Data Visualization with Leaflet

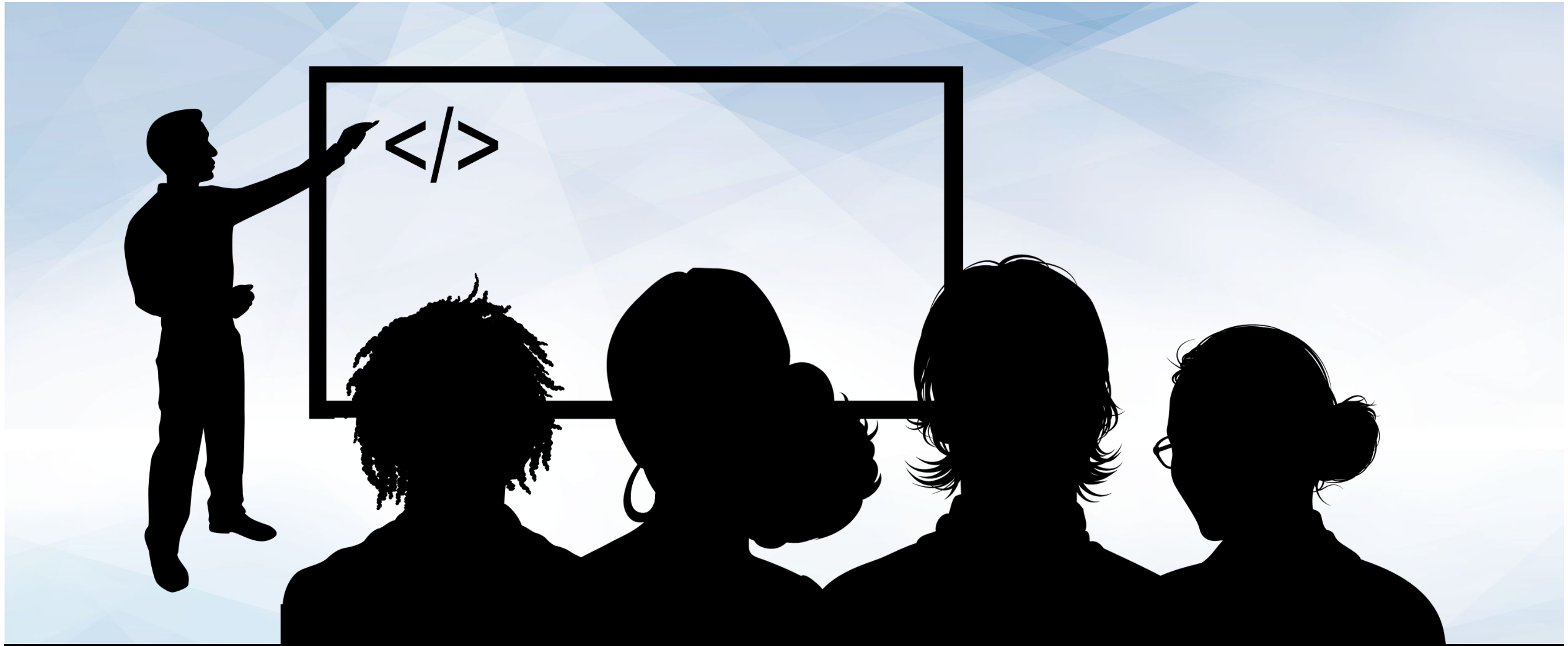
Data Boot Camp

Lesson 15.1

Class Objectives

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

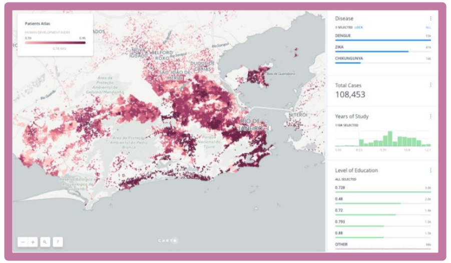
- Discuss the benefits that visualizing data with maps can provide.
- Create maps and plot data with the Leaflet.js library.
- Parse data from GeoJSON format in to create map-based data visualizations.
- Describe the concept of layers and layer controls and they can be applied to add interactivity.

