety of geographic data structures.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

Javascript is not very suitable for storing your spatial data

What if you had hundreds of objects?

What if your objects were changing all the time?

You would have to be editing your Javascript code all the time - which is not a good solution.

GeoJSON is a text-based format for exchanging spatial data on the Internet and between devices

GeoJSON has become very popular over the past number of years.

All serious GIS software can import and export GeoJSON data

Most programming languages can handle GeoJSON natively

Flexibility and portability are the key advantages of using GeoJSON

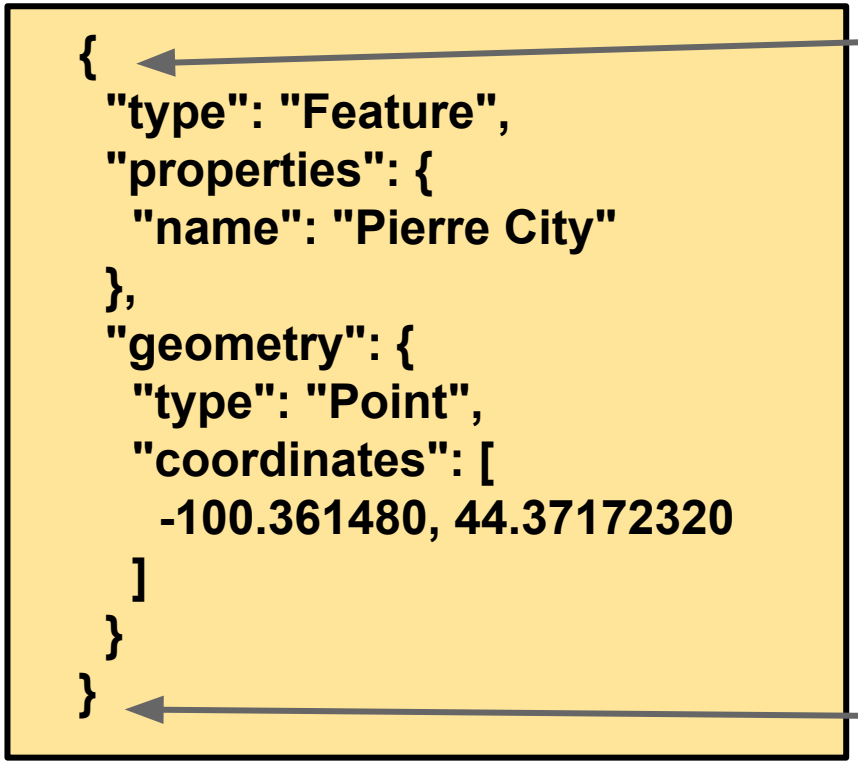
1. **You link to the GeoJSON file** from your web map Javascript
2. **You can change the GeoJSON file** as often as you like (you can even get software to do it automatically)
3. **There are tools available** to manage very big GeoJSON files (small ones can be handled by your favourite text editor)

Geojson example (6.html)

Let's look at **6.html** and **southdakota.json** in the Geany editor.

For simplicity - the json file is in the same directory as your HTML files. But they can be in any directory on your own webserver.

southdakota.json - one object



Start Feature

NAME is one of the properties

The Coordinates: Longitude, Latitude

End Feature

Things to watch for in GeoJSON

JSON is a data format. **You must obey the formatting rules** (syntax) of JSON when you are creating or editing files.

BRACKETS matter

All entities in the file are **Case Sensitive**.

