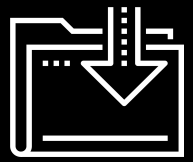
Data Visualization with Leaflet

Data Boot Camp

Lesson 15.1



© 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.

Class Objectives

By the end of this lesson, you will be able to:

Discuss the benefits of visualizing data with maps.

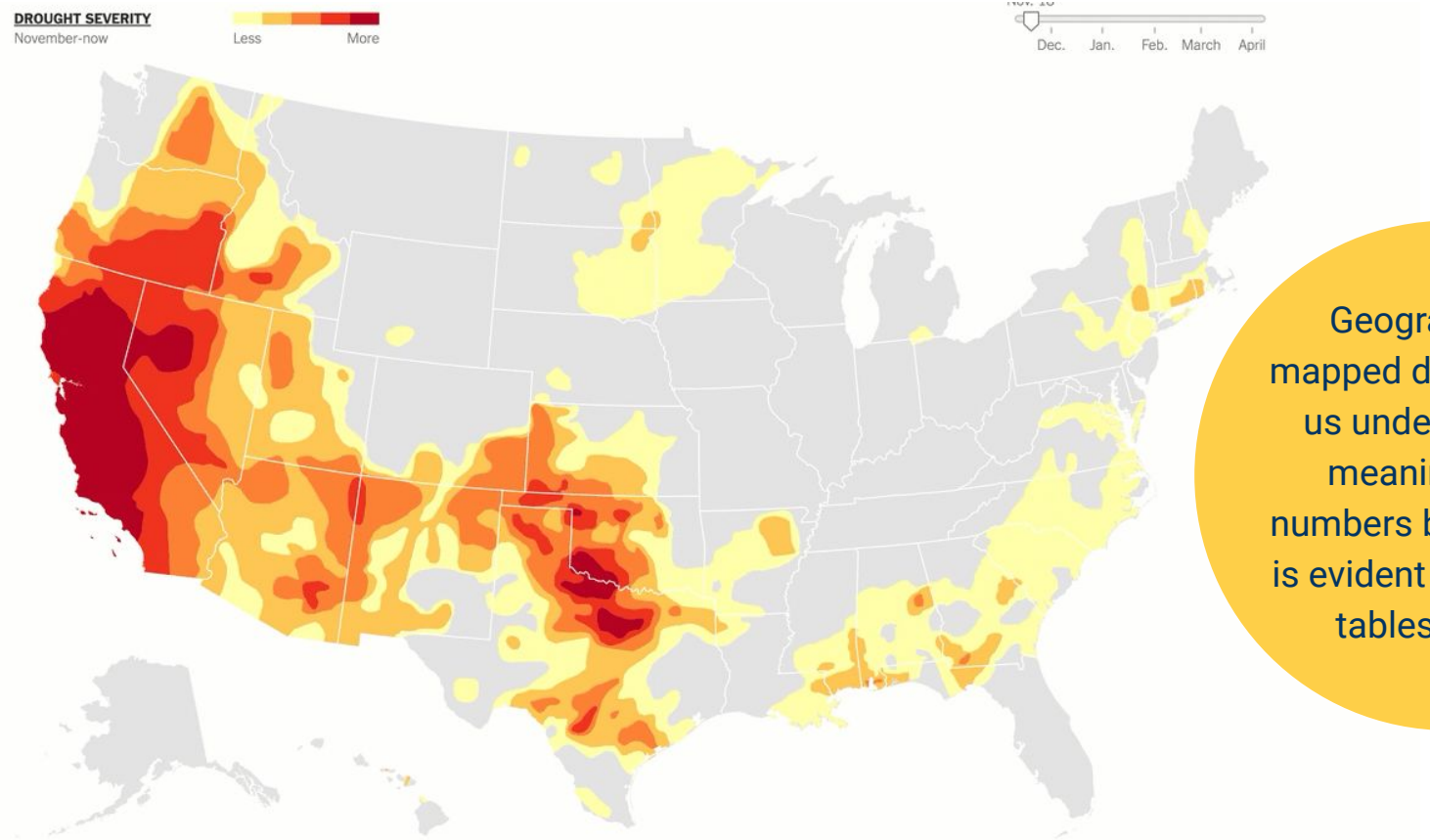
Create maps and plot data with the Leaflet.js library.

Parse data from the GeoJSON format to create map-based data visualizations.

Describe the concepts of layers and layer controls and how they are applied to add interactivity to maps.

