

CS621B/C Spatial Databases

Web-based Mapping

December 11th 2014

Dr. Peter Mooney
peter.mooney@nuim.ie

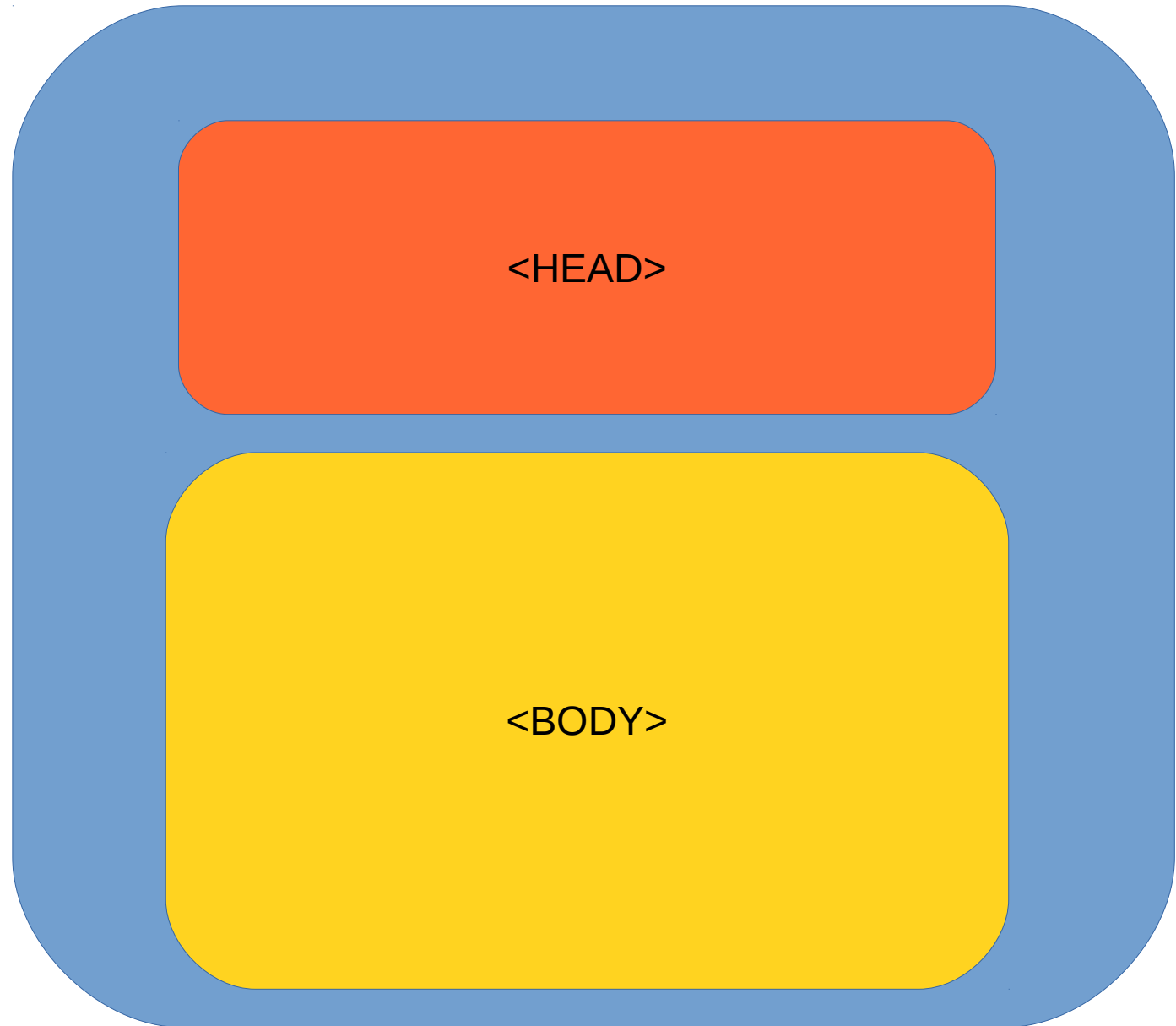
Agenda for today

- To go from a blank web-page to a web-page which includes an interactive web-based map
- Examples driven lectures

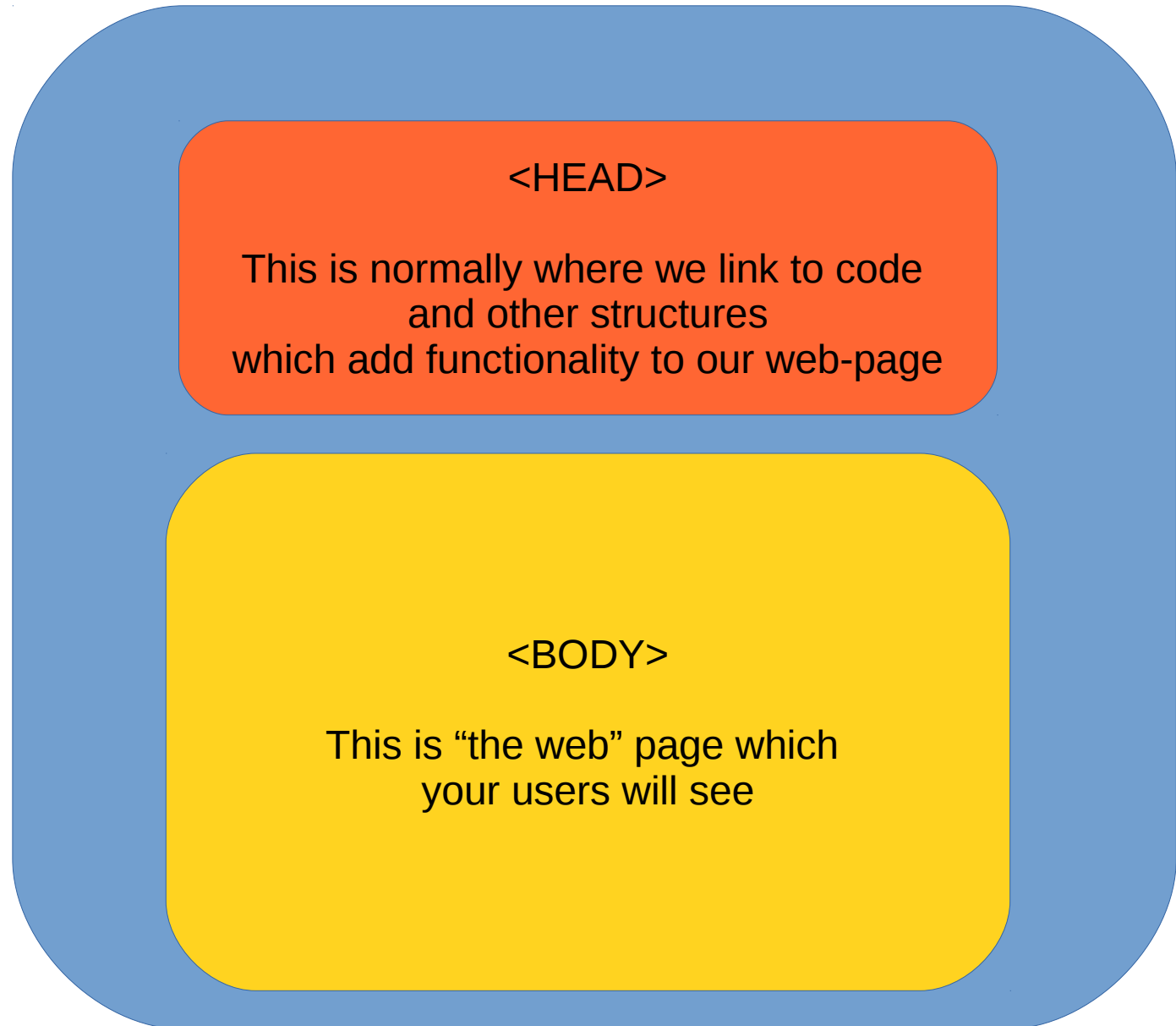
What these lectures WONT do!

- We are not trying to build high-end web-pages and web-sites
- We are not going to be covering complex interactions including client-server interactions
-
- **We are trying to understand the processes around displaying spatial data on a map on a web-page.**

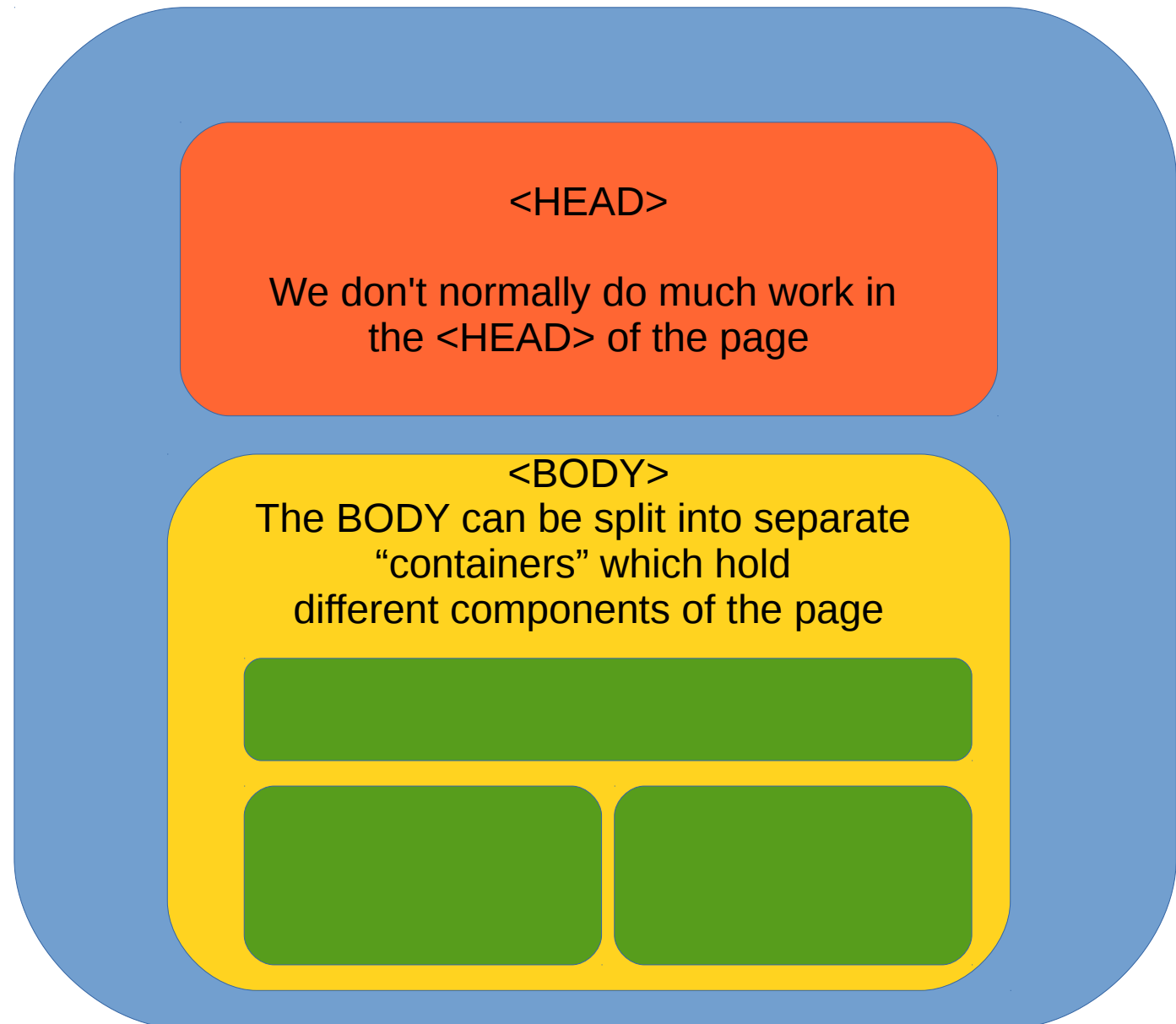
Understanding Web Page Structure



Understanding Web Page Structure



Understanding Web Page Structure



We will use Twitter Bootstrap to help us develop our web-pages

[Bootstrap](#)[Getting started](#)[CSS](#)[Components](#)[JavaScript](#)[Customize](#)[Expo](#)[Blog](#)

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

[Download Bootstrap](#)

Currently v3.3.1

I've downloaded BOOTSTRAP for you already to use today

- Inside the folder(s) you have downloaded you will notice a folder called “bootstrap”.
- Inside this folder – there are 3 sub folders (CSS, Fonts, JS)
- We will need the files inside these folders to build our web-pages today.
- In fact we will need additional files to include our mapping functionality also.

