



VIS
2018

Tutorial: Urban Trajectory Visualization

Visual System Implementation

Shamal AL-Dohuki and Ye Zhao





- **Shamal AL-Dohuki**
- **Ph.D. candidate** in the Department of Computer Science at Kent State University, Ohio, USA.
- **Software Development Lead** of **TrajAnalytics**
- My current **research interest** includes:
 - Implementing visual analytics of big urban data
 - Urban data management and visualization
 - Visual query of trajectory data
 - Semantic data query and analytics



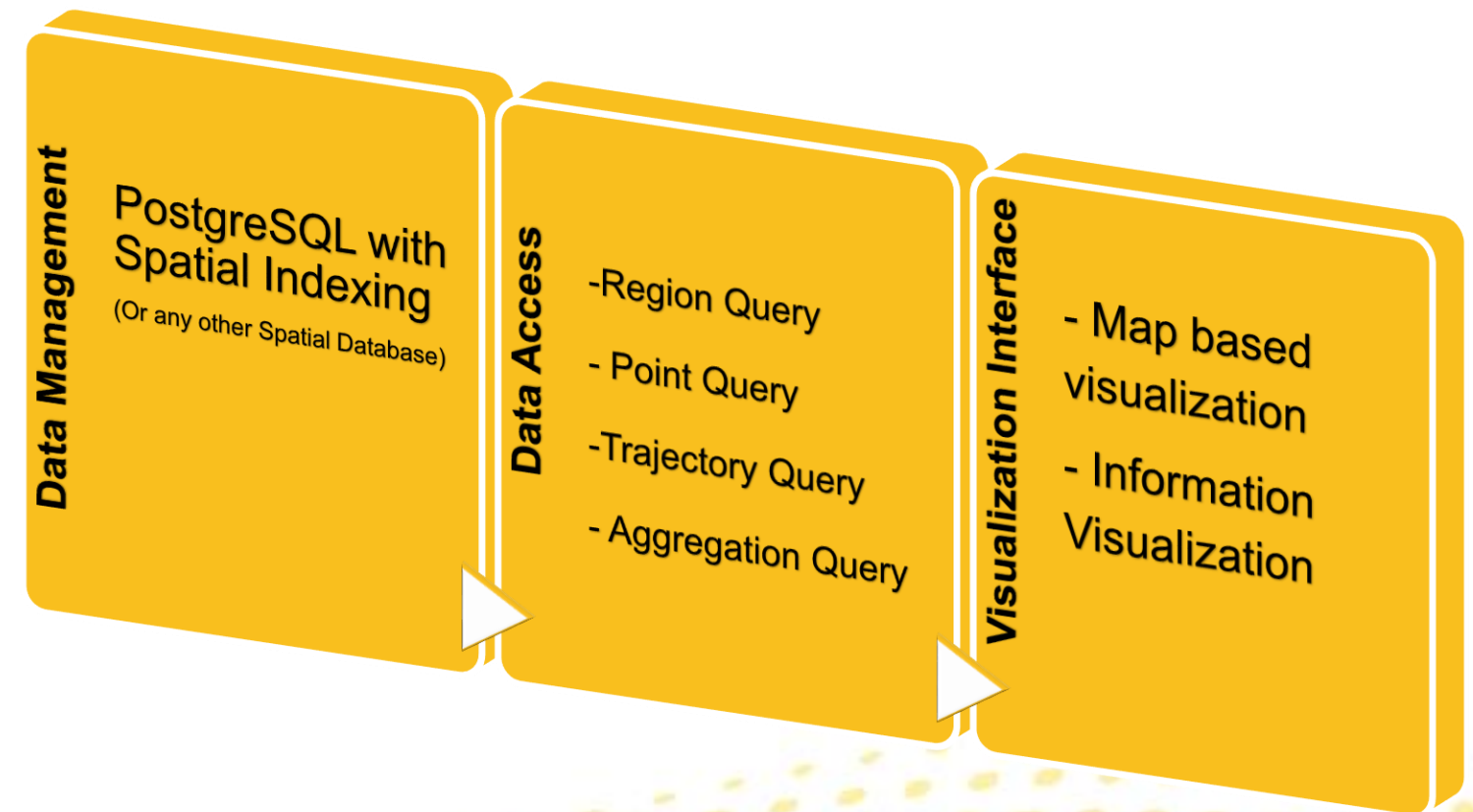
Overview

- Exploratory visualization systems for urban trajectory data
 - Efficient user interaction
 - Instant visual feedback.
- Support practitioners, researchers, and decision-makers to **store**, **query** and **visualize** the data



Overview Cont.

- The system should integrate **scalable data management** and **interactive visualization** with **powerful computational capability**.
 - Powerful computing platform.
 - Easy access gateway.
 - Scalable data storage and management.
 - Exploratory visualization.



Web based Visual System

- Web browser based visual analytics is the trend
 - Cost effective and fit for multiple platforms
 - Easy installation and maintenance
 - Secure with easy account management and access control
- Built on client-server scheme
- Data uploading and remote management
- We focus on web-based visual system in this tutorial



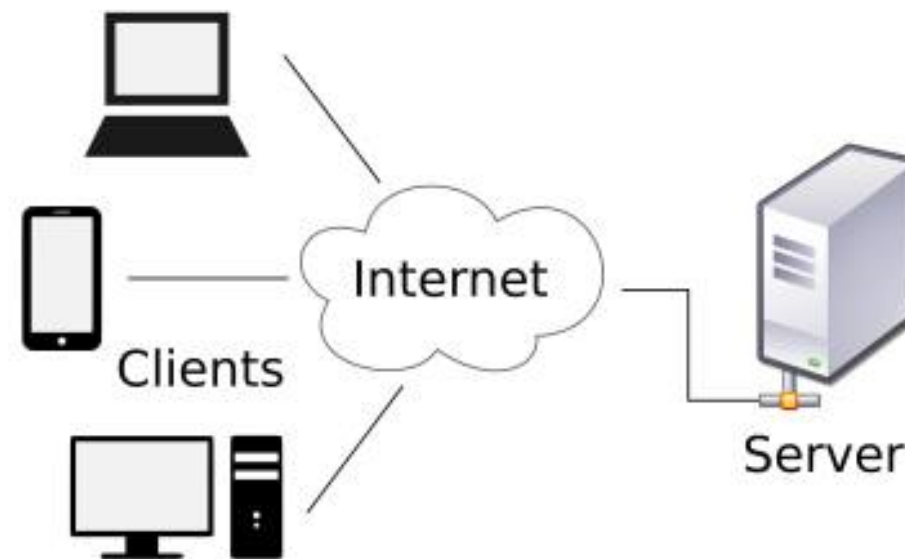
Web System Challenges

- Web Services are web accessible applications and application components that exchange data, share tasks, and automate processes over the Internet.
- To build a web service for visual exploration of trajectory data, researchers face some general issues:
 - How to transfer data between the client and the server and in what format?
 - How to visualize trajectory data on the client side?
 - What views and interactions are needed?
 - What server side and database supports are needed?



Client and Server Scheme

- Datasets are stored and controlled by the remote data server.
- Users send requests and receive information from the server via a client.
- Some functions are performed on both server and client side.
- There is a trade-off between performing functions on the database server and on the client side.

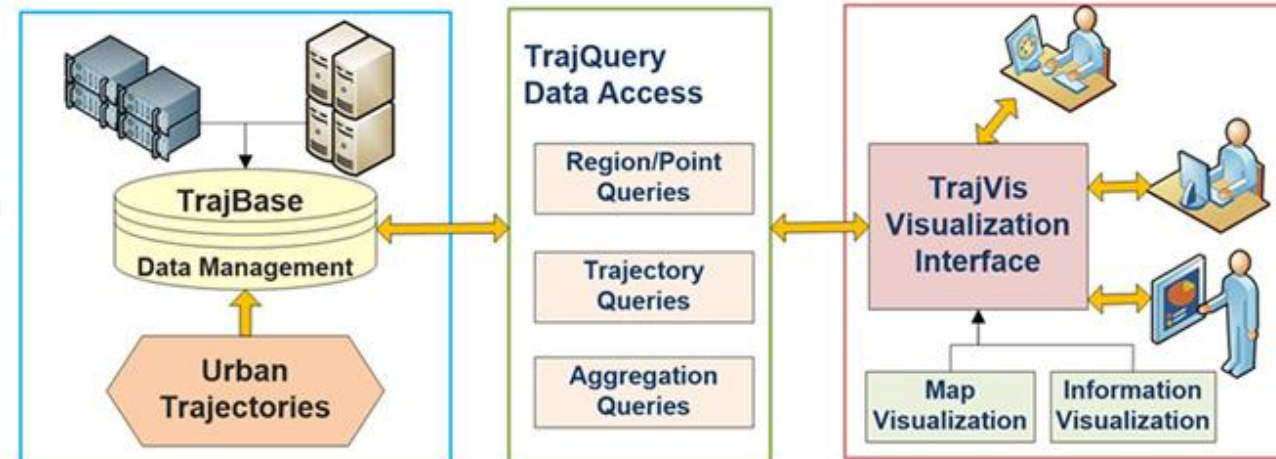


An Example Web-based Software System

- **TrajAnalytics** is a web based software for visually exploring urban trajectories

<http://vis.cs.kent.edu/TrajAnalytics/>

TrajAnalytics Software



Scalable Data Management - Interactive Visualization - Powerful Computational Capability



TrajAnalytics Features

- A cloud-based software which can be accessed through Web browsers.
- Users can
 - Load raw trajectory data
 - Perform map matching and aggregation
 - Conduct extensive visual analysis at any time
- A downloadable version is available for users to use it in local machines
 - Localhost is used



TrajAnalytics Demo



The screenshot shows the TrajAnalytics website interface. At the top, it says "TRAJECTORY ANALYTICS PROJECT" with navigation links for HOME, FEATURES, SUPPLEMENTAL VIDEOS, and TEAM. On the left, there is a logo for vis.cs.kent.edu. The main content area features the title "TrajAnalytics" and a subtitle "Web-based Software System for Visual Analysis of Urban Trajectory Data". Below this is a screenshot of the software's interface, which includes a map of a city with numerous colored trajectory lines, a table of statistics, and two bar charts. The table has columns for "City", "The Number of Trajectories", and "Number of Stops". The bar charts show the distribution of trajectories and stops. Below the screenshot, there is a section titled "TrajAnalytics: A Free Software for Visually Exploring Urban Trajectories" with a paragraph of text and a link to a guideline of usage. At the bottom, there is a logo for the National Science Foundation and the text "This work is supported by National Science Foundation ACI-1535031,1535081."

vis.cs.kent.edu

TrajAnalytics

Web-based Software System for Visual Analysis of Urban Trajectory Data

TrajAnalytics: A Free Software for Visually Exploring Urban Trajectories

Thanks to advanced technologies in sensing and computing, the mobility patterns and dynamics of urban cities and their citizen are recorded and manifested in a variety of urban trajectory datasets, which include the moving paths of human, taxi, bus, fleets, cars, and so on. Understanding and analyzing such large-scale, complex data is of great importance to enhance both human lives and urban environments. Supported by National Science Foundation, TrajAnalytics aims to provide exploratory data visualization tools for researchers, administrations, practitioners and general public to understand the data and to reveal knowledge intuitively.

A guideline of usage can be accessed [HERE](#) and can be downloaded in .pdf format [HERE](#).

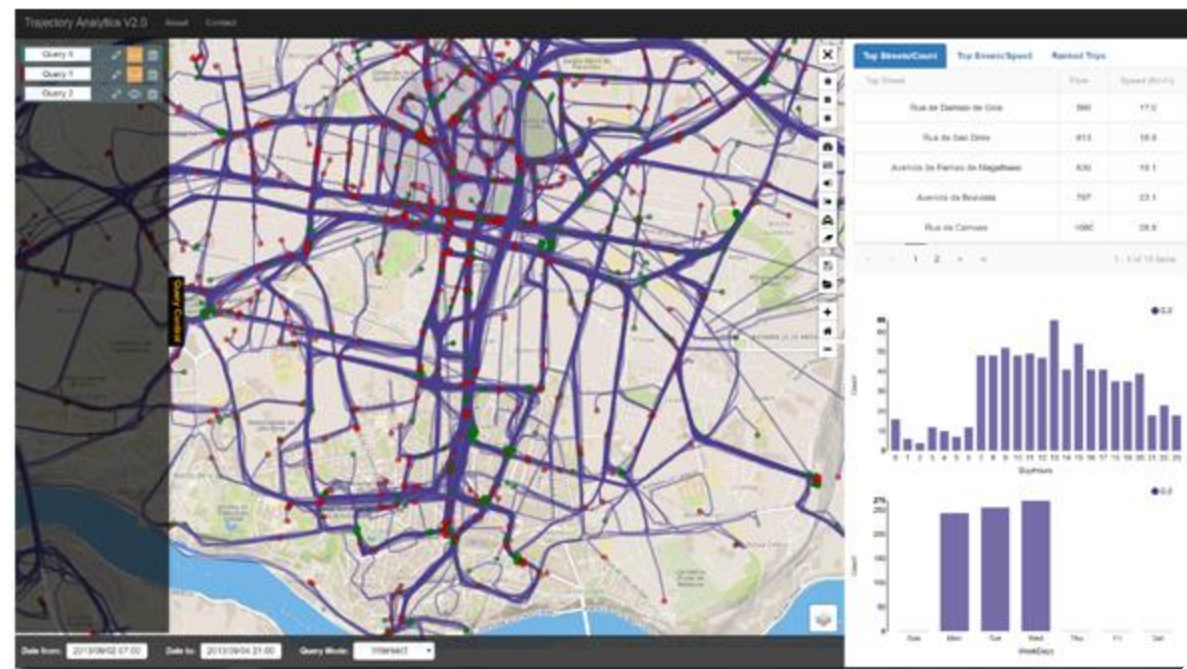
This work is supported by National Science Foundation ACI-1535031,1535081.



Visual System Implementation

- In next slides, we describe the details of implementing web based visualization system for urban trajectory datasets
 - Mostly based on our work in TrajAnalytics design and implementation

TrajAnalytics
Web-based
Software System
for Visual Analysis of
Urban Trajectory Data



Web Based System

- **Server:** Spatial Database with Spatial Indexing.
 - PostgreSQL, MySQL, MongoDB
- **Client:** HTML, PHP with HTTP, accessing with any internet browser.
- **Data Transfer:** jQuery AJAX request to PHP and PostgreSQL.
 - The data will be restored in the .json object and will be transferred to the client side.
- **Web interface:** JavaScript with multiple libraries.
 - Leaflet.js, D3.js, and more.



PostgreSQL Database

- **PostgreSQL** is an object-relational database system.
- It has the features of a traditional proprietary database system with enhancements to be found in next-generation DBMS systems.
- **PostgreSQL** has most features that are present in large proprietary DBMSs, like transactions, sub selects, triggers, views, foreign key referential integrity, and sophisticated locking.