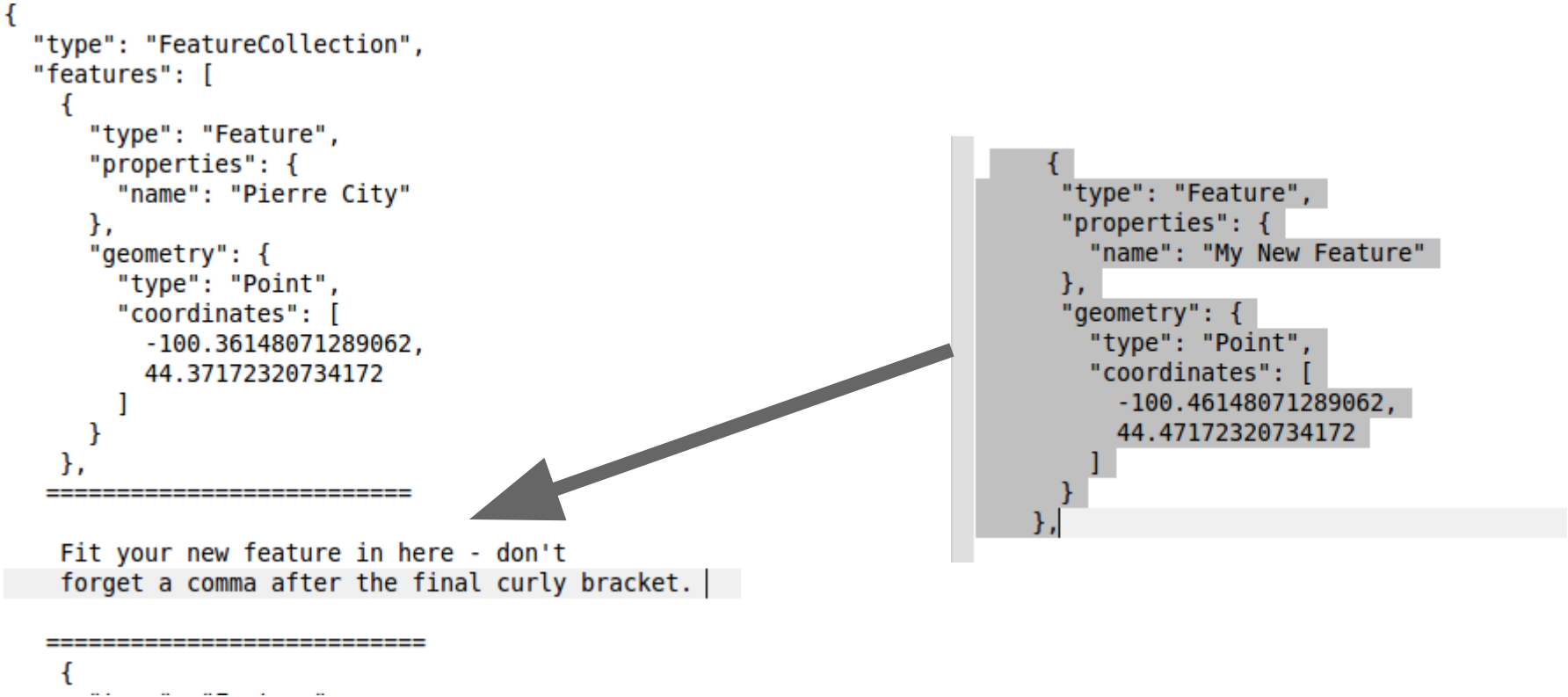
So a simple GeoJSON file has the following structure. It has n features

```
{
  "type": "FeatureCollection",
  "features": [
{feature 1}, {feature 2}, {feature 3}, ….. {feature n -1}, {feature n}
      ]
}
```

Let’s try to edit some geojson

Save 6.html as 6a.html

TASK: Use GetLatLon to find South Dakota. Open southdakota.json in Geany and **add ONE new feature** to the json file. So you can just copy one of the features already there and update the coordinates and name.


{
"type": "FeatureCollection",
"features": [
{
"type": "Feature",
"properties": {
"name": "Pierre City"
},
"geometry": {
"type": "Point",
"coordinates": [
-100.36148071289062,
44.37172320734172
]
}
},
==========
Fit your new feature in here - don't
forget a comma after the final curly bracket.
==========
{
{
"type": "Feature",
"properties": {
"name": "My New Feature"
},
"geometry": {
"type": "Point",
"coordinates": [
-100.46148071289062,
44.47172320734172
]
}
},

