Data Visualization with Leaflet

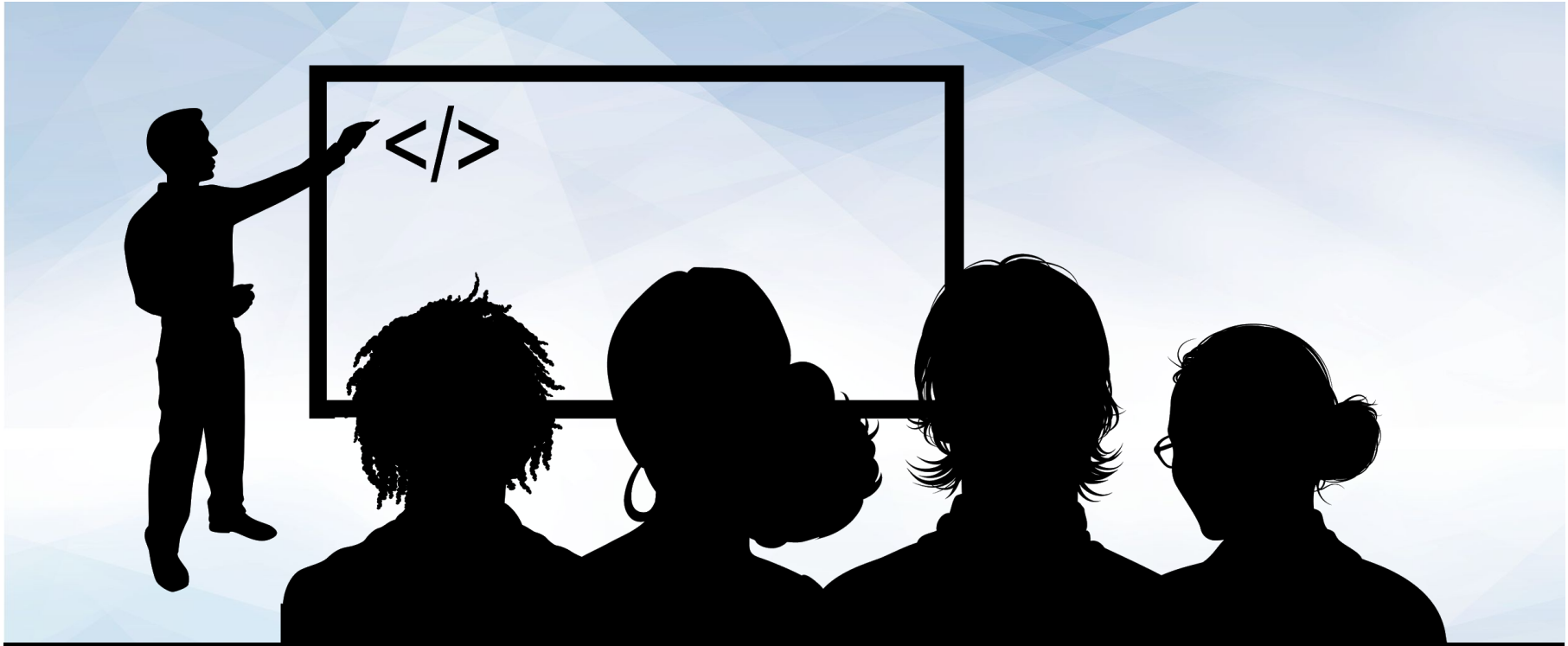
Data Boot Camp

Lesson 17.1

Class Objectives

By the end of today's class you will be able to:

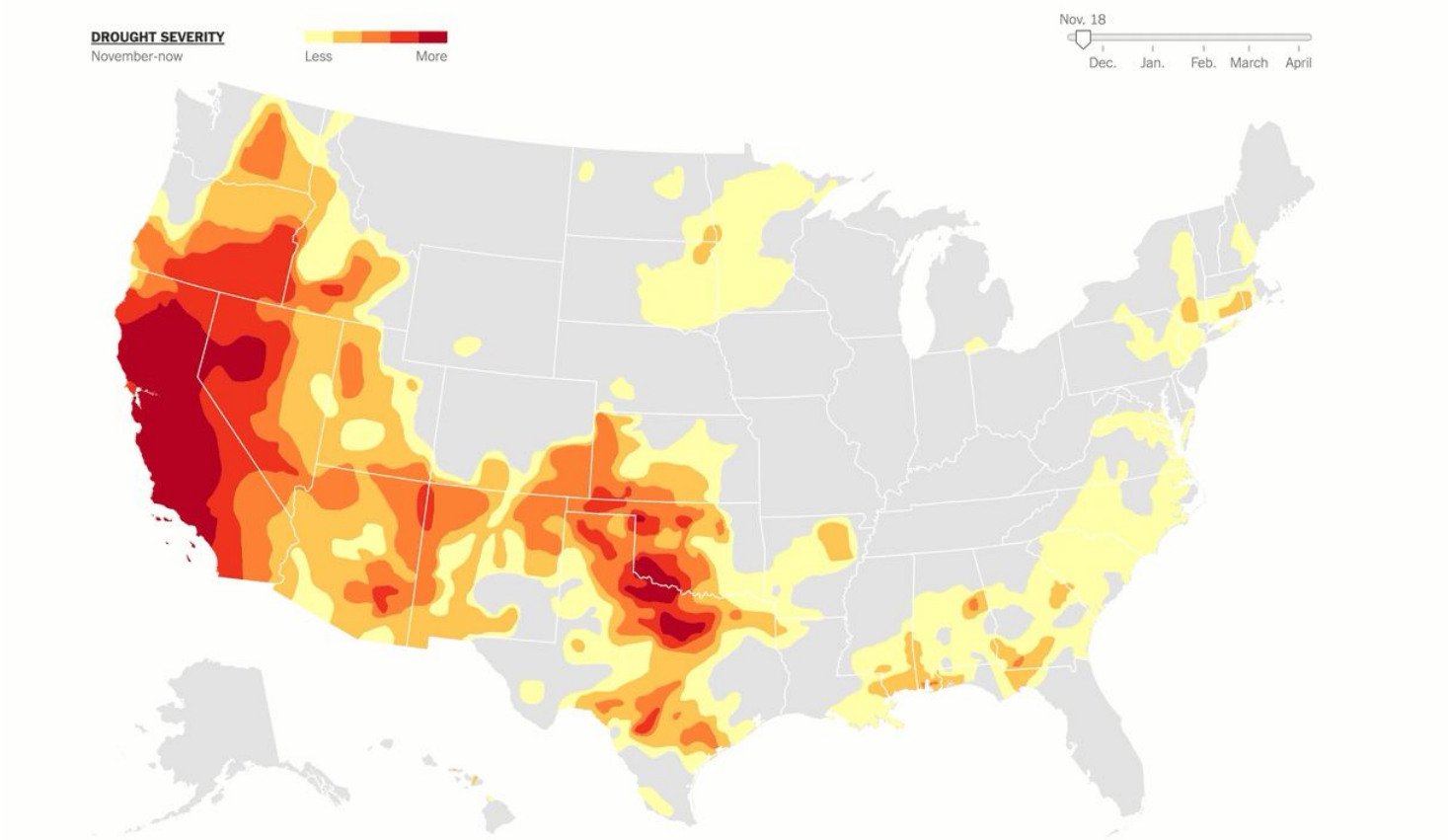
- Understand the benefits that visualizing data with maps can provide.
- Learn the basics of creating maps and plotting data with Leaflet.js library.
- Gain an understand of the GeoJSON format.
- Understand the concept of layers controls and how we can use them to add interactivity to our maps.



