Getting Started with Web Development and the ArcGIS API for JavaScript

Heather Gonzago and Anne Fitz

👉 Slides & demos: `https://bit.ly/3cnPptf` 👈

2020 ESRI DEVELOPER SUMMIT | Palm Springs, CA

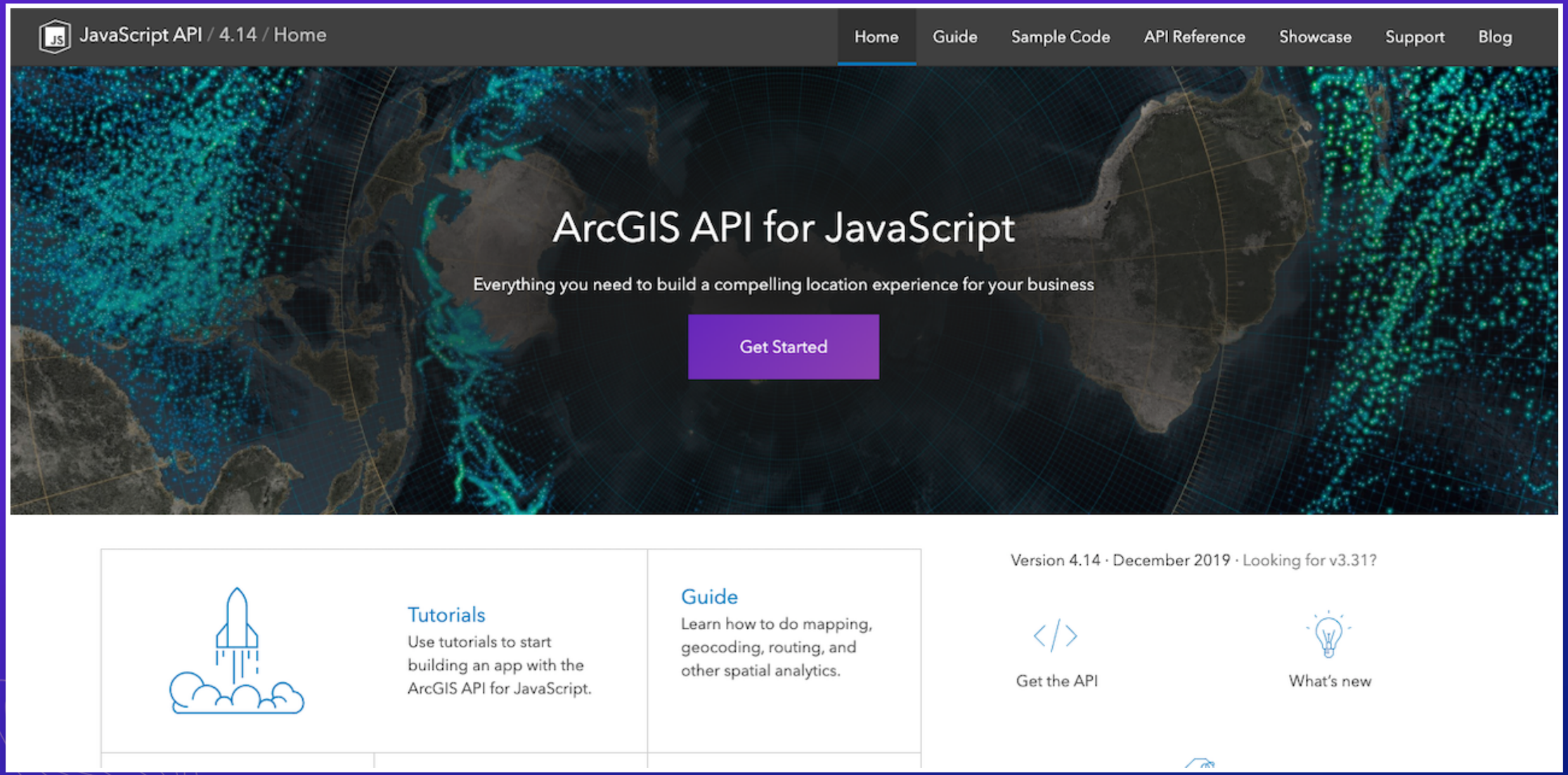
Agenda

- Setup
- First steps
- Working with layers
- Symbols and renderers
- Make the map interactive
- Filtering data
- Widgets