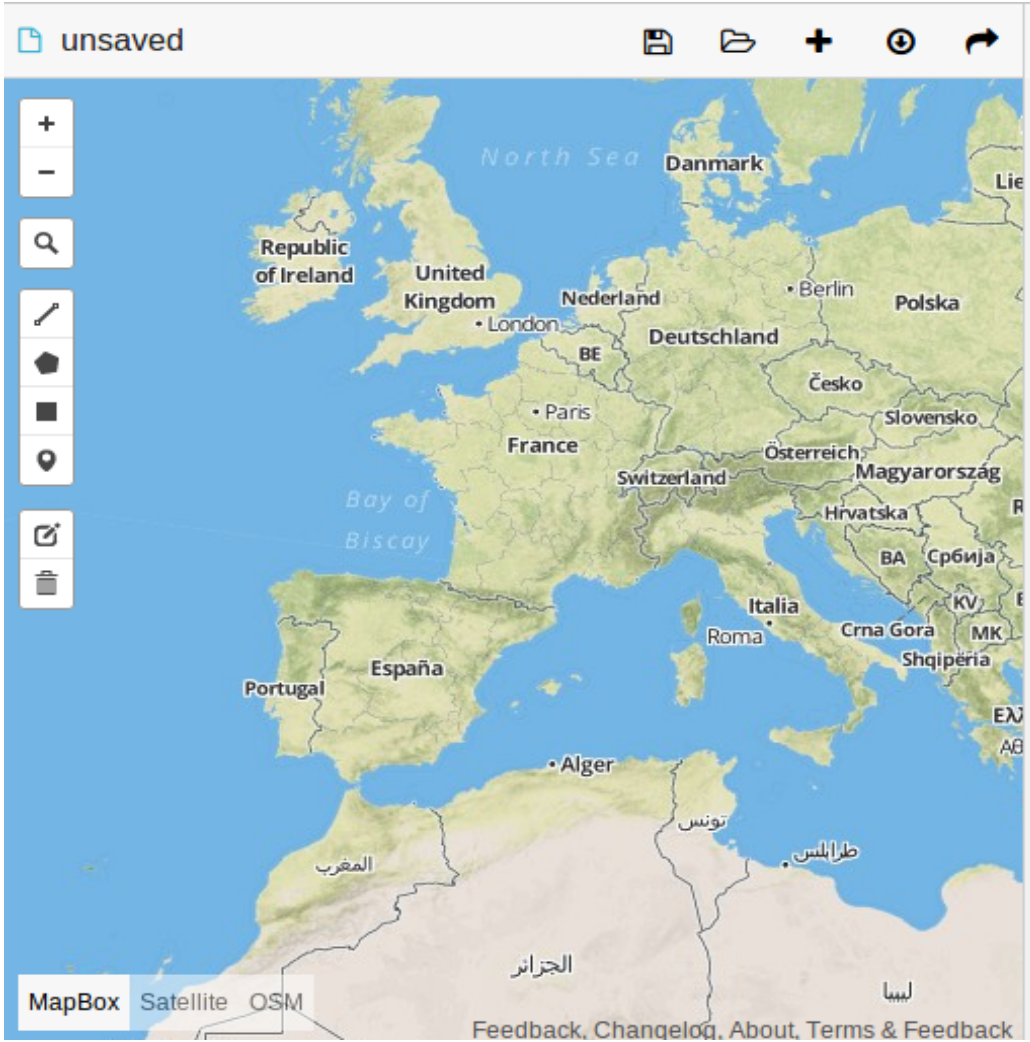
OK - so editing GeoJSON is a bit of a pain using text editors.

Luckily …. there are a few solutions to this.

1. We could use a GIS to go the management of the GeoJSON files for us.

2. We could find some JSON software to manage the files for us.

3. We could use cool applications like **geojson.io** to do the work for us.



no features

geojson.io

DEMO

Feature with ‘name’ properties

GeoJSON Task

TASK: Open “7.html” - this is going to be our map web page.

Use geojson.io to create a GeoJSON file - remember to save this file into c:\xampp\htdocs\webmapping

In the Javascript link to this filename - center and zoom the map and display in browser

For each GeoJSON feature - include a name property!

In reality - we don’t usually make up our own data for map display

If we are programmers or GIS professionals we are usually using someone elses data!

We could write software to generate the GeoJSON for us (EPA H2020 example)

But let’s look at using **OPEN DATA GeoJSON**

Formats
- ATOM
- GeoJSON Summary
- GeoJSON Detail
- KML
- Spreadsheets
- QuakeML

Web Service
- API Documentation
- Search Earthquake Archives

Glossary

Change Log

Feed Life Cycle Policy

GeoJSON Summary Format

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

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

Feeds

Past Hour

Updated every minute.

- Significant Earthquakes
- M4.5+ Earthquakes
- M2.5+ Earthquakes
- M1.0+ Earthquakes
- All Earthquakes

Past Day

Updated every minute.

```
        "type": "Feature",
        "properties": {
            "mag": 4.39,
            "place": "9km NNW of Westwood, California",
            "time": 1395062736900,
            "updated": 1395395897936,
            "tz": -420,
            "url": "http://earthquake.usgs.gov/earthquakes/eventpage/ci15476961",
            "detail": "http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/ci15476961.geojson",
            "sources": ",at,ci,us,",
            "types": ",cap,dyfi,focal-mechanism,general-link,geoserve,impact-link,losspager,moment-tensor,nearby-
ies,origin,phase-data,scitech-link,shakemap,",
            "nst": 209,
            "dmin": 0.0382,
            "rms": 0.34,
            "gap": 39,
            "magType": "mw",
            "type": "earthquake",
            "title": "M 4.4 - 9km NNW of Westwood, California"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [
                -118.4858333,
                34.1346667,
                9.88
            ]
        },
```

usgs.html - using external GeoJSON

Go to the USGS web-page and have a look at the GeoJSON feeds.

