Web Mapping Using OpenLayers

PS Singh
Scientist/Engineer-SE

Definition of a Web Mapping

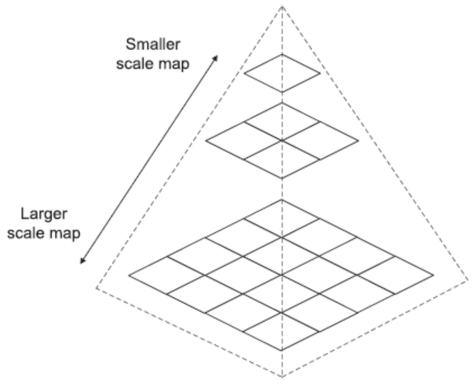
Initial Attempts on web mapping:

- Mapserver and
- ArcIMS

The two matrices on which dynamically drawn web maps have challenges are:

- Speed and
- Scalability (the ability to handle many simultaneous users/adapt or extend application based on future requirements).

<u>Example Solution</u>: Tiling and Caching using Asynchronous JavaScript and XML (AJAX)



Pyramid Tiles

Tiles from OpenStreetMap data, rendered by MapQuest

Presently most of the webmaping scenarios are based on the concept of Web Service

Definition of a web services

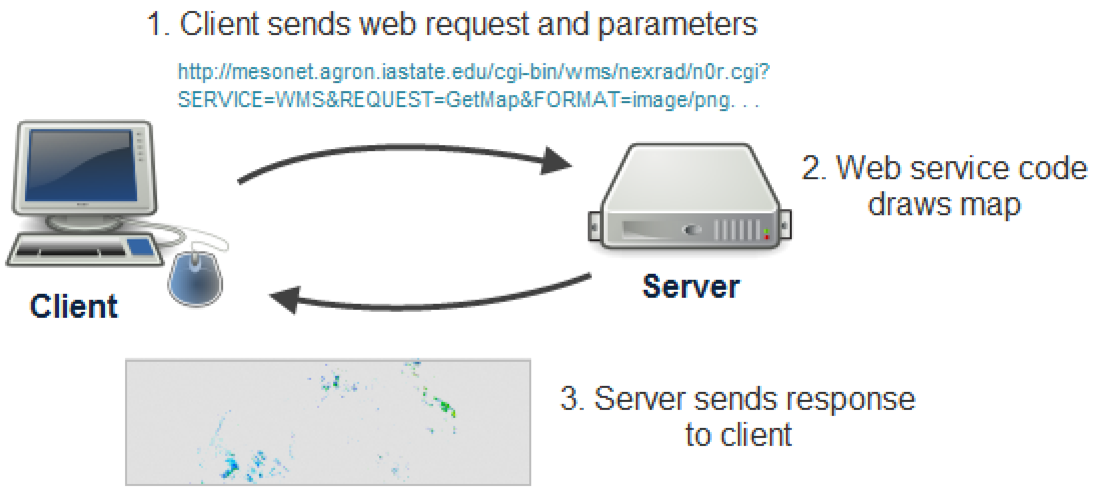
A web service as **a focused task that a specialized computer (the server) knows how to do and allows other computers to invoke**.

<u>You work with the web service like this:</u>

1. You invoke the web service by making a request from an application (the client). To make this request, you use HTTP, a standard protocol that web browsers use for communicating between clients and servers. The request contains structured pieces of information called parameters. These give specific instructions about how the task should be executed.

2. The server reads the request and runs its web service code, considering all the parameters while doing so. This produces a response, which is usually a string of information or an image.