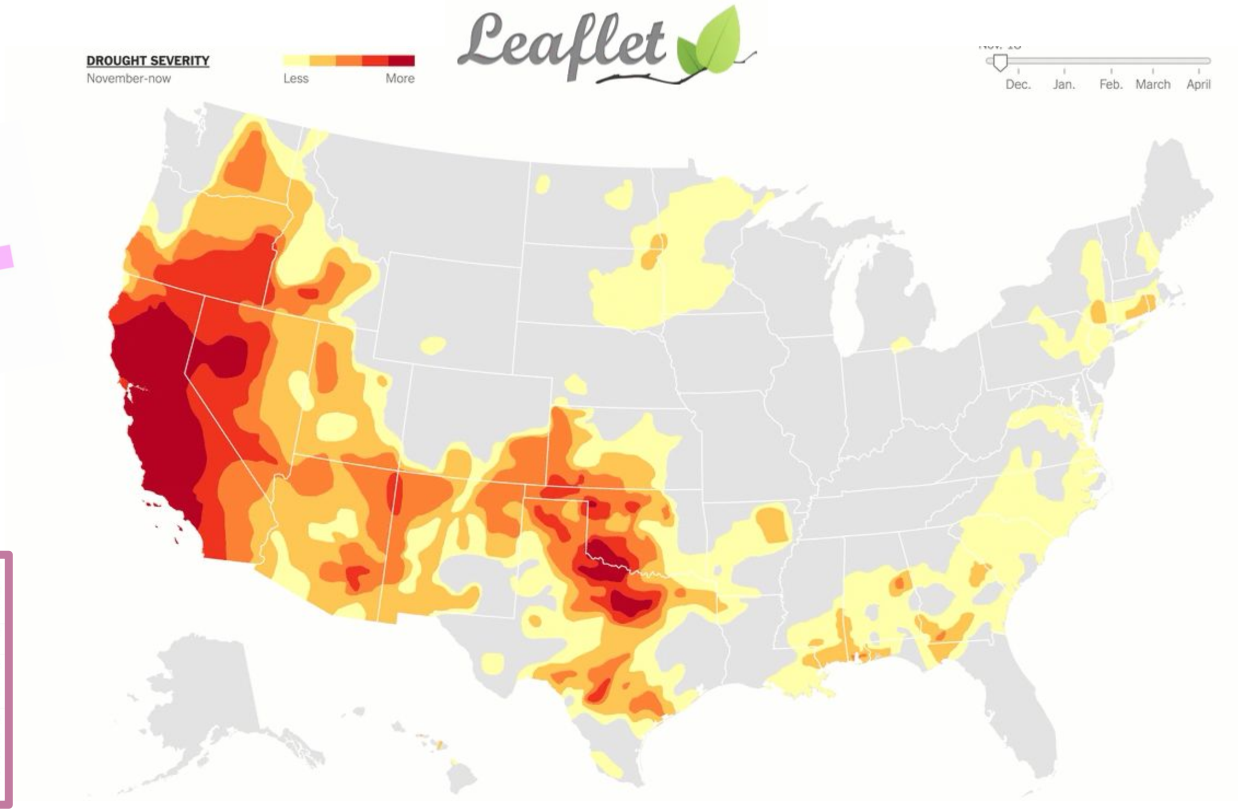


Instructor Demonstration

Welcome Students and Introduce This Week’s Topic

Welcome to Leaflet!