Have a look at usgs.html in the browser.

TASK: Save usgs.html as “8.html”. Download the GeoJSON for **ALL earthquakes from the past seven days**. Center map over most active region

Tomorrow’s workshop

We will continue to explore web mapping with Leaflet

1: Changing the background layer

2: Adding GeoJSON feeds from other locations

3: Adding LAYERS so that people can switch between views

4. Does this look good on my smartphone?

Day 2

Day 1 Recap

We covered a lot of material

1. How to display a map
2. How to place Javascript Markers and Polygons on the Map

1. How to create GeoJSON data
2. How to link GeoJSON data to our maps
3. Advantages of using GeoJSON

Go to the GitHUB page

Download as ZIP file - download to the C:\xampp\htdocs folder

Rename the zip file as “webday2” - and then unzip the file.

So all our weblinks today will be http://localhost/webday2

Some assumptions for today

- You - the web developer - are responsible for picking and setting a suitable center and zoom level for every map we create in today's workshop.
- We must use Chrome or Firefox
- To view HTML files - we must view via http://localhost in your browser

Let’s look at changing the background layers of the map

Up to now we have focussed on putting pins and polygons on the base-layer map.

There are many situations where it is useful to change the background layer. This concept is very common in desktop GIS.

Limitations

There are some limitations

Leaflet can only handle certain maps from certain providers - while there is still a huge variety of maps - we can only add layers which are compatible with Leaflet

As with desktop GIS there are two types of layers

There are **BACKGROUND** or **BASE** layers and then there are **OVERLAYS**

Generally in web mapping BASE layers are from sources such as Bing, Google, OpenStreetMap, etc. OVERLAYS are usually things like GeoJSON and KML

Let’s look at “layers.html”

As before let’s open it up in The Geany and look at it in the web-browser

Let’s concentrate on how the code is allowing us to have two different base layers

```
<script>
    <!-- We must list out the layers which we are going to use as background maps -->
    <!-- Copied directly from http://leaflet-extras.github.io/leaflet-providers/preview/ -->

    var MapQuestOpen_OSM = L.tileLayer('http://otile{s}.mqcdn.com/tiles/1.0.0/map/{z}/{x}/{y}.jpeg', {
    attribution: 'Mapquest and OSM',    subdomains: '1234'  });

    var Esri_WorldShadedRelief = L.tileLayer(
    'http://server.arcgisonline.com/ArcGIS/rest/services/World_Shaded_Relief/MapServer/tile/{z}/{y}/{x}', {
    attribution: 'Tiles &copy; Esri &mdash; Source: Esri',  maxZoom: 13     });

    <!-- Our map initialisation code is a little different -->
    <!-- Notice how we add the layer variables from above -->
    <!-- The layer variables are added to a list - comma separated list

    var map = L.map('map', {
        center: new L.LatLng(54.7887878, -7.9987623),
        zoom: 5,
        layers: [MapQuestOpen_OSM, Esri_WorldShadedRelief]
        });

    <!-- now declare the base layer maps variable -->
    var baseLayers = {"MapQuest": MapQuestOpen_OSM, "ESRI Relief": Esri_WorldShadedRelief};

    <!-- We must now get Leaflet to give us a layer switcher icon -->
    L.control.layers(baseLayers).addTo(map);

    </script>
```

Working with layers

Choosing different layers

Go to http://leaflet-extras.github.io/leaflet-providers/preview/

Which has lots of cool layers - you will only need to copy the ‘Plain Javascript’ into your code.

Task: Adding extra base layers

Save “layers.html” as “layers1a.html”

Go to the leaflet providers page - and add a 3rd baselayer to the layers page - so we should have the choice of three base-layers

Task 2: Adding extra base layers with pins or markers

We have just created a map with three base layers in the previous example.

TASK: Can you add two or three markers/pins (with popups) to this code? This will help us visualise the change in base-layers easier. .. Hint.. look at some code from yesterday for your markers/pins code (just copy)...

What about adding Google Maps?