3. The server sends you the response, and your application uses it.

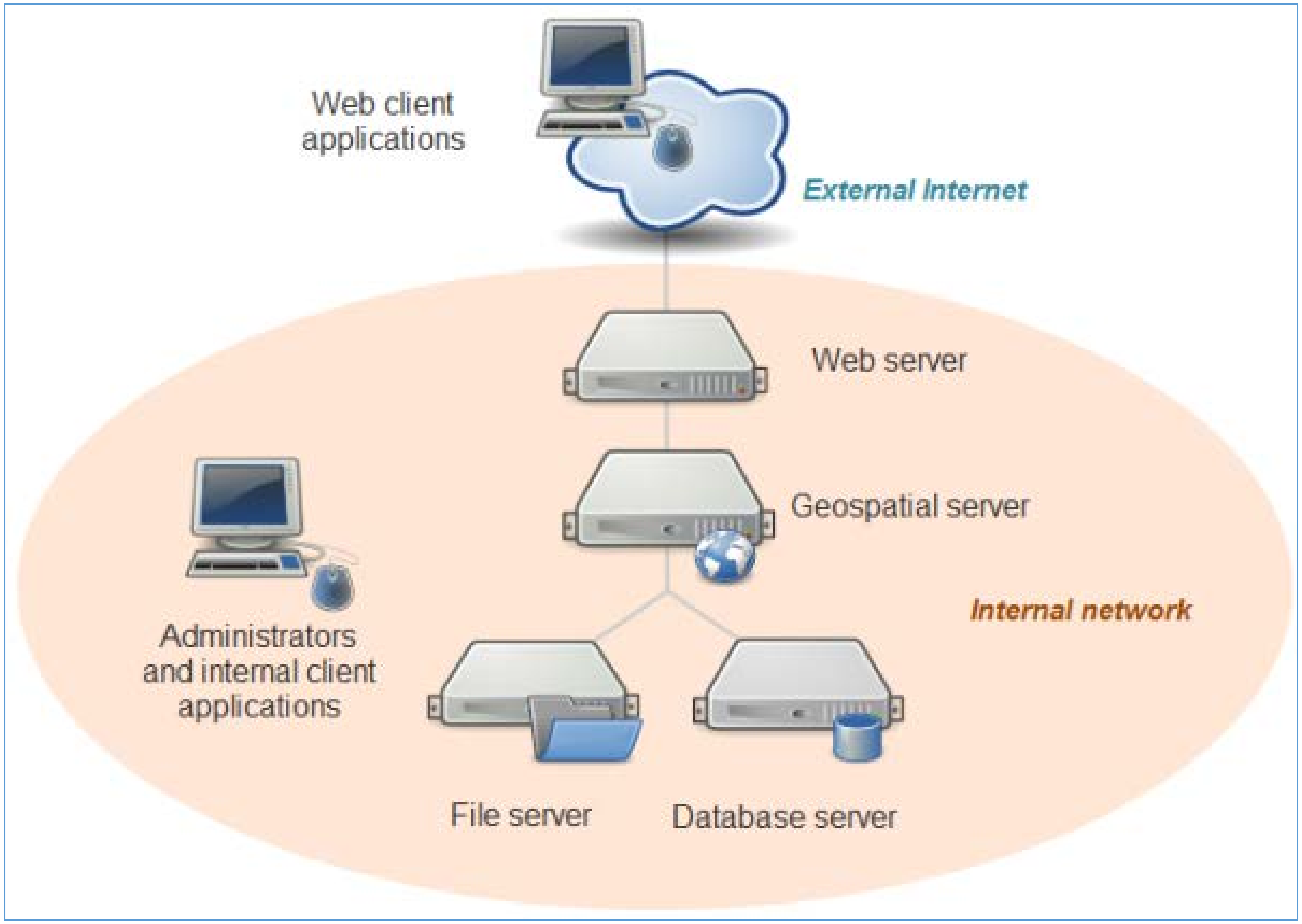

1. Client sends web request and parameters
http://mesonet.agron.iastate.edu/cgi-bin/wms/nexrad/n0r.cgi?
SERVICE=WMS&REQUEST=GetMap&FORMAT=image/png. . .
2. Web service code draws map
Client
Server
3. Server sends response to client

Sample web services Request

NEDRP Layers on ISRO Bhuvan Portal : https://bhuvan-app1.nrsc.gov.in/state/AS

```
http://mesonet.agron.iastate.edu/cgi-bin/wms/nexrad/n0r.cgi?SERVICE=WMS&REQUEST=GetMap&FORMAT=image/png&TRANSPARENT=TRUE&STYLES=&VERSION=1.3.0&LAYERS=nexrad-n0r&WIDTH=877&HEIGHT=276&CRS=EPSG:900913&BBOX=-15252263.28954773,2902486.4758432545,-6671748.242369267,5602853.811101243
```

System architecture for web mapping

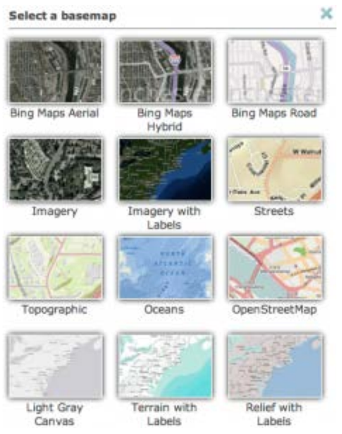


Basic Elements of a web map

- Basemaps
- Thematic layers
- Interactive elements

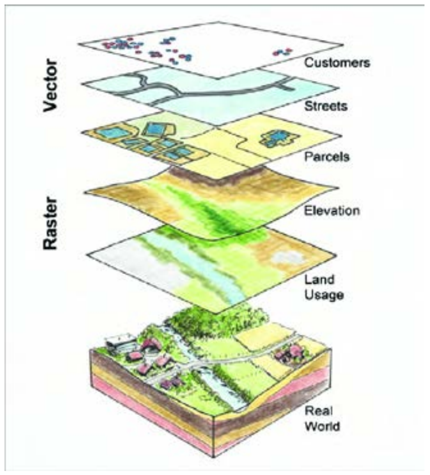
The Basemaps

The background setting for a map that provide background detail necessary to orient the location of the map. Basemaps also add to the aesthetic appeal of a map



