13.5.6 Map GeoJSON Polygons

The last example that Sadhana will walk you through is mapping GeoJSON polygon type objects. Polygons are more complex than Points or LineStrings. Sadhana is going to review the structure of a polygon's type object and then walk you through mapping these objects.

First, Sadhana wants you to create a branch on the main repository for mapping polygons. Create a branch called "Mapping_GeoJSON_Polygons" with the following folder structure. Copy the folders and files from your Mapping_GeoJSON_Linestrings branch and add them to the Mapping_GeoJSON_Polygons folder.

- Mapping_GeoJSON_Polygons
  - `index.html`
  - static
    - css
      - `style.css`
    - js
      - `config.js`
      - `logic.js`

Next, download the `torontoNeighborhoods.json` file and put it in the Mapping_Earthquakes repository.

Download torontoNeighborhoods.json (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_13/torontoNeighborhoods.json)

Let's take a look at the `torontoNeighborhoods.json` file, which contains the coordinates for each neighborhood in Toronto, Canada.

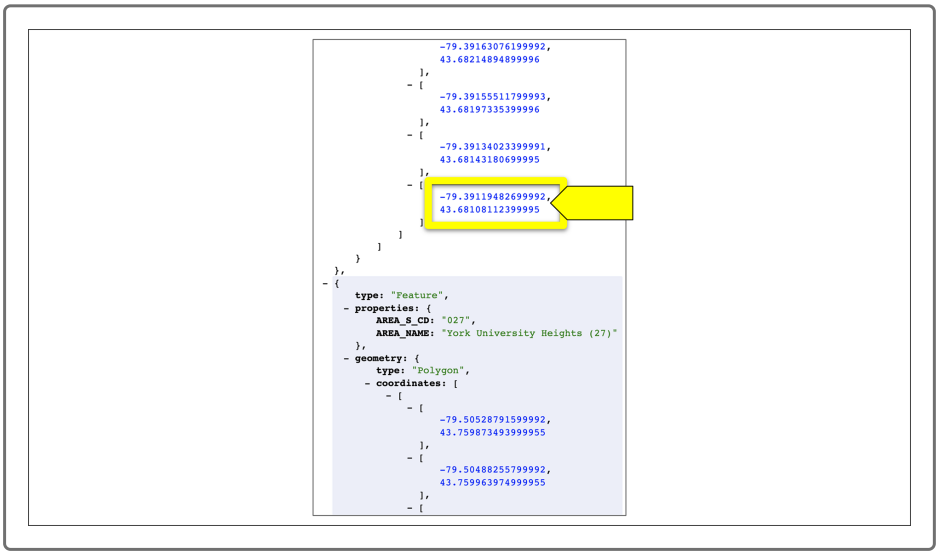
NOTE

> Depending on your computer's processor, the `torontoNeighborhoods.json` data may take awhile to load.

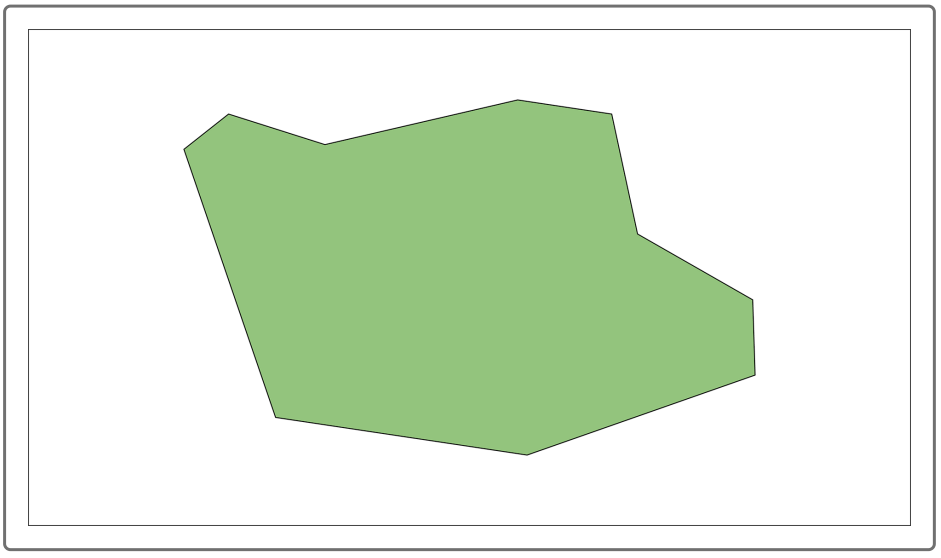
In the JSON data, we can see that the geometry type is "Polygon." To form a polygon, the coordinates have to be an array of linear ring (LinearRing) coordinate arrays. A LinearRing is a LineString with at least four or more sets of coordinates, where the starting and end points have the same coordinates.

In our first object in the `torontoNeighborhoods.json` file, the starting point coordinates are `[-79.39119482699992, 43.68108112399995]`.