Adding Google Layers requires a few more lines of code

These lines of code basically are required by Google and Leaflet to communicate to each other to ensure the maps are delivered correctly

```
<!-- to use google maps -->
<script src="http://maps.google.com/maps/api/js?v=3.2&sensor=false"></script>
<script src="leaflet-google.js"></script>
```

Open “layersGoogle.html” in Geany and Browser - this piece of code is added to the <HEAD> of the web-page. You don’t need to edit it at all. You will need this in every HTML file in which you want to add Google Map layers to

Look at the source code in Geany of “layersGoogle.html”

```
var googleLayer = new L.Google('ROADMAP');
var googleHybrid = new L.Google('HYBRID');
var googleSatellite = new L.Google('SATELLITE');
<!-- Our map initialisation code is a little different -->
<!-- Notice how we add the layer variables from above -->
<!-- The lauer variables are added to a list - comma separated list

var map = L.map('map', {
    center: new L.LatLng(54.7887878, -7.9987623),
    zoom: 5,
    layers: [MapQuestOpen_OSM, Esri_WorldShadedRelief,googleLayer]
    });

<!-- now declare the base layer maps variable -->
var baseLayers = {"MapQuest": MapQuestOpen_OSM, "ESRI Relief": Esri_WorldShadedRelief, "Google Roads": googleLayer
, "Google Hybrid": googleHybrid, "Google Satellite": googleSatellite};

<!-- We must now get Leaflet to give us a layer switcher icon -->
L.control.layers(baseLayers).addTo(map);
```

It’s very straightforward to add the Google Map layers.

In “layersGoogle.html” - can you change the layer order?

What if I wanted the Google Roads layer to appear first when someone opened up the webpage?

How do I get it to be the first layer displayed with all others needing to be switched on by the user?

Adding overlay layers

We've worked with some overlay layers already

All of our GeoJSON files can be considered overlay layers

They contain a small set of data and for those layers to be useful - they must be placed on top of a base-layer

We can have multiple base-layers and have multiple overlays

We must always specify our base-layers first and then we can specify our overlays.

This is the order we must follow in our Javascript

Let’s look at “**multiple.html**”

We have two GeoJSON files

One represents Forests in Vermont USA while the other represents Buildings in Vermont USA

Both are taken from OpenStreetMap

Our example will use these two GeoJSON files as overlays to our base maps

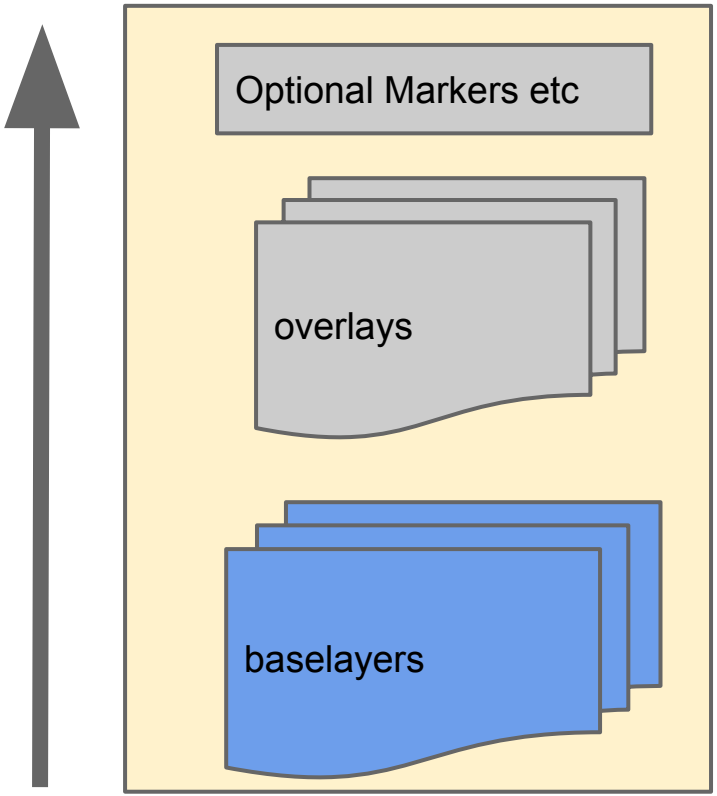
```
<!-- now we can add over-lay layers -->
<!-- add an over-lay layer of GeoJSON -->
var vermontNatural = L.geoJson(null, {
      onEachFeature: function (feature, layer) {
           layer.bindPopup(feature.properties.name);
      }
 });

$.getJSON("vermont_natural.geojson", function (data) {
vermontNatural.addData(data);
});

<!-- add another over-lay layer of GeoJSON -->
var vermontBuildings = L.geoJson(null, {
      onEachFeature: function (feature, layer) {
           layer.bindPopup(feature.properties.name);
      }
 });
$.getJSON("vermont_buildings.geojson", function (data) {
vermontBuildings.addData(data);
});

<!-- now declare the base layer maps variable -->
var baseLayers = {"ESRI Relief": Esri_WorldShadedRelief,"MapQuest": MapQuestOpen_OSM};

<!-- now declare the over lay map variables -->

var overlays = {"Vermont Natural": vermontNatural,"Vermont Buildings": vermontBuildings};

<!-- We must now get Leaflet to give us a layer switcher icon -->
L.control.layers(baseLayers,overlays).addTo(map);
```

This is use for one GeoJSON file - we have referred to it as *vermontNatural* - that is the variable name

This is the other GeoJSON file - we have called the variable *vermontBuildings*

Notice how we have added an overlays variable

This is how Leaflet thinks about layers in the Javascript Code



The large outer box represents the Layer Control which Leaflet uses to give you the option to switch layers.