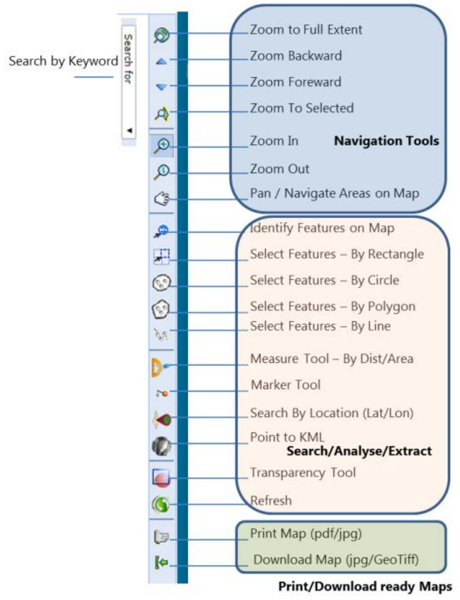
The Thematic Layers

A GIS is a collection of geographic data presented as thematic layers on a map



The Interactive Elements

Dynamic Visualize, Interact, Analyse, Extract and Export Map Info

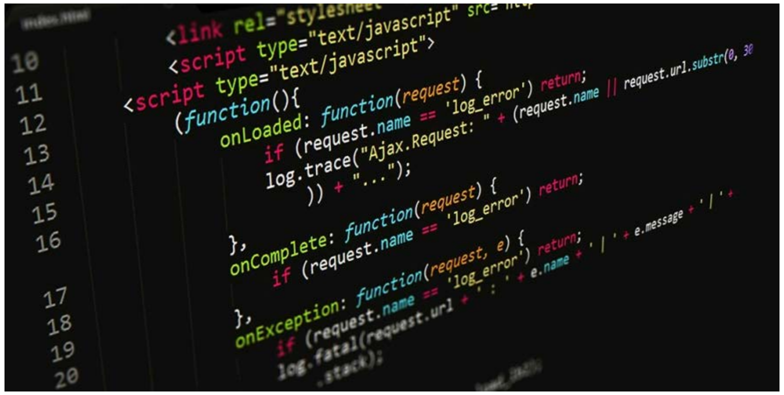

Search by Keyword
Search for
Zoom to Full Extent
Zoom Backward
Zoom Foreward
Zoom To Selected
Zoom In
Navigation Tools
Zoom Out
Pan / Navigate Areas on Map
Identify Features on Map
Select Features – By Rectangle
Select Features – By Circle
Select Features – By Polygon
Select Features – By Line
Measure Tool – By Dist/Area
Marker Tool
Search By Location (Lat/Lon)
Point to KML
Search/Analyse/Extract
Transparency Tool
Refresh
Print Map (pdf/jpg)
Download Map (jpg/GeoTiff)
Print/Download ready Maps

Basic Requirement Before Learning WebMapping

Familiarity with **HTML**, **CSS & Javascript**

It is expected that you apply this familiarity to understanding the JavaScript syntax for

- loops,
- functions,
- decision structures, and so forth.



```
<link rel="stylesheet"
<script type="text/javascript" src="http
<script type="text/javascript">
(function(){
    onLoaded: function(request) {
      if (request.name == 'log_error') return;
      log.trace("Ajax.Request: " + (request.name || request.url.substr(0, 30
      )) + "...");
    },
    onComplete: function(request) {
      if (request.name == 'log_error') return;
    },
    onException: function(request, e) {
      if (request.name == 'log_error') return;
      log.fatal(request.url + ': ' + e.name + ' | ' + e.message + ' | ' +
      stack);
```

Objectives of this Hands-On Tutorial

- Identify commonly-used web mapping APIs (both proprietary and FOSS) and recognize programming patterns that are common to each.

- Choose developer examples that relate to your web mapping task and adjust the code to meet the needs of your own application.

- Use OpenLayers to create a mashup from a tiled basemap and a WMS thematic layer.
- Create informational popups for your web map features using OpenLayers.

What is a web mapping API?

- An API (Application Program Interface) is a framework that you can use to write a program.
- It provides a set of **classes and functions** that help you avoid writing all the low-level code to perform specific actions.
- For example, web mapping APIs typically include classes for maps and layers so that you don't have to write all the low-level code for displaying an interactive map image and drawing a new layer on it.
- Instead, you can just create a new map object, create a new layer object, and call some method such as **map.addLayer(layer).**

APIs designed specifically for the purpose of making **web maps** include

- OpenLayers,
- Leaflet,
- Google Maps API, and
- ArcGIS API for JavaScript.

Examples of Open Source web mapping APIs

FOSS Based web mapping APIs for building browser-based apps with **HTML and JavaScript.**

- **OpenLayers** :

Most Matured with extensive documentation and samples

- **Leaflet:**

Light Weight and Mobile Friendliness

- **D3**:

Web app with interactive maps and charts

- **Polymaps**:

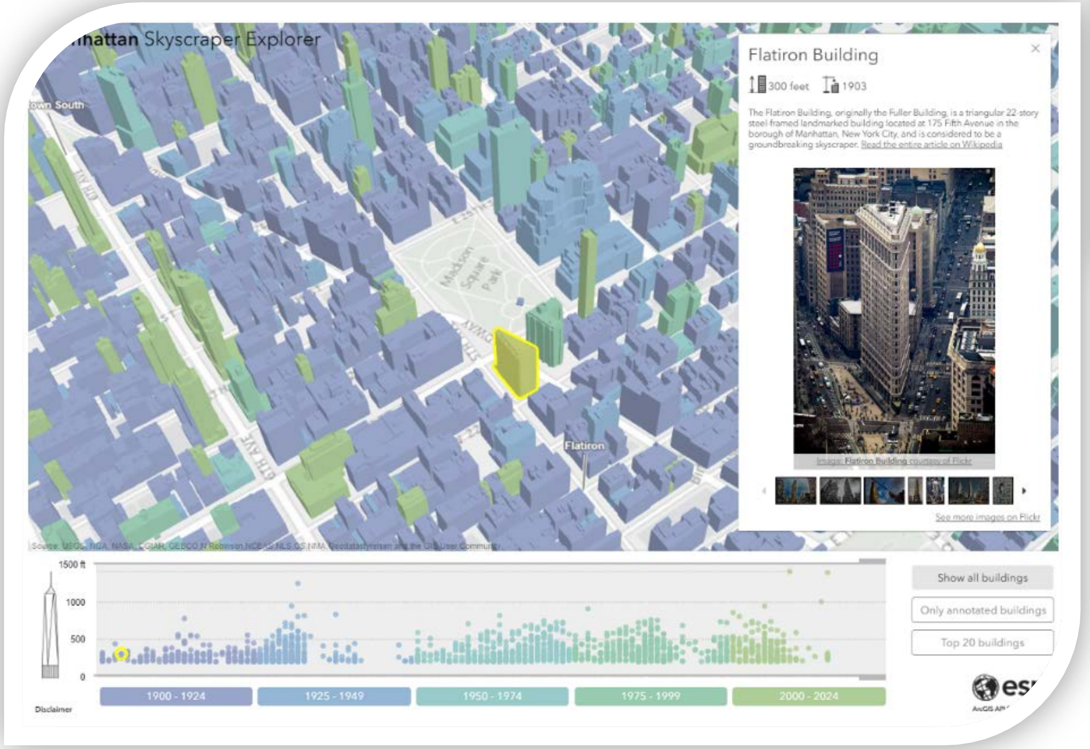
Mashing up map tiles with vector features drawn from GeoJSON and other sources.