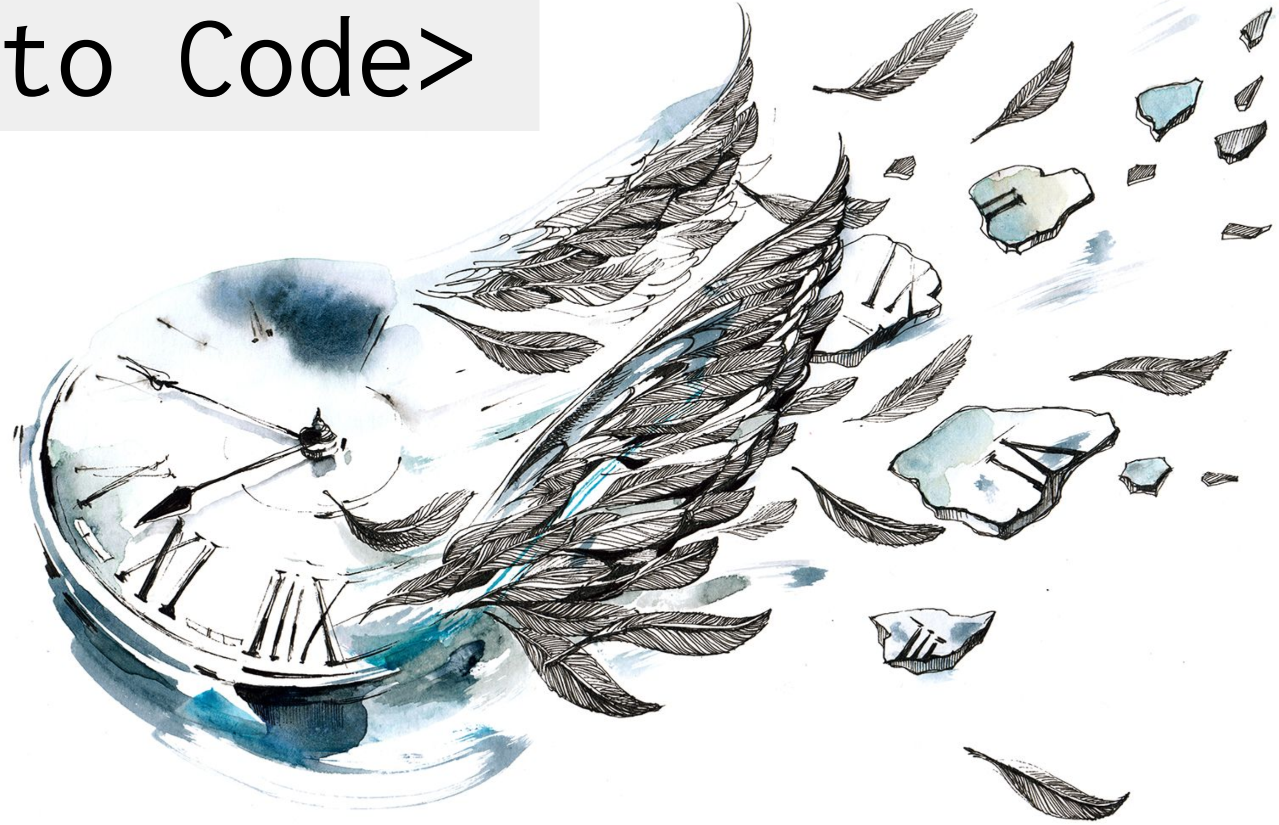


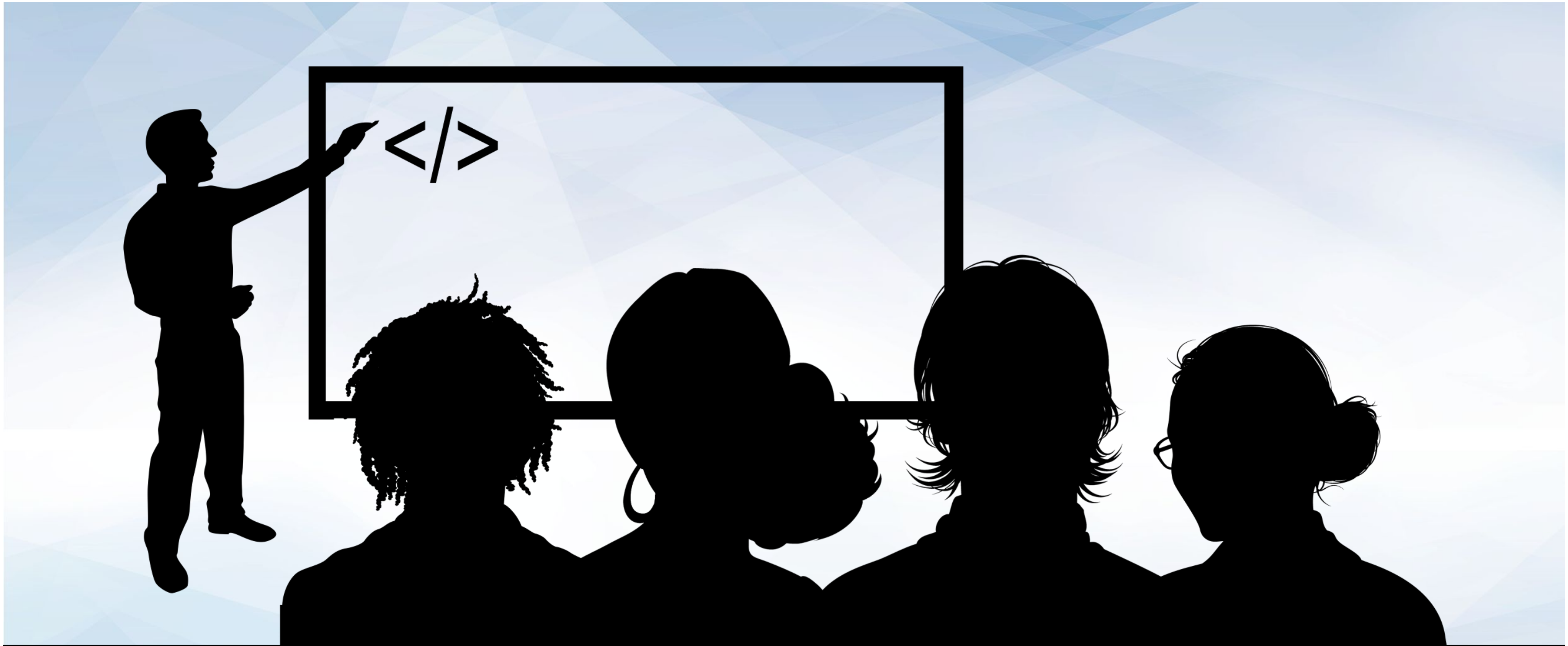
Everyone Do: Introduce Leaflet and Create Our First Map

In this activity, everyone will code along as we get introduce to Leaflet.