PostGIS for Spatial Datasets

- **PostgreSQL** also offers some special features like user-defined types, inheritance, rules, and multi-version concurrency control to reduce lock contention.
- **PostGIS** is a spatial database extender for PostgreSQL.
- It adds support for **geographic objects allowing location queries to be run in SQL.**

```
SELECT superhero.name
FROM city, superhero
WHERE ST_Contains(city.geom, superhero.geom)
AND city.name = 'Gotham';
```

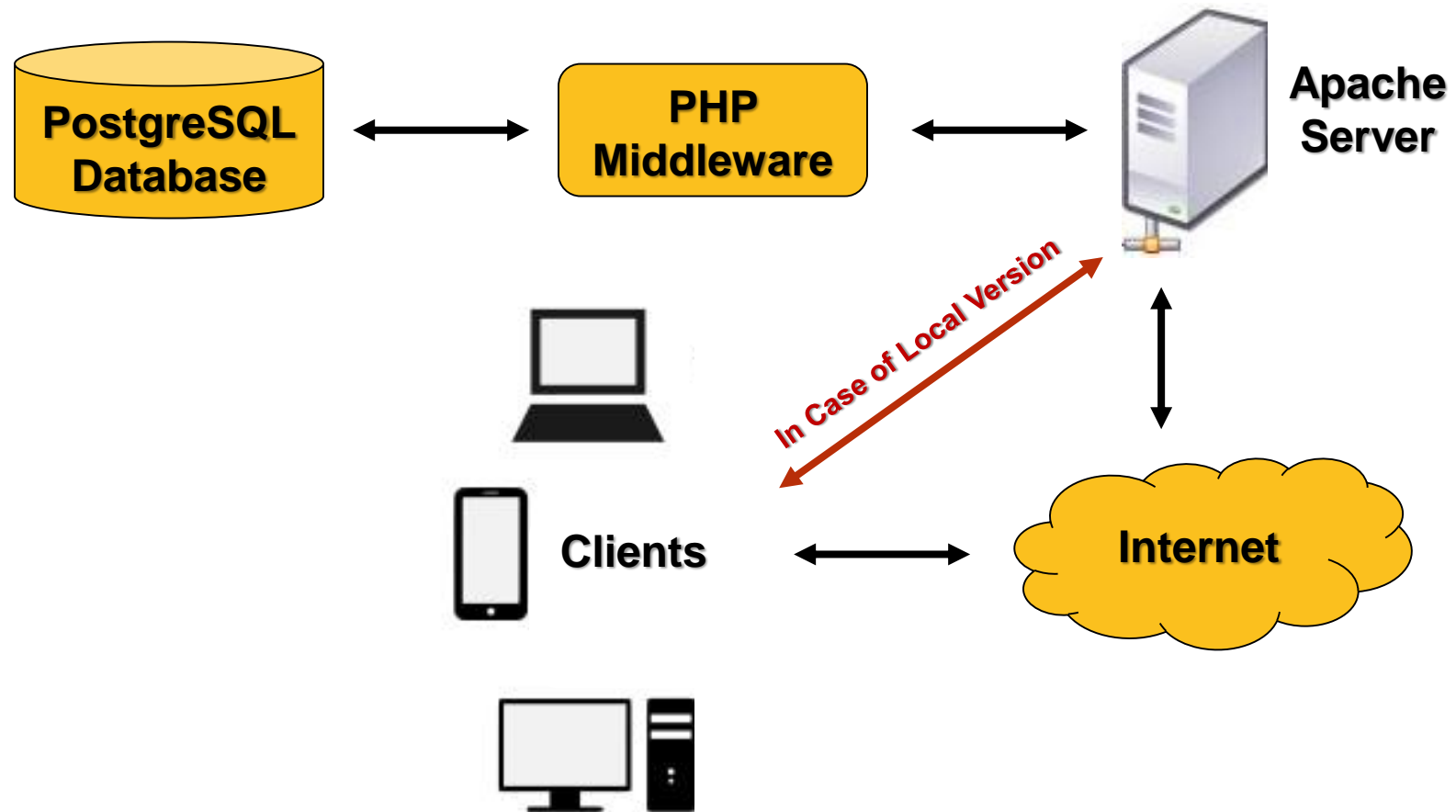


Network Communication

- Challenging issues:
 - How to connect the client with the database (**PostgreSQL**).
 - How to extract data from the database and transfer the data to the client side.
- **PHP** and the **Apache** server are used for this purpose.
- Trajectory data in the database are read by PHP through the Apache server.
- The data will be restored in the **.JSON** object and will be transferred via the internet to the client side.



System Structure



Connecting with the database by using PHP and Apache server



Web Servers

- There are many types of web servers available.
- **Apache server** is the most common server around, very easy to set up, and compatible with all major operating systems.
- **On Linux:** Installation and configuration of Apache webserver can be done in one step.

» `sudo apt-get install apache2`



Web Servers

- **On Mac OS:** Apache is installed by default. All you need to do is turn it on. Go to “Terminal ” and run the following command to turn on your already pre-installed Apache web server.
 - » **`sudo apachectl start`**
- **On Windows:** there are several install wizards that bundle things like Apache, MySQL, and PHP together to make our lives easier. One of them is **XAMPP**.
- **Note:** XAMPP is available for Linux and Mac OS X too. (**Recommended**)

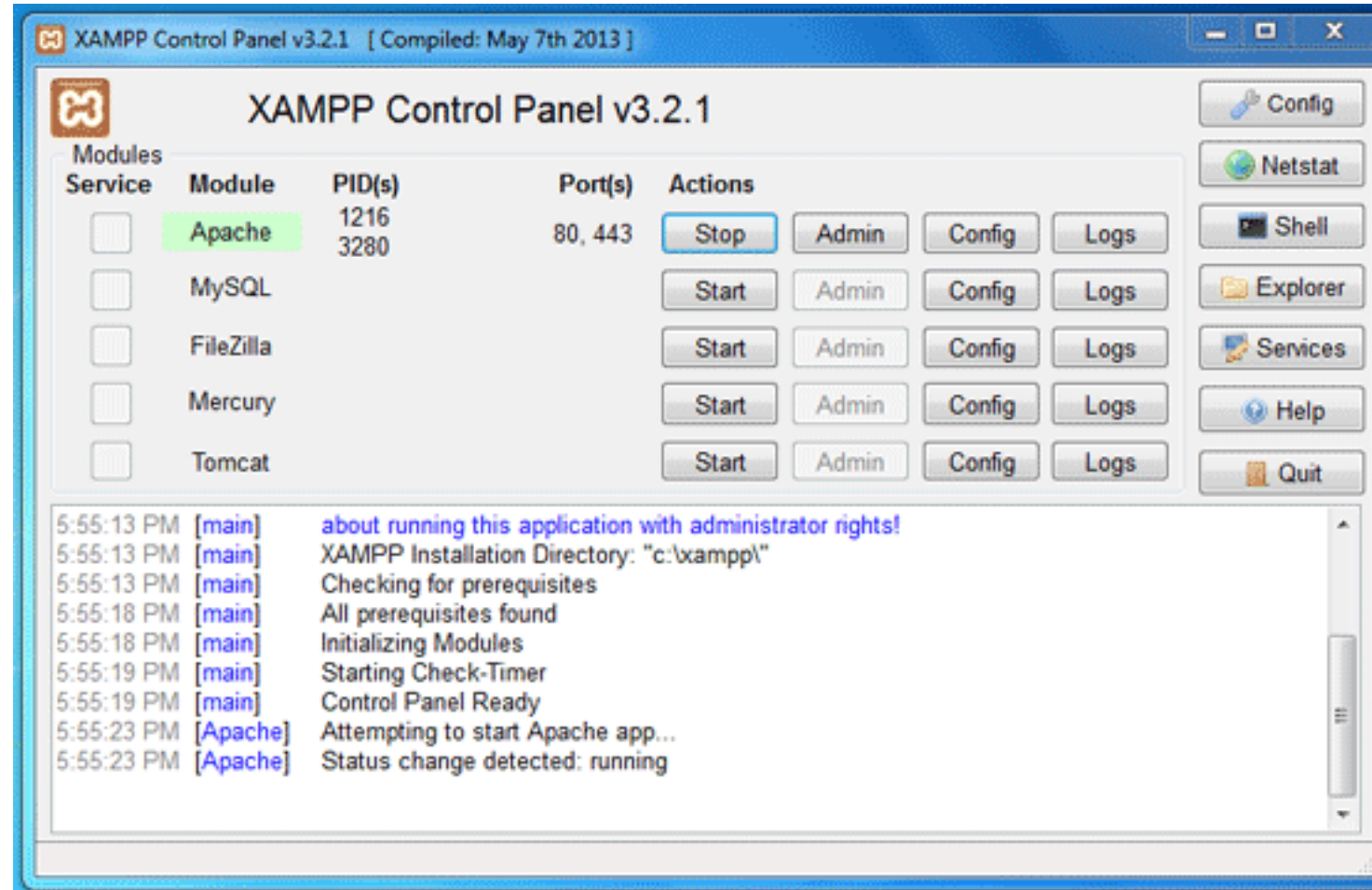


XAMPP and PHP

- **XAMPP** bundle tools like **Apache**, **MySQL**, and **PHP** together.
- **PHP** stands for Hypertext Preprocessor.
- **PHP** is a server side scripting language, like Microsoft's Active server page (ASP).
- **PHP** is open source software and widely used for making dynamic and interactive Web pages.
- **PHP** uses **pg_connect** to speak to the **PostgreSQL** database on the server.



XAMPP Control



XAMPP Control Panel: Bundle **Apache**, **MySQL**, and **PHP** together



Connect to Database

- To connect to PostgreSQL database using PHP, a **pg_connect** function with connection strings that consists of host name, name of the database, user and password was performed.
- The following codes describe how to connect to PostgreSQL using PHP:

```
<?php
```

```
$db_connection = pg_connect("host=localhost dbname=DBNAME user=USERNAME password=PASSWORD");
```

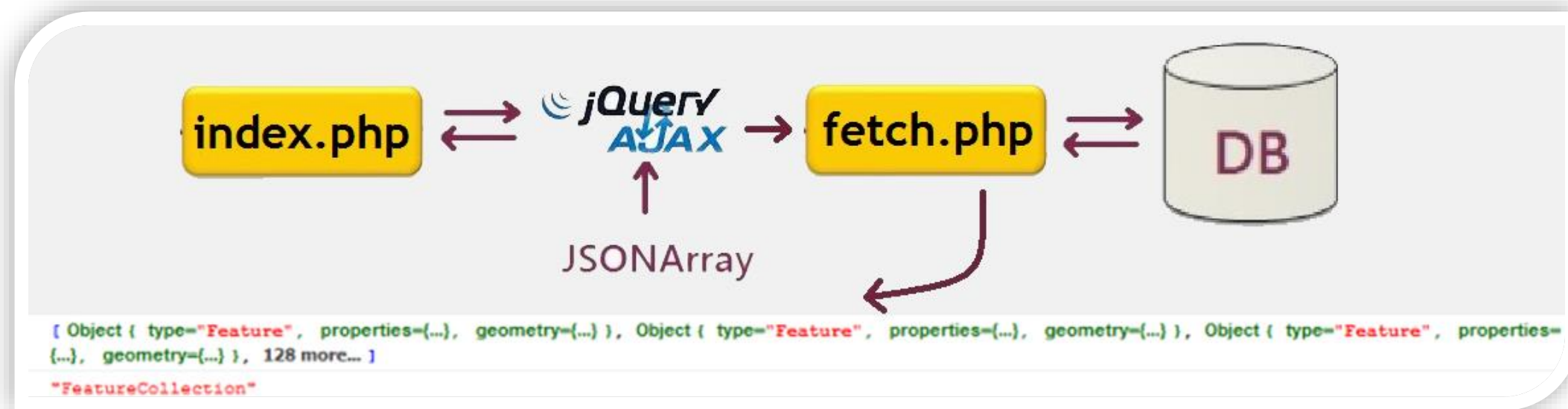
```
$result = pg_query($db_connection, "SELECT lastname FROM employees");
```

```
?>
```



Data Queries and Transfer

- In a web based visual exploration of trajectory data, jQuery AJAX request to PHP and PostgreSQL could be used.
- The data will be restored as a JSON object and will be transferred to the client side.

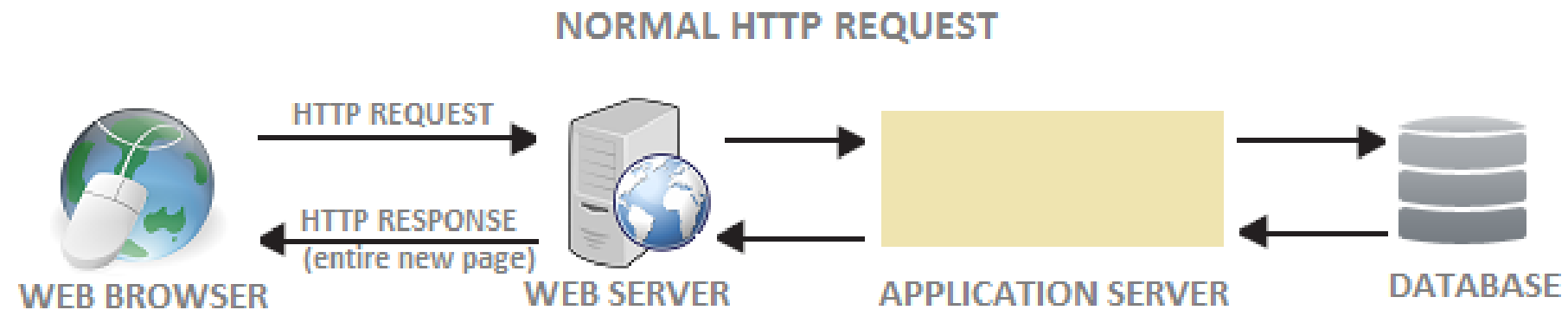


Data Queries and Transfer Cont.

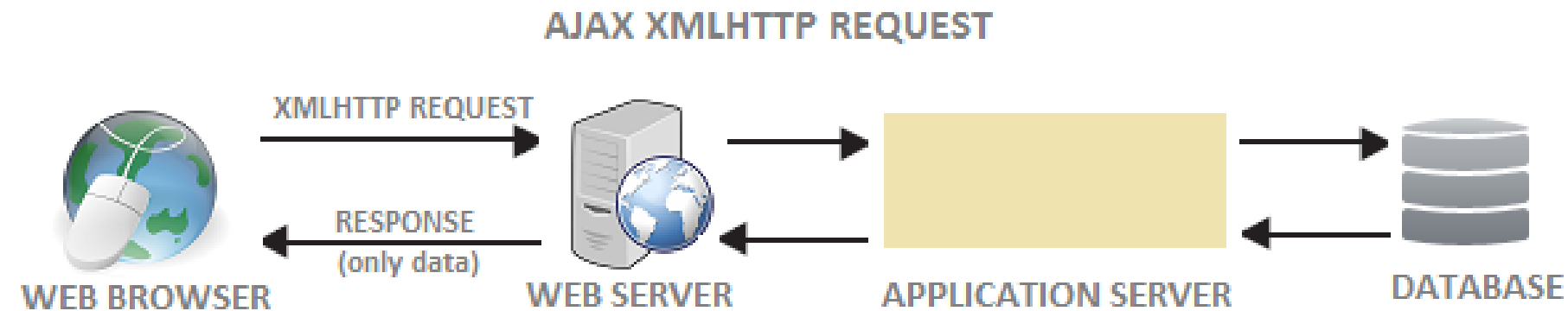
- **Ajax** (also AJAX) short for "**Asynchronous JavaScript And XML**".
- Set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications.
- With Ajax, Web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.
- Modern implementations commonly utilize **JSON** instead of **XML** due to the advantages of **JSON** being native to JavaScript.



Data Queries and Transfer Cont.



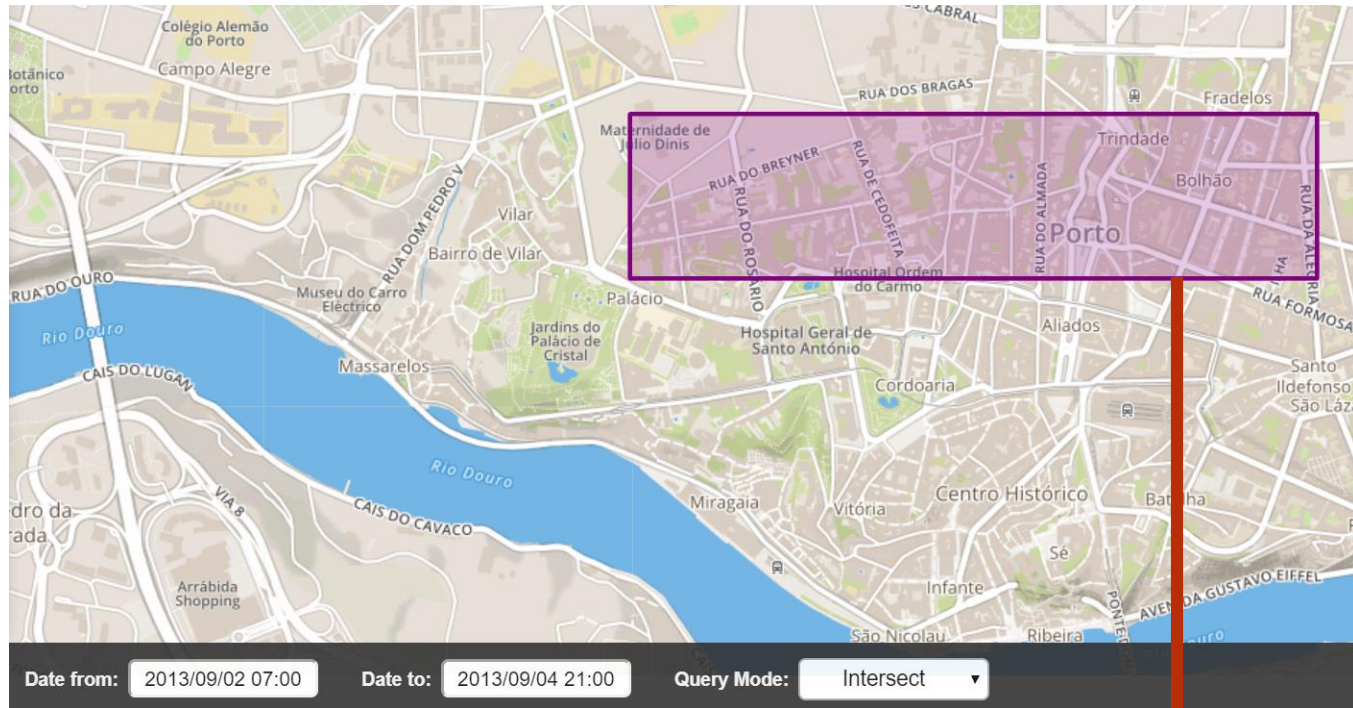
@DotNetCurry.com



Normal HTTP request VS. Ajax Request



Data Queries Example



Temporal Constraint

Spatial Constraint

```
$.post("/query.php", {  
  //Sending Spatial+Temporal Constraints to the Database for the Query  
  parameters: Spatial+Temporal Constraints,  
}, function(results) {  
  // the output of the response is now handled via a variable call 'results'  
  if (results) {  
    var obj = [];  
    obj = JSON.parse(results);  
    console.log(obj);  
  } else {  
    console.log("No Results");  
  }  
});
```