Instructor Demonstration

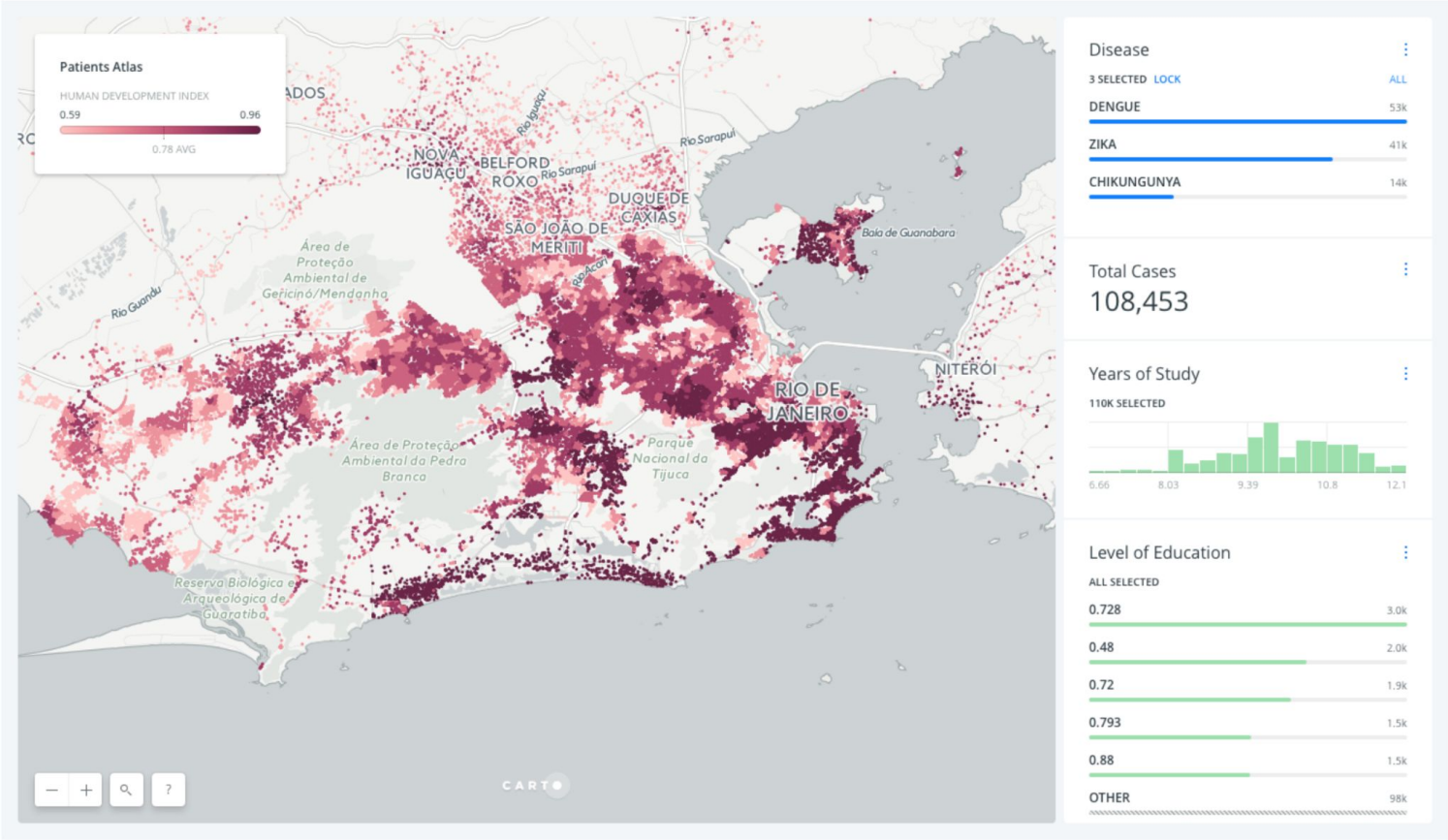
Chloro-pleth-o-rama!

Maps as a visualization tool

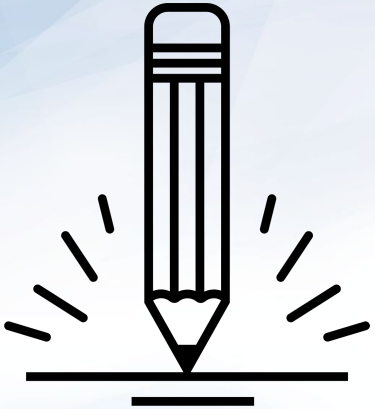


https://www.nytimes.com/interactive/2014/upshot/mapping-the-spread-of-drought-across-the-us.html?_r=0

Maps as a visualization tool



https://carto.com/blog/understand-and-predict-zika-in-brazil/



Everyone Do: Introduce Leaflet & Our first Map

In this activity, everyone will be introduced to Leaflet and Mapbox to create our very first map.