Suggested Time:
10 Minutes

<Time to Code>





Instructor Demonstration

Add Markers to the Map

Adding a new marker by adding a new marker object

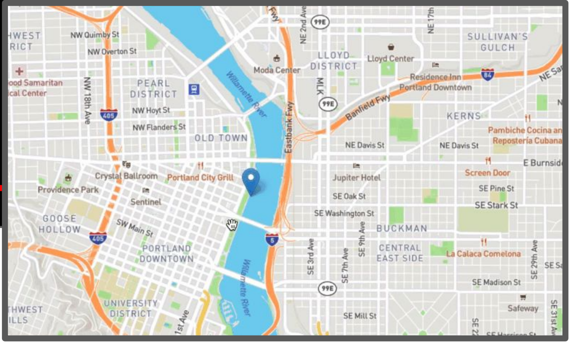
Instructor Do: Add Markers to the Map

logic.js

```
var 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);

// Create a new marker
// Pass in some initial options, and then add it to the map using the addTo method
var 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!");
```

Method used to ad each map layer

Method used to ad text to the marker when clicked



Activity: Complete a Quick Labeling Exercise

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

Suggested Time:
15 Minutes

Instructions:

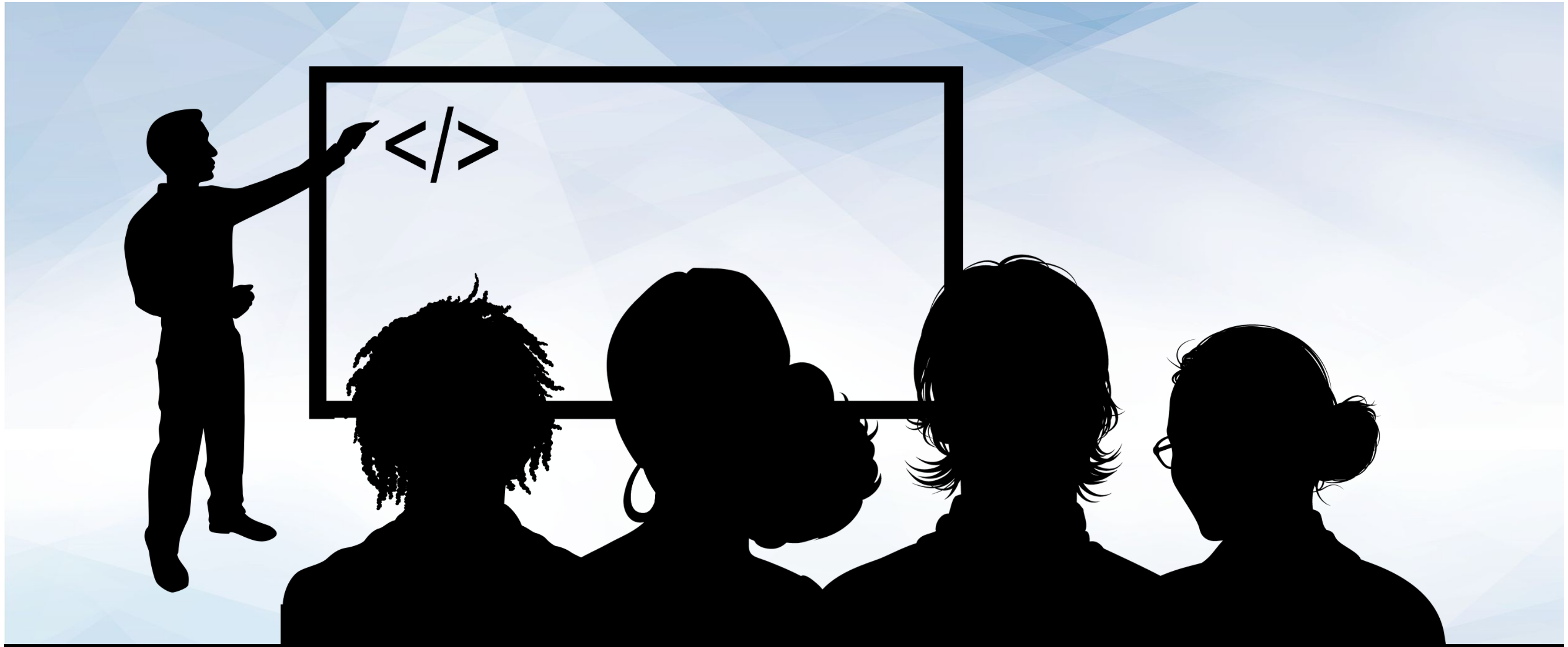
Activity: Complete a Quick Labeling Exercise

- Find the latitude and longitude for the following US cities:
  - New York
  - Los Angeles
  - Houston
  - Omaha
  - Chicago

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

- **Hint:**
  - Don’t forget to add the marker to your map after creating.

Let's Review



Instructor Demonstration

Add Other Types of Markers

Instructor Do: Add Other Types of Markers

SVG shapes to use as markers by adding these blocks of code into the .js file

```
L.marker([45.52, -122.67]).addTo(myMap); > Creates a new marker
```