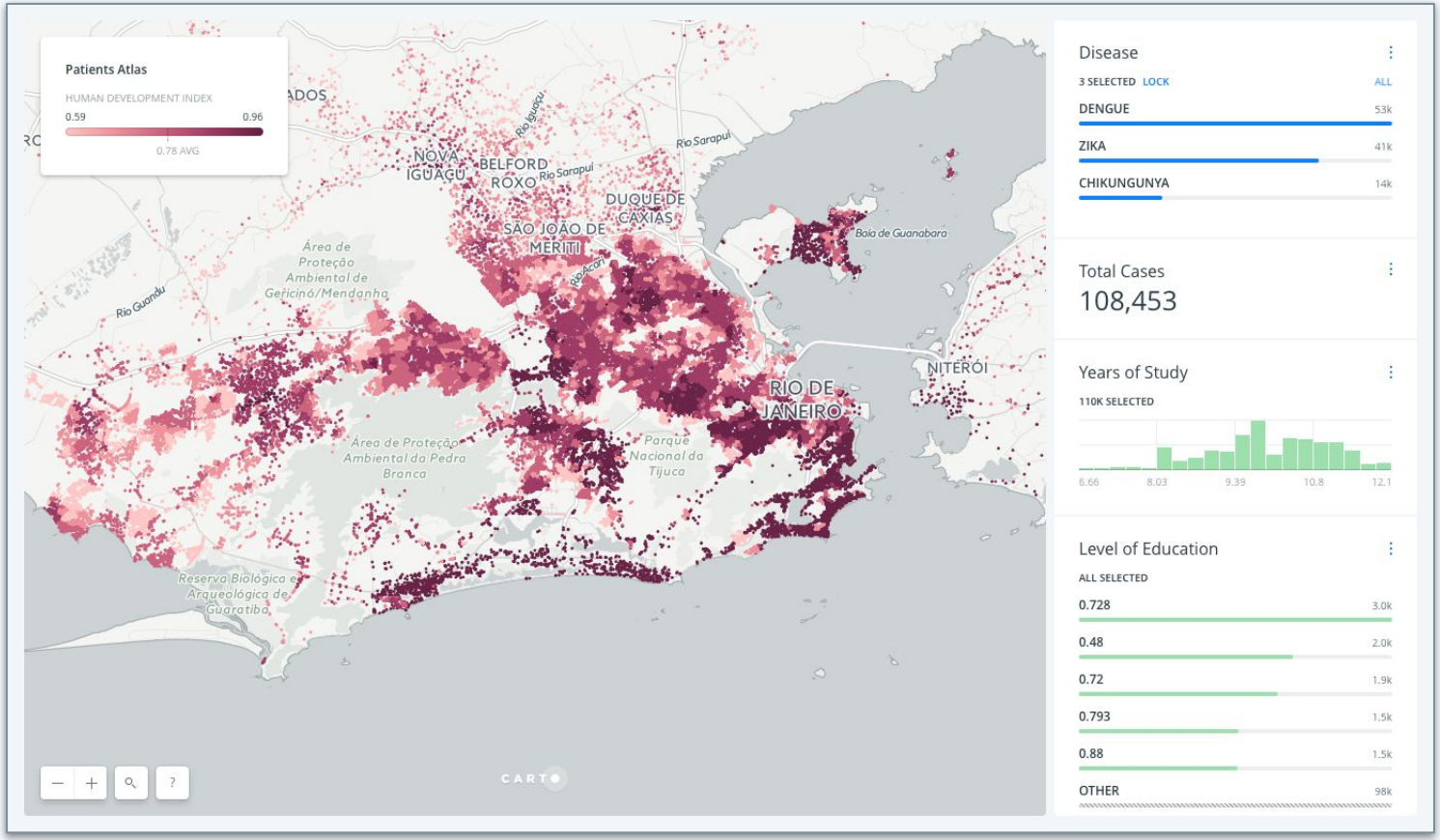
Why is visualizing data in a geographical context beneficial?

Data Visualization with Leaflet



Geographically mapped data can help us understand the meaning of the numbers beyond what is evident in analyzing tables or lists.

Data Visualization with Leaflet



What other kinds of datasets or problems seem suitable for this type of data visualization?

Time to Code

Create Our First Map using Leaflet

Suggested Time:

10 minutes

Questions?

Instructor Demonstration

Add Markers to the Map

Activity: City Markers

In this activity, you'll create a map and plot markers for five United States cities.

Suggested Time:

15 minutes

Activity: City Markers

Instructions

Use the starter files `index.html`, `logic.js`, and `style.css` provided in the Unsolved folder.