What's with the CSS, Fonts and JS folders?

- We don't really need to go into much details about what these files are today
- But
- CSS: This is Cascading Style Sheet and it controls the “look and feel” of your web-page
- Fonts: well ... these helps with Bootstrap fonts
- JS: For Javascript – this is required to make some functionality provided by Bootstrap work

Encapsulation!

- The idea behind using a framework such as BOOTSTRAP is that – YOU DON'T WORRY ABOUT CSS, FONTS and JAVASCRIPT
- The murky details of these are hidden or encapsulated away!
- We will have to work with Javascript later – but that is when we want to add our own functionality which is additional to what Bootstrap gives us

We will need to link to the Bootstrap CSS and JS on every page

- This will be done in the <HEAD> section of the webpage
- Without this link we will not have a nice responsive web-page.

Bootstrap is “Mobile First”

- What does this mean?
- Bootstrap has been developed by the people at Twitter to be “responsive across all devices”
- This means that if you use Bootstrap (and use it correctly) – then your webpage(s) should look great from small smartphone screens right up to huge screen displays!

After each example – check out the webpage you create on your PC and your smartphone/tablet



JQuery

[Plugins](#)[Contribute](#)[Events](#)[Support](#)[jQuery Foundation](#)

JQUERY UK 2015

OXFORD, UK
6 MARCH

The UK's largest front-end developer conference

[TICKETS](#)[Download](#) [API Documentation](#) [Blog](#) [Plugins](#) [Browser Support](#)

Lightweight Footprint

Only 32kB minified and gzipped. Can also be included as an AMD module



CSS3 Compliant

Supports CSS3 selectors to find elements as well as in style property manipulation



Cross-Browser

IE, Firefox, Safari, Opera, Chrome, and more



Download jQuery

v1.11.1 or v2.1.1

[View Source on GitHub →](#)[How jQuery Works →](#)

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

Resources


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

What is JQuery?


- JQuery is one of the most useful Javascript libraries on the Internet today
- You'll see it inside the JQuery folder in the todays download
- Like Bootstrap – there is no need to look at it.
- But Bootstrap needs JQuery AND our mapping application will also need it.
- Suffice to say – JQuery is very important


Let's start building some pages


GITHUB Page for today's lectures


 This repository Search


Explore Gist Blog Help

 petermooney + ▾







 petermooney / December2014

Unwatch ▾ 1


★ Star 0


🍴 Fork 0

https://github.com/petermooney/December2014



MSc Lectures December 2014 — Edit










4 commits 1 branch 0 releases 1 contributor

 branch: master ▾ December2014 / +



Initial Setup of web-pages and Bootstrap files locally

 petermooney authored 2 minutes ago latest commit 709e804cc7 

 bootstrap	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
 images	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
 jquery	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
 1.html	Test 123	3 hours ago
 1.html~	Test 123	3 hours ago
 LICENSE	Initial commit	3 hours ago
 LectureNotes.odp	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
 README.md	Initial commit	3 hours ago
 example1.html	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago

Code

Issues 0

Pull Requests 0


Wiki


Pulse


Graphs

Settings

HTTPS clone URL

https://github.com 

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#). 

 Download ZIP

You can download as ZIP

The screenshot shows the GitHub interface for the repository 'petermooney / December2014'. The repository has 4 commits, 1 branch, 0 releases, and 1 contributor. The 'Code' tab is selected, showing a list of files and folders. A yellow box highlights the instruction: 'Download as ZIP – just unzip to any location on your computer. Do not change or move any of the folders within the extracted zip file'. The 'Download ZIP' button is circled in red.

https://github.com/petermooney/December2014

MSc Lectures December 2014 — Edit

4 commits 1 branch 0 releases 1 contributor

branch: master December2014 / +

Initial Setup of web-pages and Bootstrap files locally

petermooney authored 2 minutes ago latest commit 709e804cc7

bootstrap	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
images	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
jquery	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
LectureNotes.odp	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
README.md	Initial commit	3 hours ago
example1.html	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago

Download as ZIP – just unzip to any location on your computer. Do not change or move any of the folders within the extracted zip file

Issues 0 Pull Requests 0 Wiki

Pulse Graphs Settings

HTTPS clone URL

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Download ZIP

You can always just do `git clone`

The screenshot shows the GitHub interface for the repository 'petermooney / December2014'. The repository has 4 commits, 1 branch, 0 releases, and 1 contributor. The main branch is 'master'. The commit history shows a recent commit by 'petermooney' titled 'Initial Setup of web-pages and Bootstrap files locally' with files 'bootstrap', 'images', and 'jquery'. The commit message is 'Initial Setup of web-pages and Bootstrap files locally'. The commit hash is '709e804cc7'. The commit was made 2 minutes ago. The repository also has a 'Code' button, 'Issues' (0), 'Pull Requests' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'.

https://github.com/petermooney/December2014

MSc Lectures December 2014 — Edit

4 commits 1 branch 0 releases 1 contributor

branch: master December2014 / +

Initial Setup of web-pages and Bootstrap files locally

petermooney authored 2 minutes ago latest commit 709e804cc7

File	Commit Message	Time
bootstrap	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
images	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago
jquery	Initial Setup of web-pages and Bootstrap files locally	2 minutes ago

example1.html Initial Setup of web-pages and Bootstrap files locally 2 minutes ago

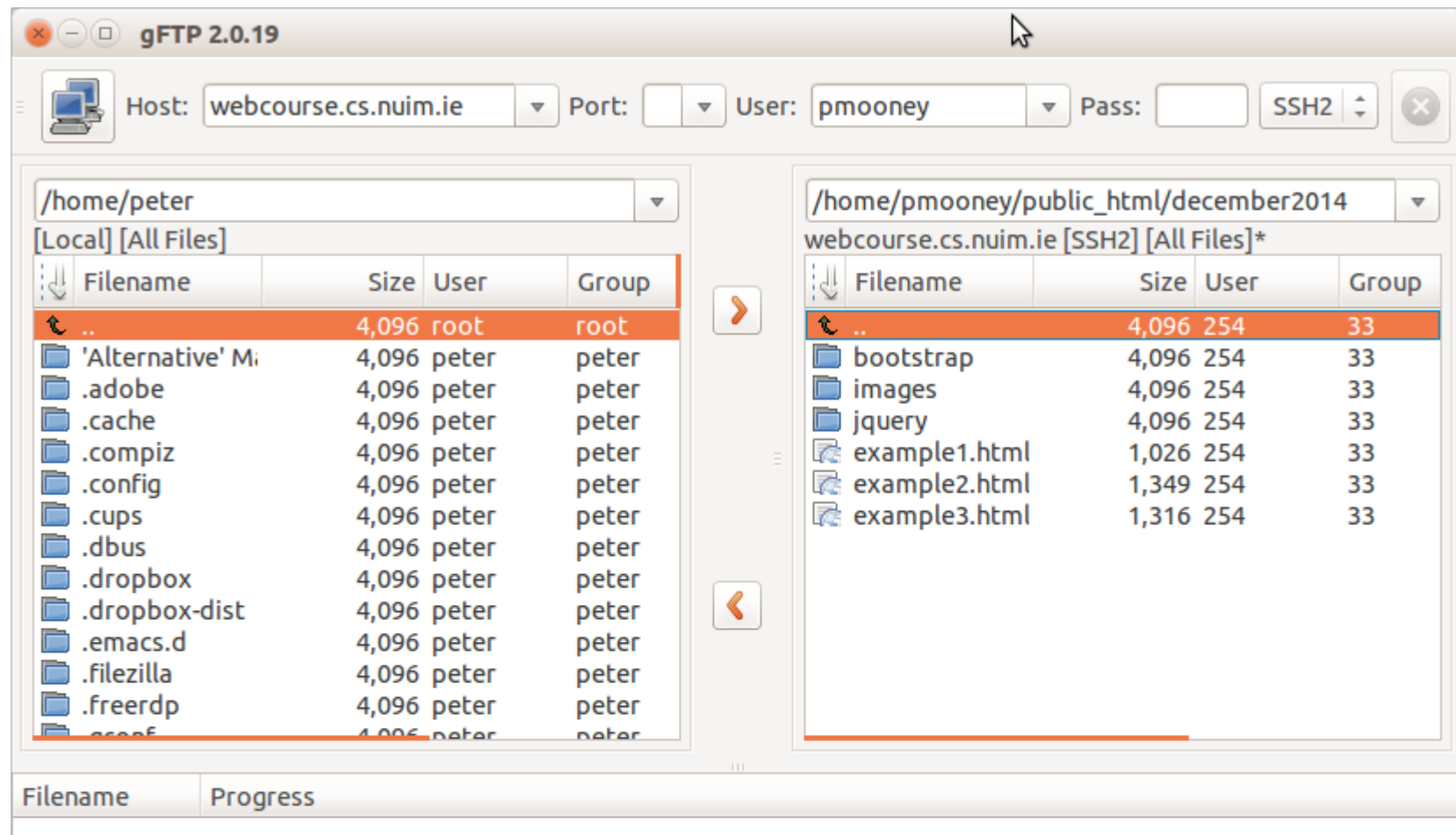
HTTPS clone URL
https://github.com

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

[Download ZIP](#)

Open a command/terminal prompt

```
git clone https://github.com/petermooney/December2014.git
```



Before we start – let's upload ALL of the folders you see in your download folder to your WEBCOURSE.CS.NUIM.IE account

```

12: Open Directory /home/pmooney/public_html/december2014
12: File handle
13: Read Directory
13: Filenames (8 entries)
14: Read Directory
14: EOF
15: Close
15: OK

```

What to do next.

- Make a new folder (you decide the name) inside the public_html folder of your web-course account.
- Your base url will then be
- <https://webcourse.cs.nuim.ie/~youname/yourfoldername>
-
- This is the folder we will use for the rest of the lectures today

All web-pages will have the same code structure today

```
<html>  
<head>  
  <title> Your page title </title>  
  Your links to CSS and JS go here  
</head>  
  <body>  
    Your webpage goes here  
  </body>  
</html>
```


Example1.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- This bit here is for the web-browser - we won't be changing it -->
    <meta charset="utf-8"><meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Example 1</title>

    <!-- Bootstrap --> <!-- We shouldn't need to change these very much
    over the course of all of the examples -->
    <!-- This is where we link directly to the Bootstrap code -->
    <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
    <script src="bootstrap/js/bootstrap.min.js"></script>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="jquery/jquery-1.11.1.min.js"></script>

  </head>
  <body>

    <div class="container">
      <div class="row">

        <h1>Hello, everyone!</h1>
        <p>
          This is the first example of a working web-page in these lectures.
        </p>

        <img class = "img-responsive" src = "../images/minions.png"/>
      </div>
    </div>

  </body>
</html>
```

<https://webcourse.cs.nuim.ie/~YOURNAME/YOURFOLDER/example1.html>

Hello, everyone!

This is the first example of a working web-page in these lectures.



Trying some HTML tags

- Let's try to use some basic HTML tags to 'MARKUP' our text
- `<h1>`This is a big heading`</h1>`
- `<h2>`Second biggest heading`</h2>`
- `<h3>`, `<h4>` and `<h5>`
- `<p>` starts a paragraph which ends with `</p>`
- `` marks bold text``
- `` marks italic text ``

Open up Example2.html and upload to your webcourse folder

```
<div class="container">
<div class="row">

  <!-- some examples of different HTML tags -->
  <h1>Using HTML tags and Bootstrap</h1>
  <p>
This is a paragraph
</p>

  <h2>This is a big heading, but not the biggest</h2>
  <p>
This is <strong>some bold text</strong> within a paragraph
</p>
  <p class = "lead">
This is nice big text for a paragraph from bootstrap. You can have other
tags in here like <strong>bold text</strong> or <em>italic</em>
text. We can write text as <code>computer code font</code> if we want
</p>