Suggested Time:
20 Minutes

Everyone Do: Introduce Leaflet & Our first Map

Instructions:

- Open the unsolved folder in your text editor and lets create four files in the following order:
  - `index.html`
  - `logic.js`
  - `config.js`
  - `style.cs`

Everyone Do: Introduce Leaflet & Our first Map

- `index.html` ➔ First, we need to create a HTML file:

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Basic Map</title>

  <!-- Leaflet CSS -->
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css"
    integrity="sha512-xwE/Az9zrjBIphAcBb3F6JVqxf46+CDLwfLMHloNu6KEQCAWi6HcDUbeOfBIptF7tcCzusKFjFw2yuvEpDL9wQ==" 
    crossorigin="" />

  <!-- Our CSS -->
  <link rel="stylesheet" type="text/css" href="style.css">
</head>

<body>
  <!-- The div where we will inject our map -->
  <div id="map"></div>

  <!-- Leaflet JS -->
  <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"
    integrity="sha512-gZwIG9x3wUXg2hdXF6+rVkLF/0Vi9U8D2Ntg4Ga5I5BZpVkVxlJWbSQtXPSiUTtC0TjtG0mxa1AJPuV0CPthew==" 
    crossorigin=""></script>
  <!-- API key -->
  <script type="text/javascript" src="config.js"></script>
  <!-- JS -->
  <script type="text/javascript" src="logic.js"></script>
</body>

</html>
```

➔ Add links to Leaflet CSS and JS libraries.

➔ Link to css file.

➔ Create a `<div>` with an `<id>` to the map.

➔ Link to file with api key..

➔ Link to file with the JS map code

Everyone Do: Introduce Leaflet & Our first Map

- `logic.js` ➔ Secondly, we need to create the first js file:

```
var myMap = L.map("map", {
 center: [45.52, -122.67],
 zoom: 13`
});
```

➔ This first block of code creates a map object, which is defined with `L.map` method. It accepts two arguments: "`map`" and an object where it sets the initial coordinates of the `center` property.

```
L.tileLayer("https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token={accessToken}", {
 attribution: "© <a href='https://www.mapbox.com/about/maps/'>Mapbox</a> © <a
href='http://www.openstreetmap.org/copyright'>OpenStreetMap</a> <strong><a
href='https://www.mapbox.com/map-feedback/' target='_blank'>Improve this map</a></strong>",
 tileSize: 512,
 maxZoom: 18,
 zoomOffset: -1,
 id: "mapbox/streets-v11",
 accessToken: API_KEY
}).addTo(myMap);
```

➔ The second block code it's just the map image show in the background. Go to https://leafletjs.com/examples/quick-start/ and copy and paste the block code.

Everyone Do: Introduce Leaflet & Our first Map

- `config.js` ➔ Next, we need to create a JS file with the API key

```
// API key
const API_KEY = "tl.jkJ1IjoidGxlZTEiLCJhIjoiY2tnZGZnamEzMHZ5NTJ4cW96aDBhbjA0ZyJ9.zGXiyl62TL22xaQmG6hcol"
```

- In order to create this file we need to first create a FREE account with Mapbox and generate a token.
  - Navigate to mapbox.com and create your free account
  - Once your FREE account is created sign in and you will be directed directly to a page where token is created.
  - Copy and paste the token in the variable called 'API_KEY' in a new file named `config.js`

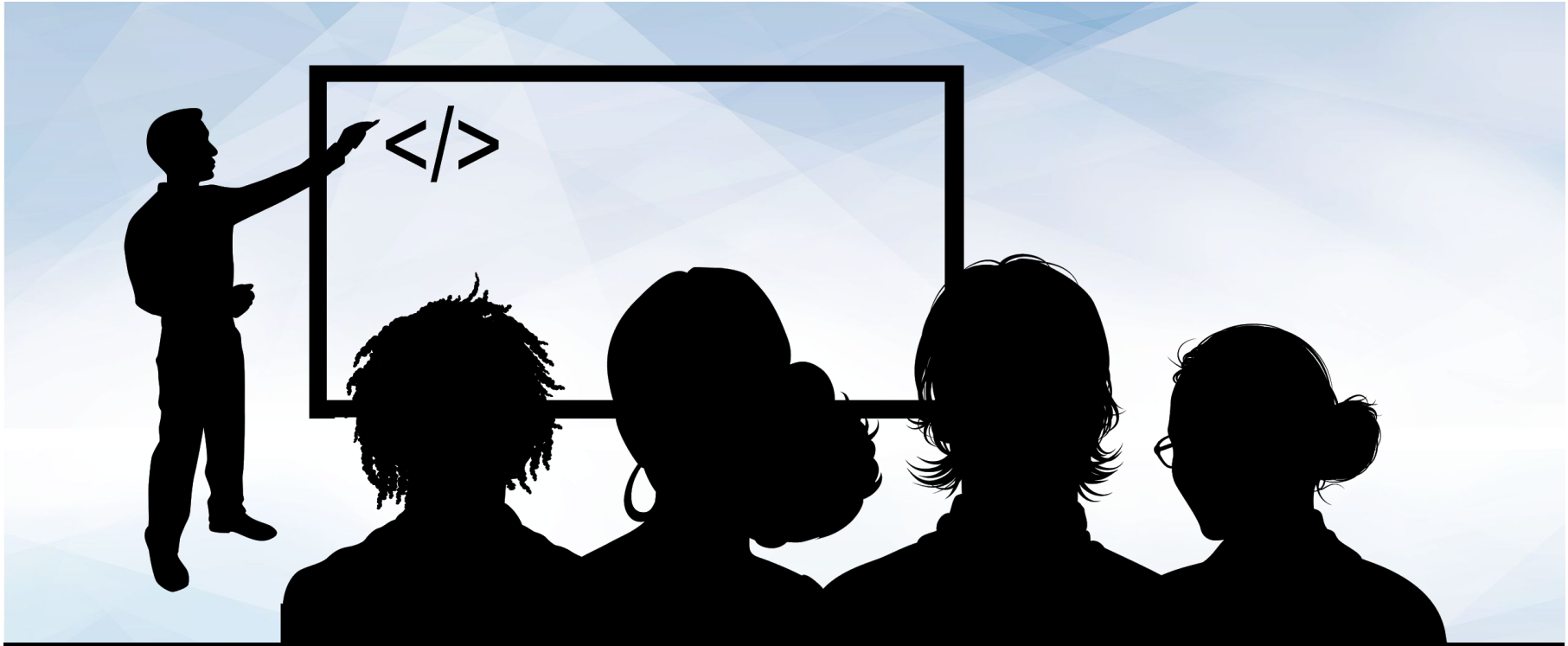
NOTE: Fictitious API KEY

Everyone Do: Introduce Leaflet & Our first Map

- `style.css` ➔ Finally, a CSS file is created to give some styling in our map.