Example: Using AJAX to do Region Query



Data Queries Example Cont.

```
<?php
ini_set('display_errors', 1);

//database login info
$host = 'localhost';
$port = '5432';
$dbname = 'sandwikimap2';
$user = 'postgres';
$password = 'postgis';

$conn = pg_connect("host=$host port=$port dbname=$dbname user=$user password=$password");
if (!$conn) {
    echo "Not connected : " . pg_error();
    exit;
}
```

Connect to the Database

Insert Your Query here

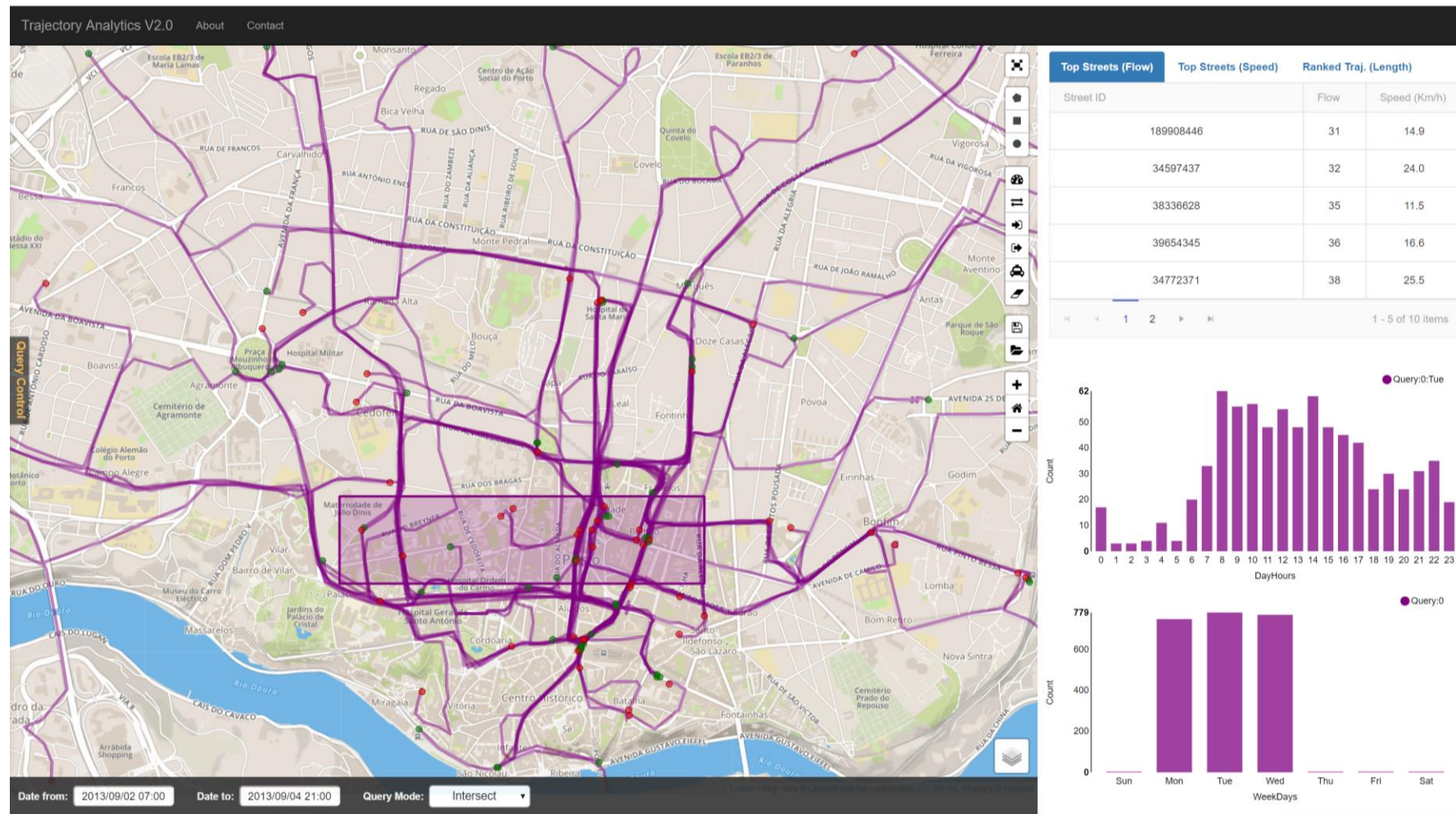
query.php

```
44, prengaman, Unimin 1, mine, plant, operational, 720, {"type": "Point", "coordinates": [-89.426681460011793
, 43.5254328998351]}, ;80, prengaman, Unimin 2, mine, plant, operational, 436, {"type": "Point", "coordinates"
: [-90.567510790073698, 44.005761689830784]}, ;46, prengaman, Pattison Sand Co. Rail Facility, rail, operational
, {"type": "Point", "coordinates": [-91.148600000151561, 43.04122999959845]}, ;88, prengaman, Milestone
Materials (Mathy Construction Co.) 3, mine, operational, {"type": "Point", "coordinates": [-92.155731219911189
, 44.95352732842084]}, ;89, prengaman, Milestone Materials (Mathy Construction Co.) 4, mine, operational
, 114, {"type": "Point", "coordinates": [-92.147588999916849, 44.950630998435763]}, ;84, prengaman, Wisconsin
Industrial Sands Hager City, plant, operational, {"type": "Point", "coordinates": [-92.424654189718012
, 44.664129017884846]}, ;85, prengaman, Wisconsin Industrial Sands Trenton, plant, permitted, {"type":
"Point", "coordinates": [-92.539716999584343, 44.602942997609439]}, ;87, prengaman, Wisconsin Industrial
Sands Diamond Bluff, mine, permitted, 1867, {"type": "Point", "coordinates": [-92.635515999421017, 44.666168897347717
]}, ;76, prengaman, Hungry Run Cranberry, mine, operational, 9, {"type": "Point", "coordinates": [-90.536993150070373
, 44.081993949838804]}, ;13, prengaman, Superior Silica Sands Mine, mine, proposed, {"type": "Point"
, "coordinates": [-91.943090989979822, 45.398717688752974]}, ;14, prengaman, Superior Silica Sands Plant
1, plant, rail, operational, 84, {"type": "Point", "coordinates": [-91.93568269998211, 45.408911798763675
]}, ;15, prengaman, Superior Silica Sands Plant 2, plant, rail, operational, 40, {"type": "Point", "coordinates"
: [-91.566731520086748, 45.209859959227074]}, ;4, prengaman, Chieftain Sand and Proppants Mine, mine, operational
, 40, {"type": "Point", "coordinates": [-91.578394840084755, 45.223613349214865]}, ;5, prengaman, Chieftain
Sand and Proppants Plant, plant, rail, operational, {"type": "Point", "coordinates": [-91.602908000080021
, 45.257065999188562]}, ;9, prengaman, Sandy Bruder Mine, mine, permitted, 40, {"type": "Point", "coordinates"
: [-91.981810049968516, 45.330327618696508]}, ;10, prengaman, Sandy Bruder Plant, mine, plant, permitted
, 160, {"type": "Point", "coordinates": [-92.003533719964892, 45.258903628665372]}, ;1, prengaman, 10K International
Arland, mine, permitted, 105, {"type": "Point", "coordinates": [-92.009255289957608, 45.304682148654258
]}, ;2, prengaman, Cameron rail Site, plant, rail, proposed, {"type": "Point", "coordinates": [-91.740582000046842
, 45.395454999028274]}, ;3, prengaman, Canadian Sand and Proppants, mine, plant, operational, 160, {"type":
"Point", "coordinates": [-91.607389700071238, 45.410280689179814]}, ;6, prengaman, EOG Resources, mine
, permitted, 336, {"type": "Point", "coordinates": [-92.024298499943669, 45.366790608627291]}, ;7, prengaman
, Great Northern Sand mine, mine, plant, operational, 420, {"type": "Point", "coordinates": [-91.591067780082166
, 45.245203969201242]}, ;8, prengaman, Midwest Frac, mine, plant, operational, 80, {"type": "Point", "coordinates"
: [-92.010319599954911, 45.326539788651623]}, ;11, prengaman, Robert Nelson, mine, operational, 11, {"type":
"Point", "coordinates": [-92.002207529960671, 45.309328158665302]}, ;12, prengaman, Sioux Creek Silica
, mine, plant, proposed, 952, {"type": "Point", "coordinates": [-91.66639363007198, 45.23172948912017]},
;16, prengaman, Gregory Weber, mine, permitted, 40, {"type": "Point", "coordinates": [-91.572008020123761
, 44.463425139241387]}, ;17, prengaman, Kendall Klevgard, mine, proposed, 180, {"type": "Point", "coordinates"
: [-91.679525380114313, 44.482075879127947]}, ;18, prengaman, Ryan Barth- Platts Valley, mine, operational
, 130, {"type": "Point", "coordinates": [-91.582367660140605, 44.1214938889240668]}, ;19, prengaman, Larson
/Stanton/Johnson, mine, plant, permitted, 998, {"type": "Point", "coordinates": [-91.701808810108261, 44.538990829101053
]}, ;20, prengaman, R & J Rolling Acres, mine, plant, permitted, 205, {"type": "Point", "coordinates": [-91
.690313850114933, 44.451525049116853]}, ;21, prengaman, Seven Sands, mine, plant, denied, 1390, {"type":
"Point", "coordinates": [-91.660185650123481, 44.3570616991530641]}, ;22, prengaman, Steve Stamm, mine
```

The received response is JSON object



Data Queries Results



The parsed JSON object is visualized by using different visualization methods

TrajAnalytics: A Free Software for Visually Exploring Urban Trajectories
<http://vis.cs.kent.edu/TrajAnalytics/>



Data Queries: Demo

The screenshot displays the Trajectory Analytics V2.0 web application. The interface includes a navigation bar at the top with 'Trajectory Analytics V2.0', 'About', 'Contact', and a 'Home' button. The main area is a map of Porto, Portugal, with various districts labeled such as Matosinhos, Senhora da Hora, São Mamede de Infesta, Rio Tinto, Fânzeres, Gondomar, Valbom, Avintes, Madalena, and Jovim. The Douro River is visible flowing through the city. On the right side, there is a control panel with three tabs: 'Ranked Records (Len)', 'Top Records (Flow)', and 'Top Records (Speed)'. The 'Ranked Records (Len)' tab is active, showing a table with two columns: 'Trajectory ID' and 'Trajectory Length (Km)'. The table currently contains one row with the value '0' and a message 'No items to display'. Below the table, the text 'No Data Available.' is displayed. At the bottom of the map, there is a 'Query Control' panel with fields for 'Date from: 2013/07/01 00:04', 'Date to: 2013/07/01 06:37', and a 'Query Mode: Intersect' dropdown menu. The bottom right corner of the map area contains the text 'Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox'.

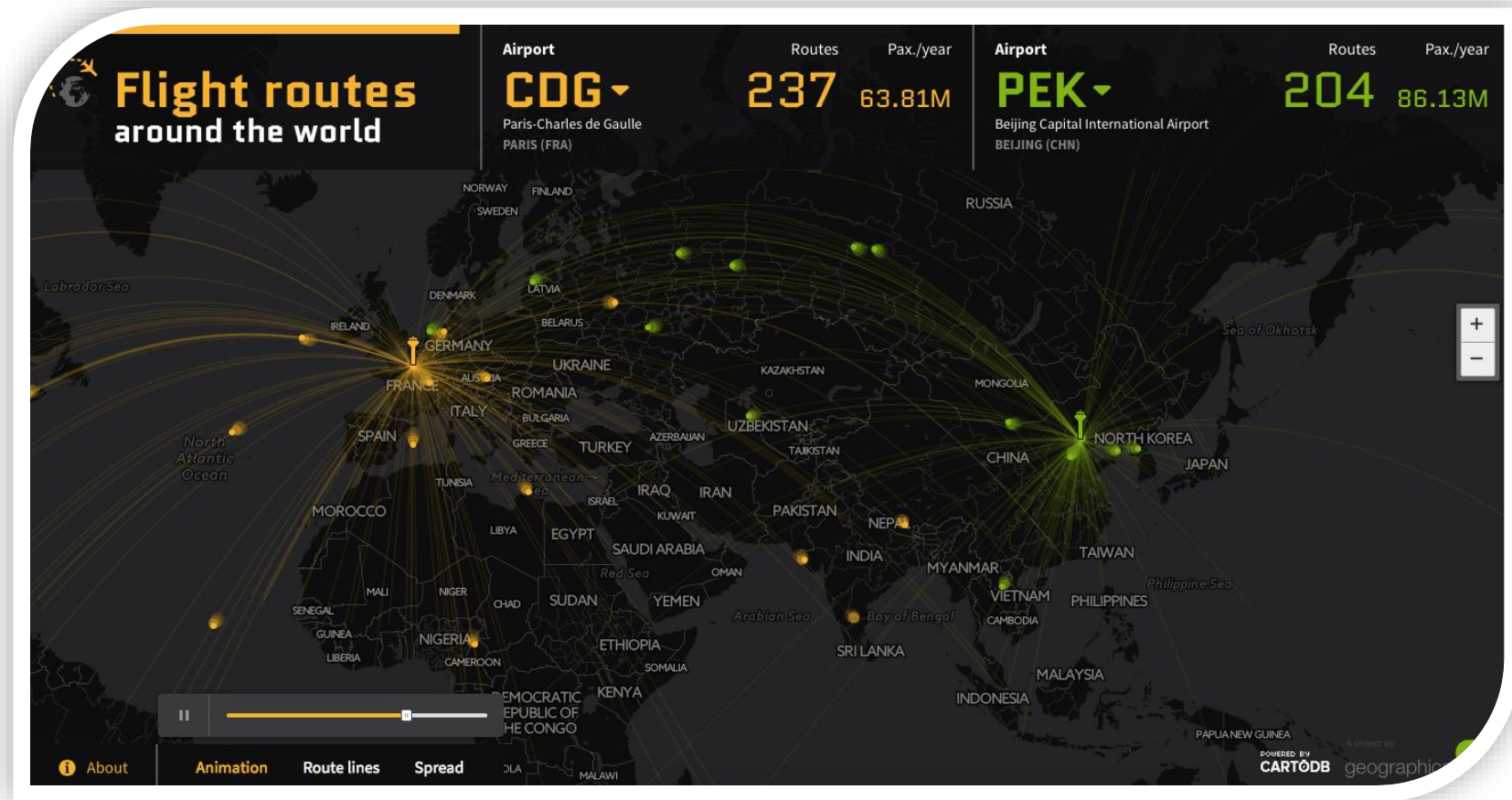


Map Visualization and Tools

- The common visualization method used for urban trajectory data is static and animated maps.
- This method provides some interaction techniques as well.
- Creating a good web map is not an easy task but over the years we've seen plenty of amazing geovisualization examples.
- All of them have been developed with the use of one of the following tools, APIs or libraries.



