KTH Royal Institute of Technology
Urban Planning and Environment - Geoinformatics

Lab 2: Interactive Web Map applications using open-source technologies-Leaflet

Due: See Assignment schedule on Canvas

Created by: Győző Gidófalvi

Teacher: Győző Gidófalvi

Updated by: Ehsan Saqib, Silvino Pedro Cumbane and Can Yang

Goal

The goal of this exercise is to learn how to create interactive, dynamic web maps using an open-source client-side web mapping library (Leaflet).

Download `lab2.zip` from Canvas, which contains the following files

- app.js: the template of the node server.
- lab2part1.html: an example HTML to get started.
- public (folder): to place your files

Preparation

- Create a folder and name it `lab2` and under that folder run

```
npm init
npm install express --save
```

Copy the `app.js` and `lab2part1.html` to `lab2` and run

```
node app.js
```

Visit http://localhost:3000/. Ensure that it displayers `This is web and mobile gis course, lab2!`.

References

- Leaflet https://leafletjs.com/

1 Creating web maps with Leaflet

1.1 Create a simple map with Leaflet

In order to use Leaflet in your web app, you will need to first add the JavaScript library to your project. There are 3 ways to do it:

(1) Use a hosted version (Recommended). You can set the following tags in `lab2part1.html`.

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/
    dist/leaflet.css"/>
<script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js">
    </script>
```

(2) Use a downloaded local version. Download Leaftlet from https://leafletjs.com/download.html. Extract the file under the **public folder** in your project. (The leaflet script file must be accessible from the HTML file so that they are placed under the public folder.) Update the tags in `lab2part1.html` .

```
<link rel="stylesheet" href="/path/to/leaflet.css" />
<script src="/path/to/leaflet.js"></script>
```

(3) Use npm to install the dependency. In a terminal under your project folder, execute

```
npm install leaflet
```

You will find a copy of the Leaflet release files in `node_modules/leaflet/dist`

Add the line to your app.js

```
app.use('/scripts', express.static(__dirname + '/node_modules/
    leaflet/dist/'));
```

Update the tag as

```
<link rel="stylesheet" href="/scripts/leaflet.css" />
<script src="/scripts/leaflet.js"></script>
```

Once the Leaflet Javascript library is added, there is a need to add Leaflet map and set the basemap layer. To acheive this add the following lines of code in script tag in `lab2part1.html`

```
var map = L.map ('mapid'). setView ([59.324608,18.06736], 12);
osm=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);
```

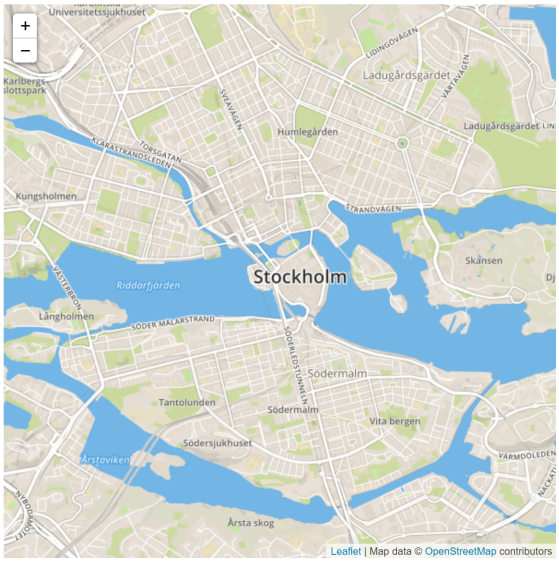

Figure 1.1: A simple map created with Leaflet

Run your program with the

```
node app.js
```

Visit http://localhost:3000/lab2part1. Make sure that you see the map in Figure 1.1.

Using Leaflet, the user to add markers and shapes on the map. The user can also link popup to the shapes using **bindPopup** function. Add a marker, circle and a polygon using the code:

```
L.marker([59.324608, 18.06736]).addTo(map).bindPopup("<b>Welcome to
    Stockholm!</b><br />I am a popup.").openPopup();

L.circle([59.346155, 18.049538], 500, {
color: 'red',
fillColor: '#f03',
fillOpacity: 0.5
}).addTo(map).bindPopup("I am a circle.");