</div>
</div>
```

Open up Example3.html and upload to your webcourse folder

```
<div class="container">
<div class="row">

  <!-- some examples of different HTML tags -->
  <h1>We can do some nice markup with lists</h1>

  <p>
    <strong>We can have normal item lists</strong>
    <ul>
      <li>First item</li>
      <li>Second item</li>
      <li>And so on....</li>
    </ul>

    or we can have <strong>Ordered Numerical Lists</strong>

    <ol>
      <li>First item with numerical order</li>
      <li>Second item with numerical order</li>
      <li>and so on for other items</li>
    </ol>

  </p>
```

We can do some nice markup with lists

We can have normal item lists

- First item
- Second item
- And so on....

or we can have **Ordered Numerical Lists**

1. First item with numerical order
2. Second item with numerical order
3. and so on for other items

Task 1

- Create a file called Task1.html – you should copy Example1, example2 or example3 and change their filename – as we need all of the content in the page `<head>` and `<body>` structure to be the same.
- **Create a simple webpage – showing heading tags, paragraphs, lists to outline the your module list, your favourite web apps, anything..**
- Upload and display on your webcourse folder

Starting mapping

- Let's get drawing some maps.
- We are going to use the structure of the previous examples as a basis for our maps
- To include maps in a web-page we need (1) a container (div) to hold the map and (2) javascript to display and control the map

Leaflet



9,193



11.2K followers



5.1k

An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps

Overview

Features

Tutorials

API

Download

Plugins

Blog



GitHub



Twitter

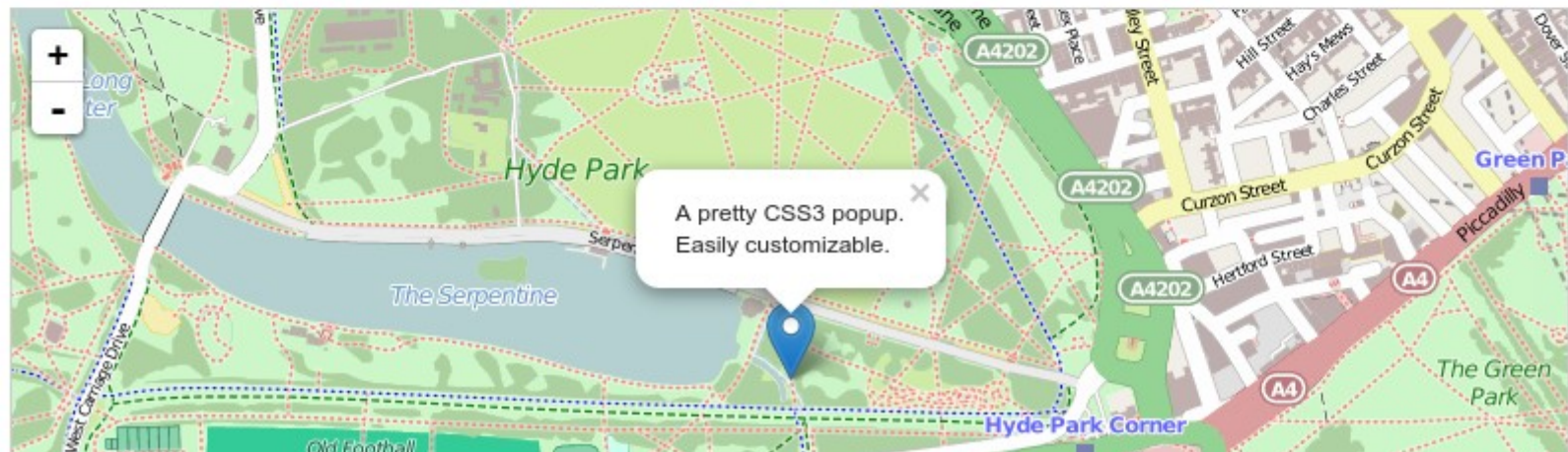


Forum

Leaflet is a modern open-source JavaScript library for mobile-friendly interactive maps. It is developed by [Vladimir Agafonkin](#) with a team of dedicated [contributors](#). Weighing just about 33 KB of JS, it has all the [features](#) most developers ever need for online maps.

Leaflet is designed with *simplicity*, *performance* and *usability* in mind. It works efficiently across all major desktop and mobile platforms out of the box, taking advantage of HTML5 and CSS3 on modern browsers while still being accessible on older ones. It can be extended with a huge amount of [plugins](#), has a beautiful, easy to use and [well-documented API](#) and a simple, readable [source code](#) that is a joy to [contribute](#) to.

Used by: Flickr foursquare Pinterest craigslist Data.gov IGN Wikimedia OSM Meetup WSJ Mapbox CartoDB GIS Cloud ...



Leaflet is easy to use

- There are many similar javascript libraries for working with maps in web-pages
- But Leaflet is a little different
- It is very easy to use – very easy to learn – and it is very mobile friendly.
- It can handle a lot of the functionality that people require from web-mapping.

Example4.html is the first Leaflet example today

- There are a few changes to Example3.html which are visible in Example4.html
- These changes reflect the introduction of Leaflet functionality
- You'll notice the LEAFLET folder in your downloaded folder for today.
-
- **Let's open up Example4.html – and we can talk through what is new.**

Change 1: We have had to make a direct link to Leaflet's CSS file and Javascript file

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- This bit here is for the web-browser - we won't be changing it -->
    <meta charset="utf-8"><meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Example 4 - Our First Map</title>

    <!-- Bootstrap --> <!-- We shouldn't need to change these very much
    over the course of all of the examples -->
    <!-- This is where we link directly to the Bootstrap code -->
    <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
    <script src="bootstrap/js/bootstrap.min.js"></script>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="jquery/jquery-1.11.1.min.js"></script>

    <!-- This is where we link to the Leaflet CSS and Leaflet JS -->
    <link rel="stylesheet" href="leaflet/leaflet.css" />
    <script src="leaflet/leaflet.js"></script>

  </head>
  <body>
```



Change 3: The Javascript to operate the map functionality is provided

```
<!-- Now for our Javascript for Leaflet and our mapping application-->
<script>
<!-- this is the map we are using -->
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'
});

<!-- this is the Javascript to setup the map -->
<!-- The [latitude longitude] is specified for the center of the map -->
<!-- zoom - Several levels 1 ... 16 -->
var map = L.map('map', {
  center: new L.LatLng(53.50765128545438, -8.734130859375),
  zoom: 5,
  layers: [MapQuestOpen_OSM]
});

<!-- This adds the base-layer map to our container ->
var baseMaps = {"MapQuest Open": MapQuestOpen_OSM};
L.control.layers(baseMaps).addTo(map);

</script>
```

Load up Example4.html to your webcourse folder – you should get output like this.

This is our first web-based map



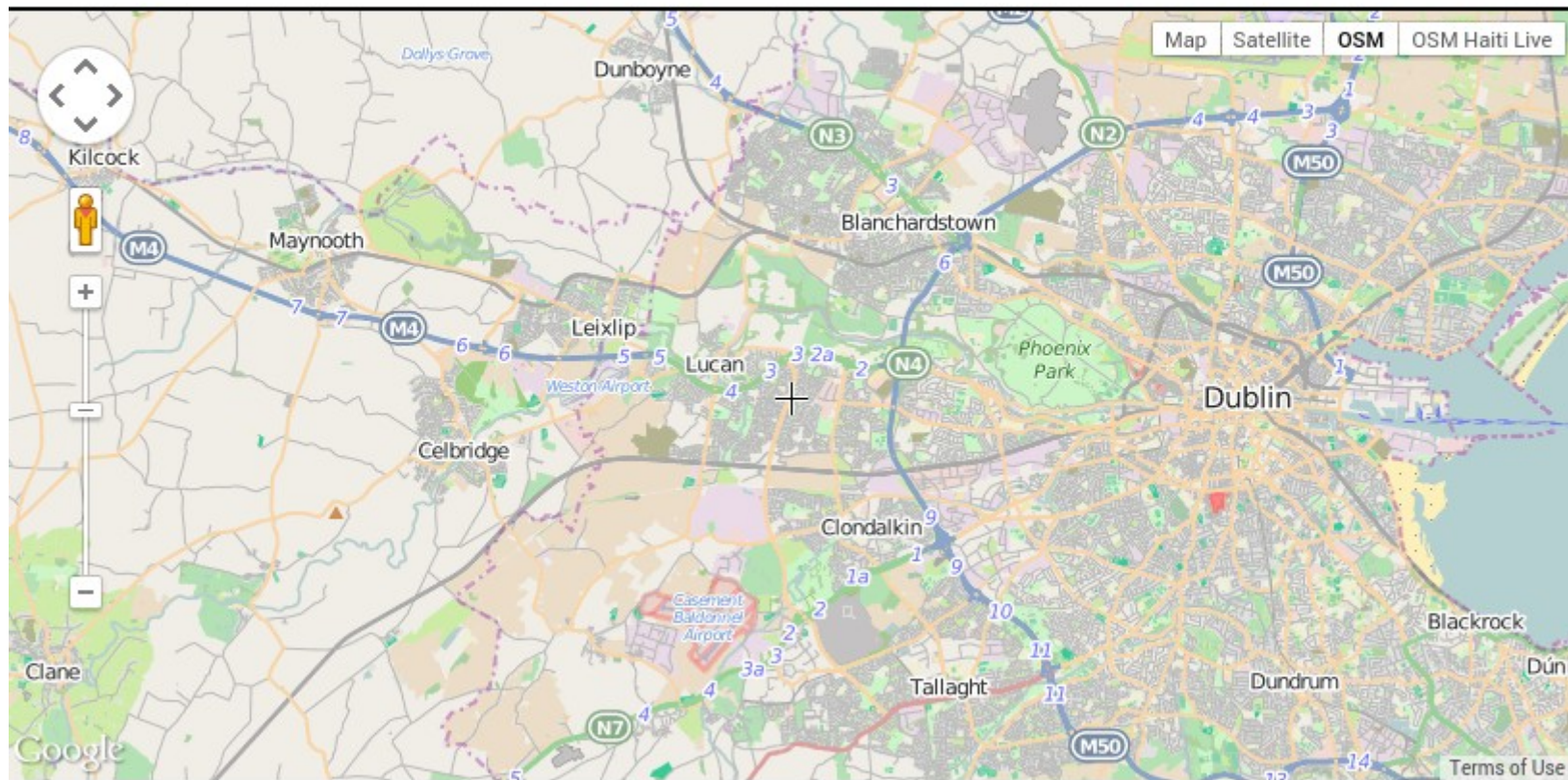
http://dbsgeo.com/latlon/

Get Lat Lon

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

Place name:

[Zoom to my location \(by IP\)](#)



Latitude, Longitude: 53.34981, -6.26031

WKT: POINT(-6.26031 53.34981)

Task 2:

- Take Example4.html and copy and rename to Task2.html
- Edit the Javascript for the map – to include a different map center (pick this from Get Lat Lon website)
- Change the zoom level – 1 ... 16
- Save, upload to webcourse folder and view

Example5.html

- Adding markers to the map
-
- We want to use the map to add our own data to
 - not just display a map canvas.
-
- Example5.html shows how to put markers onto the map

Example5.html

```
<!-- this is the Javascript to setup the map -->
<!-- The [latitude longitude] is specified for the center of the map -->
<!-- zoom - Several levels 1 ... 16 -->
var map = L.map('map', {
  center: new L.LatLng(53.38157, -6.59080),
  zoom: 15,
  layers: [MapQuestOpen_OSM]
});

var marker = L.marker([53.38450, -6.60376]).addTo(map);
var marker2 = L.marker([53.37908, -6.59730]).addTo(map);

<!-- This adds the base-layer map to our container -->
var baseMaps = {"MapQuest Open": MapQuestOpen_OSM};
L.control.layers(baseMaps).addTo(map);
```