CARTO

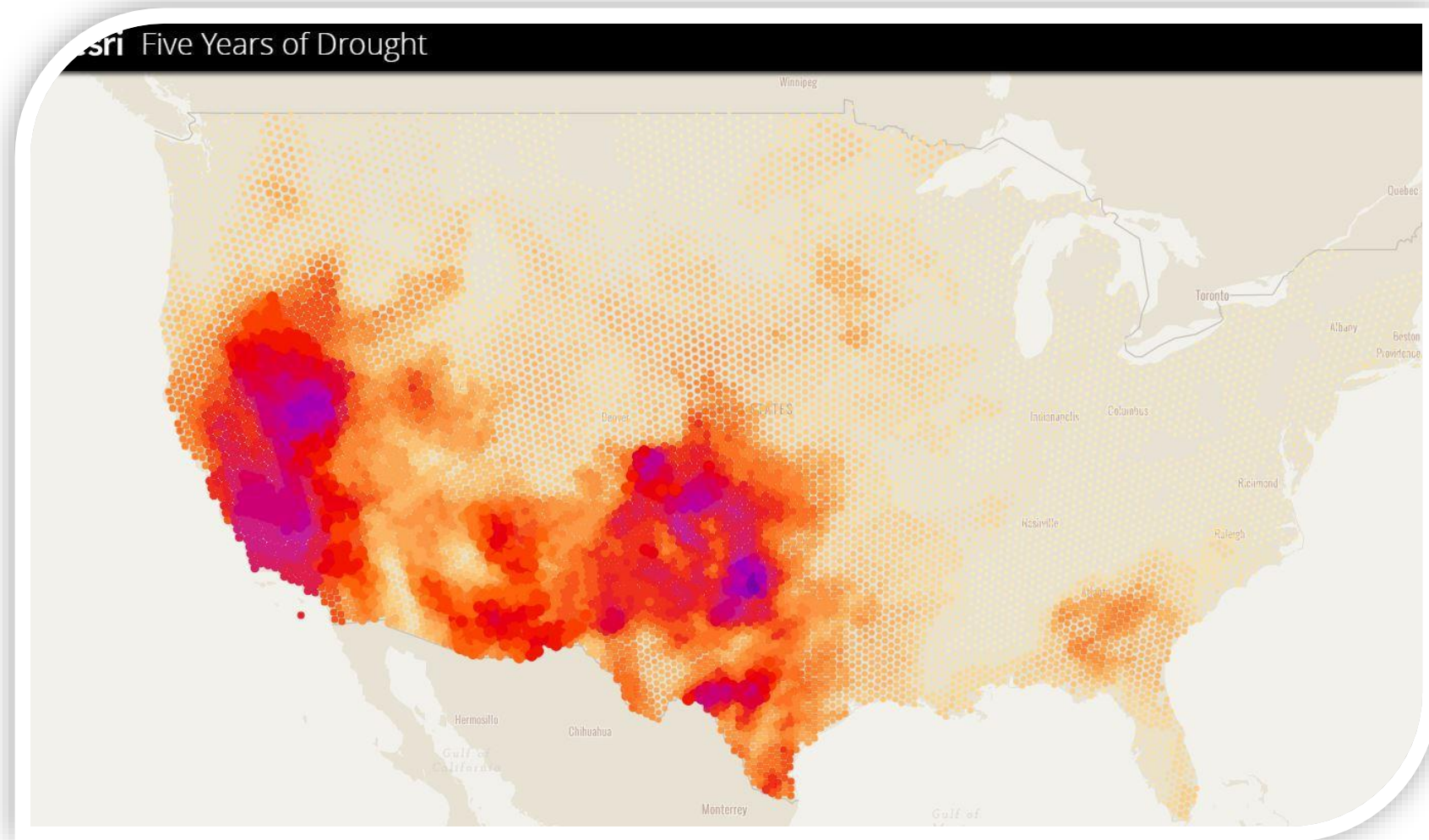


- **CARTO** previously known as CartoDB is the best platform for complex and dynamic geospatial data visualization and analysis.

example: [Flights around the world](#)



ArcGIS

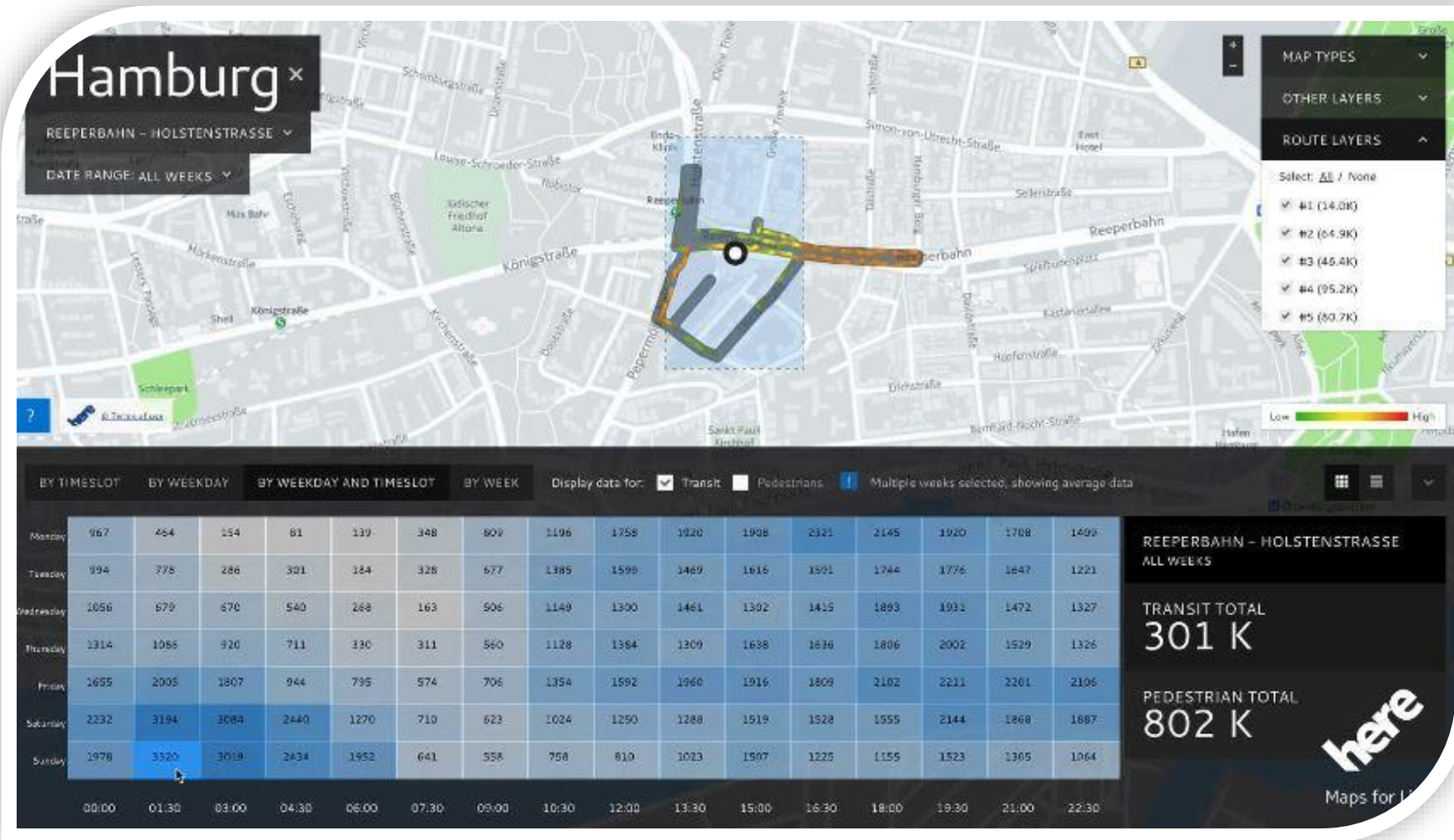


example: 5 years of drought

- [ArcGIS Online](#):
- Esri has monopolized the field of GIS and is also one of the most popular geo visualization platforms.



Data Lens

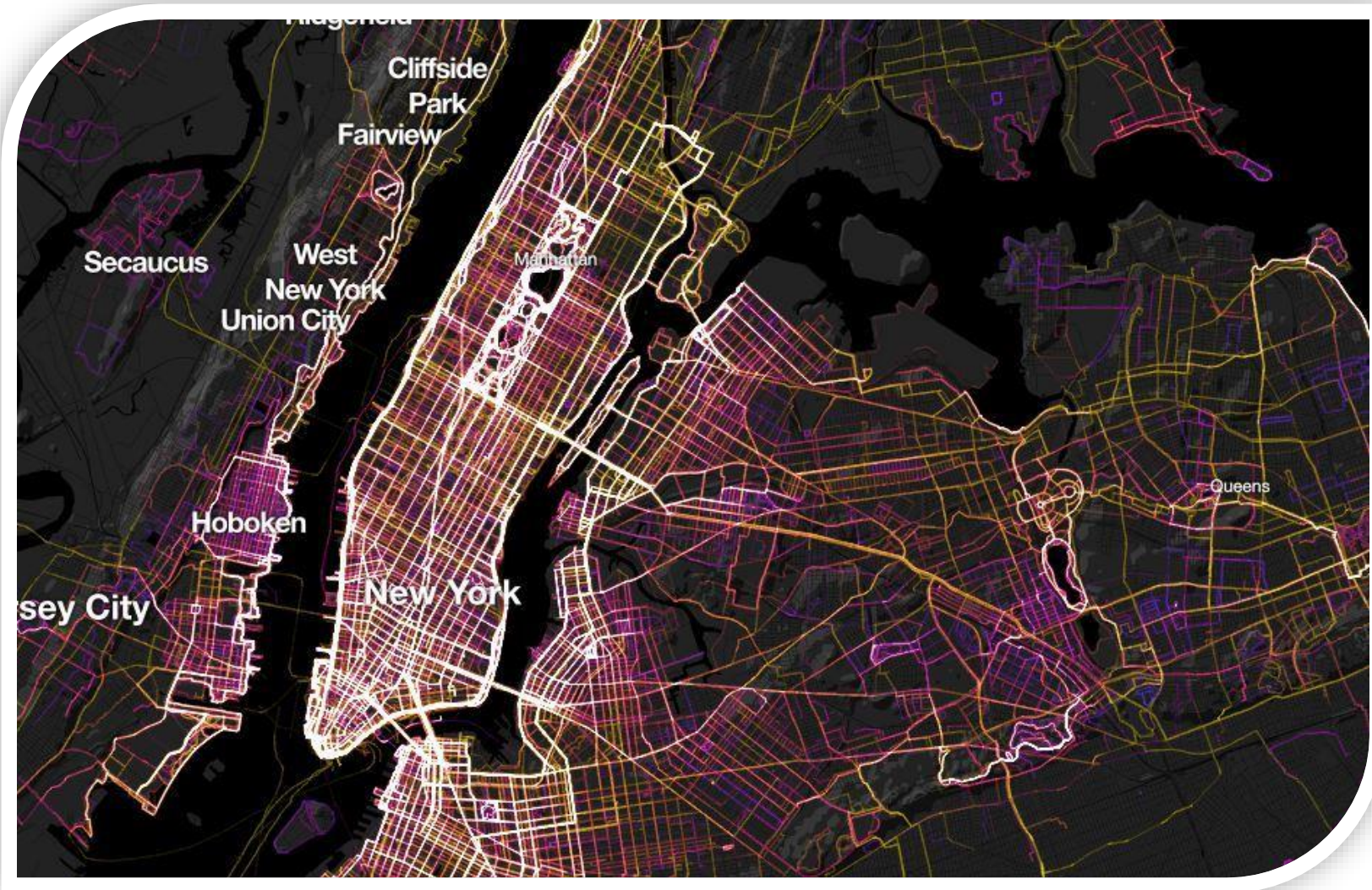


- Here Data Lens is a set of APIs that provide a seamless developer experience for creation & deployment cool data visualizations.

example: [Billboard analytics tool demo](#)



Mapbox

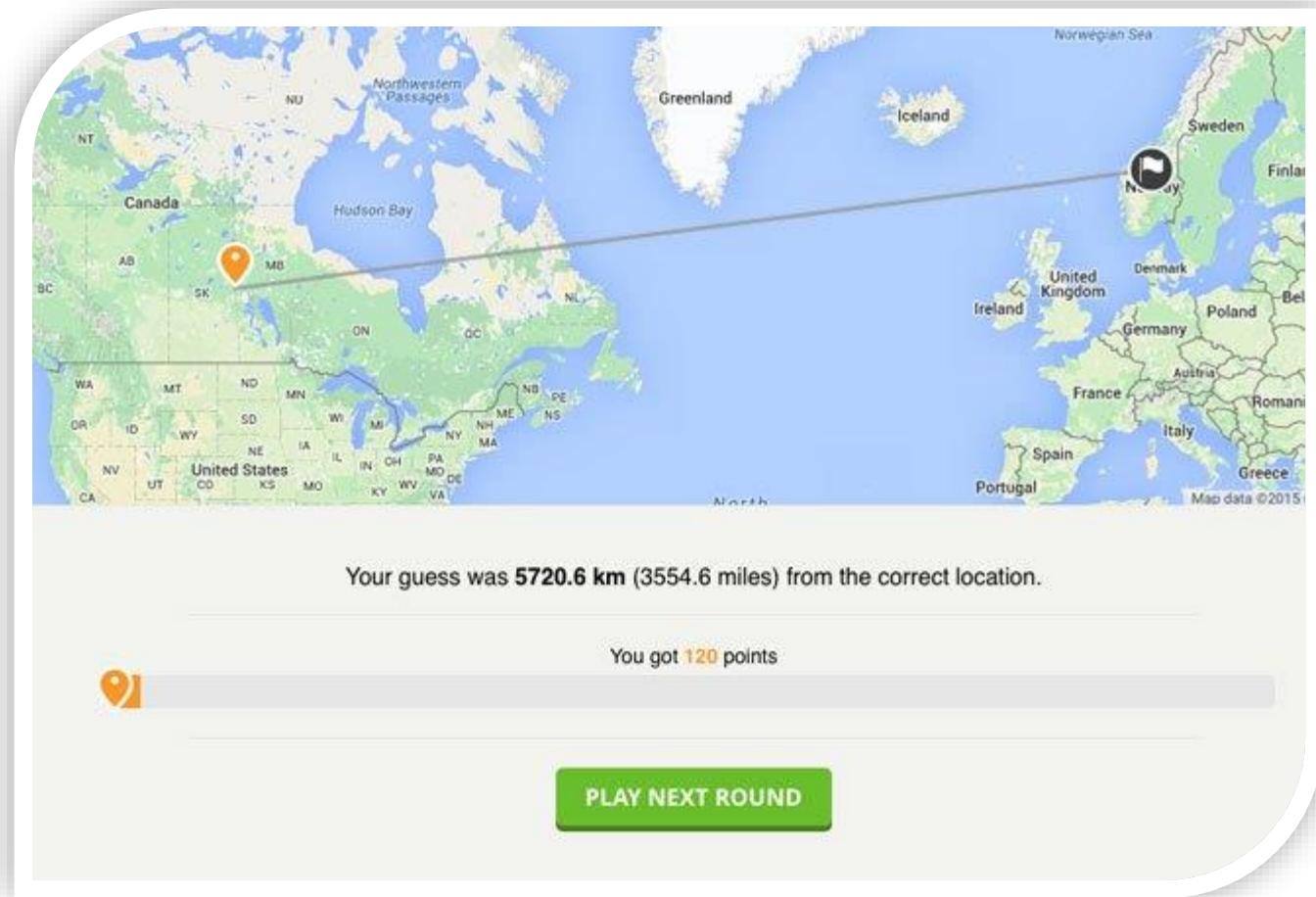


example: [1.5 Million Walks, Runs, and Bike Rides from RunKeeper mapped on Mapbox](#)

- **Mapbox** is a geo-visualization platform that gives easy to use set of tool for creating beautiful web and mobile maps.



Google Maps

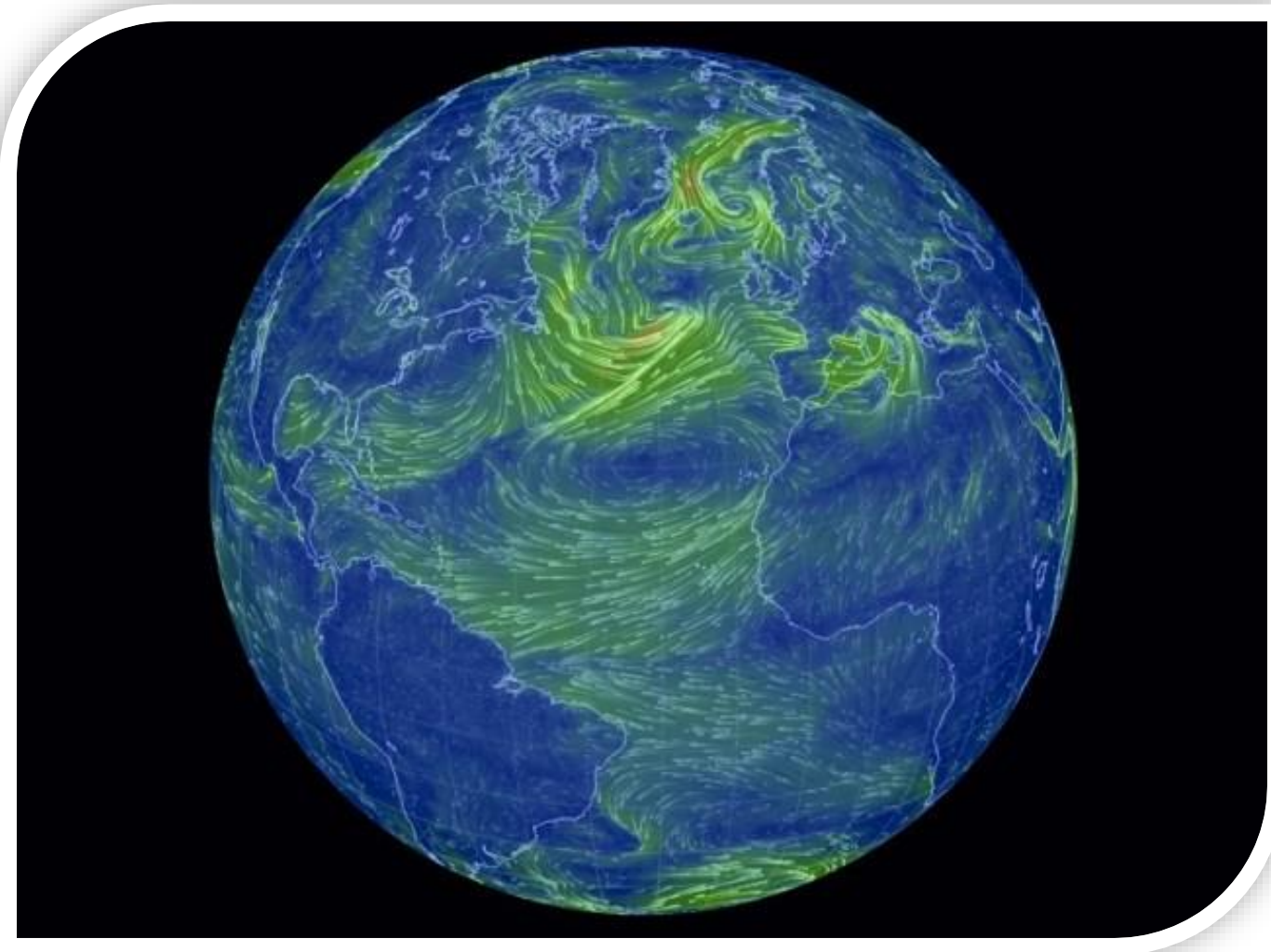


example: [GeoGuessr – StreetView based game?](#)

- Google Maps offers a set of APIs for different mapping purposes. It offers access to Google's mapping data that includes StreetView among others.