Add markers to your map for the following US cities:

City	Latitude	Longitude
New York	40.7128	-74.0059
Los Angeles	34.0522	-118.2437
Houston	29.7604	-95.3698
Omaha	41.2524	-95.9980
Chicago	41.8781	-87.6298

For each city, create a marker with a popup that displays the city's name and population. You can either look up the population for each city or make one up for this exercise.

Bonus

A popup takes a string of HTML. If you finish early, try experimenting with passing different tags or custom CSS.

Hint

Don't forget to add the marker to your map after creating it.

Time’s Up! Let’s Review.

Markers are great, but what if we want to represent an area that spans more than a single point on a map?

Leaflet allows us to define and plot Scalable Vector Graphics (SVG) shapes to use as markers. When using the Leaflet API, we refer to these SVG shapes as **vector layers**.

Instructor Demonstration

Add Other Types of Markers

Add Markers to the Map

Adding a new marker by adding a new marker object:

logic.js

```
let myMap = L.map("map", {
 center: [45.52, -122.67],
 zoom: 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(myMap);
```

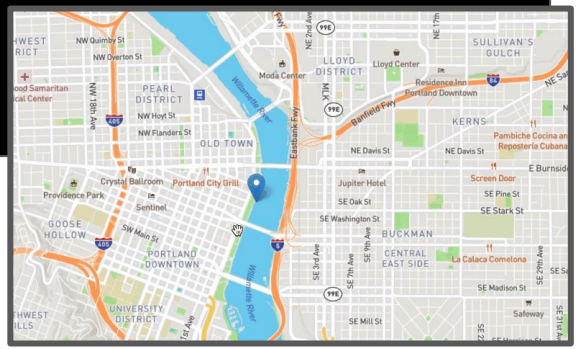
Add Markers to the Map

Method we use to add each map layer

Method we use to add text to the marker when clicked

```
// Create a new marker
// Pass in some initial options, and then add
it to the map using the addTo method
let marker = L.marker([45.52, -122.67], {
 draggable: true,
 title: "My First Marker"
}).addTo(myMap);

// Binding a pop-up to our marker
marker.bindPopup("Hello There!");
```



Activity: Other Markers

In this activity, you will work with different types of vector layers.

Suggested Time:

10 minutes

Activity: Other Markers

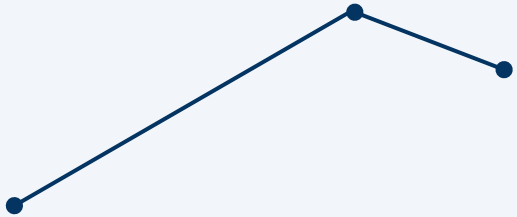
Instructions	Use the starter files `index.html`, `logic.js`, and `style.css` provided in the Unsolved folder. Create the following vector layers, and then add them to the map: • A red circle over the city of Dallas (`[32.7767, -96.7979]`) • A line connecting NYC (`[40.7128, -74.0060]`) to Toronto (`[43.6532, -79.3832]`) • A polygon that covers the area inside Atlanta (`[33.7490, -84.3880]`), Savannah (`[32.0809, -81.0912]`), Jacksonville (`[30.3322, -81.6557]`), and Montgomery (`[32.3792, -86.3077]`)
Hint	Note that the logic.js file contains some starter code. Use the Vector Layers section of the Leaflet documentation for reference.

Time’s Up! Let’s Review.

What are some of the different types of vector shapes that are available to us?

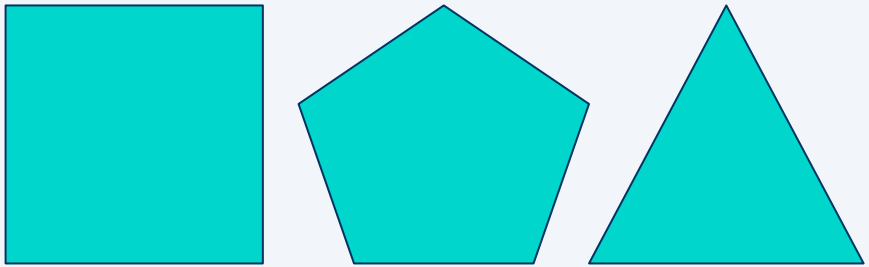
Types of Vector Shapes

Polyline

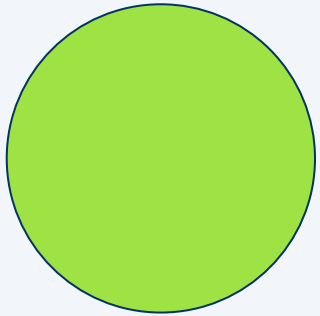


Rectangle

Polygon



Circle



What arguments do our vector layers accept when we create them?