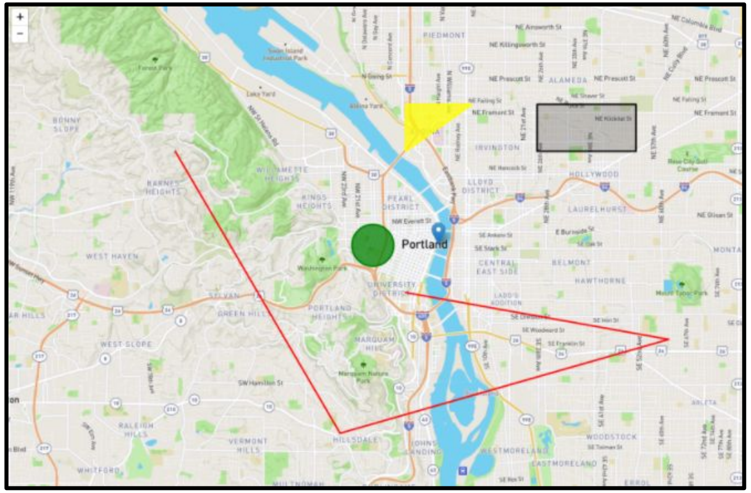
```
L.circle([45.52, -122.69], {
  color: "green",
  fillColor: "green",     > Creates a circle and pass in some initial options
  fillOpacity: 0.75,
  radius: 500
}).addTo(myMap);
```

```
L.polygon([
  [45.54, -122.68],
  [45.55, -122.68],
  [45.55, -122.66]       > Create a Polygon and pass in some initial options
], {
  color: "yellow",
  fillColor: "yellow",
  fillOpacity: 0.75
}).addTo(myMap);
```

```
var line = [    > Coordinates for each point to be used in the polyline
  [45.51, -122.68],
  [45.50, -122.60],
  [45.48, -122.70],
  [45.54, -122.
];
```

```
> Create a polyline using the line coordinates
and pass in some initial options

L.polyline(line, {
 color: "red"
}).addTo(myMap);
```

```
> Create a rectangle and pass in some initial options
L.rectangle([
 [45.55, -122.64],
 [45.54, -          122.61]
], {
 color: "black",
 weight: 3,
 stroke: true
}).addTo(myMap);
```



Activity: Add Other Type of Markers

In this activity, you will create three vector shapes to use as markers.

Suggested Time:
10 Minutes

Instructions:

Activity: Add Other Types of Markers

- Using the files in the `unsolved` folder as a starting point, create the following vector layers, and then add them to the map:
    - A red circle over the city of Dallas.
    - A line connecting NYC to Toronto.
    - A polygon that covers the area inside Atlanta, Savannah, Jacksonville, and Montgomery.

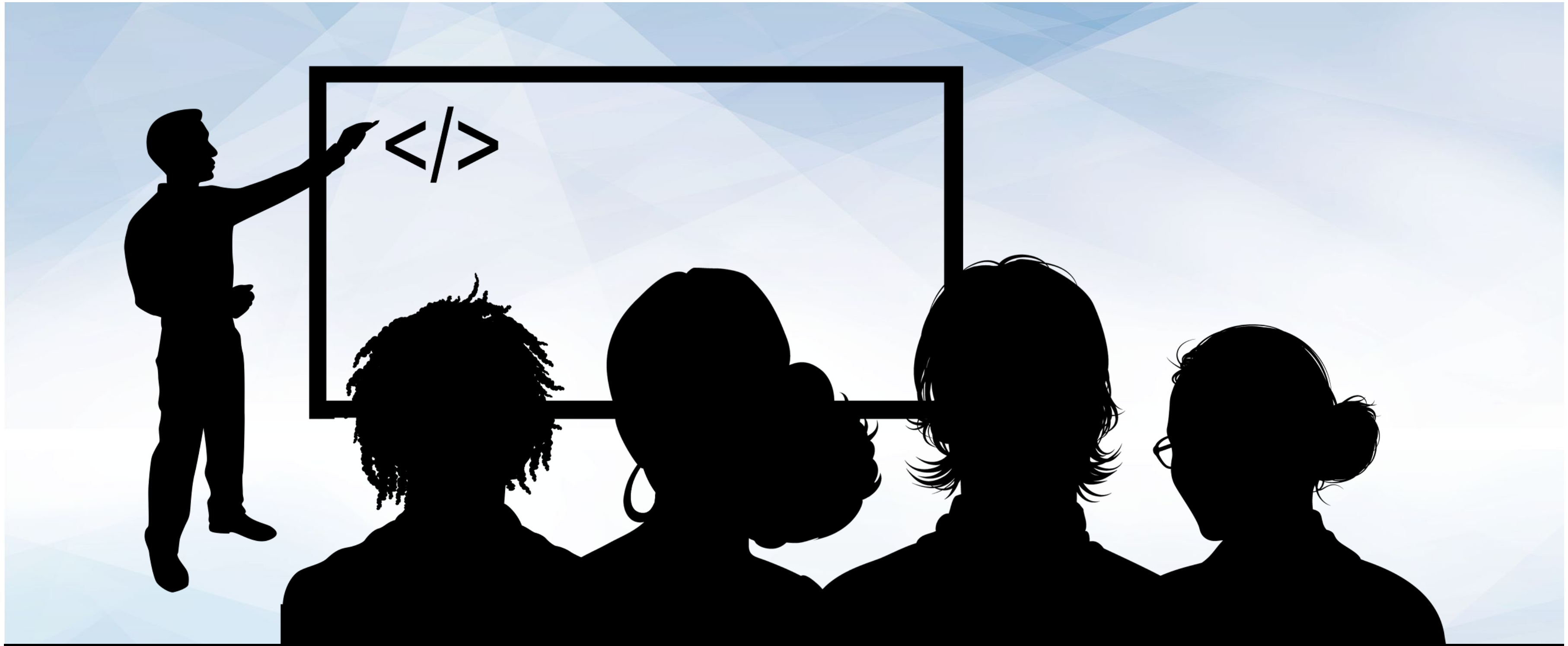
- **Hint:**
    - Note that the logic.js file contains some starter code.
    - Use the Vector Layers section of the Leaflet documentation for reference.