D3

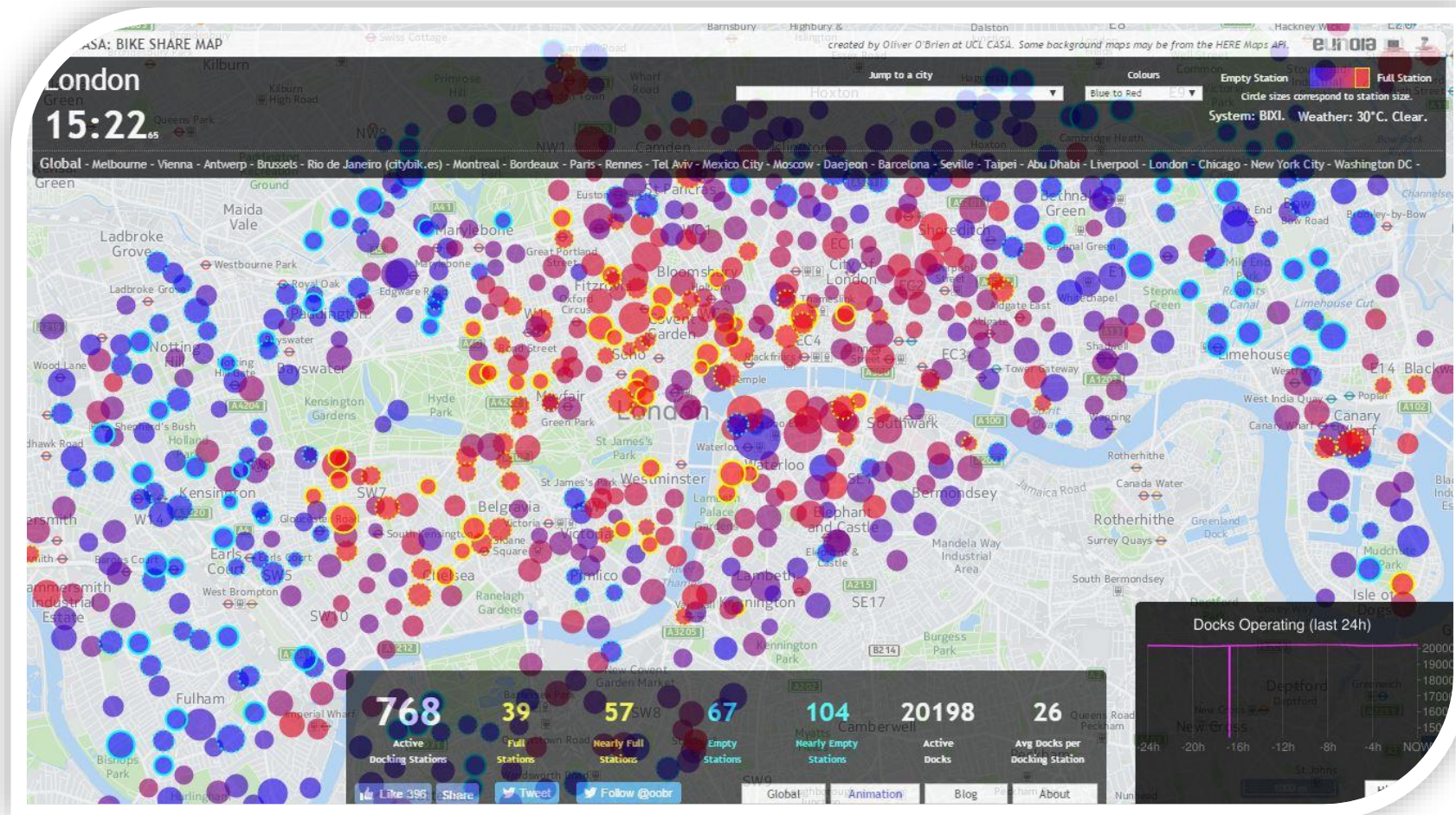


example: [Earth wind visualization](#)

- [D3.js](#) is a JavaScript library that offers powerful data visualization features for producing dynamic, interactive data visualizations in web browsers.



OpenLayers

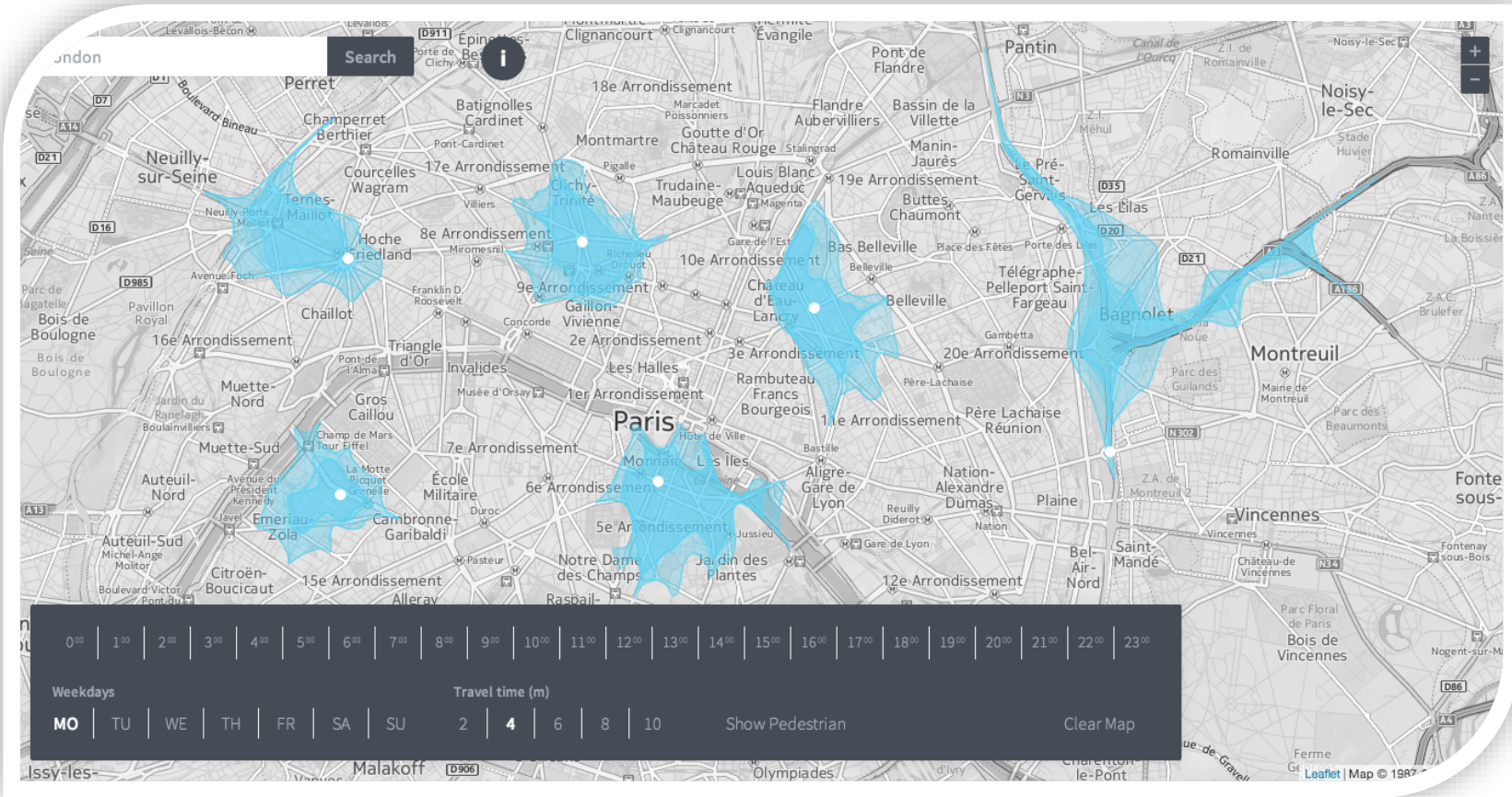


example: [Bike sharing map](#)

- OpenLayers is an open source JavaScript library for displaying map data in web browsers.
- It provides an API for building web-based geographic app. Similar to Google Maps but for free.



Leaflet



example: [Travel times map – HERE Isoline + Leaflet](#)

- [Leaflet.js API](#) an open source JavaScript library for mobile-friendly interactive maps.
- It offers really powerful mapping features that commonly used for the best and most beautiful maps.

(Recommended)



Leaflet Cont.

- Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps.
- It works efficiently across all major desktop and mobile platforms.
- It has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to.



Leaflet Cont.

- Example: Create a map in the 'map' div, add **tiles** of our choice, and then add a marker with some text in a popup:

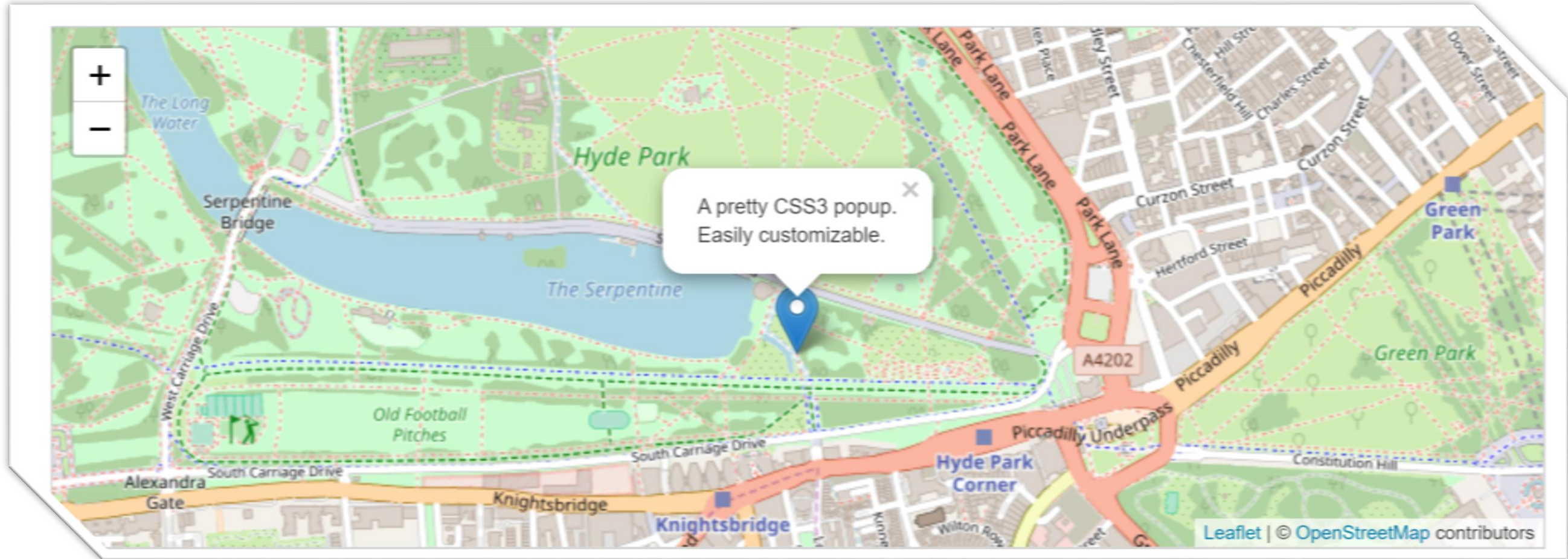
```
var map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

L.marker([51.5, -0.09]).addTo(map)
  .bindPopup('A pretty CSS3 popup.<br> Easily customizable.')
  .openPopup();
```



Leaflet Cont.

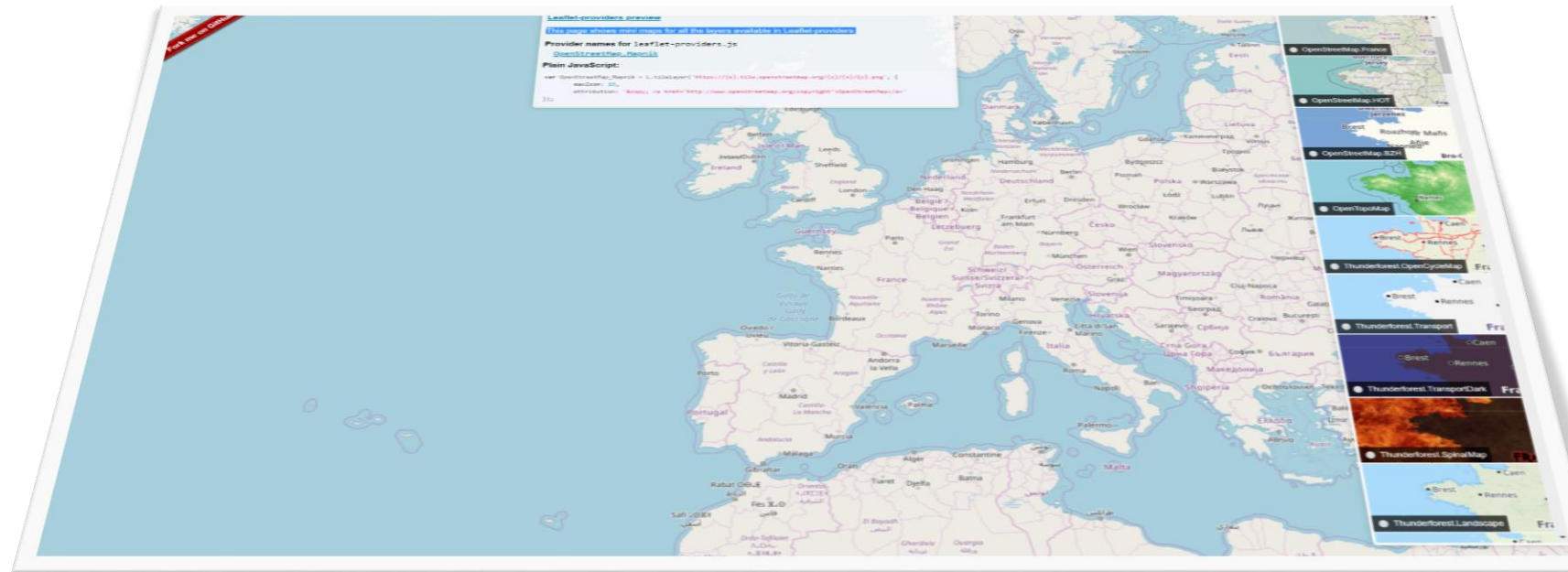


Example: Create a map in the 'map' div, add tiles of our choice, and then add a marker with some text in a popup



Leaflet Cont.

- For more map tiles, check the link below:
- <https://leaflet-extras.github.io/leaflet-providers/preview/>
- This page shows mini maps for all the layers available in Leaflet-providers.



Leaflet Cont.

- Besides tiles layers, you can easily add other things to your map, including **markers**, **polylines**, **polygons**, **circles**, and **popups**.
- **Markers** can be used to visualize **POI** on the map.
- **Polylines** can be used to visualize **trajectories** and **road network** on the map.
- **Polygons** can be used to visualize **zip code regions** on the map.
- **Circles** can be used to visualize **pick-up and drop-off locations** on the map.
- **Popups** can be used to show some **semantic information** on the map.



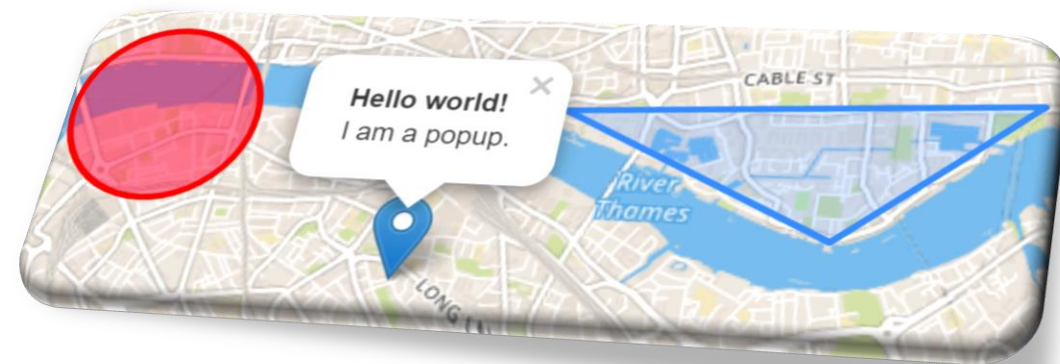
Leaflet Cont.