Arguments

They accept the following two arguments:

01 An array of coordinates that describes where our shape should appear.

02 A configuration object that describes the styles to apply to the shape.

The Leaflet documentation for Path options has a complete list of the style options that we can use for vector layers.

Break

Instructor Demonstration

Visualize City Populations

Activity: Visualize World GDP Per Capita

In this activity, you will work with different types of vector layers.

Suggested Time:

15 minutes

Activity: Visualize World GDP Per Capita

Instructions	Use the starter files `index.html`, `logic.js`, and `style.css` provided in the Unsolved folder.
	Add your code to logic.js to render the following: • A circle for each country in the dataset. • A radius size that you determine from the country's GDP per capita.
	A color for each circle that you determine as follows: • For countries with more than 100,000 GDP per capita, set the color of the circle to **yellow.** • For countries with more than 75,000 GDP per capita, set the color of the circle to blue. • For countries with more than 50,000 GDP per capita, set the color of the circle to green. • Render the remaining country circles in violet.
	Make sure that each vector layer has a popup containing the country's name and GDP per capita.
Hint	Universally adjust the radius for better visuals.
	Refer to the Leaflet documentation for Path options if you get stuck creating vector layers.

Time’s Up! Let’s Review.

Visualize World GDP Per Capita

Key concepts:

We set our marker's `radius` based on the country's GDP per capita.

We use conditionals to determine color.

We use custom map styles as a way to express data.

We use a popup to display additional information.

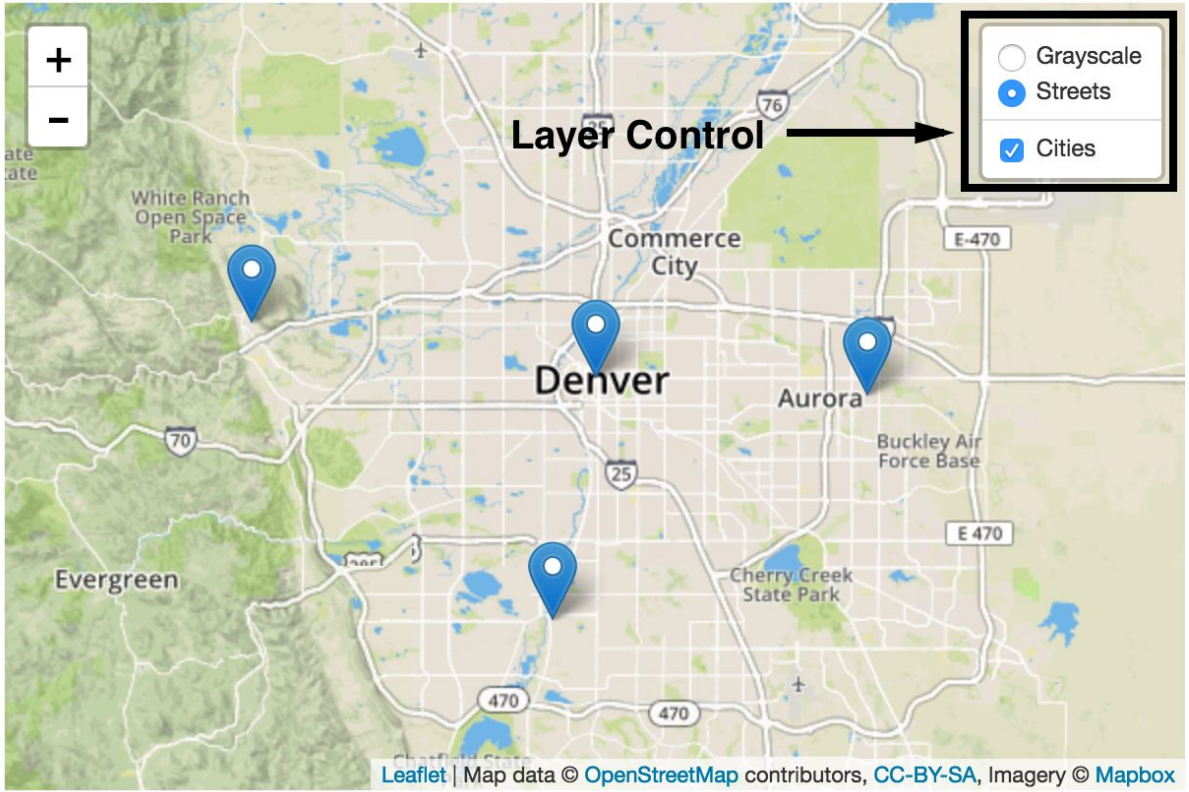
Questions?