- **ModestMaps**

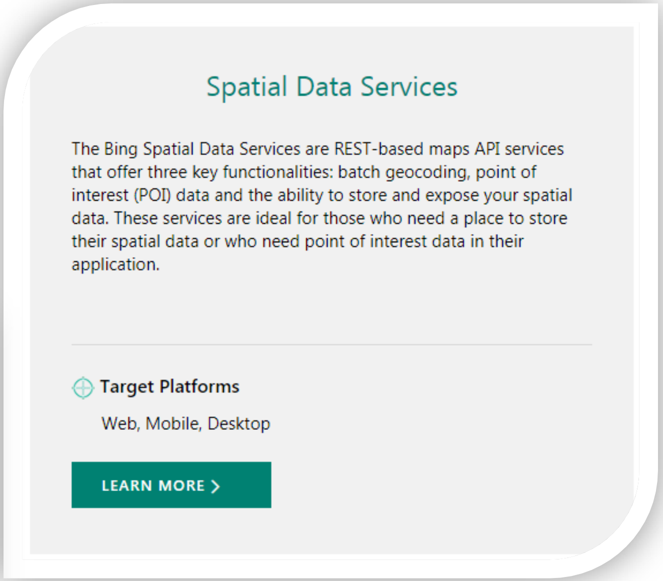
Lightweight FOSS API for displaying tiled maps

Examples of proprietary web mapping APIs

- Google Maps and Bing Maps APIs
- ArcGIS APIs
- Yahoo Map API's



The Flatiron Building, originally the Fuller Building, is a triangular 22-story steel-framed landmarked building located at 175 Fifth Avenue in the borough of Manhattan, New York City, and is considered to be a groundbreaking skyscraper. Read the entire article on Wikipedia



Spatial Data Services

The Bing Spatial Data Services are REST-based maps API services that offer three key functionalities: batch geocoding, point of interest (POI) data and the ability to store and expose your spatial data. These services are ideal for those who need a place to store their spatial data or who need point of interest data in their application.

Target Platforms

Web, Mobile, Desktop

LEARN MORE ›

What is OpenLayers?

- **OpenLayers** is a library for building mapping applications in a browser
- An API for building web map applications
- Pure client-side JavaScript
- Web 2.0 and AJAX compliant – Interoperable & Fast/Responsive
- Supports open standards.
- BSD licensed
  (implies minimum restriction on redistribution) (*Berkeley Software Distribution)*

OpenLayers History

OpenLayers appeared in the middle of 2006 as an open source alternative to Google Maps and other proprietary API providers, but it started gaining more attention in 2007, when the growing OpenStreetMap project adopted it for its website. Latest version : **v5.3.0**

Programming patterns with web mapping APIs

- Nearly all pages that use web mapping APIs include the following:

  1. References to JavaScript files and stylesheets

```
<script
src="http://cdnjs.cloudflare.com/ajax/libs/openlayers/2.12/OpenLayers.js"
></script>

<link rel="stylesheet“
href="http://cdnjs.cloudflare.com/ajax/libs/openlayers/2.12/theme/default
/style.css" type="text/css">
```