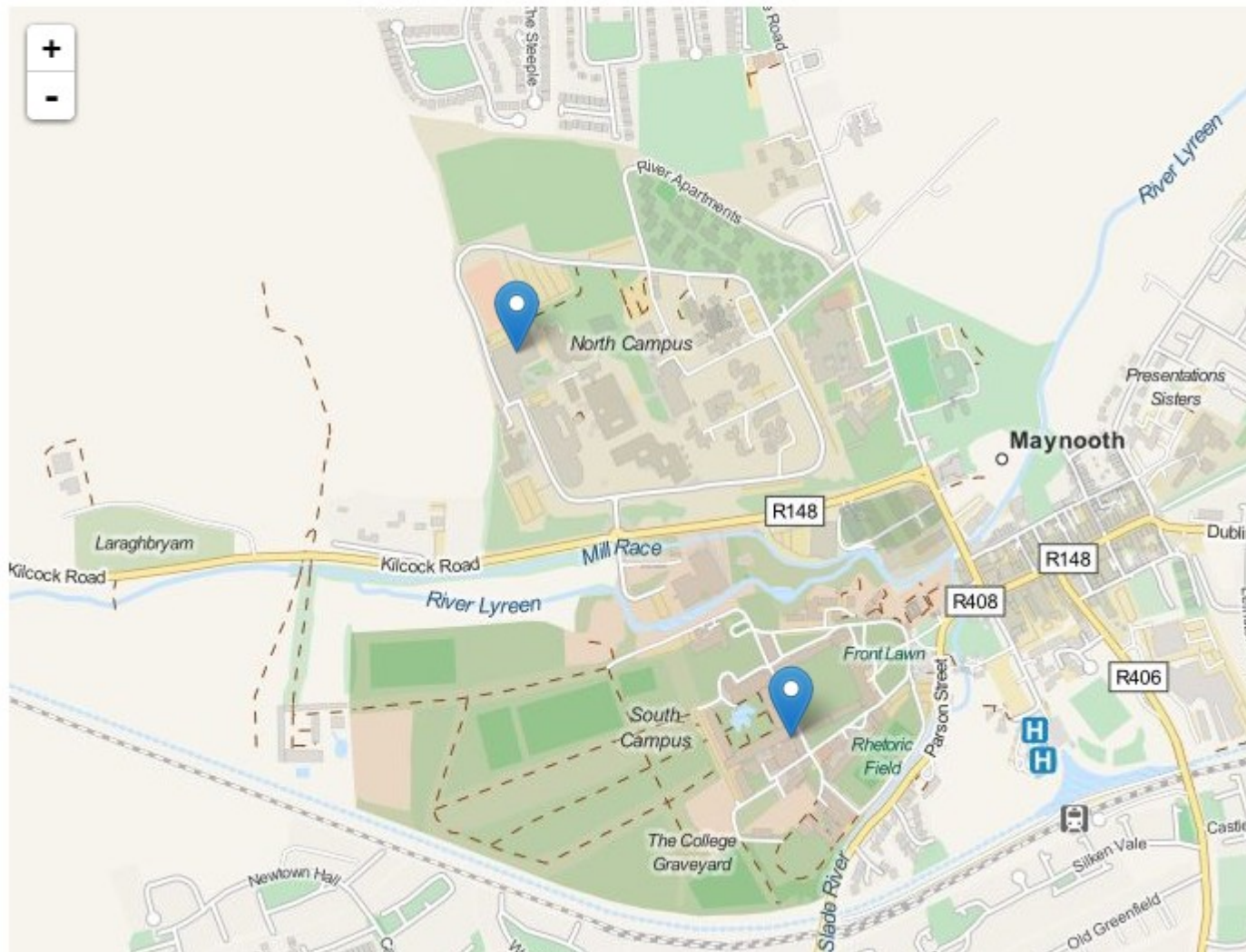
Two markers added to the map

I've changed the center of the map – to the center of the campus – and I've also changed the zoom.

Example5.html

This is our 2nd web-based map

Two places to have some dinner on the Maynooth University Campus



Example6.html – adding popups

- In the previous example – the markers didn't actually do anything – they just marked out locations on the map.
- In this example – we will supply text which can be shown when you click on the map.
- So let's open Example6.html, upload it to your webcourse folder and view it.
- Maybe check it out on your smartphone/tablet

Example6.html – adding popups

```
<!-- this is the Javascript to setup the map -->
<!-- The [latitude longitude] is specified for the center of the map -->
<!-- zoom - Several levels 1 ... 16 -->
var map = L.map('map', {
    center: new L.LatLng(53.38157, -6.59080),
    zoom: 15,
    layers: [MapQuestOpen_OSM]
});

var marker = L.marker([53.38450, -6.60376]).addTo(map);
marker.bindPopup("<b>I am the North Campus</b><br>I am a popup.");

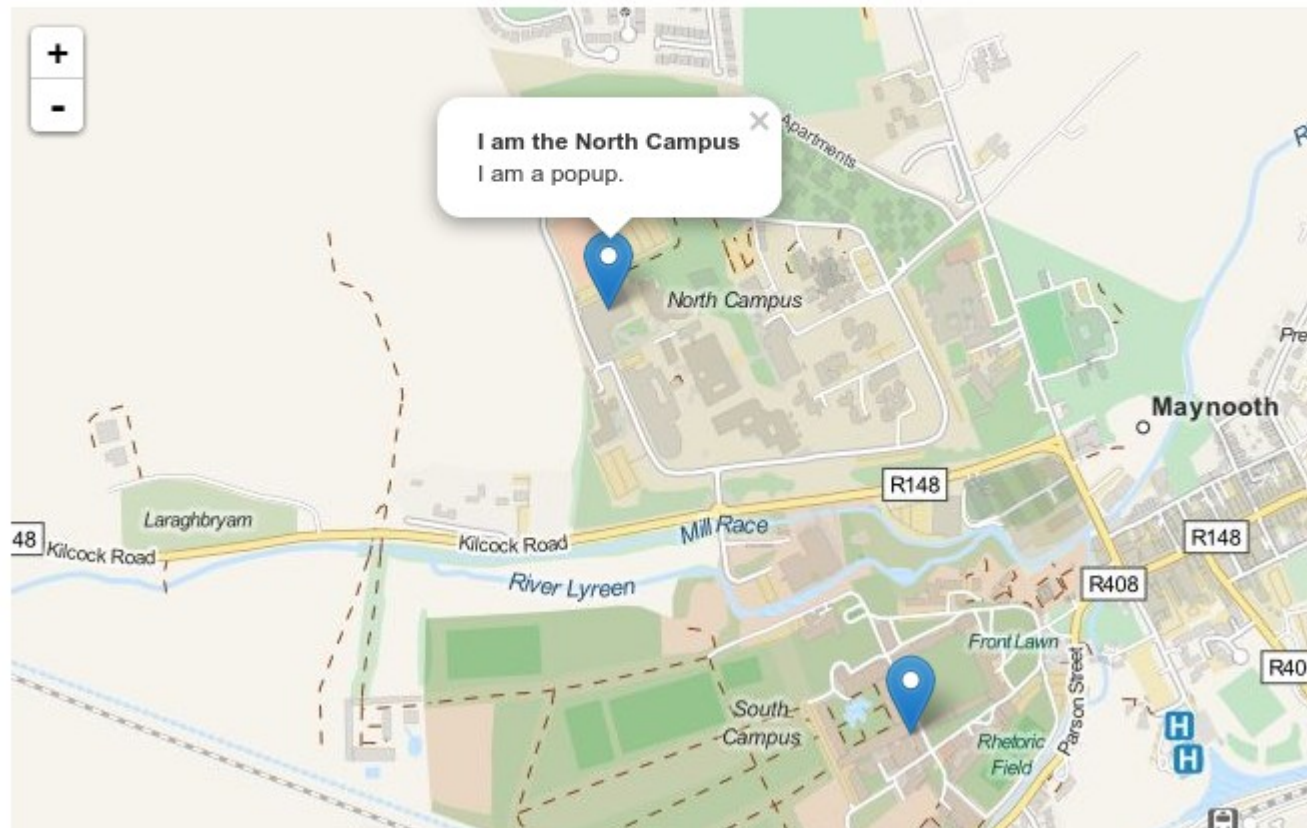
var marker2 = L.marker([53.37908, -6.59730]).addTo(map);
marker2.bindPopup("<b>I am the South Campus</b><br>I am a popup.");

<!-- This adds the base-layer map to our container ->
var baseMaps = {"MapQuest Open": MapQuestOpen_OSM};
L.control.layers(baseMaps).addTo(map);
```

Example6.html Popups

This is our 3rd web-based map

Two places to have some dinner on the Maynooth University Campus



Task 3:

- Make a copy of Example6.html
- Rename the file as Task3.html
-
- TASK: use the GetLatLon website to find coordinates
- Create 4 popups for 4 locations of Olympic Games – center the map appropriately with an appropriate zoom

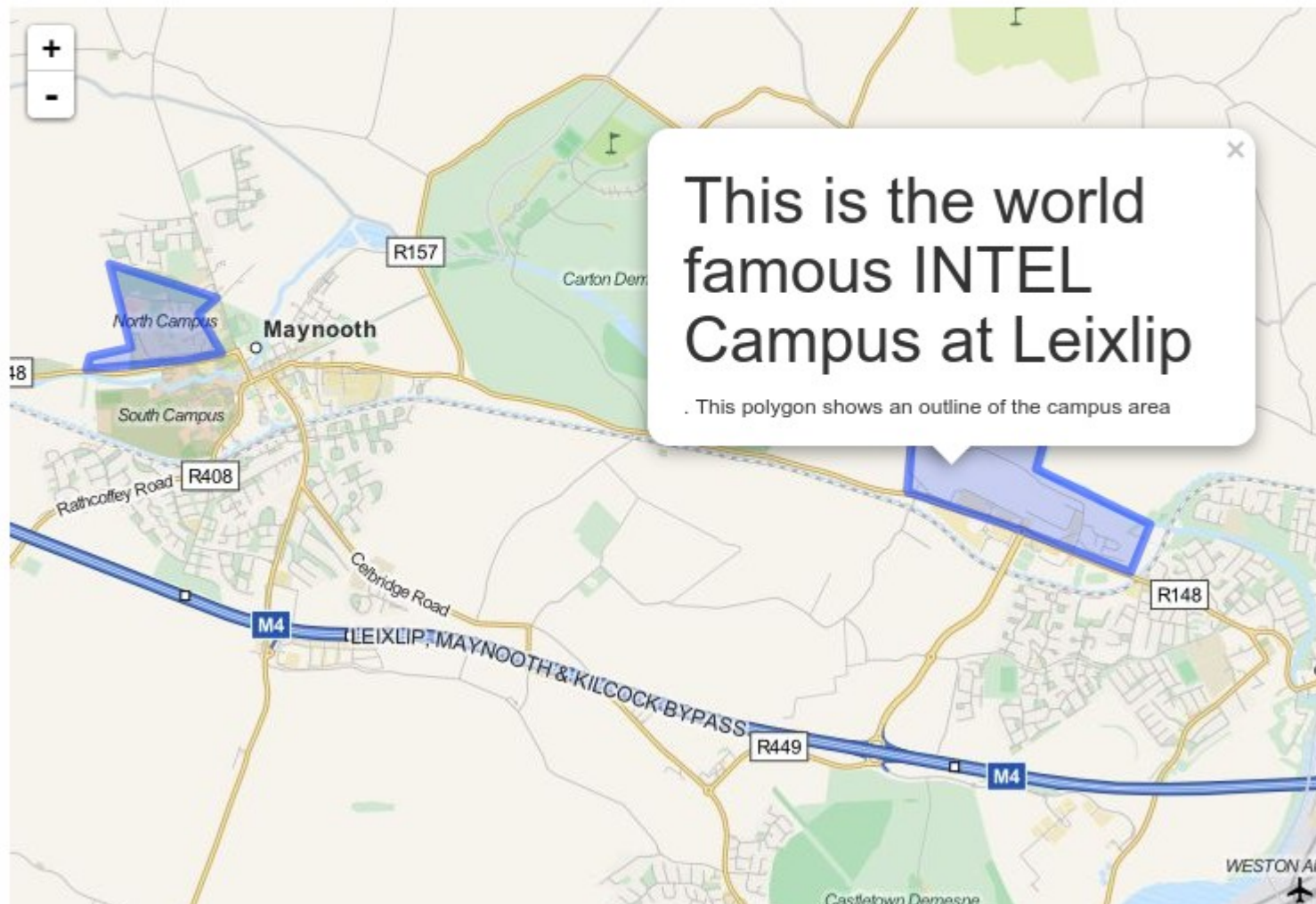
Example7.html POLYGONS

- Up to now ... we have just worked with points on the map
- How do we display polygons?
-
- Polygons can be represented in leaflet. Look at Example7.html – upload to web-course and view.