Layer Groups and Layer Controls

Layer Groups and Layer Controls

So far, we've used only one layer with our maps.

The OpenStreetMap API has supplied these maps to us. However, we can use multiple layers on the same map and then toggle between layers using layer controls.



Leaflet Has Two Types of Layers

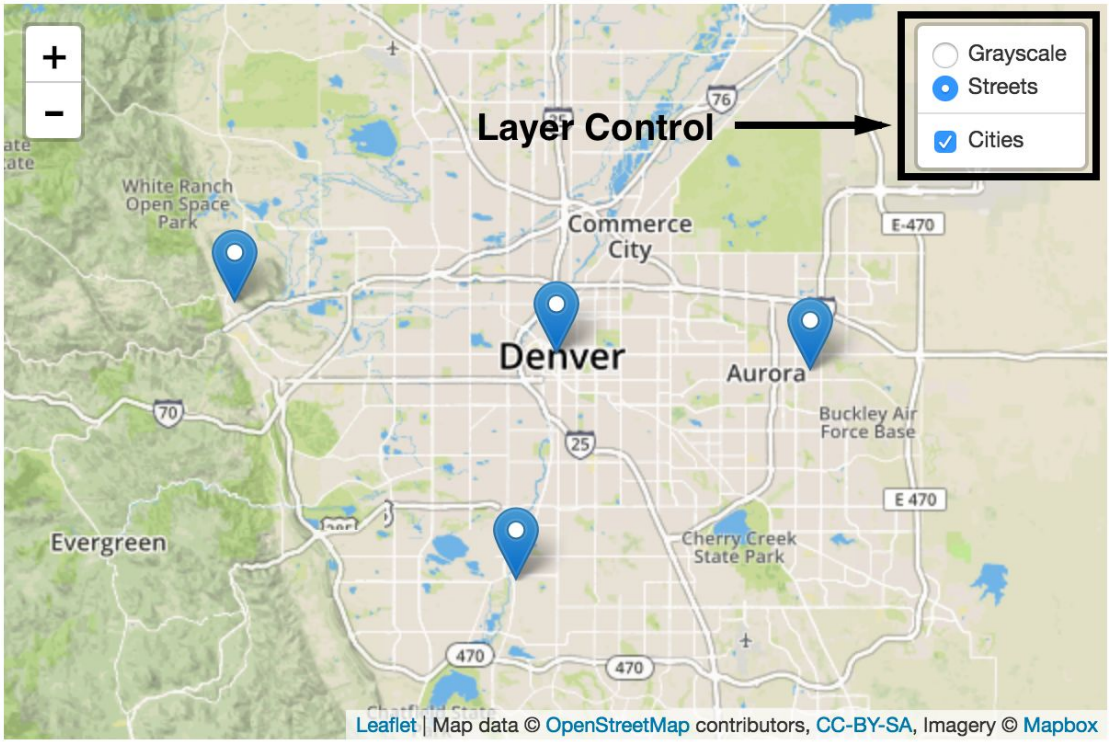
Leaflet has two types of layers.

Base Layers

Base layers are mutually exclusive (only one can be visible at a time).
In this example, they are the Streets and Grayscale layers. We can observe only one or the other at a time, and one of these layers must always be visible.

Overlays

We can turn them off entirely.
In this example, the overlay contains the city markers.



Layer Groups

We can group our markers together to create new overlays by using layer groups. We can then toggle a set of related markers on or off as a group.

Example: You have a bunch of layers you want to combine into a group to handle them as one in your code:

```
var littleton = L.marker([39.61, -105.02]).bindPopup('This is Littleton, CO.'),
    denver    = L.marker([39.74, -104.99]).bindPopup('This is Denver, CO.'),
    aurora    = L.marker([39.73, -104.8]).bindPopup('This is Aurora, CO.'),
    golden    = L.marker([39.77, -105.23]).bindPopup('This is Golden, CO.');
```

Instead of adding them directly to the map, you can do the following using the `layerGroup` class.