```
/* remove default margin and padding from body */
body {
 padding: 0;
 margin: 0;
}

/* set map, body, and html to 100% of the screen size */
#map,
body,
html {
 height: 100%;
}
```



Instructor Demonstration

Add Markers To The Map

Add Makers To The Map

- logic.js

```
var myMap = L.map("map", {
 center: [45.52, -122.67],
 zoom: 13`
});

L.tileLayer("https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token={accessToken}", {
 attribution: "© <a href='https://www.mapbox.com/about/maps/'>Mapbox</a> © <a href='http://www.openstreetmap.org/copyright'>OpenStreetMap</a> <strong><a
href='https://www.mapbox.com/map-feedback/' target='_blank'>Improve this map</a></strong>",
 tileSize: 512,
 maxZoom: 18,
 zoomOffset: -1,
 id: "mapbox/streets-v11",
 accessToken: API_KEY
}).addTo(myMap);
```

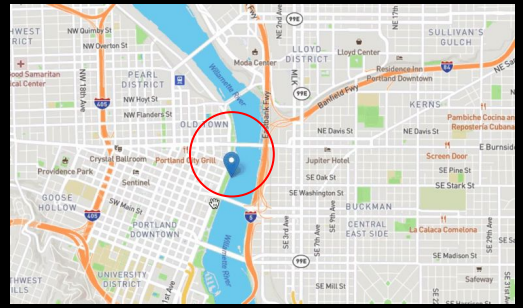
➔ Code added to logic.js to add markers to the map. ↓

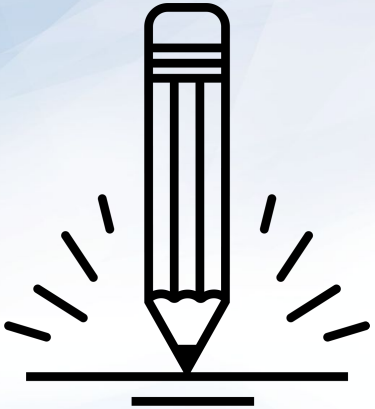
```
// Create a new marker
// Pass in some initial options, and then add it to the map using the addTo method
var marker = L.marker([45.52, -122.67], {
 draggable: true,
 title: "My First Marker"
}).addTo(myMap);

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

← Method used to add each map layer.

← Method used to add text to the marker when clicked.





Activity: Quick Labeling Exercise

In this activity, you will be plotting various US cities using Leaflet.

Suggested Time:
15 Minutes

Activity: Quick Labeling Exercise

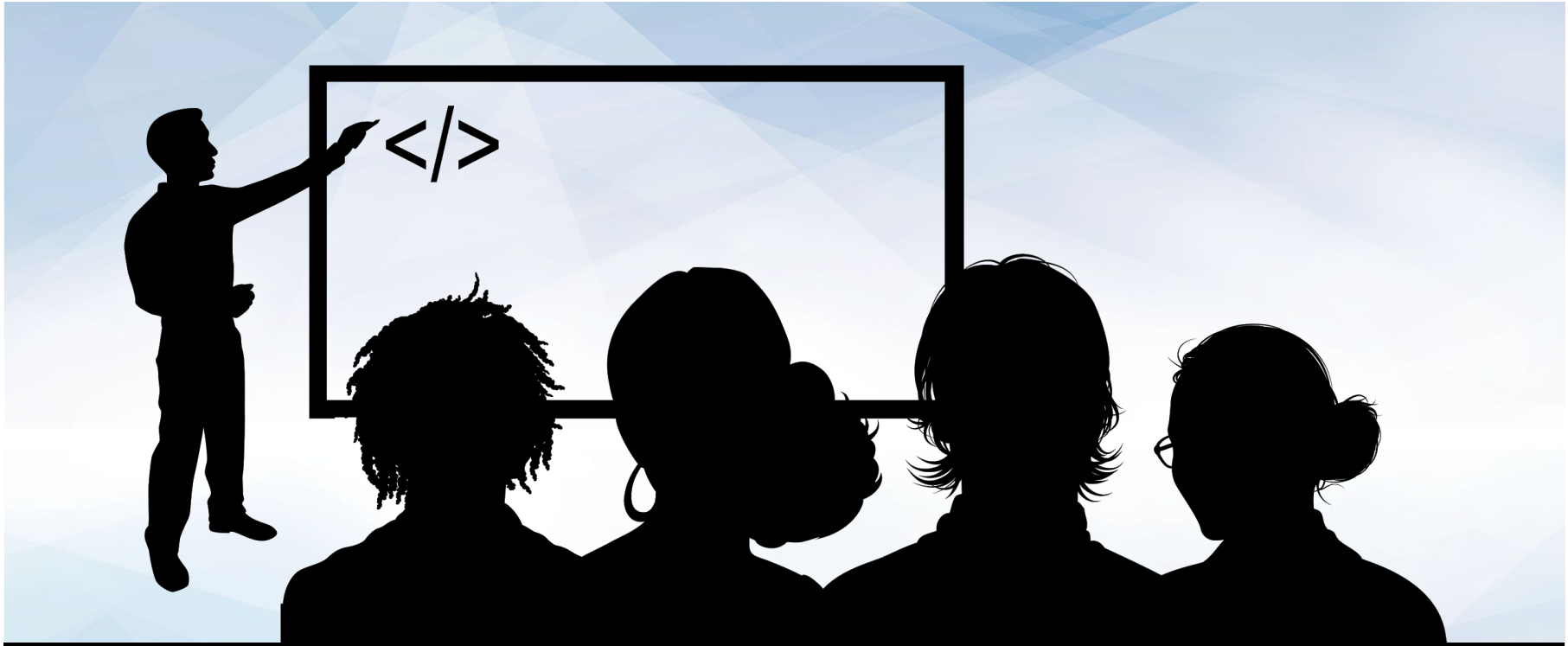
Instructions:

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

- Create a marker for each city with a pop-up that display that city's name and population. For this activity, you may either look up the population for each city, or make one up.
- **Bonus**:
  - Pop-ups take in a string of HTML. If you finish early, try experimenting with passing in different tags or custom CSS!