Example 7

This is our 4th web-based map

Looking at Polygons



```
var map = L.map('map', {  
  center: new L.LatLng(53.3774812824388, -6.516407015151344),  
  zoom: 13, layers: [MapQuestOpen_OSM]});
```

```
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");

```
var NorthCampus = L.polygon([  
  [53.38784823667833, -6.606409549713135],  
  [53.3830106009008, -6.6043925285339355],  
  [53.38253704616908, -6.608362197875977],  
  [53.38178190745818, -6.608598232269287],  
  [53.38225547058848, -6.599006652832031],  
  [53.38270343086418, -6.595659255981445],  
  [53.384917565293634, -6.59778356552124],  
  [53.38583902083523, -6.59604549407959],  
  [53.38583902083523, -6.59604549407959],  
  [53.38784823667833, -6.606409549713135]  
], {color: '#FFFF33'}).addTo(map);
```

NorthCampus.bindPopup("<h2>This is the north campus of NUIM</h2>");

Notice the color value?

- This is a color value specified as a HEXIDECIMAL value
- We don't need to go into details but this is how colors are represented accurately for web-browser software



<https://www.visibone.com/colorlab/>

Task 4

- For Task 4 – copy example7.html (rename to task4.html)
- Use GeoLatLon webpage – to get the coordinates of the south campus – create the south-campus polygon and add to the map. You might need to fix the center of the map and the zoom. You must also add a popup for the south campus polygon. Pick a suitable color for the polygon from Visibone Color Lab

How are the polygons structured?

L.polygon([

**[lat,long],
[lat,long],
etc**

.....

.....

**[lat,long]
])**

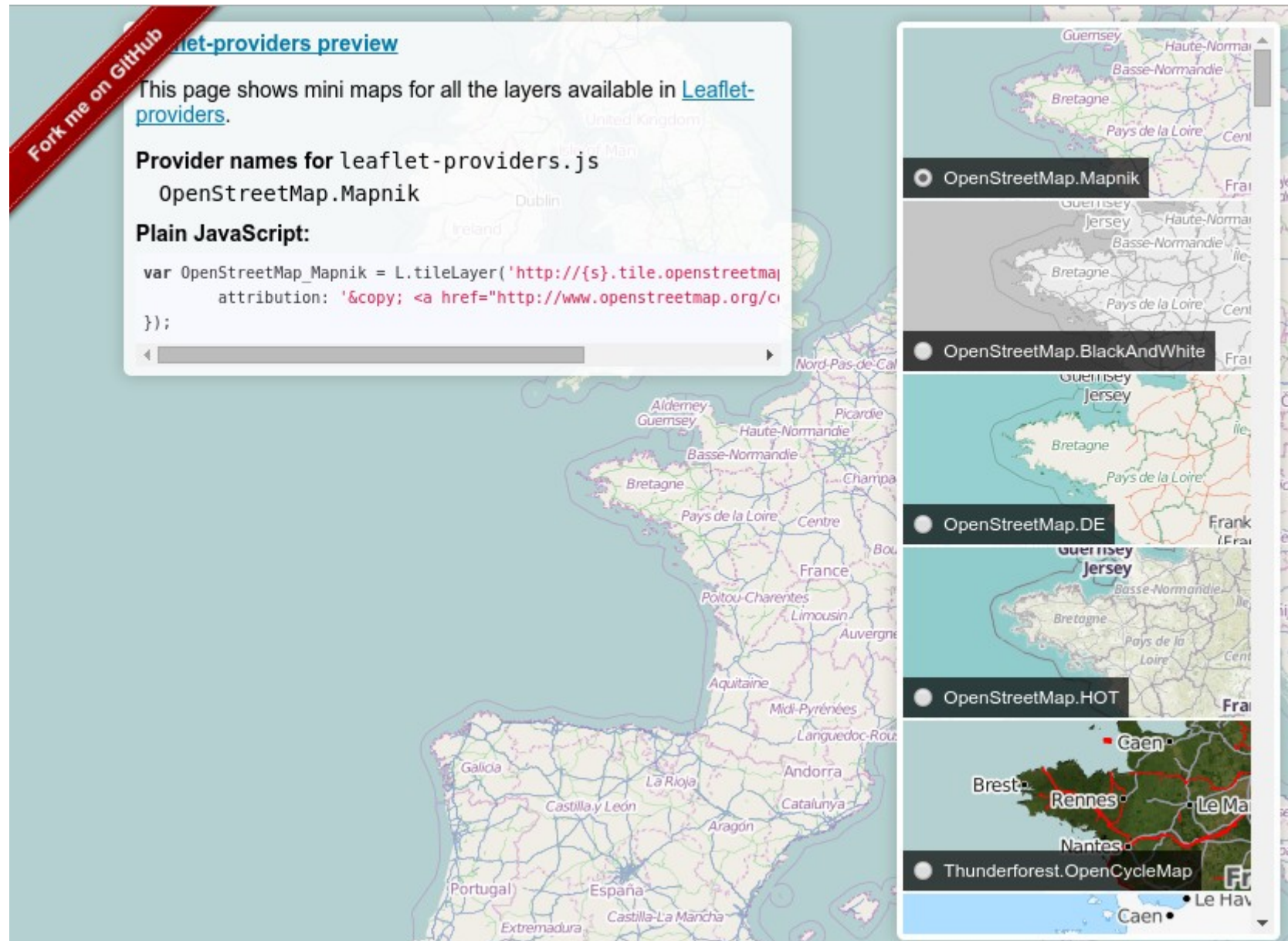
notice no comma

Switching Background Layers

Background Layers in Web Maps

- Up to this point – we have just used one background layer for all our map examples
- This is the called the “BASE LAYER” or “BASE MAP”
- In the top right hand corner there is a button icon called the “layer switcher”. This allows us to change the BASE LAYER
- In the next example we will add some new base layers

<http://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

OpenStreetMap.Mapnik

Plain JavaScript:

```
var OpenStreetMap_Mapnik = L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap contributors</a>, Imagery © Mapbox'
});
```

The background of the page is a map of France and surrounding regions. On the right side, there is a vertical list of map providers, each with a radio button and a small thumbnail map. The providers listed are:

- ☒ OpenStreetMap.Mapnik
- ☐ OpenStreetMap.BlackAndWhite
- ☐ OpenStreetMap.DE
- ☐ OpenStreetMap.HOT
- ☐ Thunderforest.OpenCycleMap

Leaflet Providers

- The **Leaflet Providers** page gives us a list of providers of base-layers which we can include in our Javascript code.
- You must carefully copy the “Plain Javascript” code from the preview into your Javascript map code.
- You can choose as many as you like – but we have to add them to the 'basemaps' variable further down our code.

Copying the Plain Javascript



[Leaflet-providers preview](#)

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

Provider names for leaflet-providers.js

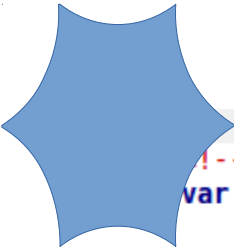
Thunderforest.OpenCycleMap

Plain JavaScript:

```
var Thunderforest_OpenCycleMap = L.tileLayer('http://{s}.tile.thunderforest.com/cycle/{z}/{x}/{y}.  
attribution: '&copy; <a href="http://www.opencyclemap.org">OpenCycleMap</a>, &copy; <a href=
```

You will need to copy this. Be careful and make sure you copy all of this text – starting at 'var'

Example8.html – adding baselayers



```
<script>
<!-- this is the map we are using -->
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'});

<!-- these are additional base layers -->
var Thunderforest_Landscape = L.tileLayer('http://{s}.tile.thunderforest.com/landscape/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="http://www.opencyclemap.org">OpenCycleMap</a>, &copy; <a
href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'});

var MapQuestOpen_Aerial = L.tileLayer('http://oatile{s}.mqcdn.com/tiles/1.0.0/sat/{z}/{x}/{y}.jpg', {
  attribution: 'Tiles Courtesy of <a href="http://www.mapquest.com/">MapQuest</a> &mdash; Portions
Courtesy NASA/JPL-Caltech and U.S. Depart. of Agriculture, Farm Service Agency',
  subdomains: '1234'});


<!-- this is the Javascript to setup the map -->
<!-- The [latitude longitude] is specified for the center of the map -->
<!-- zoom - Several levels 1 ... 16 -->
var map = L.map('map', {
  center: new L.LatLng(53.3774812824388, -6.516407015151344),
  zoom: 5, layers: [MapQuestOpen_OSM]});

var marker12 = L.marker([53.37908, -6.59730]).addTo(map);
marker12.bindPopup("<b>I am the South Campus</b><br>I am a popup.");

<!-- This adds the base-layer map to our container -->
<!-- you can have as many as you want - but the first one in this list is the default when the map is
loaded in the browser -->

var baseMaps = {"MapQuest Open": MapQuestOpen_OSM, "ThunderForest Map": Thunderforest_Landscape,
"MapQuest Aerial": MapQuestOpen_Aerial};
L.control.layers(baseMaps).addTo(map);

</script>
```



Task 5: Switching Base Layers

- Copy Example8.html and rename to Task5.html
- Use <http://leaflet-extras.github.io/leaflet-providers/preview/>
- Choose ANY 3 layers from the Providers page – which must be different to the three layers in Example8.html
- Save Task5.html – upload to webcourse – and view the results

Warnings about base-layers

- Having lots of choice with base-layers is really fantastic. There seems to be a really vast amount of options.
- However....
- Not every base layer is suitable for every situation.
- You must be careful that the base-layer provides proper/adequate spatial coverage for your application, provides tiles at all zoom levels, is up-to-date, etc.

Where's Google Maps?

- We haven't looked at including Google Maps because there are problems with the license to use Google Maps from Leaflet.
- Leaflet does provide a plugin – but it shouldn't be used in production environments
- Overall – these lectures and my approach is to use completely free and open sources of base-layer mapping and spatial data

OK – Let's see where we are!

We have got some maps working

- We can add markers, we can add popups
- We can add polygons
-
- However – all this seems pretty cumbersome.
- There is a lot of manual work involved
- **It also mixes our Javascript with our spatial data which is actually very bad practice**

Introducing GeoJSON

- For the remainder of the lectures we are going to focus on a spatial data format called GeoJSON
- You might already be familiar with JSON from programming on the web.
- GeoJSON will allow us to store our geographical information in a simple text-based format which can then be natively consumed by web-applications such as Leaflet

What is GeoJSON?

- GeoJSON is a format for encoding a variety of geographic data structures.
- It is based on JSON and is human readable
- It is navitely consumed by many web applications

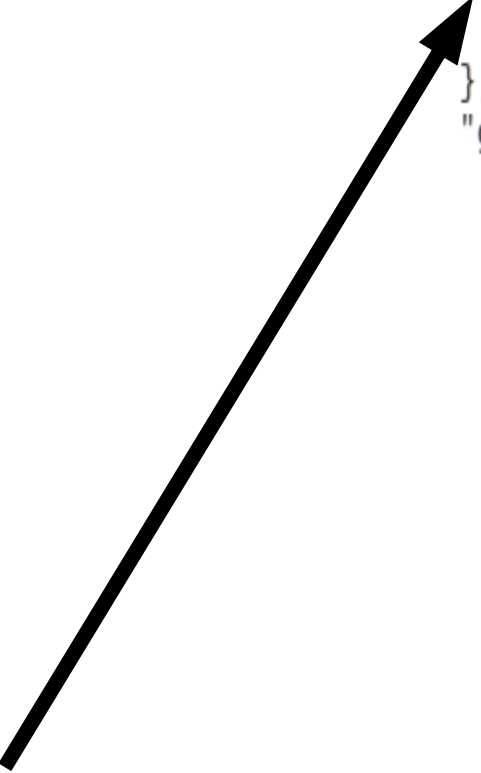
```
{  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [-6.25583, 53.34441]  
  },  
  "properties": {  
    "name": "Trinity College Dublin",  
    "numberStudents": 8000  
    "address": "Nassau Street, Dublin 2, Ireland"  
  }  
}
```


What can GeoJSON encode?

- GeoJSON can encode POINTS, LINES and POLYGONS
- It also allows us to specify PROPERTIES (or attributes/metadata) about spatial objects
- This means that we can have a rich store of spatial data within GeoJSON

Main Street – Maynooth – as GeoJSON