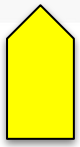
```
var cities = L.layerGroup([littleton, denver, aurora, golden]);
```

Layer Groups

Easy enough! Now you have `cities` layer that combines your city markers into one layer you can add or remove from the map at once.

```
var littleton = L.marker([39.61, -105.02]).bindPopup('This is Littleton, CO.'),
    denver    = L.marker([39.74, -104.99]).bindPopup('This is Denver, CO.'),
    aurora    = L.marker([39.73, -104.8]).bindPopup('This is Aurora, CO.'),
    golden    = L.marker([39.77, -105.23]).bindPopup('This is Golden, CO.');
```

```
var cities = L.layerGroup([littleton, denver, aurora, golden]);
```



We are creating markers in the same way as before, but instead of directly applying the markers individually to the map, we add these markers to a layer group named cities.

Activity: Perform a Layer

In this activity, you'll expand on a previous activity by adding an overlay to represent state populations. This layer will appear on the map as white circle vectors.

Suggested Time:

15 minutes

Activity: Perform a Layer

Instructions

Use the starter files `index.html`, `logic.js`, and `style.css` provided in the Unsolved folder.

Open the logic.js file in the Unsolved folder.

Add logic to this file as follows:

- Create a layer group for city markers and a separate layer group for state markers. Note that the `cityMarkers` and `stateMarkers` arrays contain all the markers, which have been created for you. Store these layer groups in variables named `cities` and `states`.
- Create a `baseMaps` object to contain the `street` and `topo` tiles, which have already been defined.
- Create an `overlayMaps` object to contain the `State Population` and `City Population` layers.
- Add a `layers` key to the options object in the `L.map` method, and set its value to an array that contains our `street`, `states`, and `cities` layers. These will determine which layers display when the map first loads.
- Create a layer control, and pass it the `baseMaps` and `overlayMaps` objects. Add the layer control to the map.

Hint

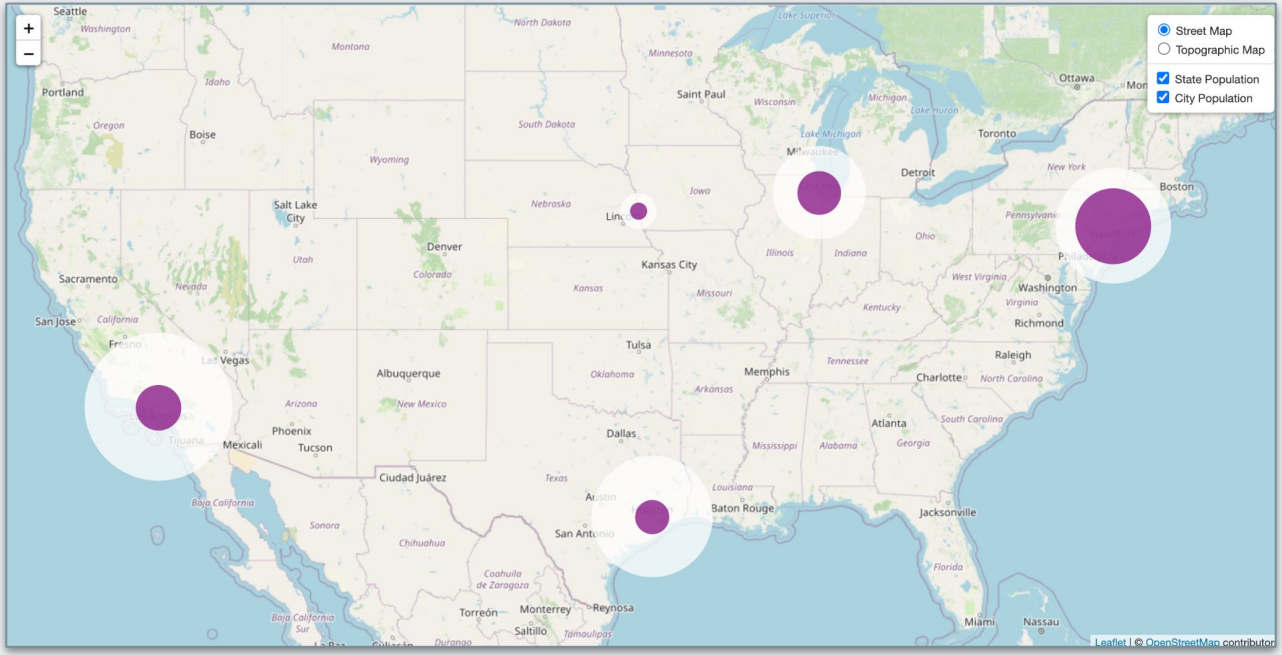
Refer to the Leaflet Layer Groups and Layers Control tutorial if you need help.

If everything is correct, you can toggle between the Street Map and Topographic Map base layers and turn the State Population and City Population overlays on and off.

Time’s Up! Let’s Review.