- **Hints**:
  - Don't forget add the marker to your map after creating it!

Time’s Up! Let’s Review.



Instructor Demonstration

Other Types of Markers

Other Types of Markers

➔ Leaflet allows SVG shapes to be used as markers by adding these blocks of code in the .js file.

- logic.js

```
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);
```

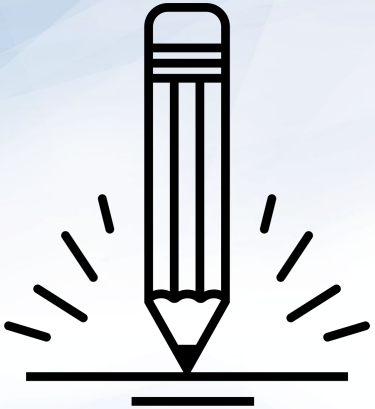
```
L.polygon([
 [45.54, -122.68],
 [45.55, -122.68],
 [45.55, -122.66]       > Create a Polygon and pass in some initial options
], {
 color: "yellow",
 fillColor: "yellow",
 fillOpacity: 0.75
}).addTo(myMap);
```

```
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: Other Types of Markers

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

Suggested Time:
10 Minutes

Activity: Quick Labeling Exercise

Instructions:

- Using the files in the `Unsolved` folder as a starting point, create the following vector layers and 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.

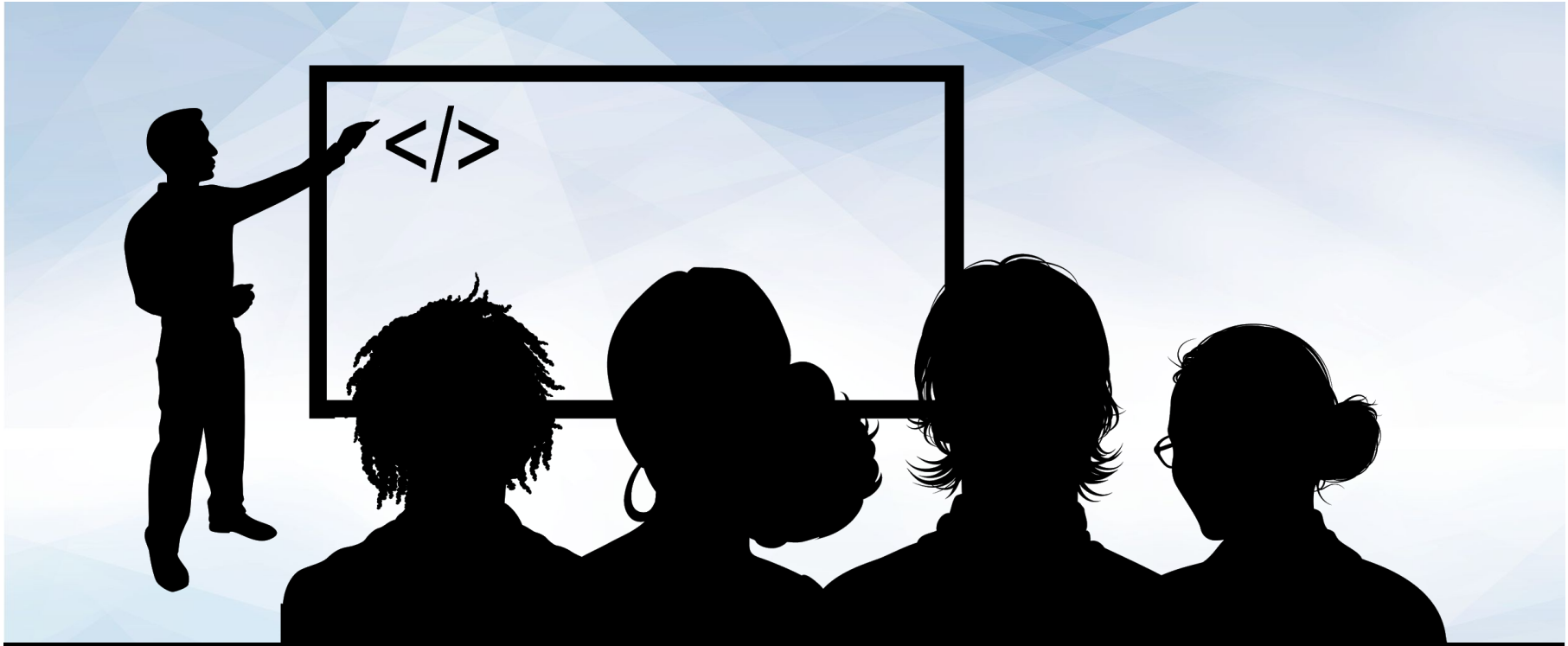
- **Hints**:
  - The logic.js file contains some starter code.
  - Use the ‘Vector Layers’ section of the Leaflet documentation for reference.

Time’s Up! Let’s Review.

Countdown timer

15:00

(with alarm)



Instructor Demonstration

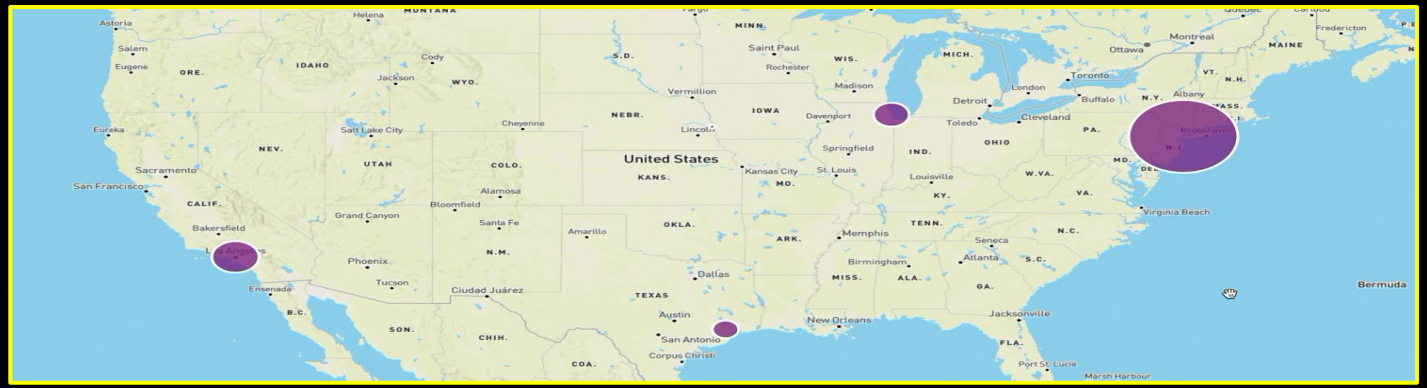
City Population Visualized

City Population Visualized

➔ We can control the size of a circle vector layer by the population size of each city.

- logic.js