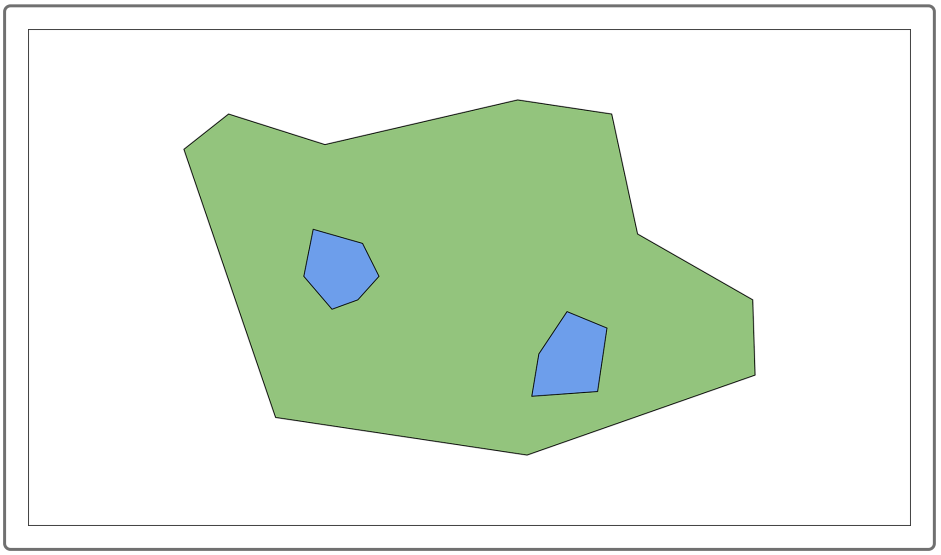
If we scroll to the end of the first object, the end point coordinates are also `[-79.39119482699992, 43.68108112399995]`.



In addition, the polygon can have multiple LinearRings, each with a different shape. The outer boundary of the polygon is one LinearRing shape, and there can be one or more LinearRing shapes inside that polygon. Consider the following solid polygon:



This is a solid polygon, like the neighborhood boundaries we'll map. There are no LinearRing shapes inside. However, if we add boundaries around the livable areas of a neighborhood with two bodies of water, like a lake or pond, then the polygon might look like the following, where the blue areas represent those bodies of water. This polygon would have multiple LinearRing shapes inside the outer boundary LinearRing:



Now that we know more about polygons, let's map this data. Since we have copied the `logic.js` file from the Mapping_GeoJSON_Linestrings and added it to our Mapping_GeoJSON_Polygons folder, we'll make some changes to the `logic.js` file.

First, we'll use the `Streets` and `Satellite Streets` map styles, so we'll need to change the variables and the API code for each style:

- Use `streets` as the variable for the `Streets` style.
- Use `satelliteStreets` as the variable for the `Satellite Streets` style.

Next, change the center of our map to Toronto with the coordinates `[43.7, -79.3]`, with a zoom level of 11, and we'll make the default layer `satelliteStreets`.

Our `logic.js` file should look like the following:

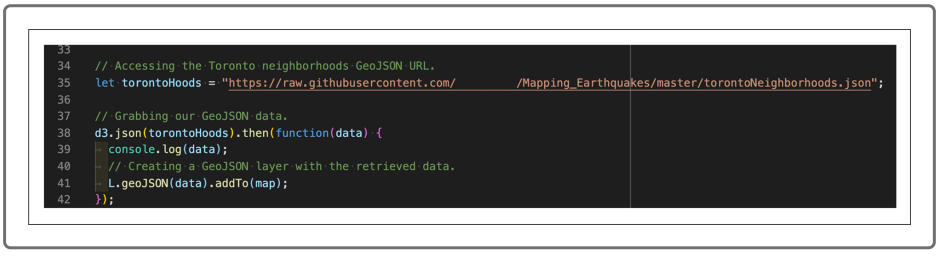
```
// Add console.log to check to see if our code is working.
console.log("working");

// We create the tile layer that will be the background of our map.
let streets = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/streets-v11/tiles/{z}/{x}/{y}?access_token={accessToken}', {
  attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https://creativecommons.or
  maxZoom: 18,
  accessToken: API_KEY
});

// We create the tile layer that will be the background of our map.
let satelliteStreets = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/satellite-streets-v11/tiles/{z}/{x}/{y}?access_token={accessToke
  attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https://creativecommons.or
  maxZoom: 18,
  accessToken: API_KEY
});

// Create a base layer that holds both maps.
let baseMaps = {
  "Streets": streets,
  "Satellite Streets": satelliteStreets
};

// Create the map object with center, zoom level and default layer.
let map = L.map('mapid', {
  center: [43.7, -79.3],
  zoom: 11,
  layers: [satelliteStreets]
});

// Pass our map layers into our layer control and add the layer control to the map.
L.control.layers(baseMaps).addTo(map);
```