Activity: Perform a Layer

With this map, we can find out the portion of a state's population that lives in its largest city! Besides having the largest population, New York also has the largest percentage of its state's population.



Fun Fact
Over 40% of New York State's population lives in New York City.

Questions?

GeoJSON

GeoJSON

Most applications that we build will pull data from existing datasets. And, one of the easiest ways to deliver geographical data is by using the GeoJSON format.


```
{"type":"FeatureCollection","metadata":{"generated":1650905924000,"url":"https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_hour.geojson","title":"USGS All Earthquakes, Past
Hour","status":200,"api":"1.10.3","count":9},"features":[{"type":"Feature","properties":{"mag":2.27999997,"place":"24 km E of Honaunau-Napoopoo,
Hawaii","time":1650905648720,"updated":1650905847330,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/hv72994952","detail":"https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/h
v72994952.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"automatic","tsunami":0,"sig":80,"net":"hv","code":"72994952","ids":",hv72994952,","sources":",hv,","types":",origin,phase-
data,","nst":25,"dmin":null,"rms":0.289999992,"gap":114,"magType":"md","type":"earthquake","title":"M 2.3 - 24 km E of Honaunau-Napoopoo, Hawaii"},"geometry":{"type":"Point","coordinates":
[-155.629669189453,19.4268341064453,0.0199999995529652]},"id":"hv72994952"},
{"type":"Feature","properties":{"mag":2.2,"place":"55 km NNE of Petersville,
Alaska","time":1650905398262,"updated":1650905580613,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/ak0225agmgzq","detail":"https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail
/ak0225agmgzq.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"automatic","tsunami":0,"sig":74,"net":"ak","code":"0225agmgzq","ids":",ak0225agmgzq,","sources":",ak,","types":",origin,
phase-data,","nst":null,"dmin":null,"rms":0.62,"gap":null,"magType":"ml","type":"earthquake","title":"M 2.2 - 55 km NNE of Petersville, Alaska"},"geometry":{"type":"Point","coordinates":
[-150.3388,62.9583,72.9]},"id":"ak0225agmgzq"},
{"type":"Feature","properties":{"mag":2.5,"place":"5 km NNW of Point MacKenzie,
Alaska","time":1650904899330,"updated":1650905072426,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/ak0225agkpuz","detail":"https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail
/ak0225agkpuz.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"automatic","tsunami":0,"sig":96,"net":"ak","code":"0225agkpuz","ids":",ak0225agkpuz,","sources":",ak,","types":",origin,
phase-data,","nst":null,"dmin":null,"rms":0.61,"gap":null,"magType":"ml","type":"earthquake","title":"M 2.5 - 5 km NNW of Point MacKenzie, Alaska"},"geometry":{"type":"Point","coordinates":
[-150.0251,61.4034,41.1]},"id":"ak0225agkpuz"},
{"type":"Feature","properties":{"mag":1.27,"place":"12km ENE of Cloverdale,
CA","time":1650904728960,"updated":1650905231081,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/nc73723250","detail":"https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/nc737
23250.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"automatic","tsunami":0,"sig":25,"net":"nc","code":"73723250","ids":",nc73723250,","sources":",nc,","types":",nearby-
cities,origin,phase-data,scitech-link,","nst":24,"dmin":0.004918,"rms":0.02,"gap":127,"magType":"md","type":"earthquake","title":"M 1.3 - 12km ENE of Cloverdale, CA"},"geometry":
{"type":"Point","coordinates":[-122.8816681,38.8334999,2.32]},"id":"nc73723250"},
{"type":"Feature","properties":{"mag":4.6,"place":"250 km SE of Chignik,
Alaska","time":1650904288110,"updated":1650904843390,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/ak0225agijwj","detail":"https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail
/ak0225agijwj.geojson","felt":null,"cdi":null,"mmi":1.777,"alert":null,"status":"automatic","tsunami":1,"sig":326,"net":"ak","code":"0225agijwj","ids":",at00rawlwl,ak0225agijwj,","sources":",at,ak,","
types":",impact-link,origin,phase-data,shakemap,","nst":null,"dmin":null,"rms":1,"gap":null,"magType":"ml","type":"earthquake","title":"M 4.6 - 250 km SE of Chignik, Alaska"},"geometry":
{"type":"Point","coordinates":[-156.1538,54.4498,1.9]},"id":"ak0225agijwj"},
{"type":"Feature","properties":{"mag":2.68,"place":"23 km W of Volcano,
Hawaii","time":1650903668240,"updated":1650905296993,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/hv72994947","detail":"https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/h
v72994947.geojson","felt":1,"cdi":2.2,"mmi":null,"alert":null,"status":"reviewed","tsunami":0,"sig":111,"net":"hv","code":"72994947","ids":",hv72994947,","sources":",hv,","types":",dyfi,origin,phase-
data,","nst":29,"dmin":null,"rms":0.11,"gap":59,"magType":"ml","type":"earthquake","title":"M 2.7 - 23 km W of Volcano, Hawaii"},"geometry":{"type":"Point","coordinates":
[-155.4605,19.422,7.78]},"id":"hv72994947"},
{"type":"Feature","properties":{"mag":0.69,"place":"18km W of Searles Valley,
CA","time":1650903212590,"updated":1650903424169,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/ci40245784","detail":"https://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/ci402
45784.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"automatic","tsunami":0,"sig":7,"net":"ci","code":"40245784","ids":",ci40245784,","sources":",ci,","types":",nearby-
cities,origin,phase-data,scitech-link,","nst":10,"dmin":0.03018,"rms":0.15,"gap":149,"magType":"ml","type":"earthquake","title":"M 0.7 - 18km W of Searles Valley, CA"},"geometry":
{"type":"Point","coordinates":[-117.5981667,35.7855,7.32]},"id":"ci40245784"},
{"type":"Feature","properties":{"mag":0.89,"place":"7km WNW of Cobb,
```