  2. The map div and object

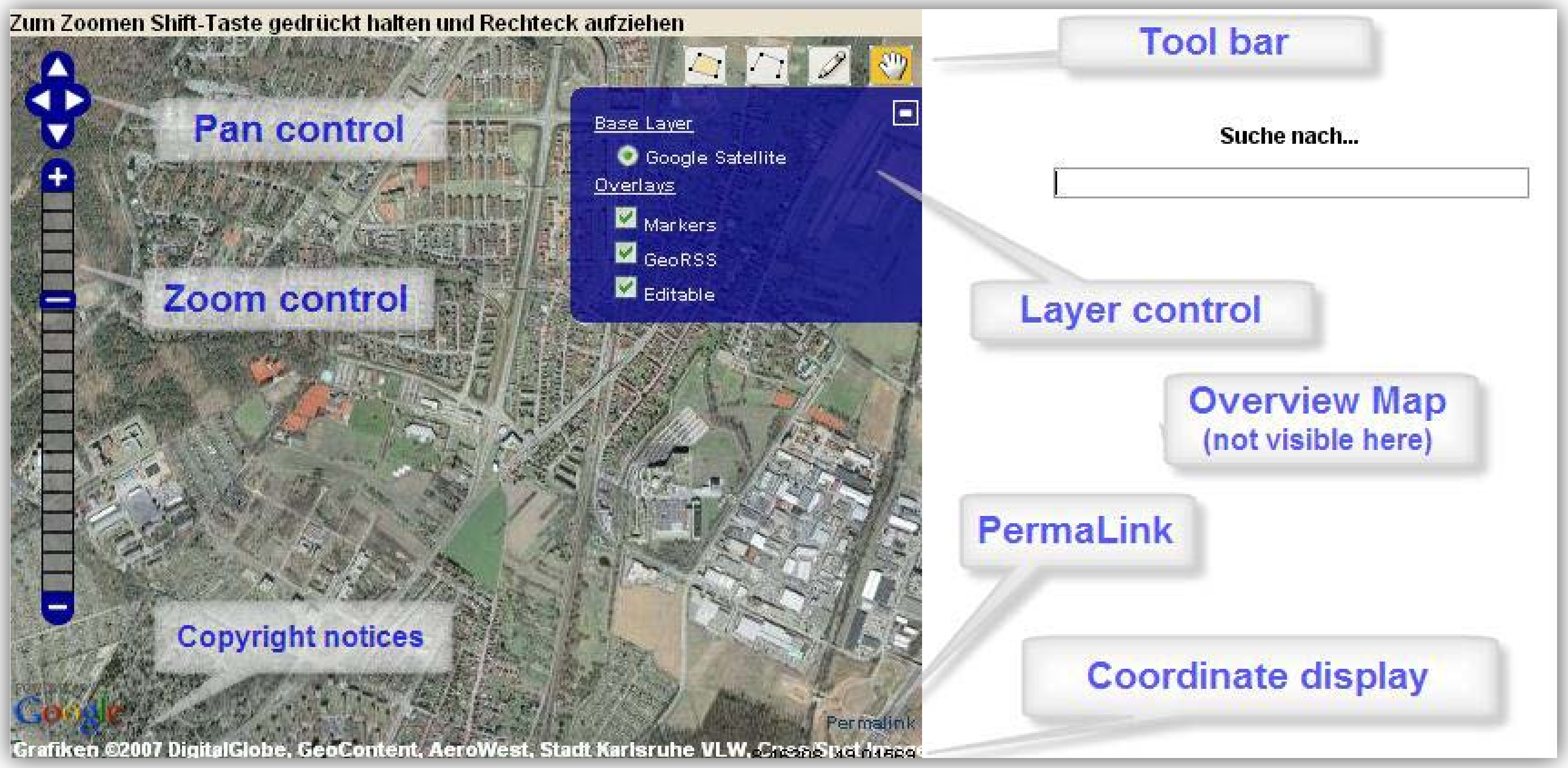
```
<div id="map"></div>
```

Elsewhere in your page, in your JavaScript code, you can create an `OpenLayers.Map` object and relate it to the html `div.`

The `OpenLayers.Map` constructor takes the `div` name as an argument.

```
<script>
var myMap;
myMap = new OpenLayers.Map("map");
</script>
```

Components of OpenLayers



OpenLayers: Data Source Support

- OGC WMS
- OGC WFS
- GeoRSS
- CSV
- ka-Map
- WorldWind
- Canvas
- WebGL
- Google Maps
- MSN Virtual Earth
- Yahoo! Maps
- Multimap

Making our first map

The process for creating a map with OpenLayers requires, at a **minimum**, the following things:

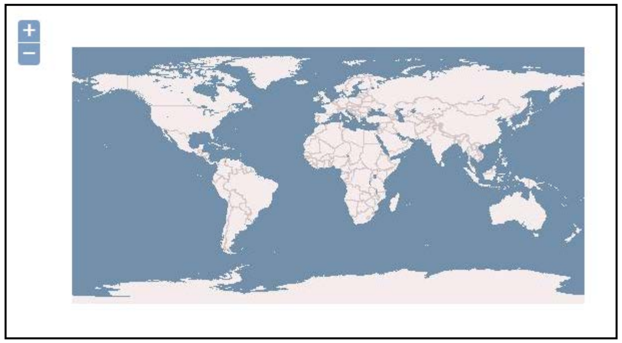
1. Including the OpenLayers library files
2. Creating an HTML element that the map will appear in
3. Creating a map object from the Map class
4. Creating a layer object from a Layer class
5. Adding the layer to the map
6. Defining the map's extent (setting the area the map will initially be displaying)

Making our first map

```
<!DOCTYPE html>
<html lang='en'>
<head>
<meta charset='utf-8' />
<title>My OpenLayers Map</title>
<script type='text/javascript'
src='https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/OpenLayers.js'></script>
<script type='text/javascript'>
 var map;
 function init() {
map = new OpenLayers.Map('map_element', {});
var wms = new OpenLayers.Layer.WMS(
'OpenLayers WMS',
'http://vmap0.tiles.osgeo.org/wms/vmap0',
{layers: 'basic'},
{}
);
map.addLayer(wms);
if(!map.getCenter()){
map.zoomToMaxExtent();
}
 }

</script>
</head>

<body onload='init();'>
<div id='map_element' style='width: 1000px; height: 500px;'>
</div>
</body>
</html>
```


Output

Making our first map

1. Including the OpenLayers library files:

Line [6]

```
 <script type='text/javascript' src='OpenLayers.js'></script>
```

2. Creating an HTML element for our map:

Lines [30] and [31]

```
 <div id='map_element' style='width: 500px; height: 500px'>
</div>
```

3. Creating a map object from the Map class:

Line [12]

```
 map = new OpenLayers.Map('map_element', { });
```

4. Creating a layer object from a Layer class:

Lines [13] to [18]

```
 var wms_layer = new OpenLayers.Layer.WMS(
 'WMS Layer Title',
 'http://vmap0.tiles.osgeo.org/wms/vmap0',
 {layers: 'basic'},
 {}
 );
```