- To add a marker:

```
var marker = L.marker([51.5, -0.09]).addTo(mymap);
```

- Adding a circle is the same (except for specifying the radius in meters as a second argument), but lets you control how it looks by passing options as the last argument when creating the object:

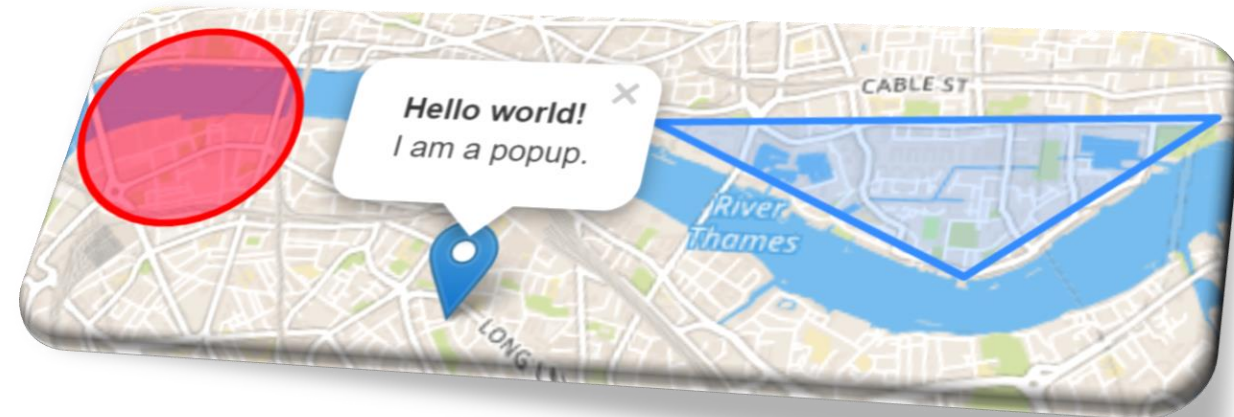
```
var circle = L.circle([51.508, -0.11], {  
  color: 'red',  
  fillColor: '#f03',  
  fillOpacity: 0.5,  
  radius: 500  
}).addTo(mymap);
```



Leaflet Cont.

- Adding a polygon is as easy:

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



- Popups are usually used when you want to attach some information to a particular object on a map. Leaflet has a very handy shortcut for this:

```
marker.bindPopup("<b>Hello world!</b><br>I am a popup.").openPopup();
circle.bindPopup("I am a circle.");
polygon.bindPopup("I am a polygon.");
```



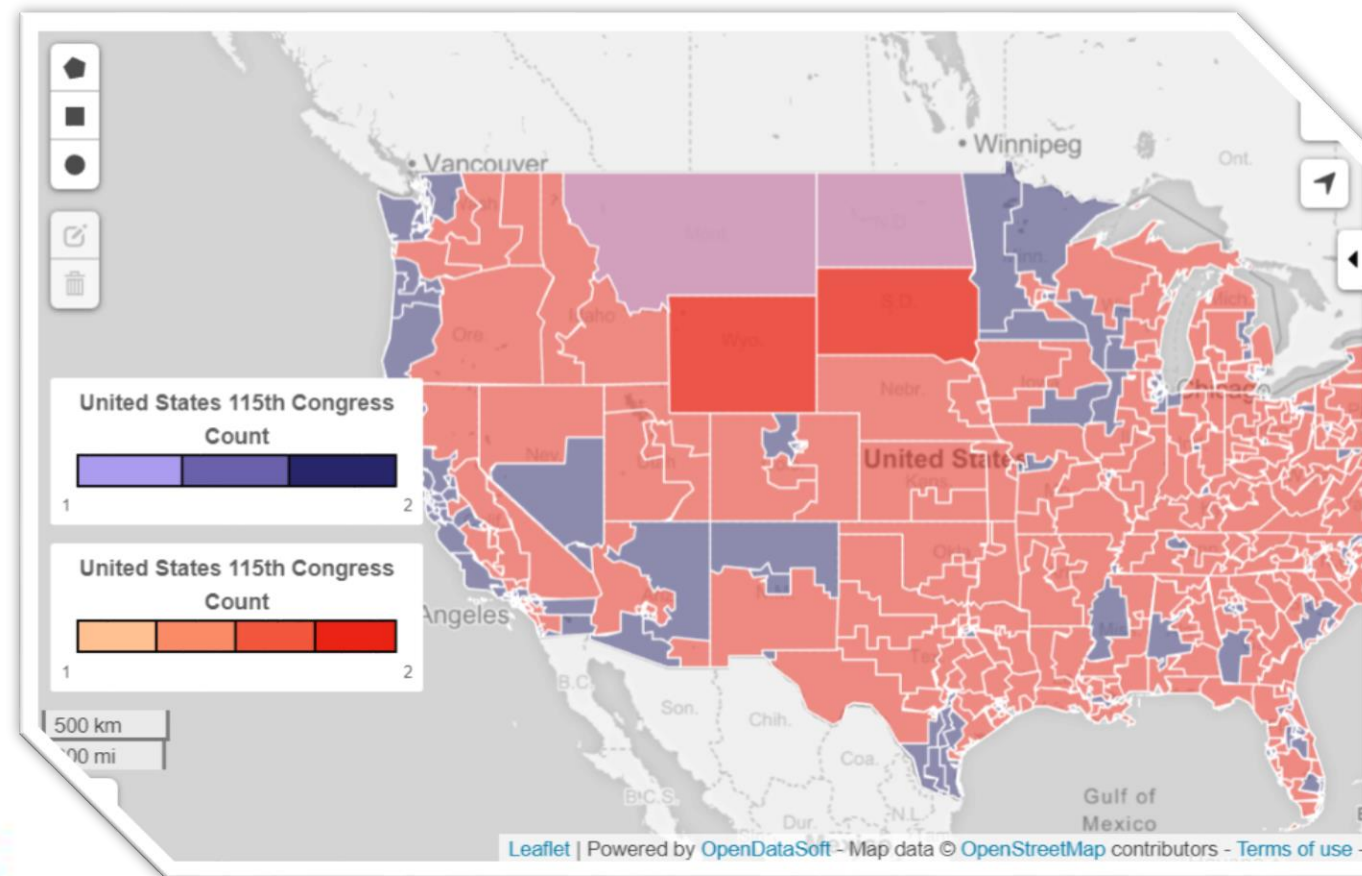
GeoJson

- **GeoJSON** is a format for encoding a variety of geographic data structures.
- **GeoJSON** is becoming a very popular data format among many **GIS** technologies and services.
- A **GeoJSON** object may represent a geometry, a feature, or a collection of features. **For instance**, the **road network** that extracted from **OSM**.
- **GeoJSON** uses the **JSON** standard.
- **Leaflet Map** is quite good at handling it.



GeoJson Cont.

- In this example, we check the "party" property and style our polygons accordingly:



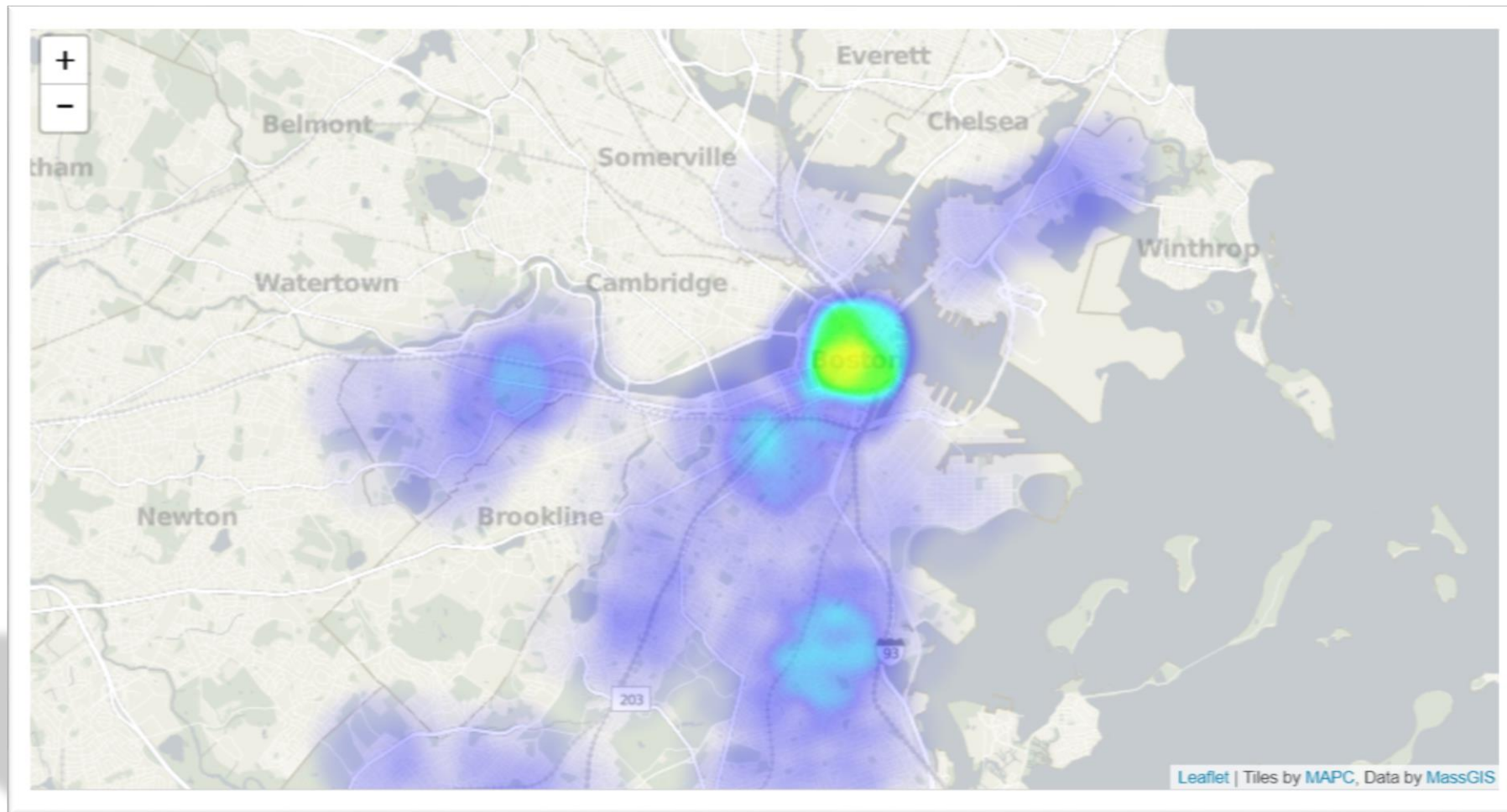
```
var states = [{
  "type": "Feature",
  "properties": {"party": "Republican"},
  "geometry": {
    "type": "Polygon",
    "coordinates": [[
      [-104.05, 48.99],
      [-97.22, 48.98],
      [-96.58, 45.94],
      [-104.03, 45.94],
      [-104.05, 48.99]
    ]]
  }
}, {
  "type": "Feature",
  "properties": {"party": "Democrat"},
  "geometry": {
    "type": "Polygon",
    "coordinates": [[
      [-109.05, 41.00],
      [-102.06, 40.99],
      [-102.03, 36.99],
      [-109.04, 36.99],
      [-109.05, 41.00]
    ]]
  }
}
];

L.geoJSON(states, {
  style: function(feature) {
    switch (feature.properties.party) {
      case 'Republican': return {color: "#ff0000"};
      case 'Democrat':   return {color: "#0000ff"};
    }
  }
}).addTo(map);
```



Heatmap

- [Leaflet Heatmap](#) is a plugin to create a **heatmap layer** for Leaflet map.



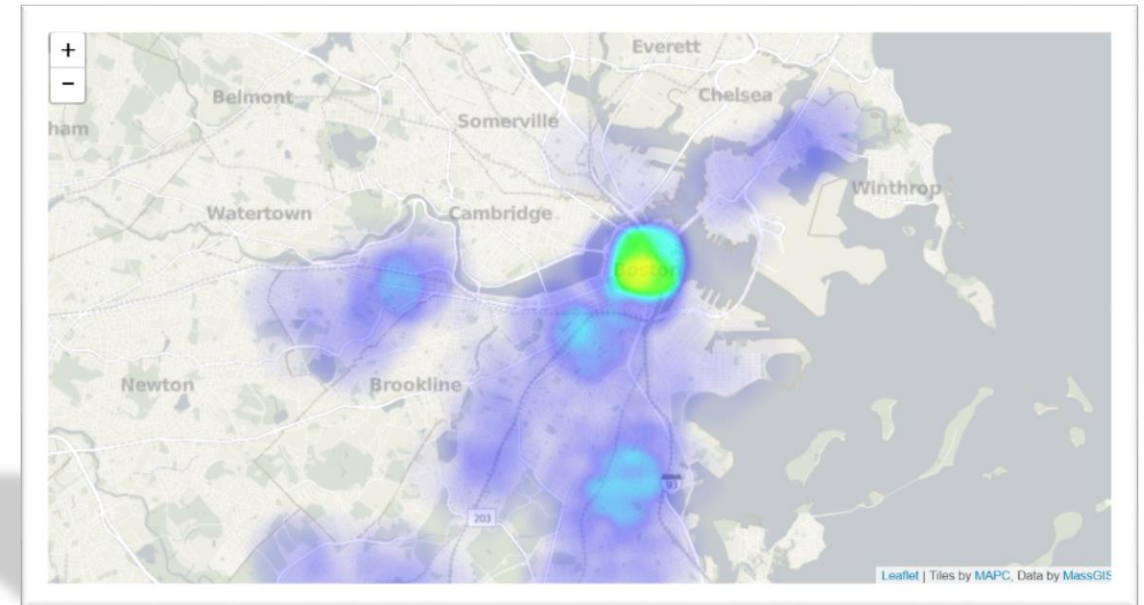
Heatmap Cont.

```
// initialize the map
var map = L.map('map').setView([42.35, -71.08], 13);

// load a tile layer
L.tileLayer('http://tiles.mapc.org/basemap/{z}/{x}/{y}.png',
  {
    attribution: 'Tiles by <a href="http://mapc.org">MAPC</a>, Data by <a href="http://mass.gov/mgis">MassG
    maxZoom: 17,
    minZoom: 9
  }).addTo(map);

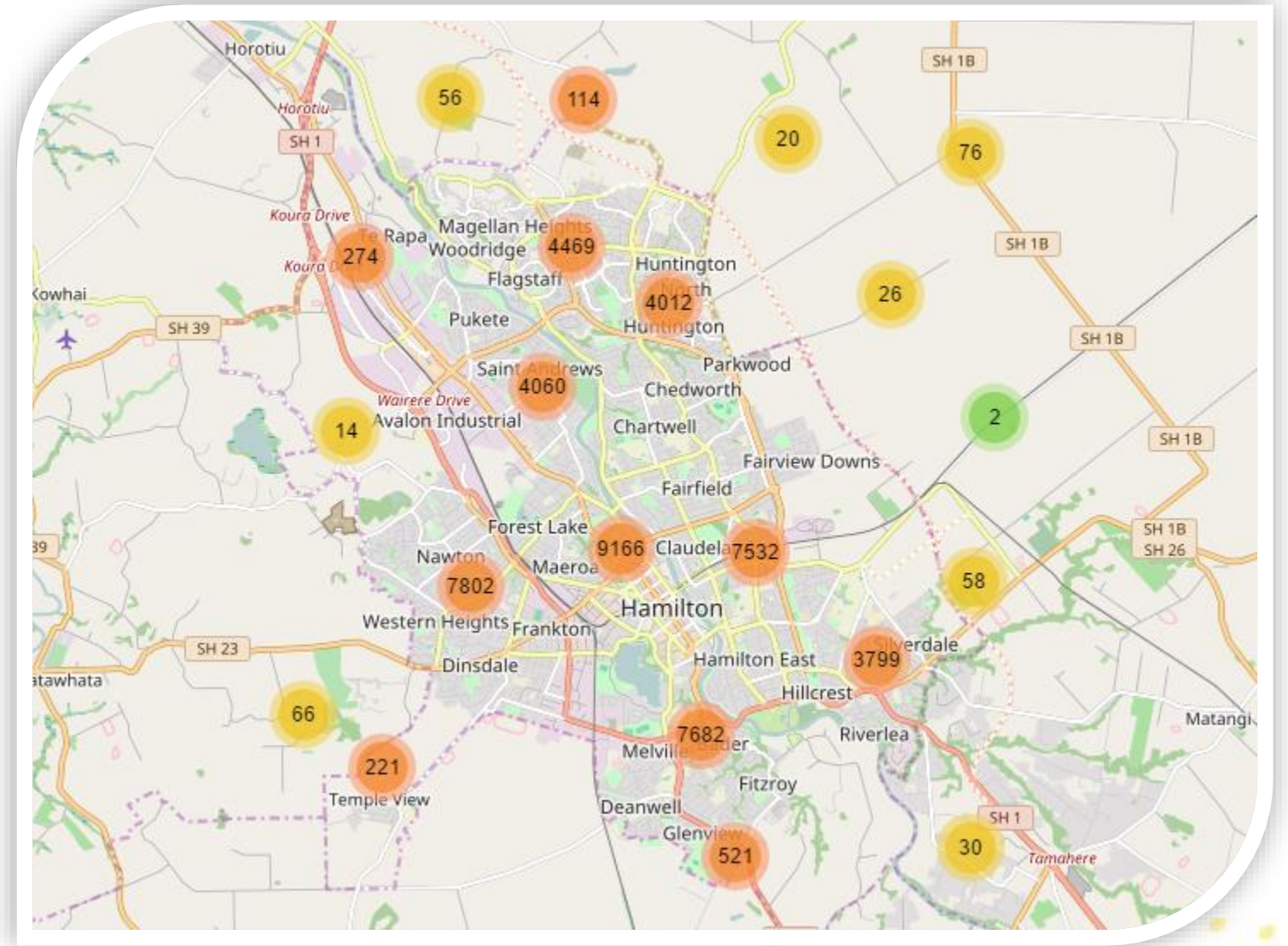
map.setZoom(12);
$.getJSON("rodents.geojson",function(data){
  var locations = data.features.map(function(rat) {
    // the heatmap plugin wants an array of each location
    var location = rat.geometry.coordinates.reverse();
    location.push(0.5);
    return location; // e.g. [50.5, 30.5, 0.2], // lat, lng, intensity
  });

  var heat = L.heatLayer(locations, { radius: 35 });
  map.addLayer(heat);
});
```