Let's Review

Countdown timer

15:00

(with alarm)

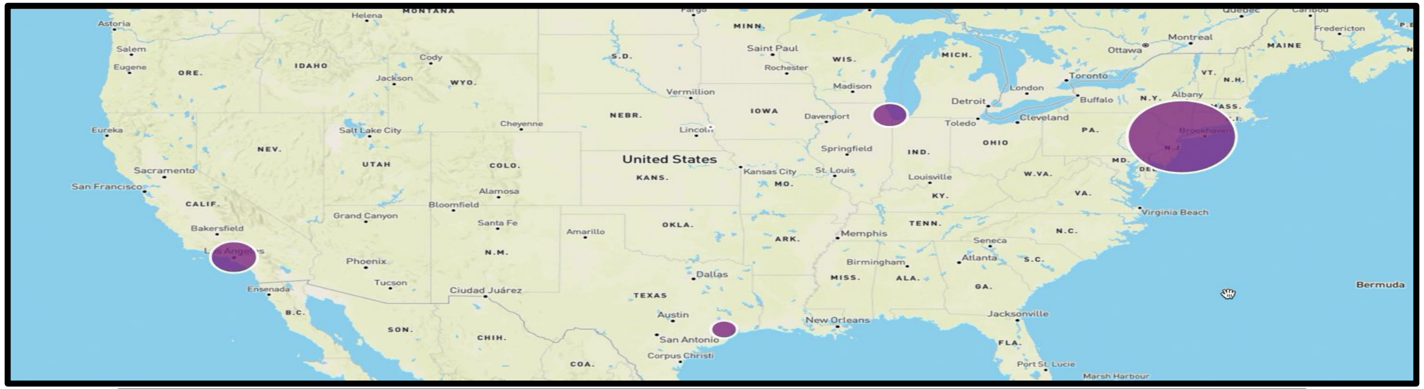


Instructor Demonstration

Visualize City Population

Controlling the size of the circle vector by the population size

Instructor Do: Visualize City Populations



logic.js

We run the `markerSize` function we defined above to calculate each city's circle radius using its population

```
// Loop through the cities array and create one marker for each city object
for (var i = 0; i < cities.length; i++) {
 L.circle(cities[i].location, {
   fillOpacity: 0.75,
   color: "white",
   fillColor: "purple",
   // Setting our circle's radius equal to the output of our markerSize function
   // This will make our maker's size proportionate to its population
   radius: markerSize(cities[i].population)
 }).bindPopup("<h1>" + cities[i].name + "</h1> <hr> <h3>Population: " + cities[i].population +
"</h3>").addTo(myMap);
}
```

Activity: Visualize the Word Cup

In this activity, you will create graduated circle maps to represent the total amount of all-time 3 point wins for the top ten countries in the FIFA World Cup.