5. Adding the layer to the map:

Line [20]

```
 map.addLayer(wms_layer);
```

6. Defining the map's extent:

Lines [21] to [23]

```
 if(!map.getCenter()){
 map.zoomToMaxExtent();
 }
```

OpenLayers Code : Adding our layers

```
<html>
<head>
<title>My Map</title>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/OpenLayers.js"></scr
ipt>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/theme/default/style
.css" type="text/css">
</head>
<body>
<h1>My Map</h1>
<div id="map-id"></div>
  <script>
        var bounds = new OpenLayers.Bounds(89.701,24.135,96.021,27.977);
        var map = new OpenLayers.Map("map-id");
        var imagery = new OpenLayers.Layer.WMS("Wastelands","https://bhuvan-
vec2.nrsc.gov.in/bhuvan/wms", {layers: "wasteland:AS_WL50K_0809", format:
"image/png"});
        map.addLayer(imagery);
        map.zoomToExtent(bounds);
  </script>
</body>
</html>
```

Task 1 : Create an html page (use Notepad/Notepad++). Insert these code and open it using a web browser

Ingredients of a map in OpenLayers

In OpenLayers, a map is a collection of layers and various controls for dealing with user interaction.
A map is generated with three basic ingredients:

- markup
- style declarations
- initialization code

Map Markup
The markup for the map in the previous example generates a single document element:

```
<div id="map-id"></div>
```

This `<div>` element will serve as the container for our map viewport. Here we use a `<div>` element, but the container for the viewport can be any block-level element. In this case, we give the container an id attribute so we can reference it easily elsewhere.

Ingredients of a map in OpenLayers

Map Style

OpenLayers comes with a default stylesheet that specifies how map-related elements should be styled.
We've explicitly included this stylesheet in the first.html page (<link rel="stylesheet" href="openlayers/theme/default/style.css" type="text/css">).

OpenLayers doesn't make any guesses about the size of your map. Because of this, following the default stylesheet, we need to include at least one custom style declaration to give the map some room on the page.

```
<link rel="stylesheet" href="OpenLayers-
2.13.1/theme/default/style.css" type="text/css">
<style> #map-id { width: 512px; height: 256px; } </style>
```

Ingredients of a map in OpenLayers

Map Initialization

The next step in generating your map is to include some initialization code. In our case, we have included a <script> element at the bottom of our document <body> to do the work:

```
<script>
var bounds = new
OpenLayers.Bounds(68.1061,6.7604,97.4152,37.0783);
var map = new OpenLayers.Map("map-id");
var imagery = new OpenLayers.Layer.WMS("Global
Imagery","http://bhuvan3.nrsc.gov.in/cgi-bin/bhuvan_satdata.exe",
{layers: "Bhuvan_satellite", format: "image/jpeg"});
map.addLayer(imagery);
map.zoomToExtent(bounds);
</script>
```

Note: The `OpenLayers.Layer.WMS` constructor requires 3 arguments and an optional fourth.

OSM in OpenLayers

```
<html>
<head>
<title>My NE Map</title>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/Open
Layers.js"></script>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/the
me/default/style.css" type="text/css">
</head>
<body>
<h1>My Map</h1>
<div id="map-id"></div>
<script>
var center = new OpenLayers.LonLat(94.0441,
25.3408).transform('EPSG:4326', 'EPSG:3857');
var map = new OpenLayers.Map("map-id", {projection: 'EPSG:3857'});
var osm = new OpenLayers.Layer.OSM();
map.addLayer(osm);
map.setCenter(center, 9);
</script> </body> </html>
```

Task 2 : Create second html page and repeat as before

Proprietary Layers

Bing Map in OpenLayers

```
<html>
<head>
<title>My Map</title>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/OpenLayers.js"></scri
pt>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/theme/default/style.
css" type="text/css">
</head>
<body>
<h1>My Map</h1>
<div id="map-id"></div>
<script>
var center = new OpenLayers.LonLat(94.0441, 25.3408).transform('EPSG:4326',
'EPSG:3857');
var map = new OpenLayers.Map("map-id", {projection: 'EPSG:3857'});
var osm = new OpenLayers.Layer.OSM(); map.addLayer(osm);
var bing = new OpenLayers.Layer.Bing({key:
"cIOrzqir2awTVsM9SjjW~T8BBXo4E_enRaCFUdXvsGg~ArMNRFEJHx4LuNlhBbaGxzdY7zKQIJda7N4iyT
rVWObFpYQk6saPPV-CDoF8PcLh",type: "Road",});
map.addLayer(bing);
map.addControl(new OpenLayers.Control.LayerSwitcher());
map.setCenter(center, 9)
</script>
</body>
</html>
```

Adding Control to Map

Task 3 : Using Bingmap on your application and adding controls

Working With Controls

Task 4: Creating an Overview Map

```
<html>
<head>
<title>My Map</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/OpenLayers.js"></script>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/openlayers/2.13.1/theme/default/style.css"
type="text/css">
</head>
<body>
<h1>My Map</h1>
<div id="map-id"></div>
<script>
var bounds = new OpenLayers.Bounds(68.1061,6.7604,97.4152,37.0783).transform('EPSG:4326',
'EPSG:3857');
var center = new OpenLayers.LonLat(91.882754,  25.576002).transform('EPSG:4326', 'EPSG:3857');
var map = new OpenLayers.Map("map-id", {projection: new OpenLayers.Projection("EPSG:3857"),
maxResolution: 0.0005, numZoomLevels: 5});
var osm = new OpenLayers.Layer.OSM();
var overview = new OpenLayers.Control.OverviewMap({mapOptions: { projection: new
OpenLayers.Projection("EPSG:4326"), maxResolution: 0.0015, numZoomLevels: 5 } });
map.addControl(overview);
map.addLayer(osm);
map.zoomToExtent(bounds);
map.setCenter(center,17);
</script>
</body>
</html>
```