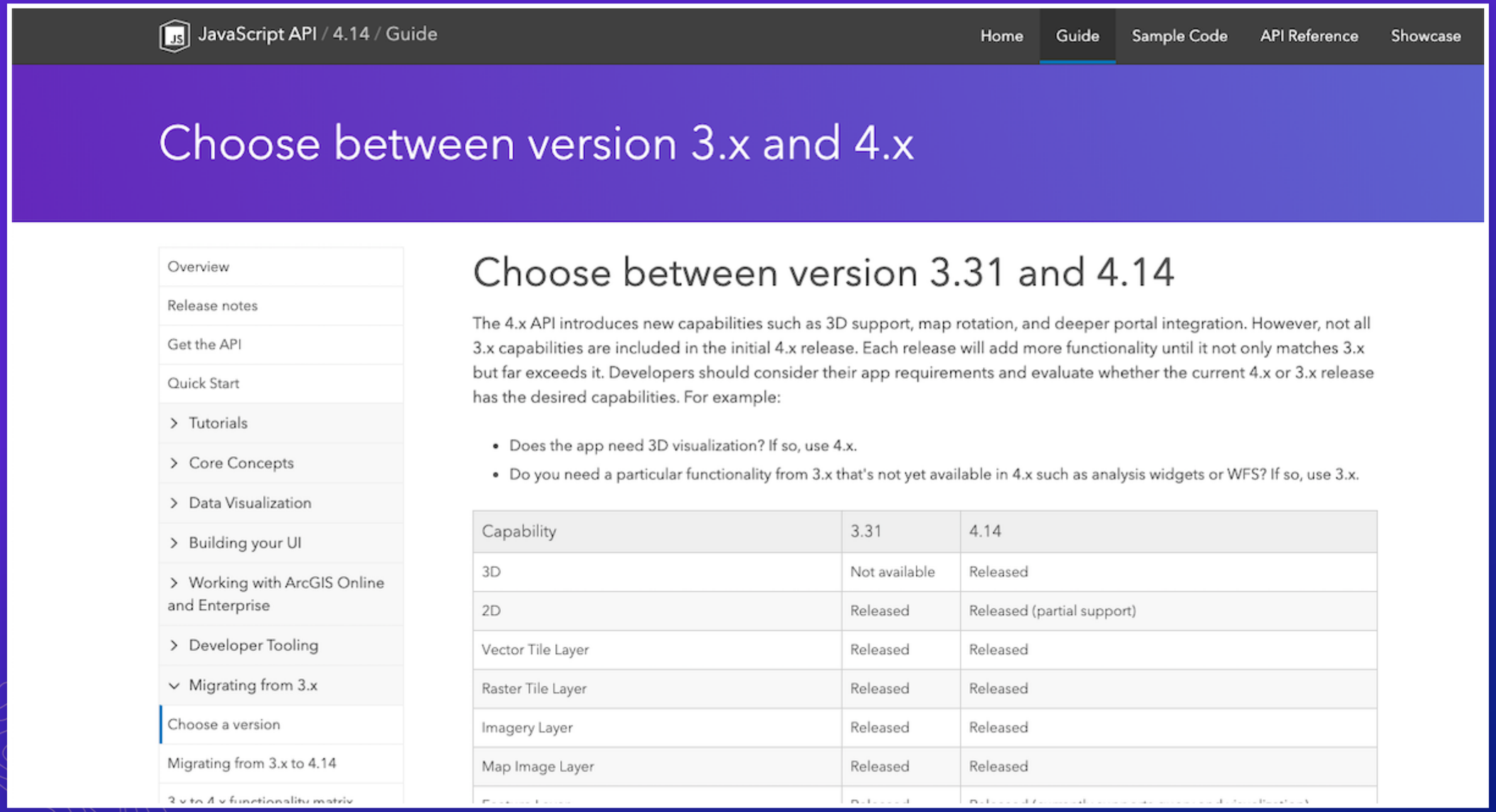
Presentations accessible via GitHub

- This session focuses on version 4.x
- Concepts remain similar between versions 3.x and 4.x
- 👉 Slides & demos: `https://bit.ly/3cnPptf` 👈

Where do I begin?



Which version of the API is best?