Now we'll create a `torontoHoods` variable in our `logic.js` file and assign it to the URL for the `torontoNeighborhoods.json` file on GitHub, which should look like the following:

```
// Accessing the Toronto neighborhoods GeoJSON URL.
let torontoHoods = "https://raw.githubusercontent.com/<GitHub_name>/Mapping_
```

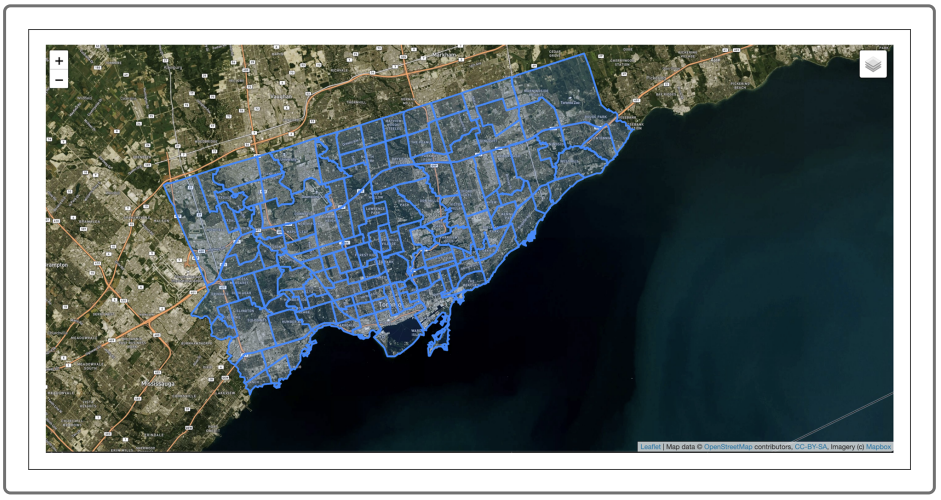
Finally, we'll need to edit the `d3.json()` method that includes the `L.geoJSON()` layer to plot the data.

After we add the code to access and parse the `torontoNeighborhoods.json` data, the `logic.js` file should look like the following:



```
33
34  // Accessing the Toronto neighborhoods GeoJSON URL.
35  let torontoHoods = "https://raw.githubusercontent.com/          /Mapping_Earthquakes/master/torontoNeighborhoods.json";
36
37  // Grabbing our GeoJSON data.
38  d3.json(torontoHoods).then(function(data) {
39    console.log(data);
40    // Creating a GeoJSON layer with the retrieved data.
41    L.geoJSON(data).addTo(map);
42  });
```

After you save your `logic.js` file and open your `index.html` file in your browser, your map should look like the following:



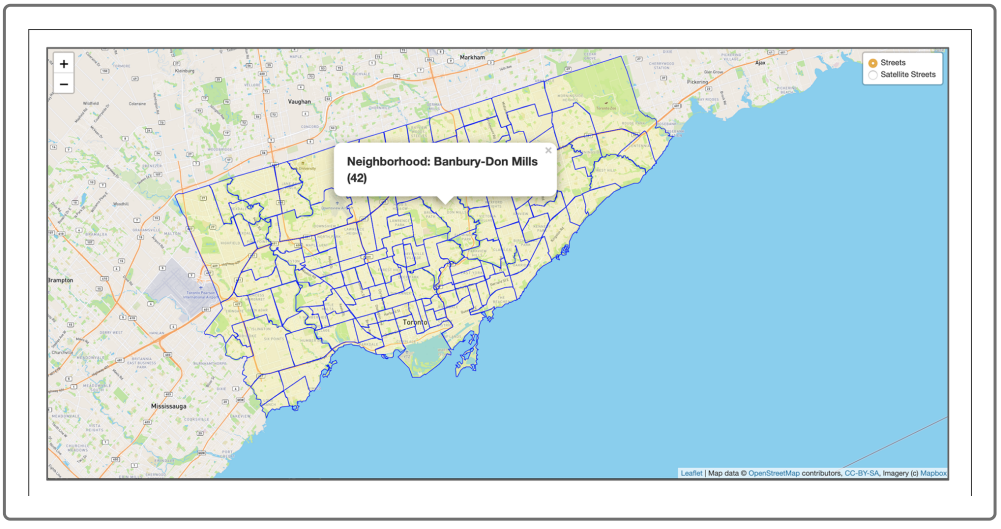
Great job! You're getting really good at mapping GeoJSON data!

SKILL DRILL

Edit your `logic.js` file with the following changes:

1. Make the lines blue, with a `weight` of 1.
2. Make the polygon fill color yellow.
3. Add a popup to show each neighborhood.
4. Make the default map layer `Streets` with `Satellite Streets` as the second option.

Your map should look like the following:



NOTE

> For more information, see the Leaflet documentation on changing **color options and other features of polygons (https://leafletjs.com/reference-1.6.0.html#path-color)** .

ADD/COMMIT/PUSH

Add, commit, and push your changes to your Mapping_GeoJSON_Polygons branch. Don't delete the branch so that others can use it to learn how to map lines.