Working With Controls

Creating a ScaleLine Control

```
var scaleline = new OpenLayers.Control.ScaleLine();
map.addControl(scaleline);
```

Editing Toolbar

```
var map = new OpenLayers.Map("map-id", {projection: 'EPSG:3857'});
var osm = new OpenLayers.Layer.OSM();
map.addControl(new OpenLayers.Control.EditingToolbar(osm));
map.addLayer(osm);
```

Overlay Example

```
<html>
<head>
<title>Assam Map</title>
<link rel="stylesheet"href="http://dev.openlayers.org/theme/default/style.css">
<style>#map-id { width: 100%; height:100%; } </style>
<script src="http://dev.openlayers.org/OpenLayers.js"type="text/javascript"> </script>
</head>
<body>
<h1>Land Use Map</h1>
<div id="map-id"></div>
<script>
var bounds =new OpenLayers.Bounds(87.577066, 20.400794, 97.863033, 29.773545);
var map =new OpenLayers.Map("map-id");
var imagery =new OpenLayers.Layer.WMS("Bhuvan Base","https://bhuvan-
ras2.nrsc.gov.in/tilecache/tilecache.py",{layers:"bhuvan_img",format:"image/png"},{isBaseLayer: true, transparent:
true, singleTile: false, visibility: true},{projection:'EPSG:4326'});
var assam = new OpenLayers.Layer.WMS("Assam Land Use","https://bhuvan-
vec2.nrsc.gov.in/bhuvan/wms",{layers:"lulc:AS_LULC50K_1112",format:"image/png"},{isBaseLayer: false, opacity: 0.5,
singleTile: true, visibility: false},{projection:'EPSG:4326'});
var megh = new OpenLayers.Layer.WMS("Megh Land Use","https://bhuvan-
vec2.nrsc.gov.in/bhuvan/wms",{layers:"lulc:ML_LULC50K_1112",format:"image/png"},{isBaseLayer: false, opacity: 0.5,
singleTile: true, visibility: false},{projection:'EPSG:4326'});

map.addLayers([assam, megh,imagery]); //map.addLayer(imagery);
map.zoomToExtent(bounds);
map.addControl(new OpenLayers.Control.LayerSwitcher());
var scaleline=new OpenLayers.Control.ScaleLine();
map.addControl(scaleline);
map.addControl(new OpenLayers.Control.EditingToolbar(map));
</script>
</body>
</html>
```

Adding Bhuvan Satellite imagery as Base Map and overlayed with Thematic layers

OSM in OpenLayers

Exercise

1. Review the OSM layer API documentation to how to load other OSM layers
2. Modify your layer initialization according

Hint:

You can go to the official OSM site to view the layers available, change to any of them and use the browser tools to look for the url pattern of those tiles.

Leaflet Example

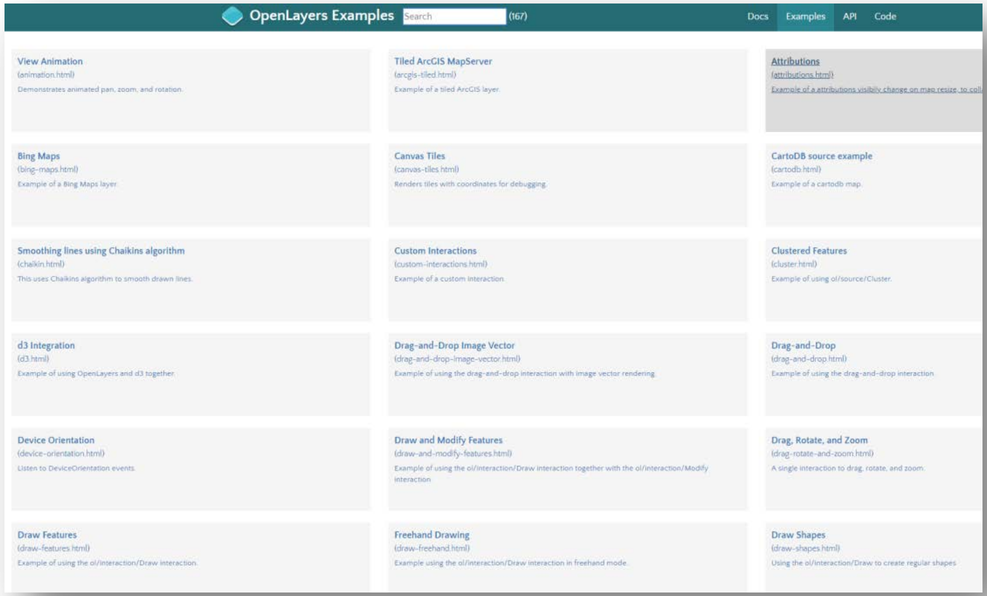
```
<html>
  <head>
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.4.0/dist/leaflet.css" integrity="sha512-
puBpdR0798OZvTTbP4A8Ix/l+A4dHDD0DGqYW6RQ+9jxkRFclaxxQb/SJAWZfWAkuyeQUytO7+7N4QKrDh+drA=="
crossorigin=""/>
    <script src="https://unpkg.com/leaflet@1.4.0/dist/leaflet.js" integrity="sha512-
QVftwZFqvtRNi0ZyCtsznlKSWOStnDORoefr1enyq5mVL4tmKB3S/EnC3rRJcxCPavG10IcrVGSmPh6Qw5lwrg=="
crossorigin=""></script>
    </head>
  <style>#mapid
    {
      width: 100%;
      height:100%;
    }
  </style>

  <body>
    <div id="mapid"></div>
    <script>
      var mymap = L.map('mapid',{zoom: 3});
      var wmsLayer = L.tileLayer.wms('https://bhuvan-vec2.nrsc.gov.in/bhuvan/wms', {
                        layers: 'lulc:MN_LULC50K_1112',
                        transparent: true,
                        maxZoom: 14,
                        minZoom: 0,
                        format:'image/png',
                        version: '1.1.1',
                        attribution: “NESAC”
            }).addTo(mymap);
     var corner1 = L.latLng(25.768523, 92.778660),corner2 = L.latLng(23.491038, 94.468081), bounds =
L.latLngBounds(corner1, corner2);
            mymap.fitBounds(bounds);
    </script>

  </body>
</html>
```