Try to understand exactly how the layering is working in the example

“**multiple.html**” is close to an Internet GIS

We have two base layers and then we have two overlay layers

The overlay layers can be switched on or off.

The base-layer can be switched.

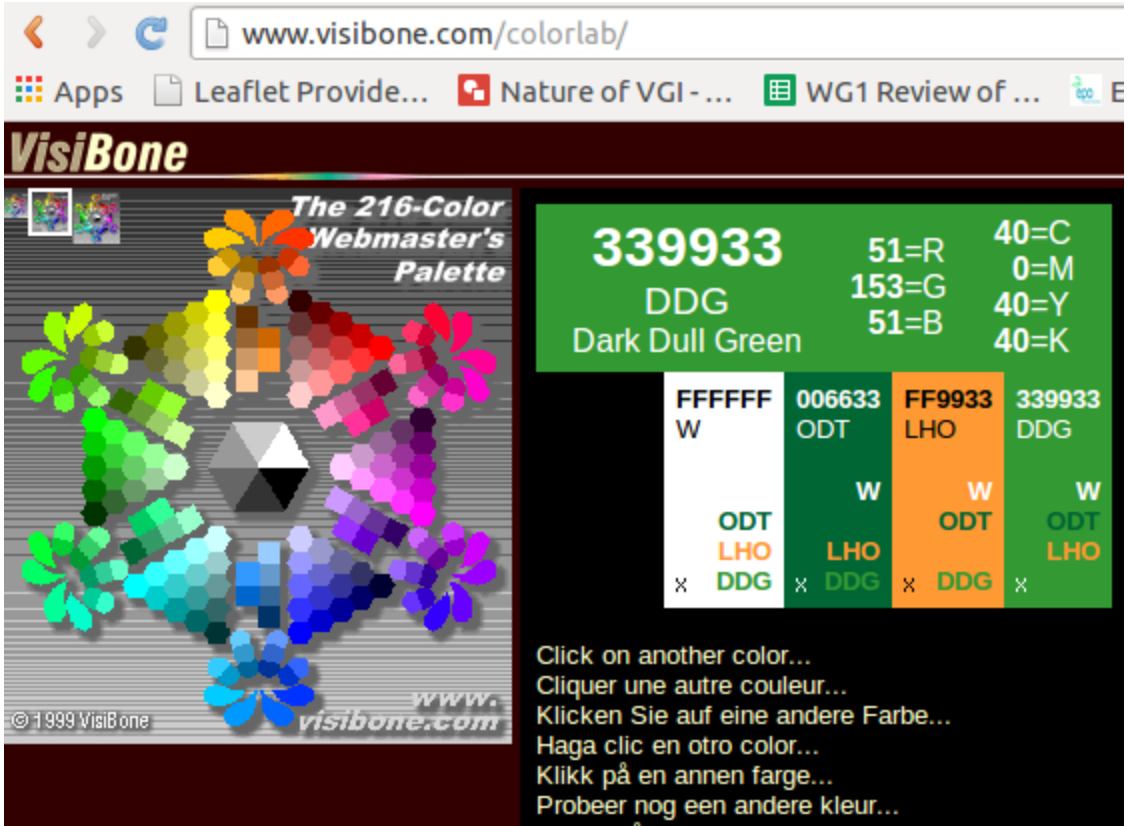
What's with the weird numbers beside color in the GeoJSON layer?

In web development colors are specified most accurately in a notation called HEXIDECIMAL.

```
<!-- now we can add over-lay layers -->
<!-- add an over-lay layer of GeoJSON -->
var vermontNatural = L.geoJson(null, {
    style: {"color": "#F339933"},
      onEachFeature: function (feature, layer) {
          layer.bindPopup(feature.properties.name);
      }
 });

$.getJSON("vermont_natural.geojson", function (data) {
vermontNatural.addData(data);
});
```

All web-browsers and web-software understand HEX and what colors they represent

Yet again - there are tools available which also help us choose HEX colors



Before we move on….