L.polygon([
[59.312686, 18.016798],
[59.330388, 18.023071],
[59.317601, 18.041611]
]).addTo(map).bindPopup("I am a polygon.");
```

Visit http://localhost:3000/lab2part1 again. Make sure that you see the map in Figure 1.2.

Task

- Briefly explain the JavaScript code in the `lab2part1.html`
- Replace the basemap with Google Map. You may take help from here
- Add a marker at KTH with a custom icon (e.g., school icon). You may reuse the code in this example.
  Hint: you can update tile layer configurations with Google Map JavaScript URL.

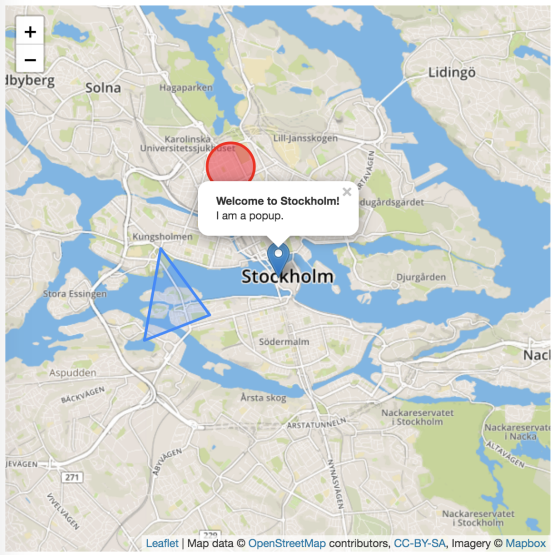

Figure 1.2: A simple map created with Leaflet

1.2 Adding Vector Layers

Leaflet provides APIs for loading vector data in form of GeoJSON. GeoJSON is a format for encoding a variety of geographic data structures. It supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon. Geometric objects with additional properties are Feature objects. Sets of features are contained by FeatureCollection objects. Take a look at the simple example below and for more information about GeoJSON see this link: https://tools.ietf.org/html/rfc7946.

In this lab we are going to use the GeoJSON file : **data.geojson** already placed in public folder, whose format is shown in List 1

Listing 1: Example of a GeoJSON format

```
{
  "type": "Feature",
  "properties": {
    "Name": "Blue line 10",
    "color": "Blue",
    "number": 10
  },
  "geometry": {
    "type": "LineString",
    "coordinates": [
    [17.887845, 59.396892],[17.901535, 59.394292],....
    ]
  }
}
```

To load the GeoJSON from a URL, we will use the `$.ajax` function provided by JQuery. Add the script tag below to the `head` tag of your HTML file.

```
<script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="
    sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="
    anonymous"></script>
```

Add the following JavaScript code to your html file and check the map in your browser.

Listing 2: Add GeoJSON data to the map

```
function onEachFeature(feature, layer) {
  var popupContent = "<p>I started out as a GeoJSON " +
  feature.geometry.type + ", but now I'm a Leaflet vector!</p>";
  layer.bindPopup(popupContent);
}

var metroline;

$.ajax({
type:'GET',
url:'/data.geojson',
dataType:'json',
async:false,
success:function(data){
metroline=L.geoJSON( data,
{
 style: function(feature)
 {
  return {color: "#438ddf"};
 }
 , onEachFeature:onEachFeature
}).addTo(map);
}
});
```

Task

- Briefly explain the codes in List 2
- Update the code to achieve the functionalities
  1. Choose a different line style for the GeoJSON layer.
  2. Show the color of each line according to the line color stored in the GeoJSON property (Common color name e.g., red, blue, can be directly assigned to the color attribute in the style).
  3. When the line is clicked, pop up a window to show its line number.

Hint: you can refer to the following documentation

- Style of path https://leafletjs.com/reference-1.6.0.html#path-option
- Leaflet GeoJSON documentation https://leafletjs.com/reference-1.6.0.html#geojson
- You can call `feature.properties.fieldname` to access the field value of a feature

1.3 Working with Events

Most of the components (objects), like map, layers or controls, trigger events to notify of changes. For example, we can be notified each time the mouse is moved over the map, a feature is clicked, when a feature is added to a vector layer, etc.

Events can be easily registered on an object with the `on()` method. Now we are going to register a *event listener* that gets notified when you click on the map. A listener gets a *handle function* that is called every time the event occurs.

Add the following snippet to your HTML file, then save and refresh in the browser.

Listing 3: Add a click event listener for the map

```
var popup = L.popup();

function onMapClick(e) {
  popup
    .setLatLng(e.latlng)
    .setContent("You clicked the map at " + e.latlng.toString())
    .openOn(map);
}
map.on('click', onMapClick);
```

Task

Read through the Leaflet tutorial at https://leafletjs.com/examples/choropleth/ carefully and try to integrate the info window and style hightlighting functions into your web map. When a line is hovered by the cursor, highlight it (with a different color or width) and display information the highlighted feature in a window as shown in Figure 1.3.

Hint:

- You will need to implement two listeners: `highlightFeature()` for the `mouseover` event and `resetHighlight()` for the `mouseout` event to achieve a natural interaction.
- You will modify some parts below to implement the functionalities.