JavaScript API / 4.14 / Guide

Home | Guide | Sample Code | API Reference | Showcase

Choose between version 3.x and 4.x

- Overview
- Release notes
- Get the API
- Quick Start
- > Tutorials
- > Core Concepts
- > Data Visualization
- > Building your UI
- > Working with ArcGIS Online and Enterprise
- > Developer Tooling
- v Migrating from 3.x
  - Choose a version
  - Migrating from 3.x to 4.14
  - 3.x to 4.x functionality matrix

Choose between version 3.31 and 4.14

The 4.x API introduces new capabilities such as 3D support, map rotation, and deeper portal integration. However, not all 3.x capabilities are included in the initial 4.x release. Each release will add more functionality until it not only matches 3.x but far exceeds it. Developers should consider their app requirements and evaluate whether the current 4.x or 3.x release has the desired capabilities. For example:

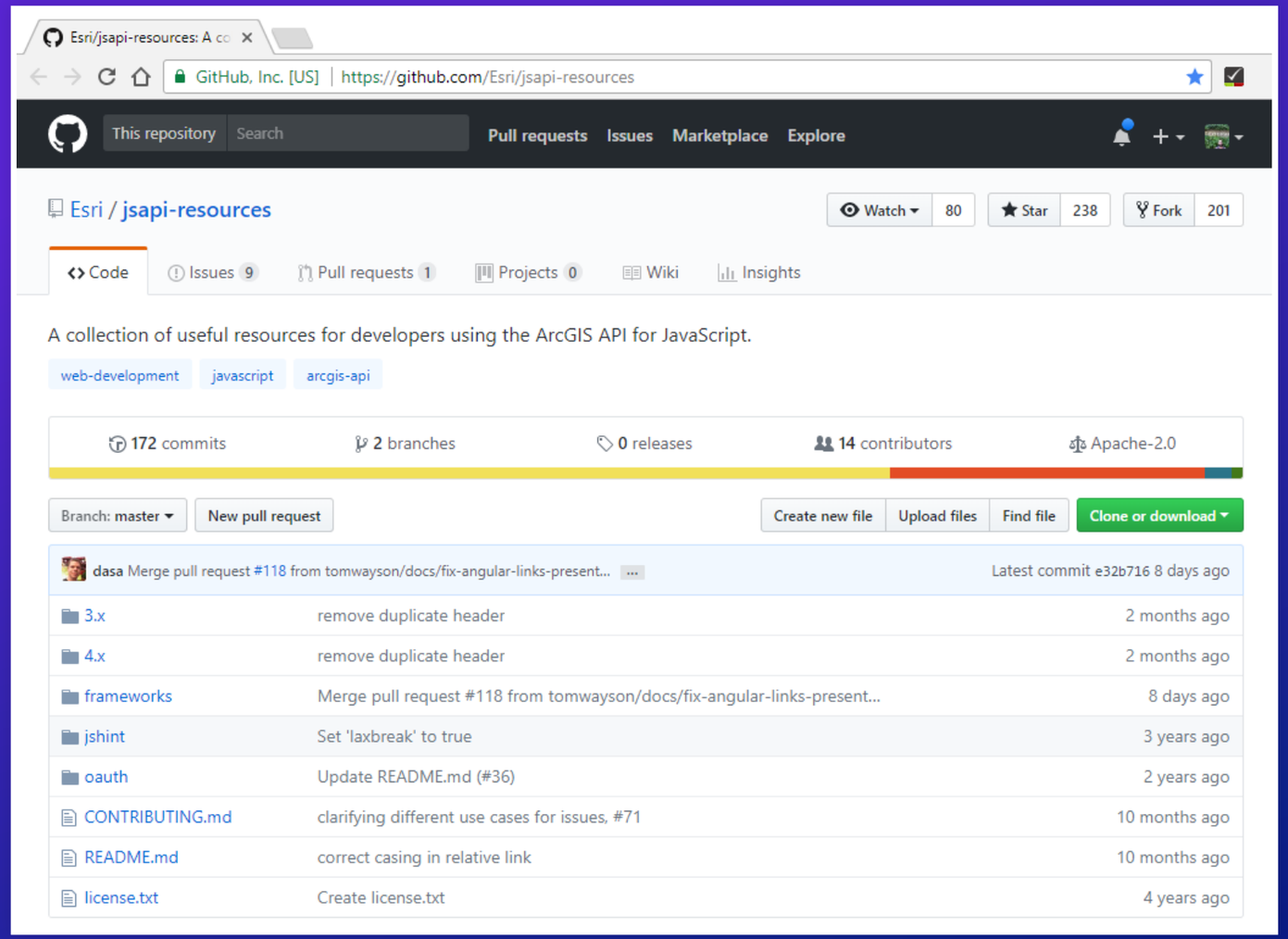
- Does the app need 3D visualization? If so, use 4.x.
- Do you need a particular functionality from 3.x that's not yet available in 4.x such as analysis widgets or WFS? If so, use 3.x.