```
"type": "Feature",  
"properties": {  
  "name": "Main Street",  
  "shopping": "yes"  
},  
"geometry": {  
  "type": "LineString",  
  "coordinates":  
    [  
      [-6.58880352973938, 53.382182876582185],  
      [-6.589779853820801, 53.38190769817839],  
      [-6.59054160118103, 53.3816773148912],  
      [-6.59102439880371, 53.381504526607806],  
      [-6.59155011177063, 53.381382934432494],  
      [-6.592065095901489, 53.38122934382003],  
      [-6.592494249343872, 53.38112694977058],  
      [-6.5927088260650635, 53.38108855193859]  
    ]  
  }  
}
```



These are the properties
or attributes

The Callan Building – NUIM Maynooth – as GeoJSON

```
- {
  "type": "Feature",
  "properties": {
    "name": "Callan Building"
  },
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [ -6.603437662124634, 53.38374432035957 ],
        [ -6.602815389633179, 53.38389790190025 ],
        [ -6.602590084075928, 53.38377631655957 ],
        [ -6.602482795715332, 53.38351394700979 ],
        [ -6.602300405502319, 53.38324517652632 ],
        [ -6.602203845977783, 53.38312358932166 ],
        [ -6.602193117141724, 53.38295080690723 ],
        [ -6.602160930633545, 53.382726828659784 ],
        [ -6.6023218631744385, 53.382630837621626 ],
        [ -6.602600812911987, 53.38261803880018 ],
        [ -6.602911949157715, 53.38304039787629 ],
        [ -6.603158712387085, 53.38332196826615 ],
        [ -6.603276729583739, 53.38356514119524 ],
        [ -6.603437662124634, 53.38374432035957 ]
      ]
    ]
  }
}
```

Some things to note about GeoJSON

- **FORMATTING** – The examples we have just seen are nicely formatted – this is to allow us humans to read it easily – normally GeoJSON removes unnecessary whitespace and line breaks.
- **SYNTAX** – GeoJSON has a VERY STRICT syntax. (curly brackets, quoting, square brackets)
- **It's always best to use an automated means of creating GeoJSON – NOT MANUALLY**
-

First GeoJSON Example

- We need to look at two files – **geojson0.html** and **example1.geojson**
- Open both of these in your text editor.
- Upload both to your webcourse folder and view the geojson0.html file in the browser.
- We can see that in the geojson file there are some polygons and a line

Let's look closer at geojson0.html

```
<!-- this is the Javascript to setup the map -->
<!-- The [latitude longitude] is specified for the center of the map -->
<!-- zoom - Several levels 1 ... 16 -->
var map = L.map('map', {
  center: new L.LatLng(53.3774812824388, -6.516407015151344),
  zoom: 13, layers: [MapQuestOpen_OSM]});

<!-- we are going to make a group for our 'top layers' -->
var topLayers = L.layerGroup();

  <!-- we juse JQuery to access our GeoJSON -->
  <!-- This will become a new layer -->
  $.getJSON("./example1.geojson",
    function(data) {
      var geojson = L.geoJson(data);
      geojson.addTo(map);
      <!-- add our new layer to the group of top layers -->
      topLayers.addLayer(geojson);
    }
  );

<!-- This adds the base-layer map to our container ->
var baseMaps = {"MapQuest Open": MapQuestOpen_OSM};
<!-- This adds our overlay or top-layers to the container -->
var overlayMaps = {"First GeoJSON": topLayers};

<!-- add both the baselayers and the overlay/top layer to the layer switcher-->
<!-- remember - this adds the final grouping of the layers -->
L.control.layers(baseMaps,overlayMaps).addTo(map);
```

1. Create a group to hold the GeoJSON layer in
2. Using JQuery – let's read the geojson in from a file
3. Add the geojson to the layer group
4. Setup the layer switcher (baselayers and overlayers)
5. Setup the layer switcher

This is a VERY important piece of Javascript – in this example we see a simple but fully featured web-based mapping application.

Here is where we use JQuery to fetch the geoJSON

```
<!-- we are going to make a group for our 'top layers' -->
var topLayers = L.layerGroup();

<!-- we juse JQuery to access our GeoJSON -->
<!-- This will become a new layer -->
$.getJSON("./example1.geojson",
  function(data) {
    var geojson = L.geoJson(data);
    geojson.addTo(map);
    <!-- add our new layer to the group of top layers -->
    topLayers.addLayer(geojson);
  }
);
```

\$.getJSON is from JQuery – this greatly simplifies our task. We apply a function then inside \$.getJSON to add the geoJSON to the map.

Geojson0.html



- Notice how we can easily switch on/off the geoJSON layer?
- This concept of switching on and off layers is very important in web-based GIS

Can we get the geoJSON layer to be interactive?

- In the earlier examples – we seen that we could attach some information to markers and objects
- These are called POPUPS – can we do the same for our geoJSON layer?
- Yes we can
- But
- It's going to need an additional piece of Javascript and an understanding of what “properties” our GeoJSON has

Let's look at **geojson1.html**



- Let's upload it to web-course and see what happens in the browser – be sure to click on the polygons.
-
- Notice that Leaflet knows what the name of each feature is
-
- This is because in the geoJSON there is a “properties” value called “name”

Inside geojson1.html

```
<!-- we juse JQuery to access our GeoJSON -->
<!-- This will become a new layer -->
$.getJSON("./example1.geojson",
    function(data) {
        // notice that we have chanded the L.geoJSON call....
        // we want to call the actionToPerformWhenClicked function when
        // something in the geoJSON data file is clicked on the map
        var geojson = L.geoJson(data,{onEachFeature: actionToPerformWhenClicked});
        geojson.addTo(map);
        <!-- add our new layer to the group of top layers -->
        topLayers.addLayer(geojson);
    }
);|
<!-- This adds the base-layer map to our container ->
var baseMaps = {"MapQuest Open": MapQuestOpen_OSM};
<!-- This adds our overlay or top-layers to the container -->
var overlayMaps = {"First GeoJSON": topLayers};

<!-- add both the baselayers and the overlay/top layer to the layer switcher-->
<!-- remember - this adds the final grouping of the layers -->
L.control.layers(baseMaps,overlayMaps).addTo(map);

function actionToPerformWhenClicked(feature, layer) {
    // does this feature have a property named name?
    if (feature.properties && feature.properties.name) {
        layer.bindPopup("The name of this geoJSON feature is " + feature.properties.name);
    }
}
```



How can we get to know the “properties” in the geoJSON?

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "name": "Main Street",
        "shopping": "yes",
        "geometry": {
          "type": "LineString",
          "coordinates": [
            [
              [-6.58880352973938, 53.382182876582185],
              [-6.589779853820801, 53.38190769817839],
              [-6.59054160118103, 53.3816773148912],
              [-6.59102439880371, 53.381504526607806],
              [-6.59155011177063, 53.381382934432494],
              [-6.592065095901489, 53.38122934382003],
              [-6.592494249343872, 53.38112694977058],
              [-6.5927088260650635, 53.38108855193859]
            ]
          ]
        }
      }
    },
    {
      "type": "Feature",
      "properties": {
        "name": "Callan Building",
        "shopping": "no",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [
              [-6.603437662124634, 53.38374432035957],
              [-6.602815389633179, 53.38389790190025],
              [-6.602590084075928, 53.38377631655957],
              [-6.602482795715332, 53.38351394700979],
              [-6.602300405502319, 53.38324517652632],
              [-6.602203845977783, 53.38312358932166],
              [-6.602193117141724, 53.38295080690723],
              [-6.602160930633545, 53.382726828659784],
              [-6.6023218631744385, 53.382630837621626],
              [-6.602600812911987, 53.38261803880018],
              [-6.602911949157715, 53.38304039787629],
              [-6.603158712387085, 53.38332196826615],
              [-6.603276729583739, 53.38356514119524],
              [-6.603437662124634, 53.38374432035957]
            ]
          ]
        }
      }
    },
    {
      "type": "Feature",
      "properties": {
        "name": "Glenroyal Shopping Centre",
        "shopping": "yes",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [
              [-6.588728427886963, 53.38065977379557],
              ...
            ]
          ]
        }
      }
    }
  ]
}
```

- You could try to read it from the geoJSON but this could be difficult
- Or you could copy/paste to an online validator for geoJSON

http://jsonformatter.curiousconcept.com

JSON Data/URL



Paste in JSON or a
URL and away you
go.

JSON Template

3 Space Tab

Validate JSON ☒



Process

Our formatted geoJSON output is nice for human reading

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "name": "Main Street",
        "shopping": "yes"
      },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            -6.58880352973938,
            53.382182876582185
          ],
          [
            -6.589779853820801,
            53.38190769817839
          ],
          [
            -6.59054160118103,
            53.3816773148912
          ],
          [
            -6.59102439880371,
            53.381504526607806
          ],
          [
            -6.59155011177063,
            53.381382934432494
          ]
        ]
      }
    }
  ]
}
```

And we can clearly see the “properties”

Remember – we will have to assume that every object has the same properties – this might not always be the case.

You must always try to understand if this is the case with any geoJSON dataset which you are working with

Let's open airports.geojson in your text editors

```
{
  "type": "Feature",
  "properties": {
    "scalerank": 2,
    "featurecla": "Airport",
    "type": "major",
    "name": "Eleftherios Venizelos Int'l",
    "abbrev": "ATH",
    "location": "terminal",
    "gps_code": "LGAV",
    "iata_code": "ATH",
    "wikipedia": "http://en.wikipedia.org/wiki/Athens_International_Airport",
    "natlscale": 150.0
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      23.947116055407307,
      37.936233129925363
    ]
  }
}
```

Let's upload airports.geojson and geojson2.html to webcourse

- This is a big geojson dataset
- It provides a marker where every airport (of a minimum size/standard) is located
- Notice from the source code of geojson2.html how we didn't really have to change much – we just changed the **actionToPerformWhenClicked** function so that it produced a nicer popup.

World Airports

```

<!-- we juse JQuery to access our GeoJSON -->
<!-- This will become a new layer -->
$.getJSON("./airports.geojson",
    function(data) {
        // notice that we have chanded the L.geoJSON call....
        // we want to call the actionToPerformWhenClicked function
        // something in the geoJSON data file is clicked on the map
        var geojson = L.geoJson(data,{onEachFeature: actionToPerformWhenClicked});
        geojson.addTo(map);
        <!-- add our new layer to the group of top layers -->
        topLayers.addLayer(geojson);
    }
);

<!-- This adds the base-layer map to our container -->
var baseMaps = {"MapQuest Open": MapQuestOpen_OSM};
<!-- This adds our overlay or top-layers to the container -->
var overlayMaps = {"Airports GeoJSON": topLayers};

<!-- add both the baselayers and the overlay/top layer to the layer switcher-->
<!-- remember - this adds the final grouping of the layers -->
L.control.layers(baseMaps,overlayMaps).addTo(map);

function actionToPerformWhenClicked(feature, layer) {
    // does this feature have a property named name?
    if (feature.properties.name) {
        // we can create a nice HTML based popup here
        layer.bindPopup("<h1>" + feature.properties.name + "</h1> Abbreviated Name " +
            feature.properties.abbrev + "<br/><b>IATA Code</b>" + feature.properties.iata_code);
    }
}