```
// 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) ➔ Run the markerSize function we defined above to calculation each city's circle radius using it's population.
 }).bindPopup("<h1>" + cities[i].name + "</h1> <hr> <h3>Population: " + cities[i].population +
"</h3>").addTo(myMap);
}
```



Activity: World Cup Visualized

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

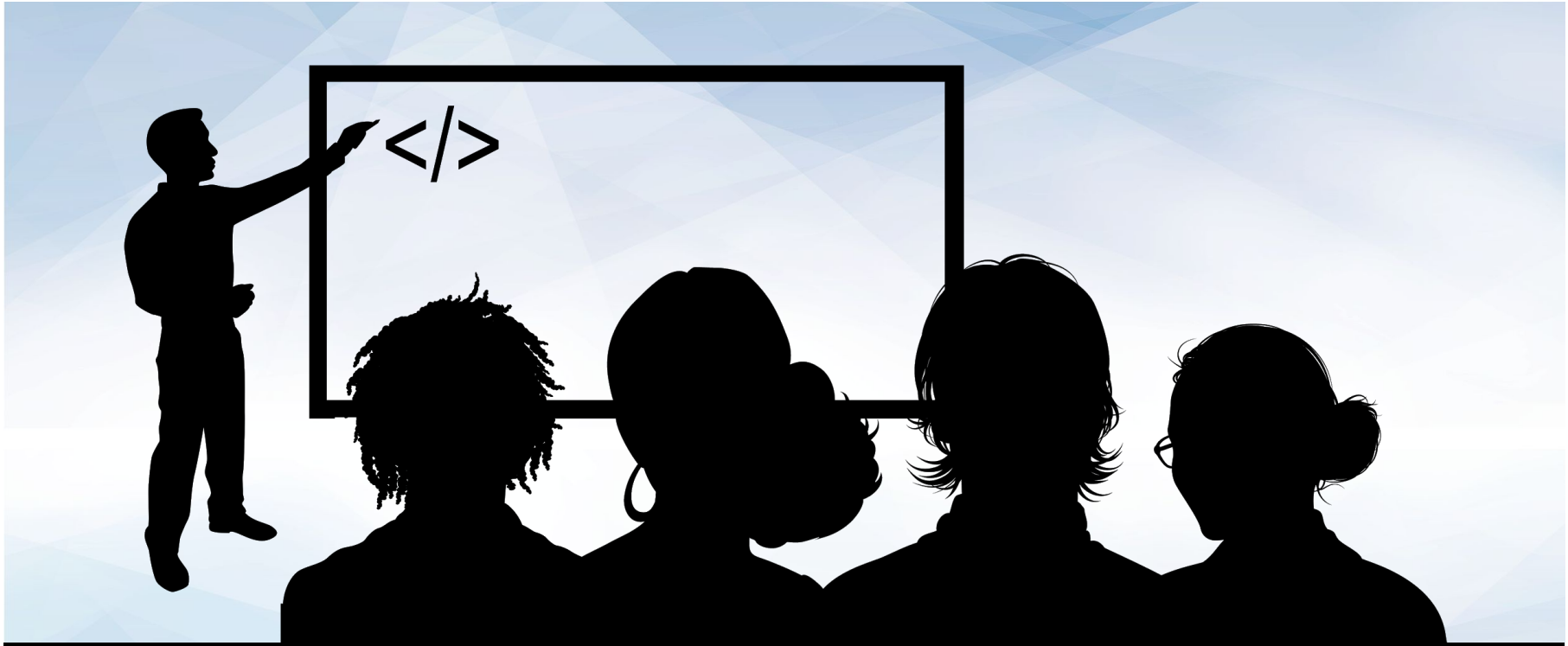
Activity: World Cup Visualized

Instructions:

- 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 3 point wins including World Cup 2018.
  - For countries with over 200 points, set the color of the circle to blue.
  - For countries with between 199 and 150 points, set the color of the circle to green.
  - For countries with with between 149 and 100 points, set the color of the circle to yellow.
  - Render the remaining country circles in red.
- Make sure that each vector layer you include has a pop-up with the country's name and points.

- **Hints**:
  - The radius will need to be adjusted universally for better visuals.
  - Refer to the Leaflet docs for Path Options if stuck creating vectors layers.

Time’s Up! Let’s Review.



Instructor Demonstration

Layers Groups & Layer Controls

Layer Groups & Layer Controls

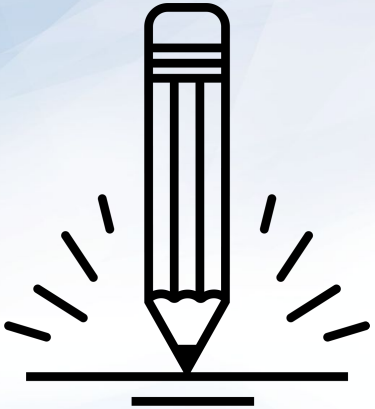
➔ Using Multiple layers within the same map.

- There are two types 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.