Suggested Time:
15 Minutes

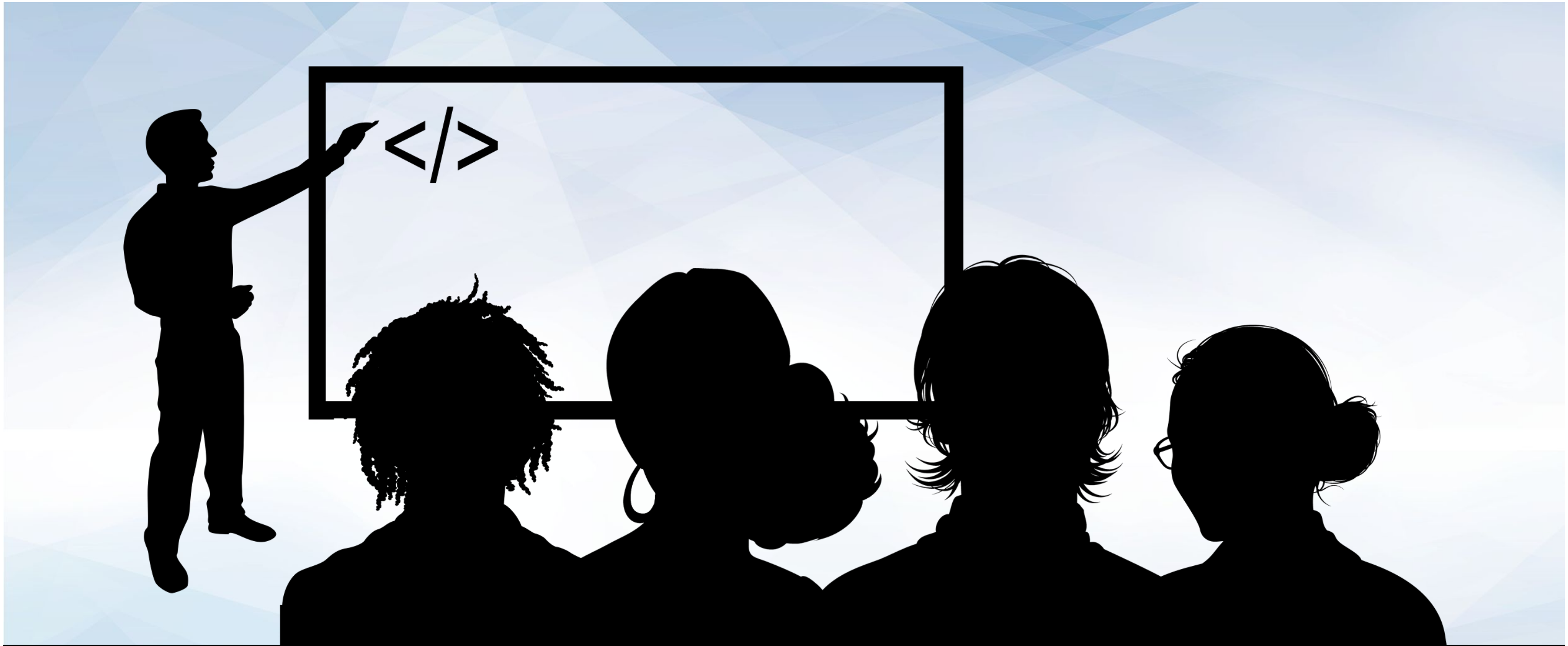
Instructions:

Activity: Visualize the World Cup

1. Add your code to logic.js to render the following:
   - A circle for each country in the data set.
   - A radius size determined by the country's all-time three-point-wins.
   - A color for each circle that you determine as follows:
     - For countries with more than 200 points, set the color of the circle to yellow.
     - For countries with more than 100 points, set the color of the circle to blue.
     - For countries with more than 90 points, set the color of the circle to green.
     - Render the remaining country circles in red.
2. Make sure that each vector layer you include has a pop-up with the country's name and points.

- **Hint:**
  - Universally adjust the radius for better visuals.
  - Refer to the Leaflet documentation for Path options if you get stuck creating vector layers.

Let's Review

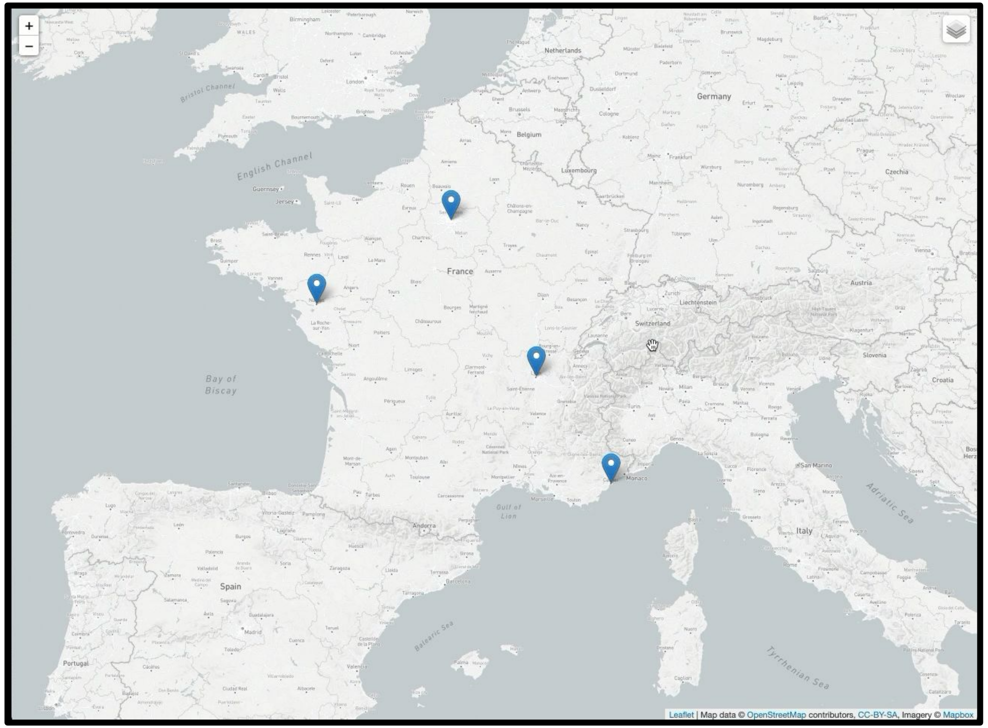


Instructor Demonstration

Demonstrate Layer Groups and Layer Controls

Using multiple layers within the same map.

Instructor Do: Demonstrate Layer Group and Layer Controls



Leaflet has two type of layers:

- ➔ **Base Layers**: These are mutually exclusive to each other, which only one can be visible at a time.
- ➔ **Overlays:** These go over the base layers and can be turned on and off.