Capability	3.31	4.14
3D	Not available	Released
2D	Released	Released (partial support)
Vector Tile Layer	Released	Released
Raster Tile Layer	Released	Released
Imagery Layer	Released	Released
Map Image Layer	Released	Released

Developer Setup

JSAPI Resources

- Includes
  - JSHint file
  - TypeScript definition file
  - Build tools (Webpack, npm)
  - Working with frameworks



Get the API

- CDN
- Custom builds
- Download builds

```
<link rel="stylesheet" href="https://js.arcgis.com/4.15/esri/css/main.css">
<script src="https://js.arcgis.com/4.15/"></script>
```

CSS

- **Main.css** contains styles for entire API

```
<link rel="stylesheet"
href="https://js.arcgis.com/4.15/esri/css/main.css">
```

- **View.css** is smaller in size but better choice if only needing basic CSS (maps, widgets, etc.)

```
<link rel="stylesheet"
href="https://js.arcgis.com/4.15/esri/css/view.css">
```

- Custom CSS (SASS)

First steps

- How will your app be written?
- Separate files or one combined file?

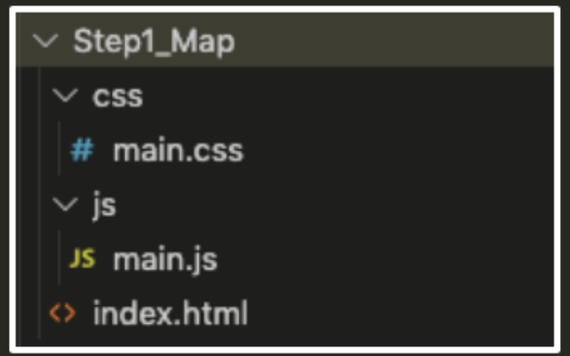
```html
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="initial-scale=1,maximum-scale=1,user-scalable=no">
  <title>Step 1: Create a map</title>
  <link rel="stylesheet" href="https://js.arcgis.com/4.14/esri/css/1main.css">
  <link rel="stylesheet" href="css/main.css">
  <script src="https://js.arcgis.com/4.14/"></script>
  <script src="js/main.js"></script>
</head>

<body>

  <div id="viewDiv"></div>
</body>

</html>
```