```
// An array which will be used to store created cityMarkers
var cityMarkers = [];

for (var i = 0; i < cities.length; i++) {
  // loop through the cities array, create a new marker, push it to the cityMarkers array
  cityMarkers.push(
    L.marker(cities[i].location).bindPopup("<h1>" + cities[i].name + "</h1>")
  );
}

// Add all the cityMarkers to a new layer group.
// Now we can handle them as one group instead of referencing each individually
var cityLayer = L.layerGroup(cityMarkers);
```



Activity: Layer Activity

In this activity, you will return to our US cities map and refactor the code to use layer groups and a layer control to be able to represent the population for the entire state as well as the city.

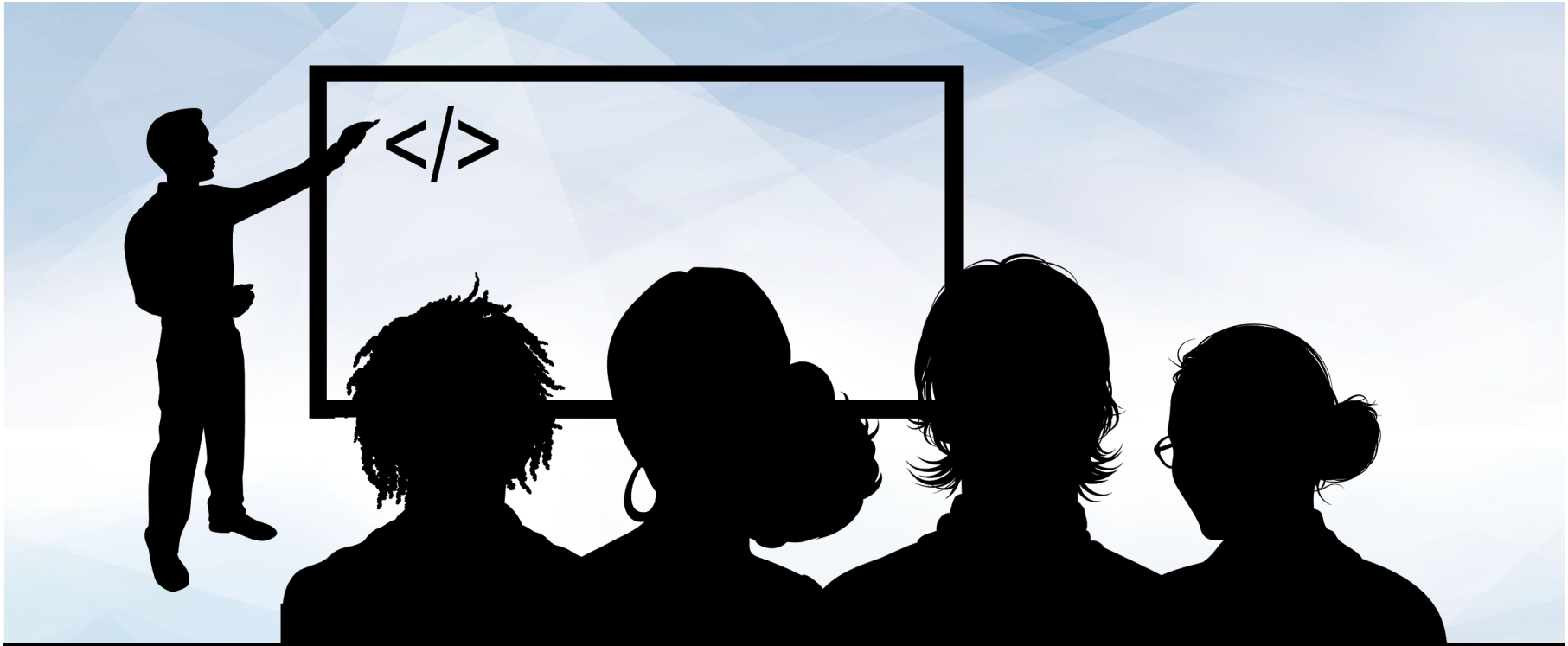
Suggested Time:
15 Minutes

Activity: Layer Activity

Instructions:

- Open the logic.js file inside of the Unsolved folder.
- Add logic to this file to accomplish the following:
  - Create a layer group for city markers and a separate layer group for state markers. All of the markers have been created for you already and stored in the `cityMarkers` and stateMarkers arrays. Store these layer groups in variables named cities and states.
  - Create a baseMaps object to contain the `streetmap` and `darkmap` tiles, which have been already defined.
  - Create an `overlayMaps` object to contain "State Population" and "City Population" layers.
  - Add a layers key to the options object inside of the L.map method and set its value to an array containing our `streetmap`, states, and cities layers. These will determine which layers are displayed when the map first loads.
  - Finally, create a layer control and pass in the `baseMaps` and `overlayMaps` objects. Add the layer control to the map.
- **Hints**:
  - If you get stuck refer to the Leaflet Layers Control Docs.
  - If successful, you should be able to toggle between Street Map and Dark Map base layers, as well as turn State Population and City Population overlay layers on and off.

Time’s Up! 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.

What is GeoJSON?

➔ Check your slack for the following link:

http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_hour.geojson