Try changing the colors of the two Vermont layers in the “multiple.html” page

Remember HEX is a 6 character code - and it must always start with a # symbol

So the structure of your Javascript code will be

1. Declare your baselayers (you can have as many as your wish)
2. Declare your map (center and zoom)
3. Declare each of your overlay layers individually
4. Then you must finally setup the layer controls
5. Save and view the map

Working with layers

Next we are going to build our own multi-layer web-page

I have put four GeoJSON files in your download ZIP file: “ireland_pubs.geojson”, “ireland_restaurants.geojson”, “ireland_forests.geojson”, “ireland_lakes.geojson”

TASK: An Irish Layers Map

This is one of our last task assignments.

What tools have we got at our disposal?

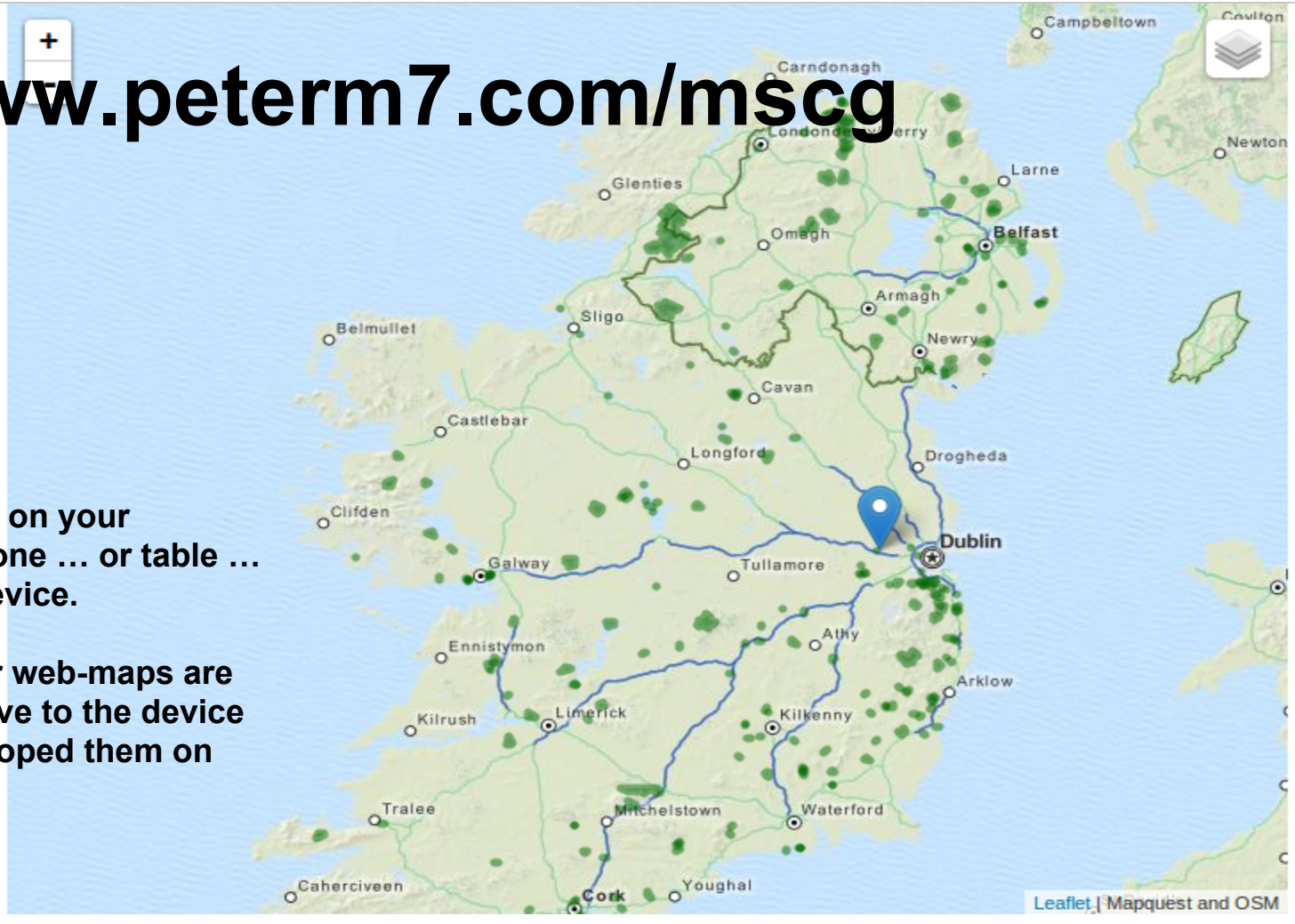
We have got lots of GeoJSON files for Ireland.
We can have many background layers
We can put pins and markers on the maps.

TASK: An Irish Layers Map

Using “multiple.html” as the starting point deliver the following webpage to these specifications.