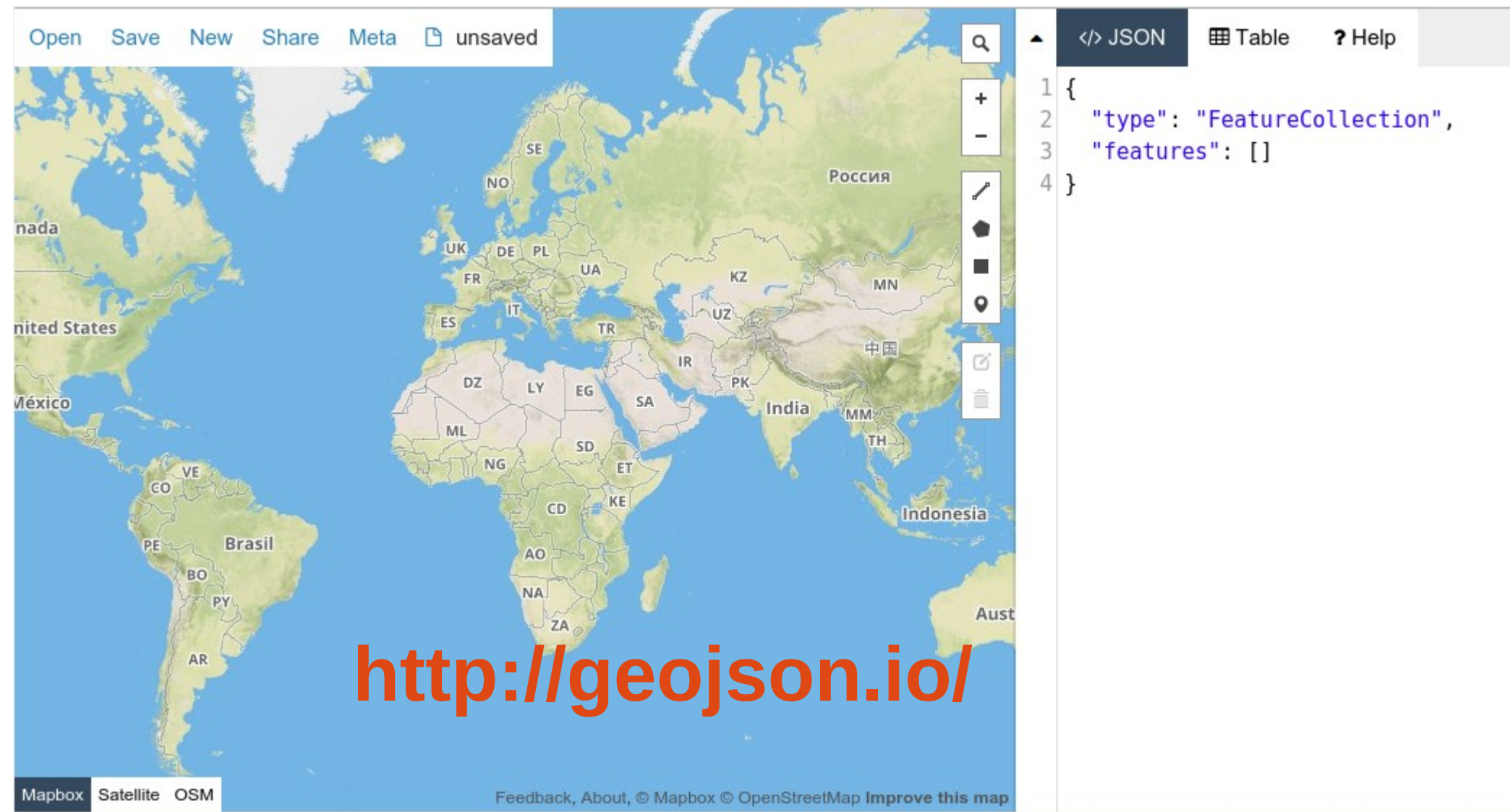
```



How do I get geoJSON?

- There are literally hundreds of ways to get your spatial data in geoJSON
 - - use a programming language such as PHP
 - - use a desktop GIS such as QGIS
 - - use a spatial database such as PostGIS
 - - download it from a web service/site
 - - create your own using an interactive tool

Creating our own geoJSON



The screenshot displays the geojson.io web application. On the left, a world map is shown with country borders and labels. The map is centered on Europe. On the right, a JSON editor is open, showing a basic geoJSON structure. The editor has tabs for 'JSON', 'Table', and 'Help'. The JSON tab is selected, and the following code is visible:

```
1 {  
2   "type": "FeatureCollection",  
3   "features": []  
4 }
```

At the bottom of the map, the URL <http://geojson.io/> is displayed in large red text. The bottom of the map interface includes a 'Mapbox' logo, 'Satellite' and 'OSM' map style buttons, and a footer with 'Feedback, About, © Mapbox © OpenStreetMap Improve this map'.

Use the geoJSON.io to create some objects on the map

- Zoom in to your region of interest
- Create an object – make sure that you add some properties or attributes.
- Choose “save” -> geoJSON and you can save it to your working folder for today.

Task 6

- **Copy either `geojson1.html` or `geojson2.html` and rename as `task6.html`**
- Use `geojson.io` to create a geojson dataset
- Dataset Purpose: *The outline of 5 university campuses in Ireland. These outlines should have `name`, `studentPop`, `cityName` as properties. Properties can be just estimates or approximations*
- The resulting webmap should be appropriately centered and zoomed
- Popups should include the three properties named above.

Adding a touch of style to geoJSON

- Up to now ... we have just added our geoJSON data to our maps without any consideration of styling or visual properties.
- How can we change the visual properties of the objects in a geoJSON dataset?
- ANS: We have to use Javascript.

Style Example: geojson3.html

- Look at geojson3.html in your text editor.
- Change the filename of the geojson file to correspond to the file you created in geojson.io a few minutes ago.
- Upload both to webcourse server and view
-
- If we look at the source code – we will see a new piece of Javascript which specifies the style we want to apply to the objects in the geojson file.

Javascript: applying a style

```
// this is the style we want to apply to the objects in our geojson dataset
var myGeoJSONStyle = {
  "color": "#ff7800",
  "opacity": 0.5,
  "weight": 4,
  "fillOpacity": 0.5,
  "fillColor": "blue"
};

<!-- we are going to make a group for our 'top layers' -->
var topLayers = L.layerGroup();

<!-- we juse JQuery to access our GeoJSON -->
<!-- This will become a new layer -->
$.getJSON("./example1.geojson",
  function(data) {
    // notice that we have chanded the L.geoJSON call....
    // we want to call the actionToPerformWhenClicked function when
    // something in the geoJSON data file is clicked on the map
    // notice that we have now applied a style to the geoJSON objects

    var geojson = L.geoJson(data,{style: myGeoJSONStyle,
      onEachFeature: actionToPerformWhenClicked});
    geojson.addTo(map);
    <!-- add our new layer to the group of top layers -->
    topLayers.addLayer(geojson);
  }
);
```

Color of outline line

Opacity of outline line [0, 1.0]

Thickness (pixels) outline line

What the fille opacity is [0,1.0]

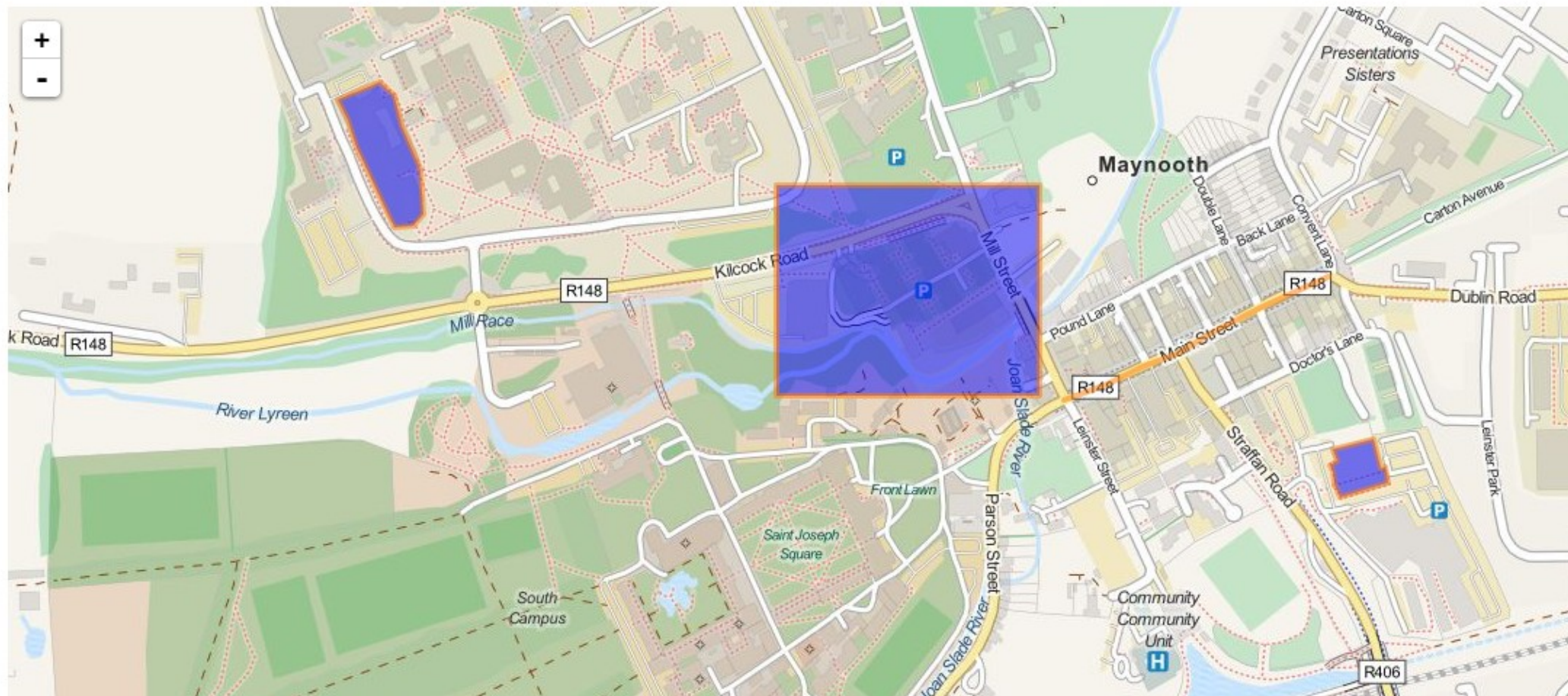
The color which will fill up polygons – we can specify this in HEX or in ordinary color names (check the VISIBONE website)

We then apply our style to this geoJSON layer – by including it in it's properties

What the styling should look like

Starting off with GeoJSON - a larger dataset - and Style

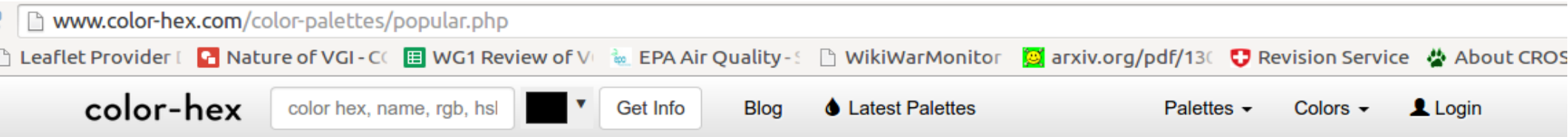
Looking at GeoJSON - seeing popups from the GeoJSON Objects from a larger dataset. And styling the polygons.



Let's experiment with the style(s)

- Why not change some of these values and see what effect they have on how your dataset looks?
- You will need to make the change in your text editor and then upload to webcourse and view
-
- Note – these styles will not effect MARKERS for points.

www.color-hex.com/color-palettes/popular.php



Popular Color Palettes

Most popular color palettes.



Facebook



Metro Style



Metro UI Colors



RGB Grey White



Caucasian Skin Tone



I Loved In Shades of Green



Shades of Purple



Cappuccino



Shades of Gray



s+b teal



Craftsman Connection



cool



Downloading Townlands.ie data

Search for a townland, civil parish, barony, electoral division or county

Search

The data from townlands.ie is available in many formats

Type of Data	Shapefile	GeoJSON	KML	CSV
Townlands	download shapefile	download GeoJSON	download KML	download CSV
Electoral Divisions	download shapefile	download GeoJSON	download KML	download CSV
Civil Parishes	download shapefile	download GeoJSON	download KML	download CSV
Baronies	download shapefile	download GeoJSON	download KML	download CSV
Counties	download shapefile	download GeoJSON	download KML	download CSV

Data format

The geometry is available in [WSG 84](#) (aka [EPSG 4326](#), aka "latitude and longitude") projection system.

There are several columns per entry:

- OSM_ID** Integer. The id of the object in the OSM database. If it's positive, it's a way; if it's negative, it's a relation. (Consult the [OSM data model](#) for more)
- NAME** String. The name of the object. Should be the "common name". Almost certainly in English, but may be in Irish. (NB: In the KML/KMZ file, this is NAME2 due to how ogr2ogr converts things. Suggestions welcome for how to fix this.)
- NAME-CA** String. The name of the object in Irish.



You will need to unzip this download to get the geoJSON

The property name is in CAPITAL LETTERS in the dataset

TASK 7: Styling the counties

- Copy geojson3.html and rename as counties.html
- YOU MUST HAVE THE COUNTIES GEOJSON DOWNLOADED from townlands.ie
- **TASK** – Display the counties dataset in a web-map. Apply a style which is suitable to this dataset. Change the text on the web-page to reflect the content. Fix the zoom and center. AND – you will need to display the name of the county (remember the property 'name' is in capitals...)

Adding multiple overlay layers

- Up to this point we have always just had one overlay layer
- We seen ealier where we could have many base layers.
- To create a layered effect to web-maps we need to be able to add multiple overlay layers

We will need to create a new layerGroup and a new \$.getJSON for each layer we want to add