MapView

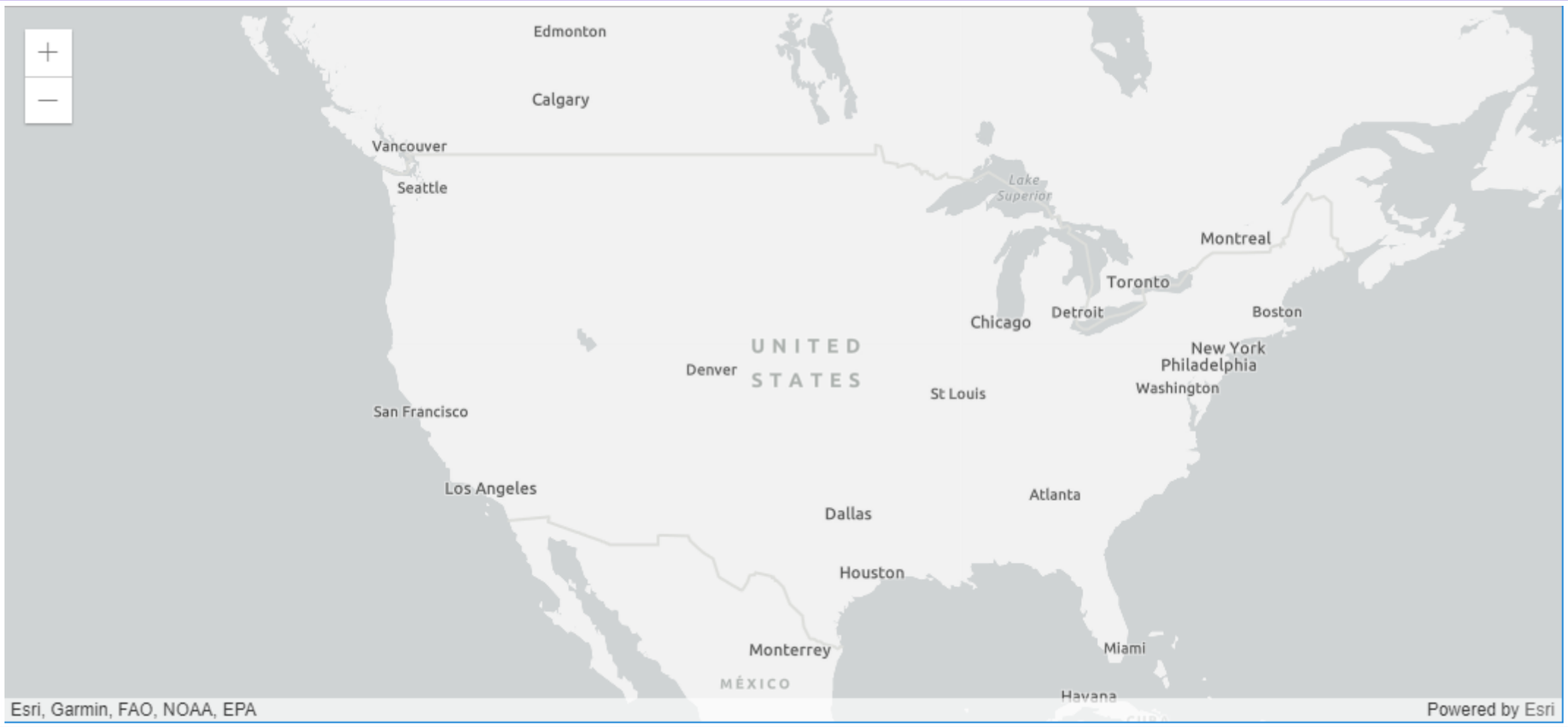
Visualize data within Map or Scene

```
const map = new Map({
  basemap: "gray-vector"
});
```

```
const view = new MapView({
  container: "viewDiv",
  map: map,
  zoom: 12,
  center: [-117.168, 32.776]
});
```

```
const view = new SceneView({
  container: "viewDiv",
  map: map,
  camera: {
    heading: 210,
    tilt: 78,
    position: {
      x: -8249335,
      y: 4832005,
      z: 50.7,
      spatialReference: {
        wkid: 3857
      }
    }
  }
});
```