Layer Groups:

- ➔ Use LayerGroup class in case you have a bunch of layers you want to combine into a group to handle as one in your code.

Activity: Perform a Layer Activity

In this activity, you will expand on our previously created US cities maps by adding an overlay to represent the state populations. This layer will appear on the map as white circle vectors.

Suggested Time:
15 Minutes

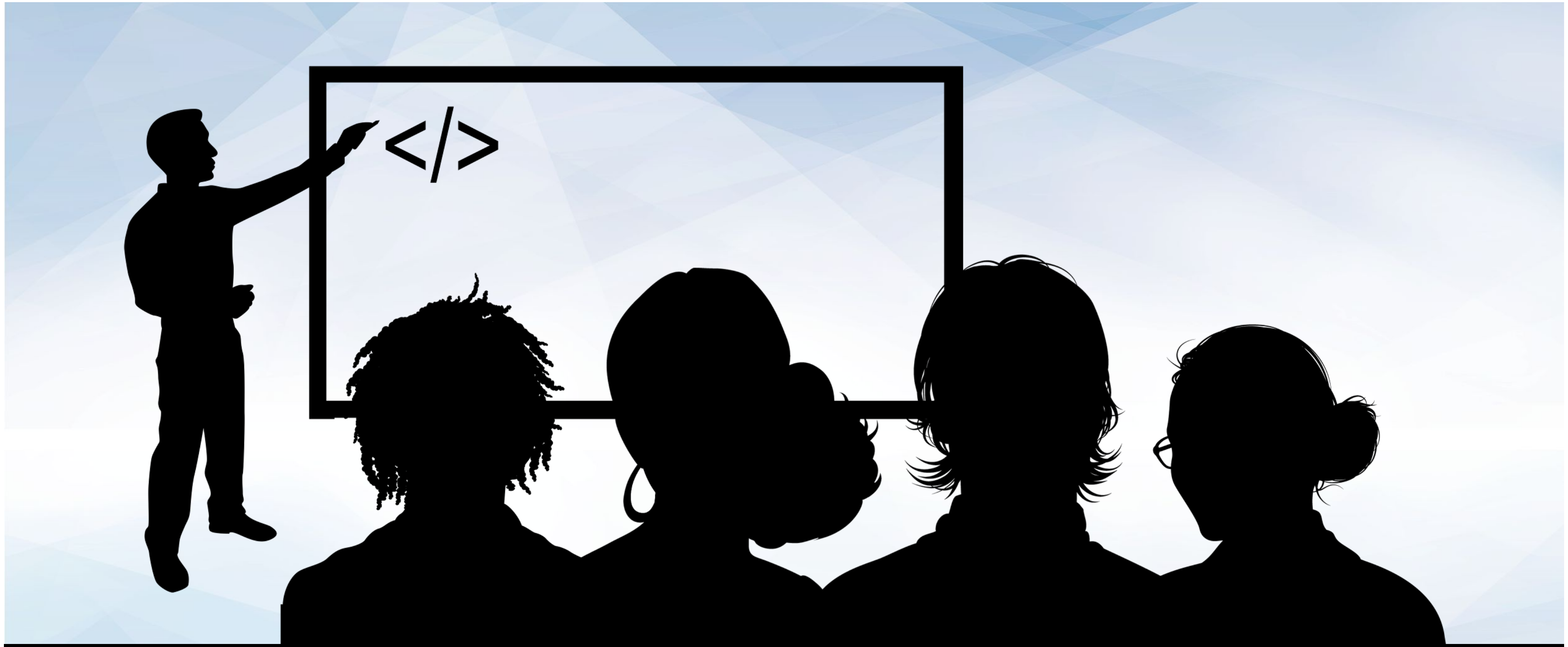
Instructions:

Activity: Perform a Layer Activity

1. Open the logic.js file in the `Unsolved` folder.
2. Add logic to the file as follows:
    - Create layers groups for city markers and a separate layer group for state markers. Note that the cityMarkers and stateMakerkers arrays contains all the markers, which have been created for you. Store these layer groups in variable 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 `layer` 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:**
    - If you get stuck, refer to the Leaflet Layer Groups and Layers Control tutorial.
    - If you succeed, you should be able to toggle between the Street Map and Topographic Map base layers and to turn the State Population and City Population overlays on and off.

Let's Review



Instructor Demonstration

What is GeoJSON?

What is GeoJSON?

GeoJSON

GeoJSON is a geospatial data interchange format based on JavaScript Object Notation (JSON). It defines several types of JSON objects and the manner in which they are combined to represent data about geographic features, their properties, and their spatial extents.

Activity: Perform a GeoJSON Activity

In this activity, you will plot markers on a map to represent occurrences of earthquakes. To achieve this, we'll work with GeoJSON data from the U.S. Geological Survey (USGS).

Suggested Time:
15 Minutes

Instructions:

Activity: Perform a GeoJSON Activity

1. Open the logic.js file.
2. 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.
3. 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.
4. 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 streetmap tile layer and the darkmap 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

Let's Review