```
<!-- we are going to make a group for our 'top layers' -->
var topLayers = L.layerGroup();

<!-- we juse JQuery to access our GeoJSON -->
<!-- This will become a new layer -->
$.getJSON("./counties.geojson",
function(data) {
    // notice that we have chanded the L.geoJSON call....
    // we want to call the actionToPerformWhenClicked function when
    // something in the geoJSON data file is clicked on the map
    // notice that we have now applied a style to the geoJSON objects

    var geojson = L.geoJson(data,{style: myGeoJSONStyle,
                                onEachFeature: actionToPerformWhenClicked});
    geojson.addTo(map);
    <!-- add our new layer to the group of top layers -->
    topLayers.addLayer(geojson);
}
);
```

This variable name will have to change – we cannot have the same variable named used twice for different layers

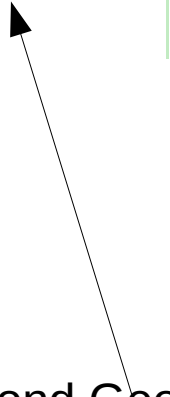
We are also going to have to add the name of this new layerGroup to the layer switcher

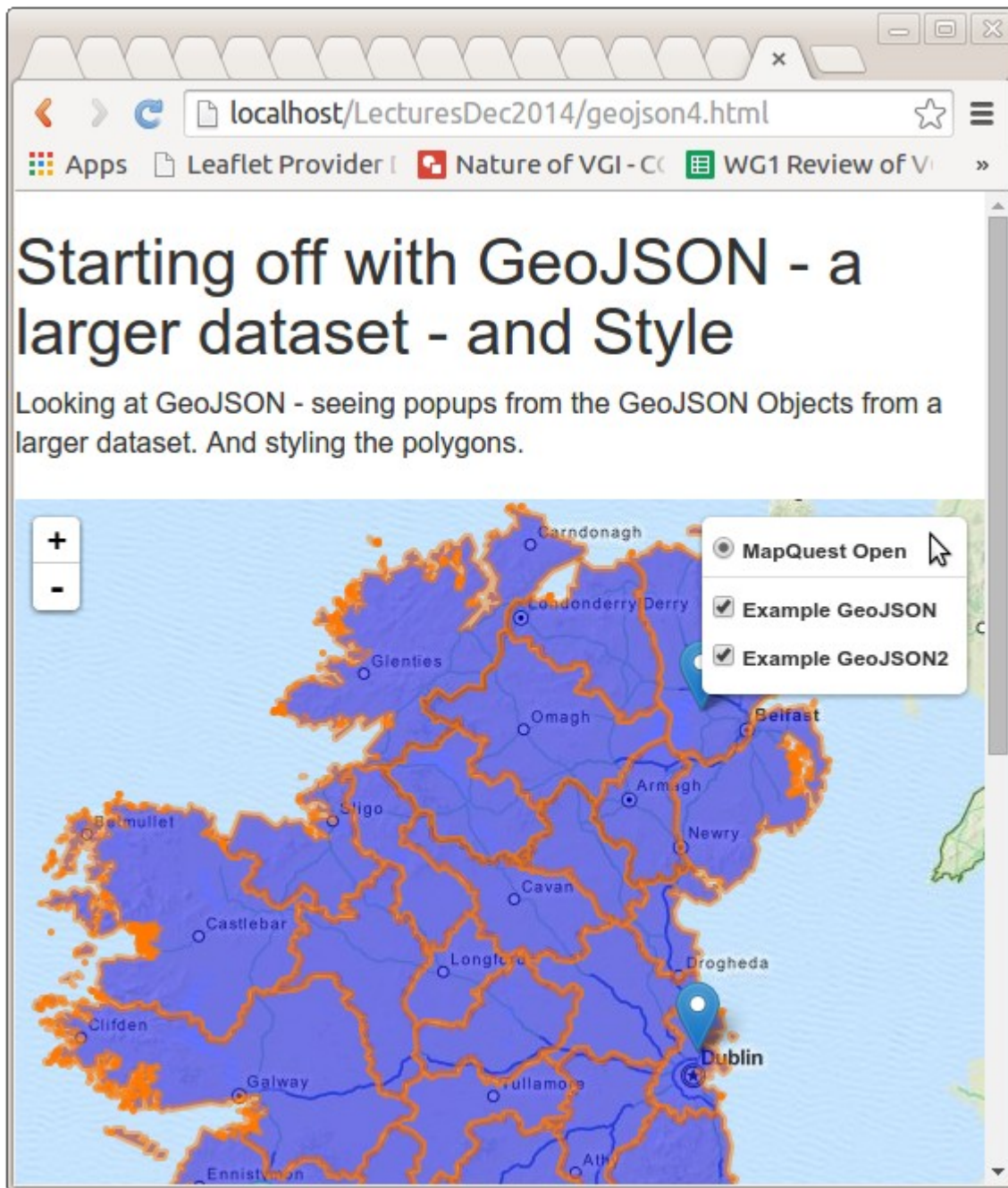
```
<!-- This adds our overlay or top-layers to the container -->  
var overlayMaps = {"Example GeoJSON": topLayers, "Example GeoJSON2": topLayers1};
```

One GeoJSON layer



A second GeoJSON layer – note the different variable name





Final Task – Multiple GeoJSON Layers

- <http://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php>
- We can download some geoJSON data from the USGS Earthquake monitoring programme.

Final Task – Multiple GeoJSON Layers

- <http://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php>
- We can download some geoJSON data from the USGS Earthquake monitoring programme.

The screenshot shows the USGS Earthquake Hazards Program website. The browser address bar displays `earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php`. The website has a dark blue header with the text "Earthquake Hazards Program" and navigation links: Home, About Us, Contact Us, and a search bar. Below the header is a light blue navigation bar with tabs: EARTHQUAKES, HAZARDS, DATA & PRODUCTS, LEARN, MONITORING, and RESEARCH. The "EARTHQUAKES" tab is selected. On the left side, there is a sidebar menu under the heading "Formats". The menu items are: ATOM, GeoJSON Summary (highlighted in green), GeoJSON Detail, KML, Spreadsheets, QuakeML, Web Service, API Documentation, Search Earthquake Archives, Glossary, Change Log, and Feed Life Cycle Policy. The main content area is titled "GeoJSON Summary Format" and contains three sections: "Description", "Usage", and "Output". The "Description" section explains that GeoJSON is a format for encoding geographic data structures and mentions the JSON standard and the GeoJSONP feed. The "Usage" section states that the feed adheres to the USGS Earthquakes Feed Life Cycle Policy. The "Output" section shows a snippet of GeoJSON data. On the right side, there is a "Feeds" section with two subsections: "Past Hour" and "Past Day". Each subsection lists five links: Significant Earthquakes, M4.5+ Earthquakes, M2.5+ Earthquakes, M1.0+ Earthquakes, and All Earthquakes. The "Past Hour" section also notes "Updated every minute."

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

Leaflet Provider | Nature of VGI - CC | WG1 Review of V | EPA Air Quality - | WikiWarMonitor | arxiv.org/pdf/13 | Revision Servi

Earthquake Hazards Program Home About Us Contact Us

EARTHQUAKES HAZARDS DATA & PRODUCTS LEARN MONITORING RESEARCH

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

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 programatic interface for applications.

Output

```
{
```

Feeds

Past Hour

Updated every minute.

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

Past Day

Updated every minute.

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

Properties for all of the Earthquake datasets

```
{  
  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  
  },  
}
```

Past Day

Updated every minute.

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

Past 7 Days

Updated every minute.

- [Significant Earthquakes](#)
- [M4.5+ Earthquakes](#)
- [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](#)

Final Task: Create multiple overlay layers with style and popup content

- Copy geojson3.html and rename as multiple.html
- Download ANY 2 of the USGS earthquake monitoring geojson datasets
- Create TWO overlay layers
- No style is necessary for markers
- Create an appropriate popup for each of the datasets including at least TWO of the properties in the dataset
- Center and zoom the map appropriately.

Can we have different styles for different layers of geoJSON?

- Yes – for every layer – we can have a different style (color, line-thickness, etc)
- It's up to you to decide what the most appropriate colours are etc.

Can we have different styles for different layers of geoJSON?

```
// this is the style we want to apply to the objects in our geojson dataset
var myGeoJSONStyle = {
  "color": "#ff7800",
  "opacity": 0.5,
  "weight": 4,
  "fillOpacity": 0.5,
  "fillColor": "blue"
};

<!-- we are going to make a group for our 'top layers' -->
var topLayers = L.layerGroup();

<!-- we juse JQuery to access our GeoJSON -->
<!-- This will become a new layer -->
$.getJSON("./counties.geojson",
  function(data) {
    // notice that we have chanded the L.geoJSON call....
    // we want to call the actionToPerformWhenClicked function when
    // something in the geoJSON data file is clicked on the map
    // notice that we have now applied a style to the geoJSON objects

    var geojson = L.geoJson(data,{style: myGeoJSONStyle,
      onEachFeature: actionToPerformWhenClicked});
    geojson.addTo(map);
    <!-- add our new layer to the group of top layers -->
    topLayers.addLayer(geojson);
  }
);
```

We can define as many styles as we like – just use the same structure as myGeoJSONStyle

You'll have to call it a different name to this.

And you'll have to change it down when you load the geoJSON

L.geoJSON()

Can we have different popups for different layers of geoJSON?

```
var topLayers1 = L.layerGroup();
$.getJSON("./airports.geojson",
  function(data) {
    // notice that we have changed the L.geoJSON call....
    // we want to call the actionToPerformWhenClicked function when
    // something in the geoJSON data file is clicked on the map
    // notice that we have now applied a style to the geoJSON objects

    var geojson = L.geoJson(data,{style: myGeoJSONStyle,
                                onEachFeature: actionToPerformWhenClicked});
    geojson.addTo(map);
    <!-- add our new layer to the group of top layers -->
    topLayers1.addLayer(geojson);
  }
);
<!-- This adds the base-layer map to our container -->
var baseMaps = {"MapQuest Open": MapQuestOpen_OSM};

<!-- This adds our overlay or top-layers to the container -->
var overlayMaps = {"Example GeoJSON1": topLayers,"Example GeoJSON2": topLayers1};

<!-- add both the baselayers and the overlay/top layer to the layer switcher-->
<!-- remember - this adds the final grouping of the layers -->
L.control.layers(baseMaps,overlayMaps).addTo(map);
```

We will need to have different `actionToPerformWhenClicked` functions if we want different popups for different layers

```
function actionToPerformWhenClicked(feature, layer) {
  // does this feature have a property named name?
  if (feature.properties && feature.properties.name) {
    // we can create a nice HTML based popup here
    layer.bindPopup("<h1>" + feature.properties.name+ "</h1>");
  }
}
```

You will have to copy this function and change the name – you can change it to any name (no spaces)

So for multiple overlays you should always have the following for each layer

- A unique name for
var topLayers = L.layerGroup();
- A unique name for
var myGeoJSONStyle = {};
- The **\$.getJSON** call – don't forget to match the variable names at **addLayer()**. And don't forget to include the variable name for the style
- Your popup content: you'll need to specify the correct function **onEachFeature: actionToPerformWhenClicked**
- Add your layer to the **var overlayMaps** grouping.

Assignment

The End