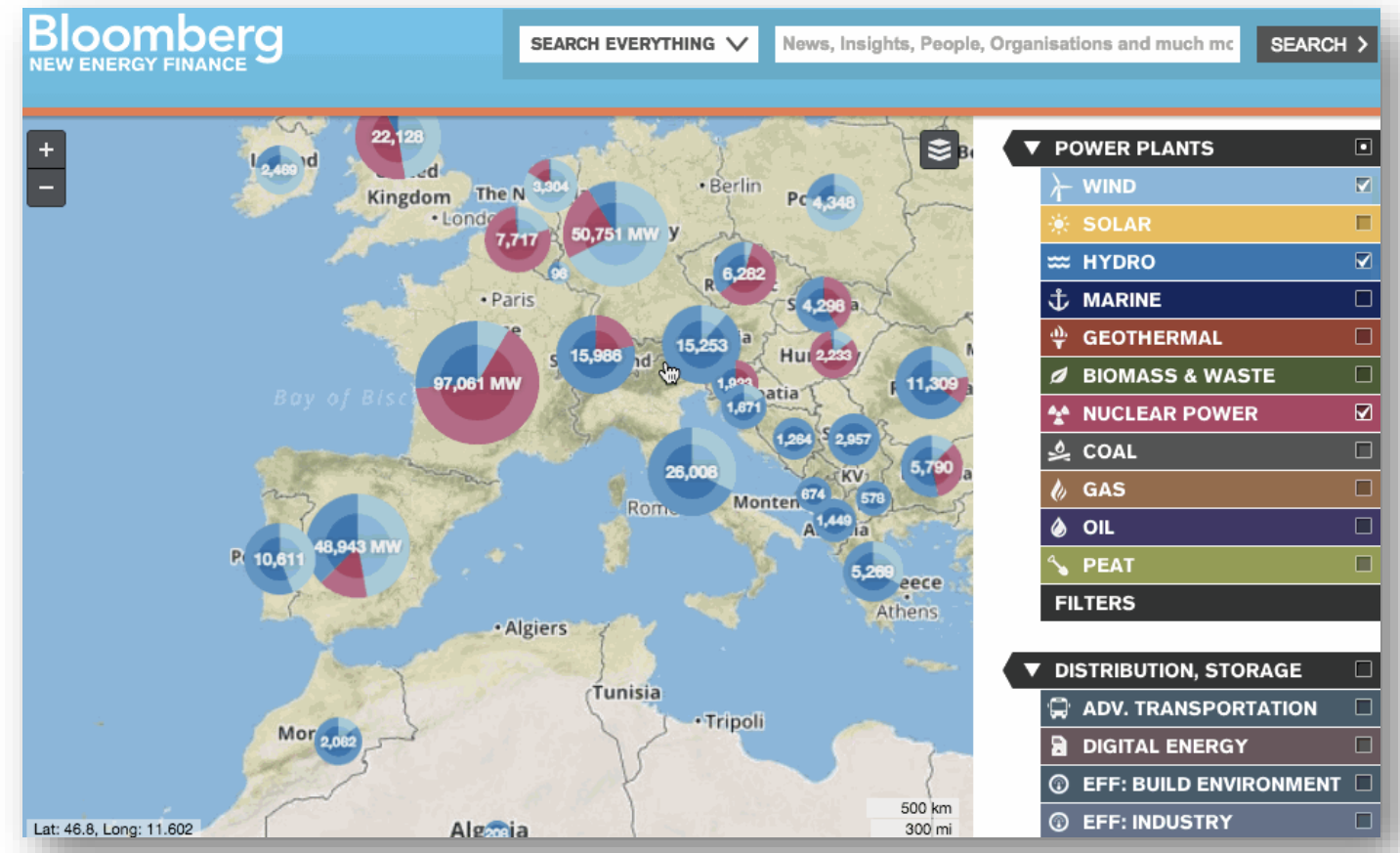
Markers

- [Marker Clustering plugin for Leaflet](#)
- Provides Beautiful Animated Marker Clustering functionality for Leaflet, a JS library for interactive maps.
- [Check this Example](#)



Mapbox

- [Bloomberg New Energy](#) Finance relaunched their Energy Asset maps to visualize everything from power plant capacity to distribution of gas pipelines and biofuel production.
- Using **Mapbox.js** with the **Leaflet Marker Cluster plugin**, the map reveals more information as you zoom in.



Drilling down on wind versus hydro versus nuclear power with Bloomberg New Energy Finance maps.



Data Attributes Visualization

- Urban trajectory data consists of time/location/other properties (e.g. Speed).
- The common visualization method used for location is static and animated maps.
- Creating charts and infographics for other attributes can be time-consuming.
- Many APIs have been implemented to make this process easier.
 - [D3.js](#), [Chart.js](#), [Google Charts](#), [Highcharts](#), [Plotly](#) and others.
- Next, we will show some of the recommended tools that can make the tedious task of making beautiful charts easier:



D3.js

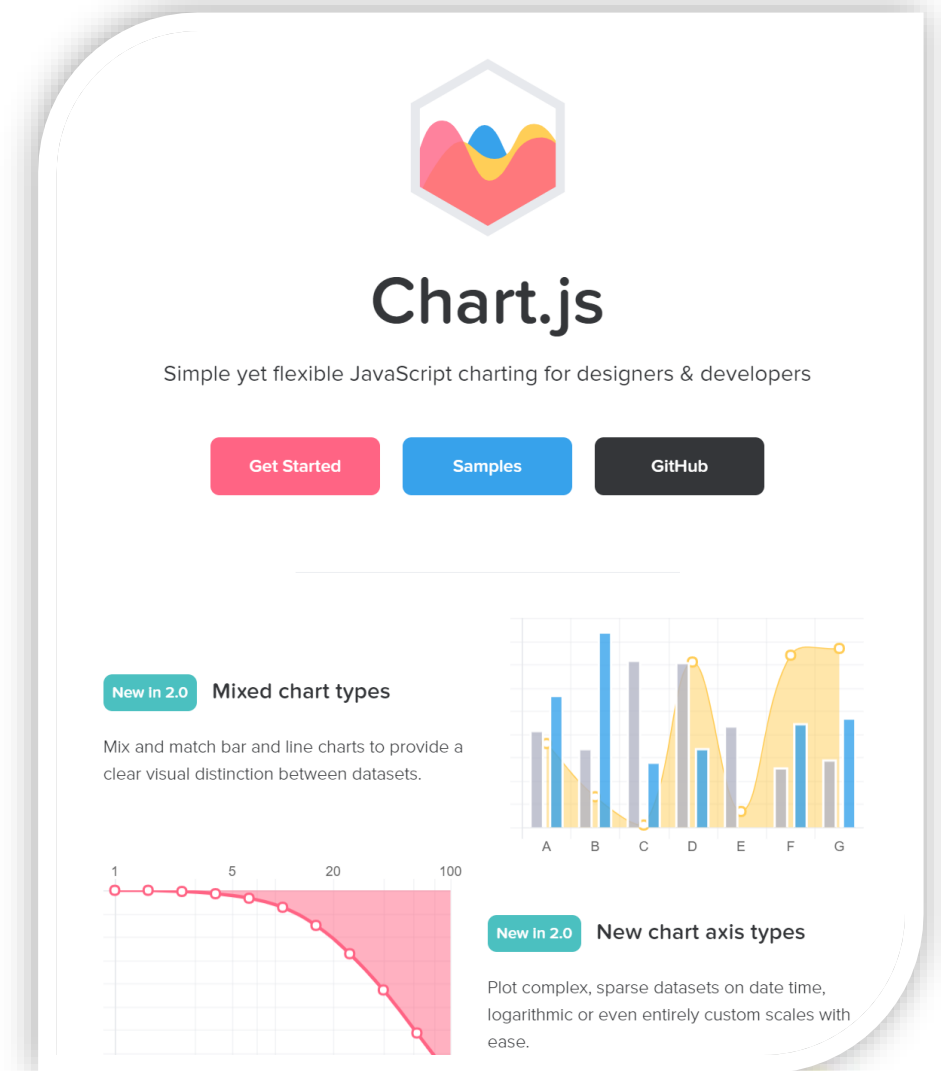
- [D3.js](#) is a JavaScript library that uses HTML, SVG, and CSS to render some amazing diagrams and charts from a variety of data sources including Urban Trajectory Data.



Chart.js

- Although armed with only six chart types, open source library [Chart.js](#) is the perfect data visualization tool for hobbies and small projects.

(Recommended)

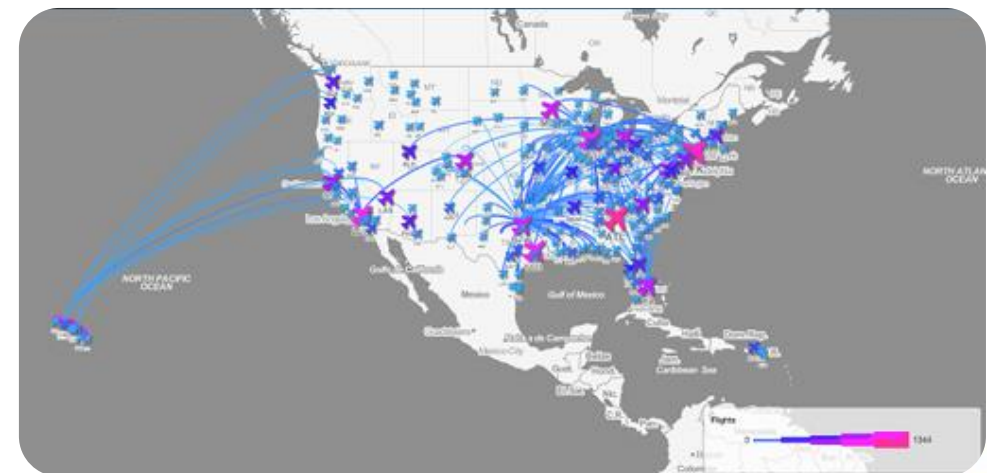
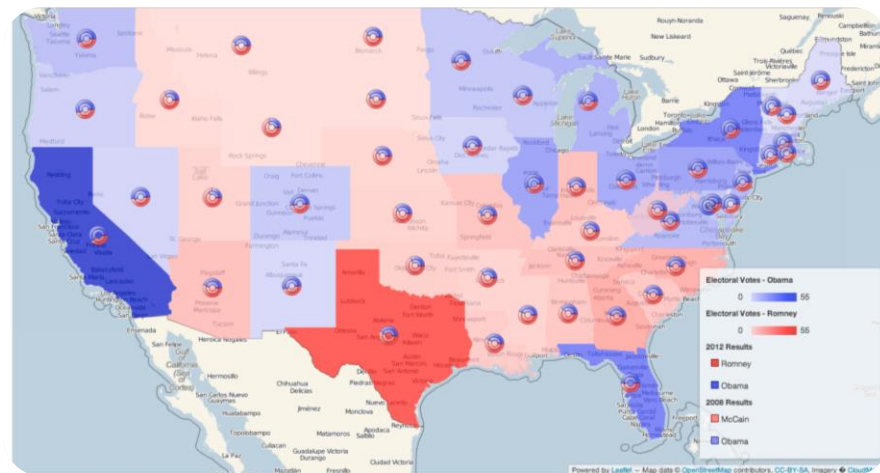


The screenshot shows the Chart.js website homepage. At the top is the Chart.js logo, a stylized 'C' with a heart shape inside, composed of red, yellow, and blue segments. Below the logo is the text 'Chart.js' in a large, bold, black font. Underneath that is the tagline 'Simple yet flexible JavaScript charting for designers & developers'. There are three buttons: 'Get Started' (pink), 'Samples' (blue), and 'GitHub' (black). Below the buttons is a section titled 'New In 2.0' with two sub-sections. The first is 'Mixed chart types', which includes a bar chart with a line chart overlaid on top. The second is 'New chart axis types', which includes a line chart with a logarithmic x-axis. The x-axis labels are 1, 5, 20, and 100. The y-axis is a simple linear scale. The line is red and has a shaded area underneath it.



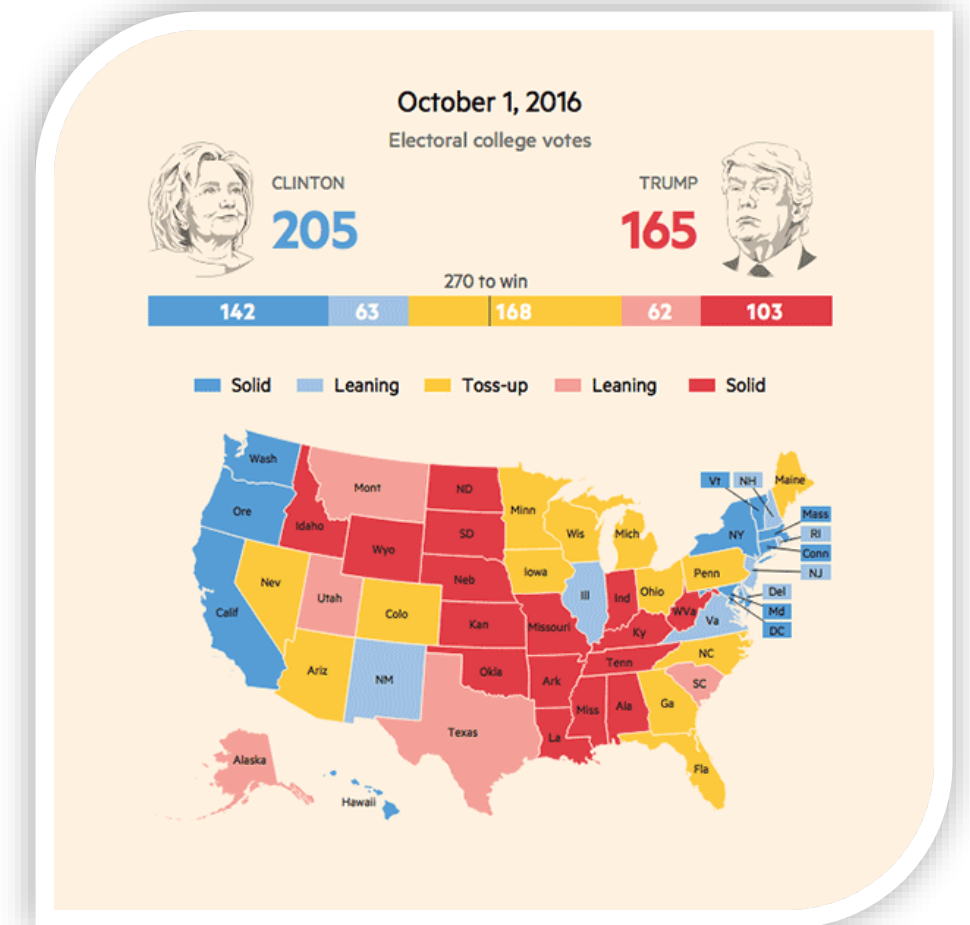
Leaflet DVF

- Leaflet Data Visualization Framework (DVF)
- The **Leaflet DVF** is an extension to the Leaflet JavaScript mapping library.
- The primary goal of the framework is to simplify data visualization on the map.
- Allow the user to draw different types of glyphs on the map.



