

Exercise 4: Interactive Web Mapping

Goal

The goal of this exercise is to practice adding additional interactive elements to your web map.

Part 1: Display the European Nuclear Power Plants Layer as Overlay

The main theme for this exercise is the distribution of nuclear power plants in Europe and the corresponding dataset can be found in the “geodata_eu” folder that you have previously unpacked.

Now transform this dataset in GeoJSON (in exercise 2 you have learned how to do this) and copy it to “C:\Workspace\wamp\mymap\geojson\” or a location that you have chosen for your GeoJSON data sources. In order to add this layer to the map, this time we will use AJAX – Asynchronous JavaScript and XML (see <https://github.com/calvinmetcalf/leaflet-ajax>) in order to simulate how dynamically generated datasources can also be embedded in a web map.

Before you start working on your new layer, in order to have a better overview of the code, feel free to remove any other overlay layers from your map that you do not need to keep. Also do not forget to check the intermediate results in the web browser from time to time.

First you need to include the Leaflet-ajax plugin in your map code:

```
<!-- AJAX.js plugin -->
<script src="leaflet-ajax-master/dist/leaflet.ajax.min.js"></script>
```

Then you can add the nuclear power plants dataset with the AJAX plugin to the map:

```
//loading the geojson for the power plants
var geojsonNPP = new L.GeoJSON.AJAX("geojson/eu_nuclear_plants.geojson").addTo(map);
```

In the next section, you are required to change the default markers into an attribute based symbolization, that highlights the “Status” of the individual power plant (if it is “active”, “closed” or “under construction”).

For this purpose, please start by consulting the documentation for the GeoJSON layer (see <http://leafletjs.com/reference.html#geojson>) and make use of the “pointToLayer” option. The symbol can be a circle marker that you can create using, for example, an L.circleMarker (see <http://leafletjs.com/reference.html#circlemarker>):

```
//loading the geojson for the power plants
var geojsonNPP = new L.GeoJSON.AJAX("geojson/eu_nuclear_plants.geojson", {
  pointToLayer: function (feature, latlng) {
    return L.circleMarker(latlng)});
```

When no styling options are given, blue features are drawn by default. Make use of the style option to change the default symbolization:

//loading the geojson for the power plants

```
var geojsonNPP = new L.GeoJSON.AJAX("geojson/eu_nuclear_plants.geojson",{
  style: function(feature) {
    return {
      radius:6,
      fillColor: "#FF006F",
      color: "#FF006F" ,
      weight: 1,
      opacity: 1,
      fillOpacity: 0.7
    }
  },
  pointToLayer: function (feature, latlng) {
    return L.circleMarker(latlng)});
```

Next, in order to achieve the attribute-based differentiation of symbols, you need to create a function that defines what colour is used for what attribute:

// Style function for the power plants layer

```
function getColor(status) {
  if (status == 'active'){return "#FF006F" }
  else if (status == 'closed'){return "#20679A" }
  else if (status == "under construction"){return "#0FB248"}
  else {return "#FCFF15"}
}
```

This function can be then used in the style for the “fillColor” and “color”:

//loading the geojson for the power plants

```
var geojsonNPP = new L.GeoJSON.AJAX("geojson/eu_nuclear_plants.geojson",{
  style: function(feature) {
    return {
      radius:6,
      fillColor: getColor(feature.properties.Status),
      color: getColor(feature.properties.Status),
      weight: 1,
      opacity: 1,
      fillOpacity: 0.7
    }
  },
  pointToLayer: function (feature, latlng) {
    return L.circleMarker(latlng)});
```

Finally check the resulted web map in the web browser.

Part 2: Add Additional Interactive Elements

In this second part of the exercise you will refine the know-how that you gained in the exercise 2, part 5, and add a pop-up to the nuclear power plants with additional information, add a zoom slider or display the coordinates of the mouse in the map. If needed, also search for, and read about additional information related to javascript “events” and “event listeners”.

In order to add a **pop-up** with additional information that is displayed on mouseover you will first define a function that defines what is displayed by the pop-up window:

```
// function for the popup window
function popUpNPP(feature,layer){
  layer.bindPopup('<b>' + feature.properties.Name + '</b><br><small>(' + feature.properties.Status + ' )
</small>');
  layer.on('mouseover', function(e){
    this.openPopup();
  });
}
```

Afterwards this pop-up window can be added to the GeoJSON layer by using the “onEachFeature” option.

For the **zoomslider** first add the plugin:

```
<!-- LEAFLET.ZOOMSLIDER.js plugin -->
<script src="Leaflet.zoomslider/src/L.Control.Zoomslider.js"></script>
<link href="Leaflet.zoomslider/src/L.Control.Zoomslider.css" rel="stylesheet" />
```

Then add the following CSS in the style element of the map index file:

```
/* Correction of a few mishaps introduced by the bootstrap CSS */
/* For the zoom slider plugin */
.leaflet-control-zoomslider-wrap leaflet-bar-part{
  color:#000;
}
.leaflet-control-zoomslider-body{
  box-sizing:content-box;
}
.leaflet-control-zoomslider-knob{
  box-sizing:content-box;
}
```

Finally, add the new control after the scale and layers stack elements:

```
map.addControl(new L.Control.Zoomslider());
```

and in the map element options, add:

```
zoomControl: false
```

For **displaying the coordinates** in the map study and apply the following code:

Link the Coordinates plugin:

```
<!-- LEAFLET.COORDINATES.js plugin and the associated stylesheet -->
<script src="Leaflet.Coordinates/dist/Leaflet.Coordinates-0.1.4.min.js"></script>
<link href="Leaflet.Coordinates/dist/Leaflet.Coordinates-0.1.4.css" rel="stylesheet"/>
```

Define an appropriate CSS:

```
/* Correction of a few mishaps introduced by the bootstrap CSS */
/* For the coordinates plugin */
.labelFirst{
  color:#333;
  font-size:10px;
  font-weight:normal;
  font:11px "Helvetica Neue", Arial, Helvetica, sans-serif;
}
.leaflet-left .leaflet-control{
  margin-left:5px;
}
.leaflet-control-coordinates{
  border-left-color:#777;
  border-left-width:2px;
  border-left-style:solid;
  border-right-color:#777;
  border-right-width:2px;
  border-right-style:solid;
}
```

Add the control to the map:

```
// Coordinates plugin
L.control.coordinates({position:"bottomleft", //optional default "bootomright"
  decimals:3, //optional default 4
  decimalSeperator:".", //optional default "."
  labelTemplateLat:"Latitude: {y}", //optional default "Lat: {y}"
  labelTemplateLng:"Longitude: {x}", //optional default "Lng: {x}"
  enableUserInput:false, //optional default true
  useDMS:true, //optional default false
  useLatLngOrder: true //ordering of labels, default false-> lng-lat
}).addTo(map);
```

Add the attribution to the geosjon layers. The attribution property is not natively supported for geojson layers in the actual version of Leaflet. However, there is an easy fix:

```
eu_nuclear_plants.getAttribution = function() { return ' your source information here';
```

If you have time left, optionally you may like to implement a **sidebar** as documented in (<https://github.com/turbo87/leaflet-sidebar/> or <https://github.com/Turbo87/sidebar-v2>).

Discussion of the Exercise

Congratulations, you have finished the exercises for the first day! You now know a range of interactive elements that can be added to a web map.

If you still have time left, please continue with the optional exercise for day 1 (exercise A).