GeoJSON

GeoJSON is an open-standard format for representing simple geographical features along with their nonspatial attributes by using JavaScript Object Notation (JSON).

This GeoJSON document depicts all the earthquakes that have taken place across the globe within the past hour.


```
{"type":"FeatureCollection","metadata":{"generated":1650905924000,"url":"https://earthquak
e.usgs.gov/earthquakes/feed/v1.0/summary/all_hour.geojson","title":"USGS All Earthquakes,
PastHour","status":200,"api":"1.10.3","count":9},"features":[{"type":"Feature","properties
":{"mag":2.27999997,"place":"24 km E of Honaunau-Napoopoo,
```

GeoJSON

GeoJSON has the following characteristics:

- It represents geographical features by coordinates and can attach other properties to these features.
- The types of features are the following:
  - `Point`
  - `LineString`
  - `Polygon`
  - `MultiPoint`
  - `MultiLineString`
  - `MultiPolygon`

Questions?

Activity: GeoJSON

In this activity, you'll work with GeoJSON data from the U.S. Geological Survey (USGS) to make plot markers representing occurrences of earthquakes.

Suggested Time:

15 minutes

Activity: GeoJSON

Instructions

Use the starter files `index.html`, `logic.js`, and `style.css` provided in the Unsolved folder.

- Open the logic.js file.
- Note that your starter code places an API call to the USGS Earthquake Hazards Program API. Take a moment to study the `features` array that we extract from the response.
- Add logic to create a GeoJSON layer that contains all the features retrieved from the API call, and add the layer directly to the map. You can reference today's previous activities and the Leaflet Using GeoJSON with Leaflet tutorial.
- Create an `overlayMaps` object by using the newly created earthquake GeoJSON layer. Pass `overlayMaps` to the layer control.

Bonus

Create a separate overlay for the GeoJSON and a base layer by using the `street` tile layer and the `topo` tile layer.
Add these to a layer control. If you get stuck, refer to the previous activity.

Add a popup to each marker to display the time and location of the earthquake at that location.

Hint

Refer to the following Leaflet documentation:

- GeoJSON
- Using GeoJSON with Leaflet tutorial

If you want to reformat the time from the GeoJSON data for the bonus, you may wish to use the JavaScript method Date()

Time's Up! Let's Review.

How do we create a GeoJSON layer?

Create a GeoJSON layer

We pass all the earthquake feature data to the `L.GeoJSON` method.

We save its return value (which is the new Leaflet GeoJSON layer) in the `earthquakes` variable.

What's happening with the `onEachFeature` function that we defined?

onEachFeature Function

During layer creation, Leaflet supplies a built-in hook named `onEachFeature`.

We can define a function that performs custom functionality with the addition of each feature object to the GeoJSON layer.

In this case, we give each layer a tooltip with the time and location of the earthquake.

What are `baseMaps` and `overlayMaps` for?
Why not just add the layers directly to the map?

If we add the GeoJSON layers directly to the map, we can't use a layer control with them.

baseMaps and overlayMaps

After creating our GeoJSON layer, we create a `baseMaps` layer and an `overlayMaps` layer.

This matches what we did in the previous activity, except here we use `earthquakes` instead of `cities` for our overlay.

Questions?