Interactions

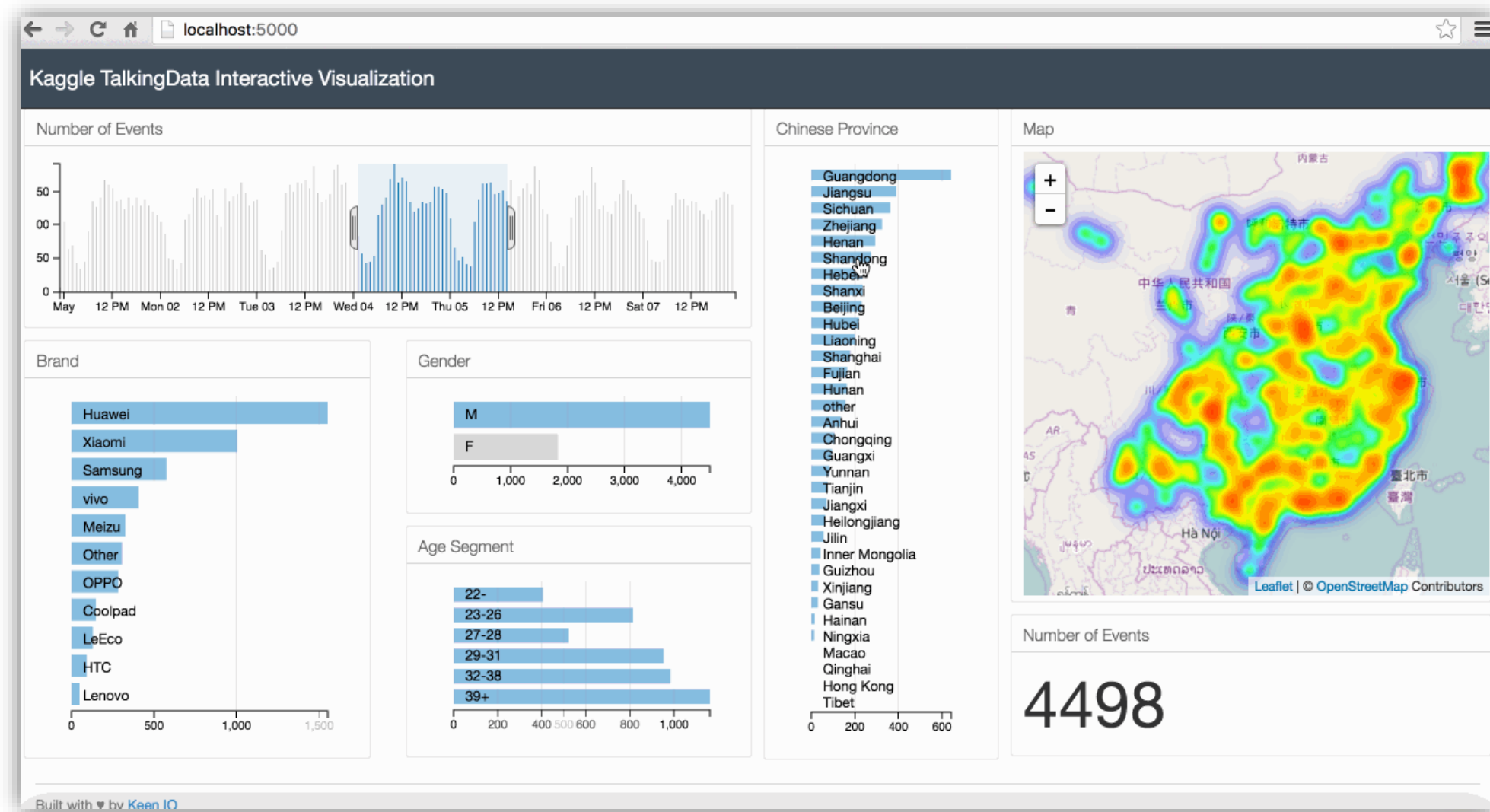
- Interactive visualizations can add an impressive animation to plain and boring datasets.
- It allows users to select specific data points to visualize the story in the way they choose.
- One of the most powerful ways to accomplish this is through JavaScript.



Created by [Joanna S. Kao](#)



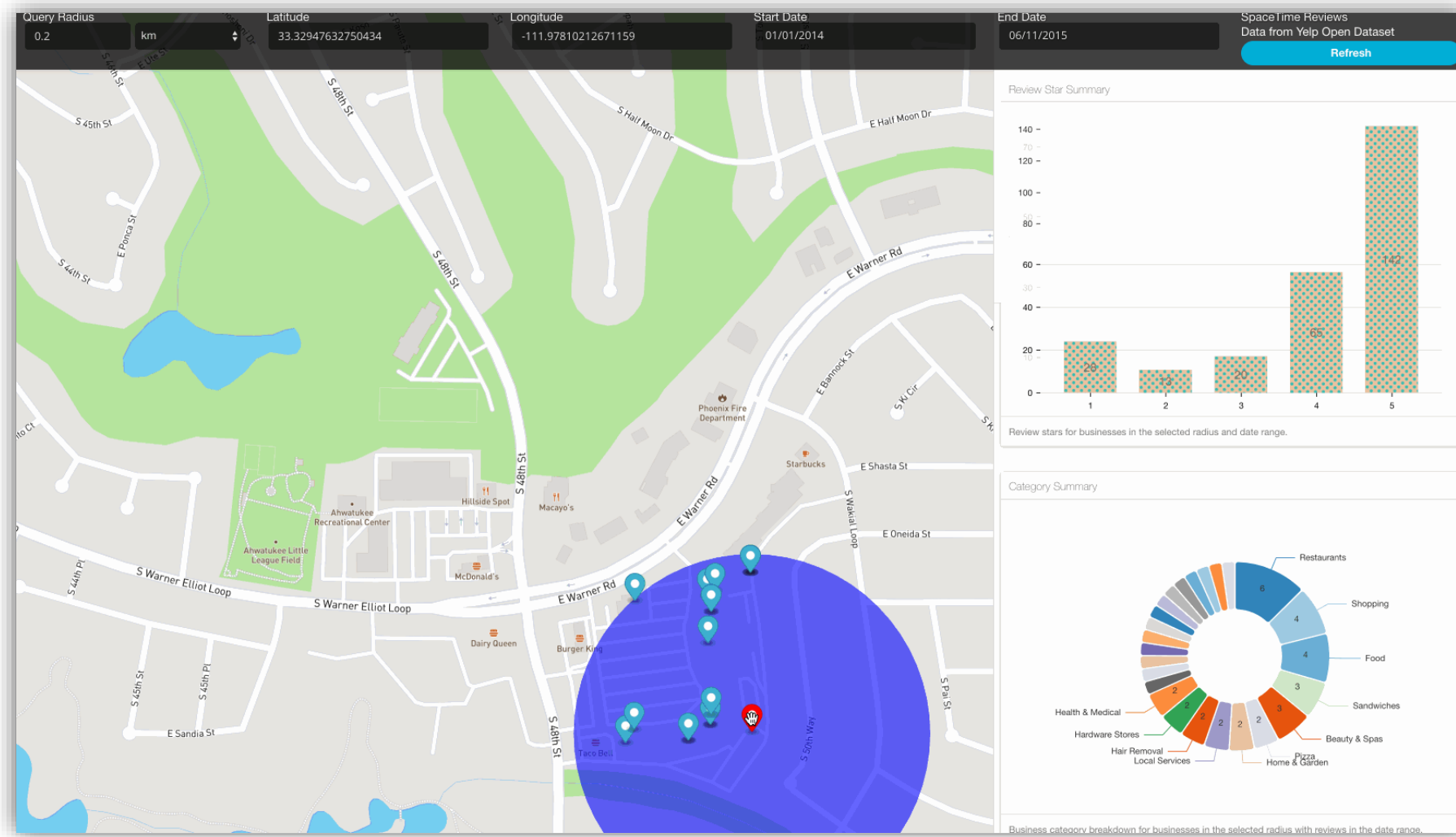
Interactions Cont.



Example: [Building an interactive visualization of geospatial data by Adil Moujahid](#)



Interactions Cont.



SpaceTime Reviews Data from Yelp Open Dataset

- A web service that allows users to search for businesses within a radius of a specific point that has received reviews within a specified date range.
- The interaction is provided for both map visualization and information visualization.
- Interaction is linked to both views.



Interaction Example

- **Leaflet.js - Event Handling**
- The Leaflet JavaScript program can respond to various events generated by the user.

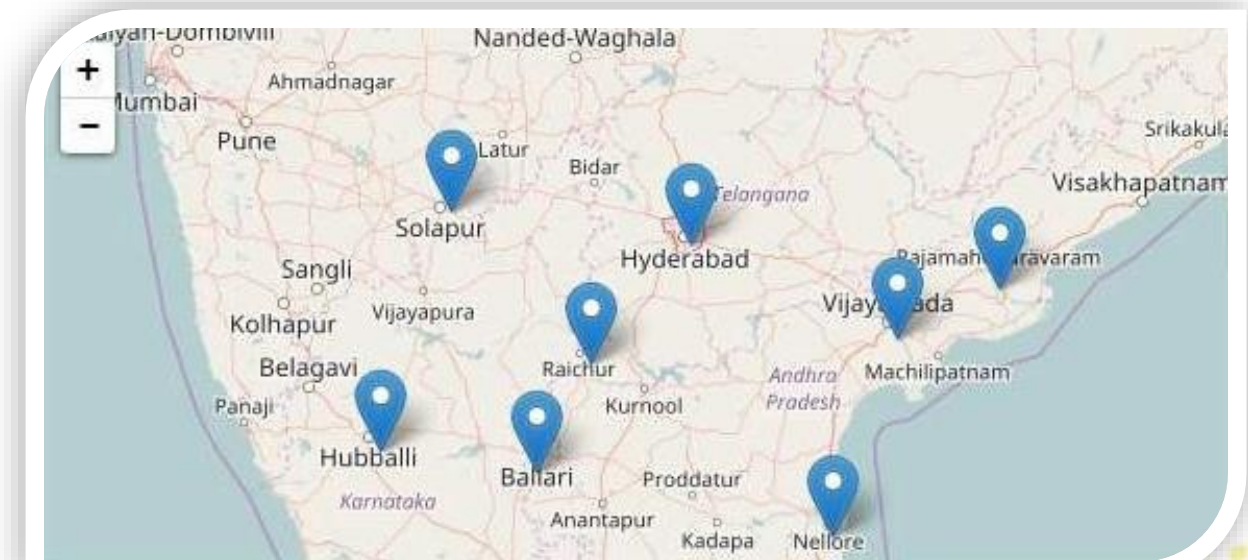
Example:

**Add Marker to the map when
user click on the map**

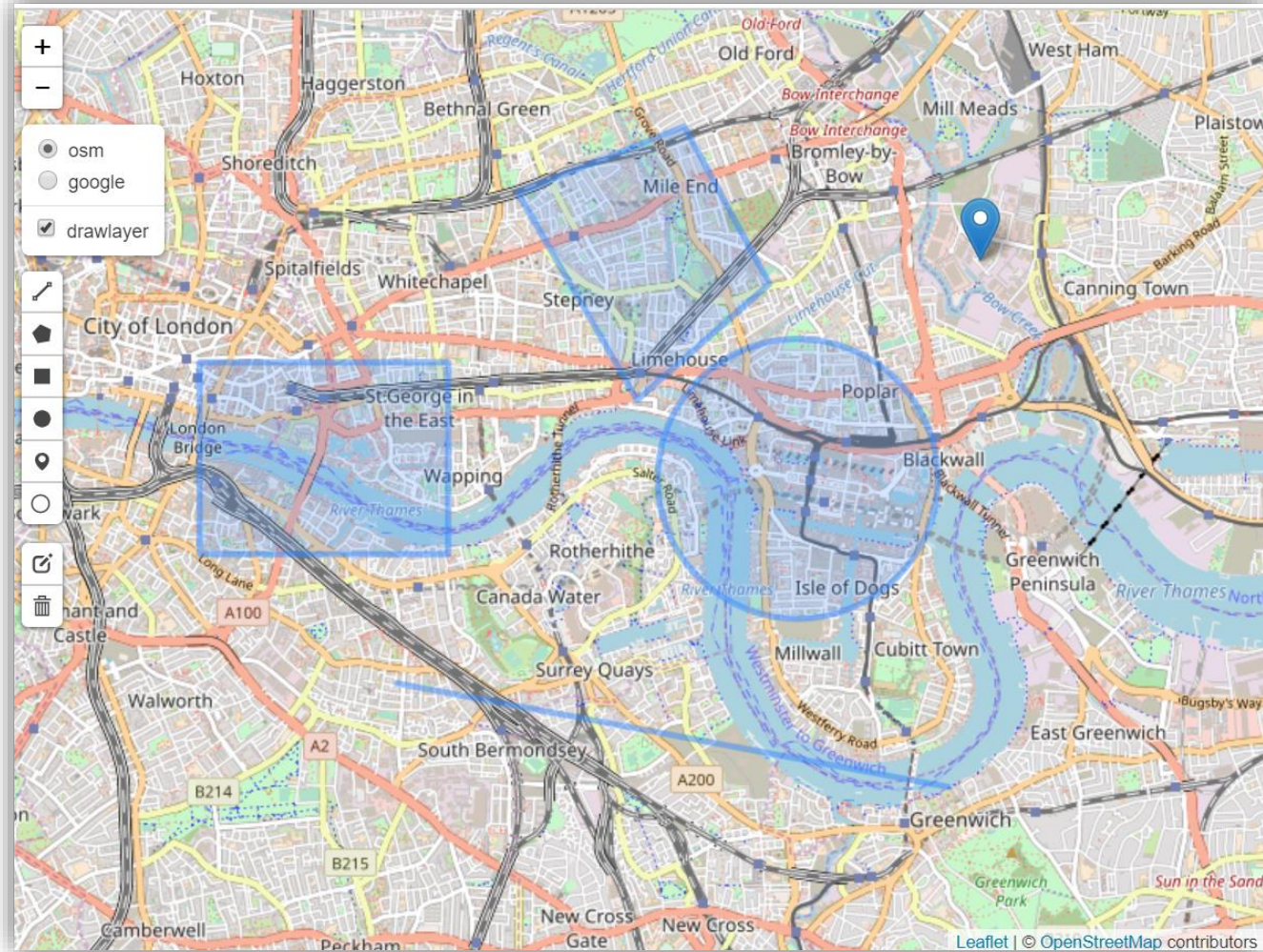
```
// Creating map options
var mapOptions = {
  center: [16.506174, 80.648015],
  zoom: 7
}
var map = new L.map('map', mapOptions); // Creating a map object

// Creating a Layer object
var layer = new L.TileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png');
map.addLayer(layer); // Adding layer to the map

map.on("click", function(e){
  new L.Marker([e.latlng.lat, e.latlng.lng]).addTo(map);
})
```



Interaction Example



- **Leaflet.draw**
- To do spatial (region) query over the map canvas, there is a need to provide users with the ability to draw different shapes (polygons, bounding box, circle, line) on the map to define the spatial constraints.



Interaction Example

- **Leaflet.draw** is designed to help users to:
 - Draw shapes on your map with easy to use drawing tools.
 - Edit and delete vectors and markers.
 - Super customizable:
 - Customize the styles of each shape to fit in with your maps theme.
 - Pick and choose the which tools you want to use.
 - Roll your own by simply using the drawing and editing handlers.
 - Event based system allows you to perform any necessary actions when shapes are created, edited or deleted.



Interaction Example Cont.

- **Leaflet.draw** is very simple to drop into your Leaflet application.
- The following example will add both the draw and edit toolbars to a map:

```
// create a map in the "map" div, set the view to a given place and zoom
var map = L.map('map').setView([175.30867, -37.77914], 13);

// add an OpenStreetMap tile layer
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);

// Initialize the FeatureGroup to store editable layers
var drawnItems = new L.FeatureGroup();
map.addLayer(drawnItems);

// Initialize the draw control and pass it the FeatureGroup of editable layers
var drawControl = new L.Control.Draw({
  edit: {
    featureGroup: drawnItems
  }
});
map.addControl(drawControl);
```



Interaction Example Cont.

- Once you have successfully added the **Leaflet.draw** plugin to your map, you will want to respond to the different actions users can trigger.

```
map.on('draw:created', function (e) {
  var type = e.layerType,
      layer = e.layer;

  if (type === 'marker') {
    // Do marker specific actions → For instance, do a point query
  }

  // Do whatever else you need to. (save to db, add to map etc)
  drawnItems.addLayer(layer);
});

map.on('draw:edited', function () {
  // Update db to save latest changes.
});

map.on('draw:deleted', function () {
  // Update db to save latest changes.
});
```



Thank You!