More OpenLayers / Leaflet Examples

https://openlayers.org/en/latest/examples/



OpenLayers Examples Search (167) Docs Examples API Code

View Animation
(animation.html)
Demonstrates animated pan, zoom, and rotation.

Tiled ArcGIS MapServer
(arcgis-tiled.html)
Example of a tiled ArcGIS layer.

Attributions
(attributions.html)
Example of a attributions visibily change on map resize, to coll

Bing Maps
(bing-maps.html)
Example of a Bing Maps layer.

Canvas Tiles
(canvas-tiles.html)
Renders tiles with coordinates for debugging.

CartoDB source example
(cartodb.html)
Example of a cartodb map.

Smoothing lines using Chaikins algorithm
(chaikin.html)
This uses Chaikins algorithm to smooth drawn lines.

Custom Interactions
(custom-interactions.html)
Example of a custom interaction.

Clustered Features
(cluster.html)
Example of using ol/source/Cluster.

d3 Integration
(d3.html)
Example of using OpenLayers and d3 together.

Drag-and-Drop Image Vector
(drag-and-drop-image-vector.html)
Example of using the drag-and-drop interaction with image vector rendering.

Drag-and-Drop
(drag-and-drop.html)
Example of using the drag-and-drop interaction.

Device Orientation
(device-orientation.html)
Listen to DeviceOrientation events.

Draw and Modify Features
(draw-and-modify-features.html)
Example of using the ol/interaction/Draw interaction together with the ol/interaction/Modify interaction.

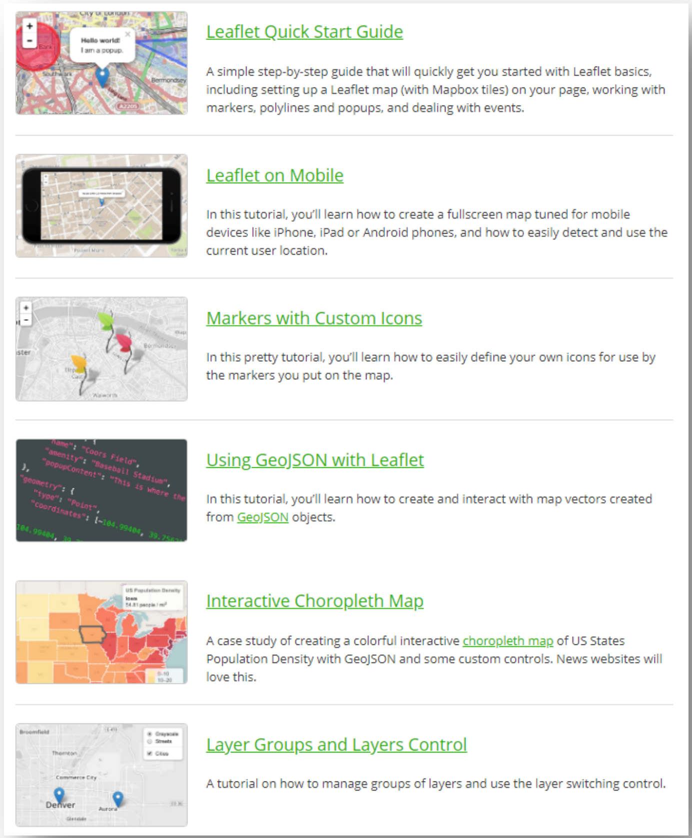
Drag, Rotate, and Zoom
(drag-rotate-and-zoom.html)
A single interaction to drag, rotate, and zoom.

Draw Features
(draw-features.html)
Example of using the ol/interaction/Draw interaction.

Freehand Drawing
(draw-freehand.html)
Example using the ol/interaction/Draw interaction in freehand mode.

Draw Shapes
(draw-shapes.html)
Using the ol/interaction/Draw to create regular shapes.

https://leafletjs.com/examples.html



Leaflet Quick Start Guide

A simple step-by-step guide that will quickly get you started with Leaflet basics, including setting up a Leaflet map (with Mapbox tiles) on your page, working with markers, polylines and popups, and dealing with events.

Leaflet on Mobile

In this tutorial, you'll learn how to create a fullscreen map tuned for mobile devices like iPhone, iPad or Android phones, and how to easily detect and use the current user location.

Markers with Custom Icons

In this pretty tutorial, you'll learn how to easily define your own icons for use by the markers you put on the map.

Using GeoJSON with Leaflet

In this tutorial, you'll learn how to create and interact with map vectors created from GeoJSON objects.

Interactive Choropleth Map

A case study of creating a colorful interactive choropleth map of US States Population Density with GeoJSON and some custom controls. News websites will love this.

Layer Groups and Layers Control

A tutorial on how to manage groups of layers and use the layer switching control.