06. LEAFLET GeoJSON.

6.1. Using GeoJSON with Leaflet.

GeoJSON is becoming a very popular data format among many GIS technologies and services — it's simple, lightweight, straightforward, and Leaflet is quite good at handling it. In this example, you'll learn how to create and interact with map vectors created from GeoJSON objects.

https://tools.ietf.org/html/rfc7946

6.2. About GeoJSON.

According to **GeoJSON Specification (RFC 7946)**:

> **GeoJSON** is a format for encoding a variety of geographic data structures […]. A GeoJSON object may represent a region of space (a Geometry), a spatially bounded entity (a Feature), or a list of Features (a FeatureCollection). GeoJSON supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection. Features in GeoJSON contain a Geometry object and additional properties, and a FeatureCollection contains a list of Features.

Leaflet supports all of the GeoJSON types above, but Features and FeatureCollections work best as they allow you to describe features with a set of properties. We can even use these properties to style our Leaflet vectors. Here's an example of a simple GeoJSON feature:

```
var geojsonFeature = {
    "type": "Feature",
    "properties": {
        "name": "Coors Field",
        "amenity": "Baseball Stadium",
        "popupContent": "This is where the Rockies play!"
    },
    "geometry": {
        "type": "Point",
        "coordinates": [-104.99404, 39.75621]
    }
};
```

6.3. The GeoJSON layer

GeoJSON objects are added to the map through a GeoJSON layer. To create it and add it to a map, we can use the following code:

Example:

```
L.geoJSON(geojsonFeature).addTo(map);
```

GeoJSON objects may also be passed as an array of valid GeoJSON objects.

Example:

```
var myLines = [{
    "type": "LineString",
    "coordinates": [[-100, 40], [-105, 45], [-110, 55]]
}, {
    "type": "LineString",
    "coordinates": [[-105, 40], [-110, 45], [-115, 55]]
}];
```

Alternatively, we could create an empty GeoJSON layer and assign it to a variable so that we can add more features to it later.

Example:

```
var myLayer = L.geoJSON().addTo(map);
myLayer.addData(geojsonFeature);
```

6.4. GeoJSON options

style

The style option can be used to style features two different ways. First, we can pass a simple object that styles all paths (polylines and polygons) the same way:

Example:

```
var myLines = [{
    "type": "LineString",
    "coordinates": [[-100, 40], [-105, 45], [-110, 55]]
}, {
    "type": "LineString",
    "coordinates": [[-105, 40], [-110, 45], [-115, 55]]
}];

var myStyle = {
    "color": "#ff7800",
    "weight": 5,
    "opacity": 0.65
};

L.geoJSON(myLines, {
    style: myStyle
}).addTo(map);
```

Alternatively, we can pass a function that styles individual features based on their properties. In the example below we check the "party" property and style our polygons accordingly:

Example:

```
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);
```

pointToLayer

Points are handled differently than polylines and polygons. By default simple markers are drawn for GeoJSON Points. We can alter this by passing a pointToLayer function in a GeoJSON options object when creating the GeoJSON layer. This function is passed a LatLng and should return an instance of ILayer, in this case likely a Marker or CircleMarker.

Here we're using the pointToLayer option to create a *CircleMarker*:

Example:

```
var geojsonMarkerOptions = {
    radius: 8,
    fillColor: "#ff7800",
    color: "#000",
    weight: 1,
    opacity: 1,
    fillOpacity: 0.8
};

L.geoJSON(someGeojsonFeature, {
    pointToLayer: function (feature, latlng) {
```

```
                return L.circleMarker(latlng, geojsonMarkerOptions);
        }
    }).addTo(map);
```

We could also set the style property in this example — Leaflet is smart enough to apply styles to GeoJSON points if you create a vector layer like circle inside the pointToLayer function.

onEachFeature

The onEachFeature option is a function that gets called on each feature before adding it to a GeoJSON layer. A common reason to use this option is to attach a popup to features when they are clicked.

Example:

```
function onEachFeature(feature, layer) {
    // does this feature have a property named popupContent?
    if (feature.properties && feature.properties.popupContent) {
        layer.bindPopup(feature.properties.popupContent);
    }
}

var geojsonFeature = {
    "type": "Feature",
    "properties": {
        "name": "Coors Field",
        "amenity": "Baseball Stadium",
        "popupContent": "This is where the Rockies play!"
    },
    "geometry": {
        "type": "Point",
        "coordinates": [-104.99404, 39.75621]
    }
};

L.geoJSON(geojsonFeature, {
    onEachFeature: onEachFeature
}).addTo(map);
```

filter

The filter option can be used to control the visibility of GeoJSON features. To accomplish this we pass a function as the filter option. This function gets called for each feature in your GeoJSON layer, and gets passed the feature and the layer. You can then utilise the values in the feature's properties to control the visibility by returning true or false.

In the example below "Busch Field" will not be shown on the map.

Example:

```
var someFeatures = [{
    "type": "Feature",
    "properties": {
        "name": "Coors Field",
        "show_on_map": true
    },
    "geometry": {
        "type": "Point",
        "coordinates": [-104.99404, 39.75621]
    }
}, {
    "type": "Feature",
    "properties": {
        "name": "Busch Field",
        "show_on_map": false
    },
    "geometry": {
        "type": "Point",
        "coordinates": [-104.98404, 39.74621]
    }
}];

L.geoJSON(someFeatures, {
    filter: function(feature, layer) {
        return feature.properties.show_on_map;
    }
}).addTo(map);
```