```
var info = L.control({position:'bottomright'});

info.onAdd = function(map) {
  this._div = L.DomUtil.create('div', 'info');
  this.update();
  return this._div;
};

info.update = function(props) {
  // To be filled
};

info.addTo(map);

function highlightFeature(e) {
   // To be filled
}

function resetHighlight(e) {
 // To be filled
}
```

You will also need to register the two listeners for each feature in the GeoJSON.

```
function onEachFeature(feature, layer) {
    // Previous content
```

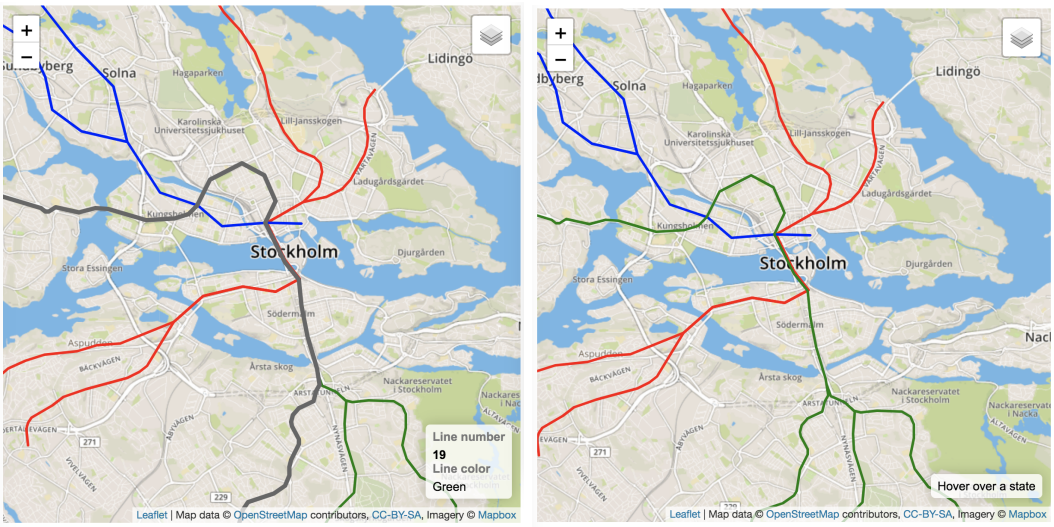

Figure 1.3: Feature highlighted (left) and unhighlighted (right)

```
        ...
        // New event listeners added.
        layer.on({
          mouseover: highlightFeature,
          mouseout: resetHighlight,
        });
    }
```

- The info window is dynamically added to the HTML, the link to the CSS style is already added in the HTML file.

1.4 Adding layer controls

As we have seen, we can add various type of layers to the map, it would be convenient to design a layer controller so that the user can turn on/off the visbility or switch between specific groups of layers. Leaflet provides an API called layer group to achieve this goal.

Study the tutorial at https://leafletjs.com/examples/layers-control/ carefully and finish the following tasks.

Tasks

- Add two base map layers Mapbox and Google Map so that you can choose one of them as the basemap.
- Add a layer group called data to include Marker layer, GeoJSON layer, etc.
- Make necessary screenshots for the maps in your report

2 Deliverables

Submit a written report in PDF format on Canvas for the following deliverables. Write the name of your group members on the report. Your deliverables should include the following.

1. Show your interactive web map to the teaching assistant.
2. Include in your submission the html file.
3. Include typical screenshots of your map in your report and explain the relevant parts (code segments) that make up your map.

3 Leaflet plugins (Optional)

Quite a lot of third party plugins have been developed for Leaflet, which are listed at https://leafletjs.com/plugins.html.
Several popular plugins include

- Leaflet draw http://leaflet.github.io/Leaflet.draw/docs/leaflet-draw-latest.html
- Leaflet marker cluster https://github.com/Leaflet/Leaflet.markercluster
- Leaflet heatmap https://www.patrick-wied.at/static/heatmapjs/example-heatmap-leaflet.html
- OSM buildings https://osmbuildings.org/?lat=59.34763&lon=18.07143&zoom=16.8&tilt=30

Try some of those plugins, it may inspire you some wonderful ideas for your project.