1. A choice of three base layers - including ONE which we have not used already
2. A choice of three GeoJSON files for Ireland - choose a suitable color for the polygon layers (if used)
3. A popup marker at Maynooth
4. The map centered and zoomed appropriately
5. Some text at the top of the page (under the <h1> tags to explain what the map is about.

www.peterm7.com/mscg

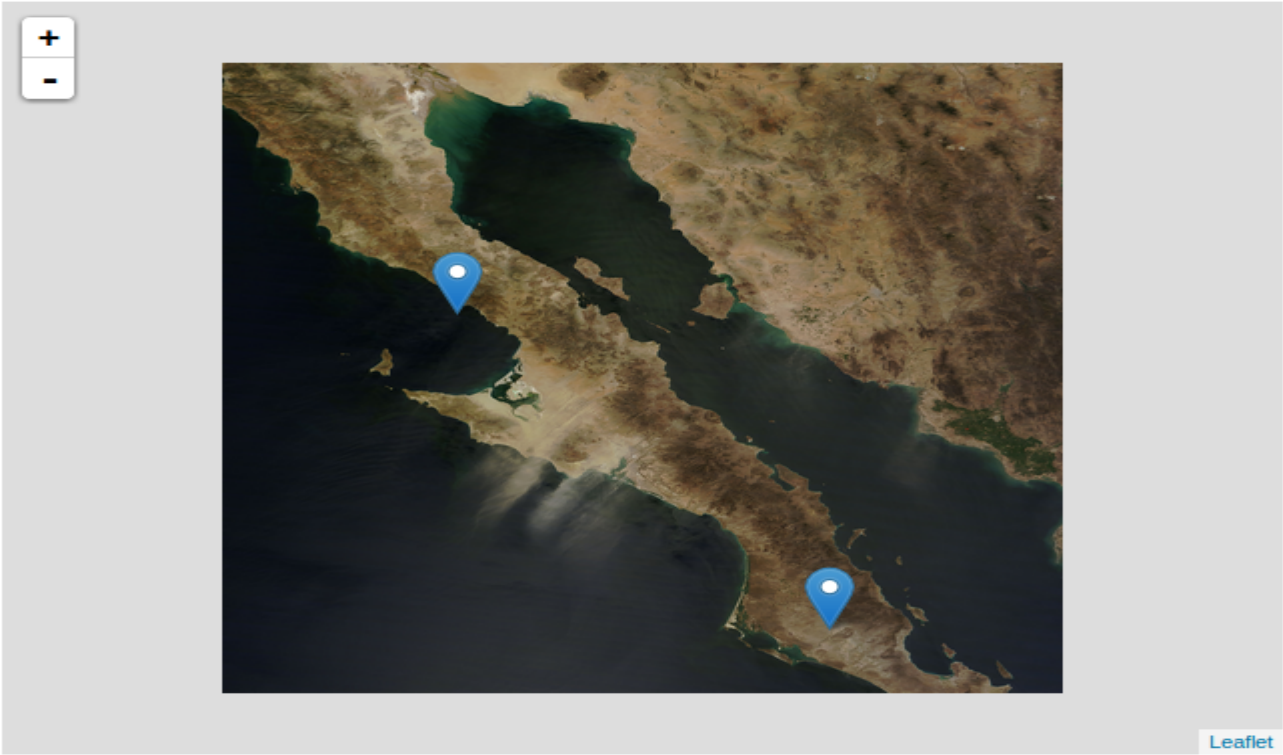


Try it out on your smartphone … or table … or any device.

All of our web-maps are responsive to the device we developed them on

Finally … “image.html”

Using Imagery



On GitHUB

By this evening the GitHUB repository will be updated and include lecture notes and all files. It will include the online example which we looked at on our smartphones.

You can use this code as a basis for your own work.

Just install XAMPP at home - use a text editor - and away you go!

Contact me

If you have any burning questions about the workshop content you can contact me at: peter.mooney@nuim.ie

Where do I go next?

We’ve managed to develop some very nice web applications over the past two days without doing any real programming.

To begin to do some really interesting things (interaction, dynamic display, databases) you will need to start programming.

Consider Python?

http://www.afterhoursprogramming.com/tutorial/Python/Overview/

Learn Python The Hard Way, 3rd Edition

Welcome to the 3rd Edition of Learn Python the Hard Way. You can visit the companion site to the book at http://learnpythonthehardway.org/ where you can purchase digital downloads and paper versions of the book. The free HTML version of the book is available at http://learnpythonthehardway.org/book/.

Table Of Contents

How about PHP?

http://www.w3schools.com/PHP/

Let’s finish by looking at “programming.php” in the browser and in the Geany

Thanks for listening!!