```
{"type":"FeatureCollection","metadata":{"generated":1603337170000,"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":7},"features":[{"type":"Feature","properties":{"mag":1.29,"place":"13km SW of Searles Valley,
CA","time":1603335918400,"updated":1603336147381,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/ci39440911","detail":"https://earthquake.usgs.
gov/earthquakes/feed/v1.0/detail/ci39440911.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"automatic","tsunami":0,"sig":26,"net":"ci","code":"3
9440911","ids":",ci39440911,","sources":",ci,","types":",nearby-cities,origin,phase-data,scitech-
link,","nst":19,"dmin":0.1353,"rms":0.17,"gap":140,"magType":"ml","type":"earthquake","title":"M 1.3 - 13km SW of Searles Valley, CA"},"geometry":
{"type":"Point","coordinates":[-117.5178333,35.6966667,6.65]},"id":"ci39440911"},
{"type":"Feature","properties":{"mag":5.1,"place":"50 km WNW of Jiangyou,
China","time":1603335819083,"updated":1603336468040,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/us6000cb4i","detail":"https://earthquake.us
gs.gov/earthquakes/feed/v1.0/detail/us6000cb4i.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"reviewed","tsunami":0,"sig":400,"net":"us","code"
:"6000cb4i","ids":",us6000cb4i,","sources":",us,","types":",origin,phase-
data,","nst":null,"dmin":11.379,"rms":0.57,"gap":41,"magType":"mb","type":"earthquake","title":"M 5.1 - 50 km WNW of Jiangyou, China"},"geometry":
{"type":"Point","coordinates":[104.2181,31.9295,16.96]},"id":"us6000cb4i"},
{"type":"Feature","properties":{"mag":1.12,"place":"15km S of Trona,
CA","time":1603334693410,"updated":1603335588520,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/ci39440895","detail":"https://earthquake.usgs.
gov/earthquakes/feed/v1.0/detail/ci39440895.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"reviewed","tsunami":0,"sig":19,"net":"ci","code":"39
440895","ids":",ci39440895,","sources":",ci,","types":",focal-mechanism,nearby-cities,origin,phase-data,scitech-
link,","nst":15,"dmin":0.1147,"rms":0.15,"gap":131,"magType":"ml","type":"earthquake","title":"M 1.1 - 15km S of Trona, CA"},"geometry":
{"type":"Point","coordinates":[-117.406,35.6348333,10.15]},"id":"ci39440895"},
{"type":"Feature","properties":{"mag":2,"place":"15km W of Ludlow,
CA","time":1603334429420,"updated":1603335569542,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/ci39440887","detail":"https://earthquake.usgs.
gov/earthquakes/feed/v1.0/detail/ci39440887.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"reviewed","tsunami":0,"sig":62,"net":"ci","code":"39
440887","ids":",ci39440887,","sources":",ci,","types":",nearby-cities,origin,phase-data,scitech-
link,","nst":19,"dmin":0.1323,"rms":0.14,"gap":48,"magType":"ml","type":"earthquake","title":"M 2.0 - 15km W of Ludlow, CA"},"geometry":
{"type":"Point","coordinates":[-116.3166667,34.6976667,3.3]},"id":"ci39440887"},
{"type":"Feature","properties":{"mag":0.3,"place":"30 km SSE of Mina,
Nevada","time":1603333972210,"updated":1603334302902,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/nn00779882","detail":"https://earthquake.u
sgs.gov/earthquakes/feed/v1.0/detail/nn00779882.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"automatic","tsunami":0,"sig":1,"net":"nn","code"
:"00779882","ids":",nn00779882,","sources":",nn,","types":",origin,phase-
data,","nst":10,"dmin":0.011,"rms":0.03,"gap":133.28,"magType":"ml","type":"earthquake","title":"M 0.3 - 30 km SSE of Mina, Nevada"},"geometry":
{"type":"Point","coordinates":[-117.9923,38.1273,10.6]},"id":"nn00779882"},
{"type":"Feature","properties":{"mag":4.7,"place":"Reykjanes
Ridge","time":1603333903888,"updated":1603334600040,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/us6000cb47","detail":"https://earthquake.us
gs.gov/earthquakes/feed/v1.0/detail/us6000cb47.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"reviewed","tsunami":0,"sig":340,"net":"us","code"
:"6000cb47","ids":",us6000cb47,","sources":",us,","types":",origin,phase-
data,","nst":null,"dmin":9.792,"rms":0.98,"gap":119,"magType":"mb","type":"earthquake","title":"M 4.7 - Reykjanes Ridge"},"geometry":
{"type":"Point","coordinates":[-35.4046,53.0278,10]},"id":"us6000cb47"},
{"type":"Feature","properties":{"mag":2,"place":"7 km NW of Fritz Creek,
Alaska","time":1603333651473,"updated":1603334659397,"tz":null,"url":"https://earthquake.usgs.gov/earthquakes/eventpage/ak020dlkfgbw","detail":"https://earthquake
.usgs.gov/earthquakes/feed/v1.0/detail/ak020dlkfgbw.geojson","felt":null,"cdi":null,"mmi":null,"alert":null,"status":"automatic","tsunami":0,"sig":62,"net":"ak","
code":"020dlkfgbw","ids":",ak020dlkfgbw,","sources":",ak,","types":",origin,","nst":null,"dmin":null,"rms":0.85,"gap":null,"magType":"ml","type":"earthquake","tit
le":"M 2.0 - 7 km NW of Fritz Creek, Alaska"},"geometry":{"type":"Point","coordinates":[-151.3941,59.784,82.6]},"id":"ak020dlkfgbw"}],"bbox":
```


- The link will open a GeoJSON document depicting all of the earthquakes that have taken place in the last hour.

Activity: GeoJSON Activity

In this activity, you will be working with GeoJSON data to plot occurrences of earthquakes.

Suggested Time:
15 Minutes

Activity: Layer Activity

Instructions:

- Open the logic.js file.
- Your starter code places an API call to the USGS Earthquake Hazards Program API. Take a moment to study the "features" array that is extracted from the response.
- Add some logic to create a GeoJSON layer containing all features retrieved from the API call and add it directly to the map. You can reference today's previous activities as well as Leaflet's Doc for GeoJSON.
- Create an `overlayMaps` object using the newly created earthquake GeoJSON layer. Pass the `overlayMaps` into the layer control.
- **Bonus**:
  - Create a separate overlay layer for the GeoJSON, as well as a base layer using the `streetmap` tile layer and the `darkmap` tile layer. Add these to a layer control. Refer to the previous activity if stuck here.
  - Add a popup to each marker to display the time and location of the earthquake at that location.

- **Hints**:
  - See Leaflet Documentation on GeoJSON:
    - http://leafletjs.com/reference.html#geojson
    - Leaflet's Doc for GeoJSON

Time’s Up! Let’s Review.