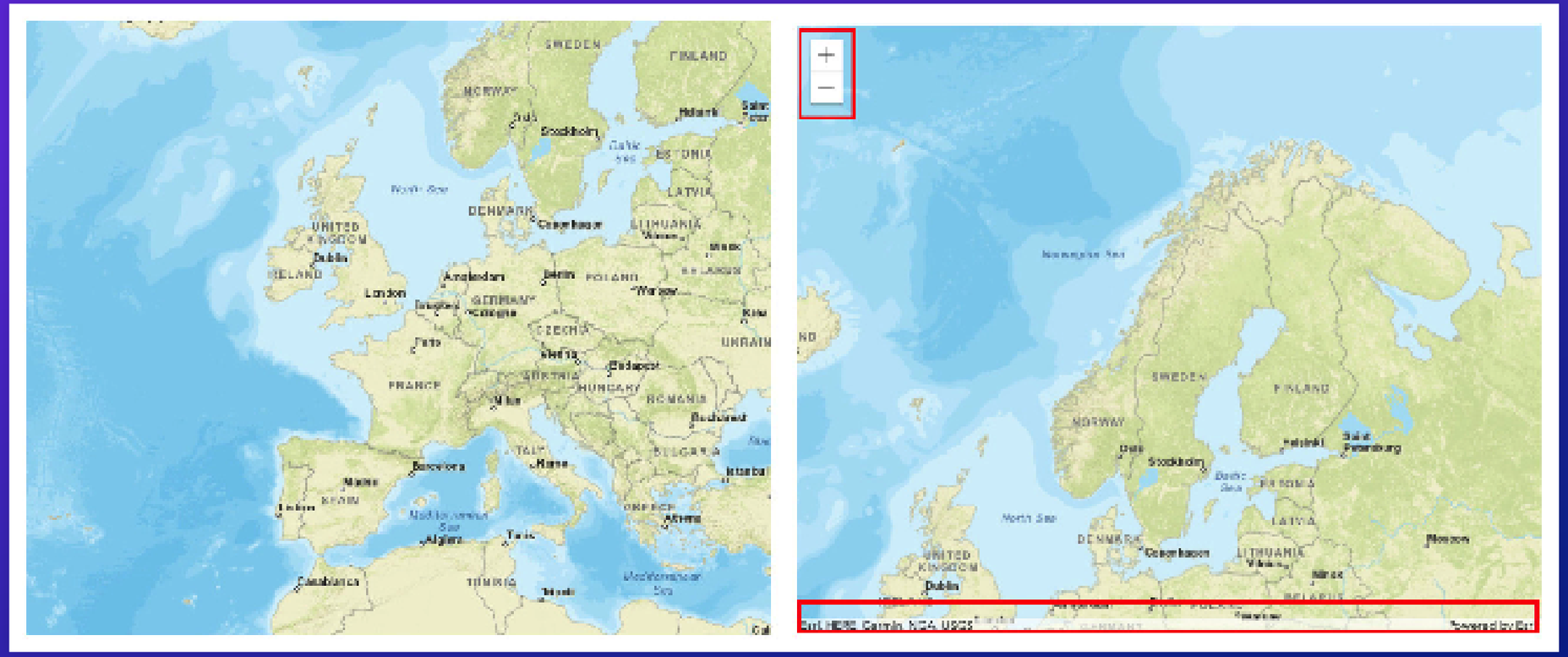
Demo: Make a map



Common Gotchas

- Module order makes a difference
- Missing module
- Missing CSS

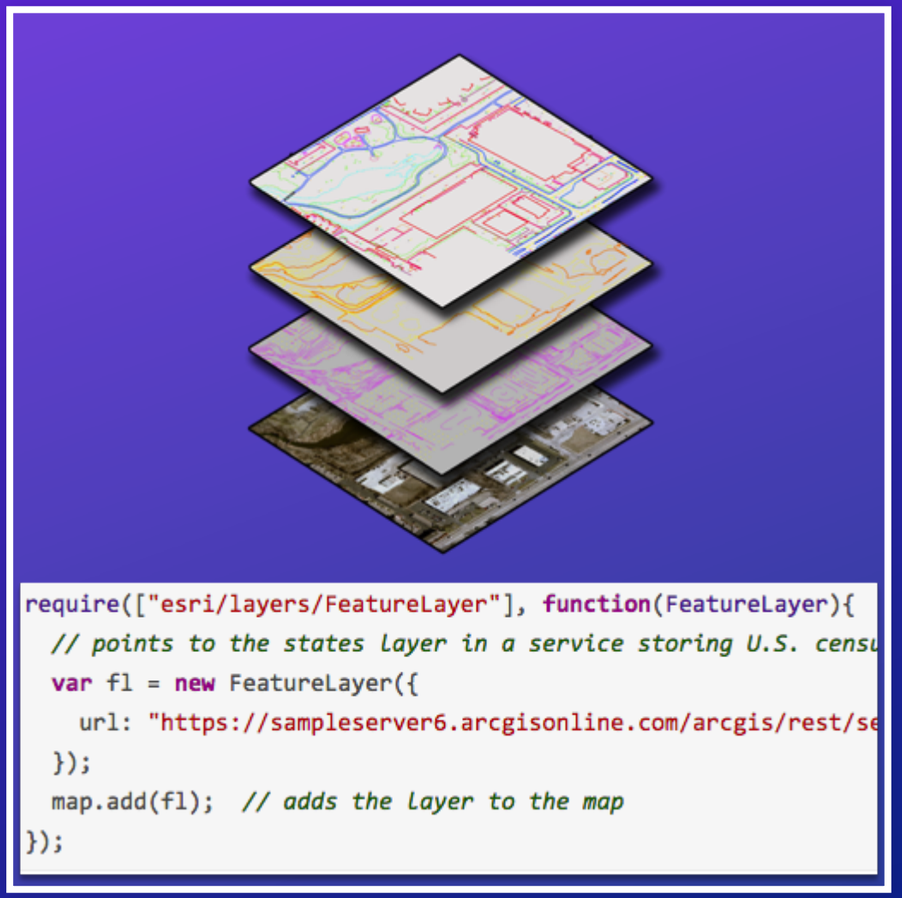
```
Uncaught ReferenceError: require is not defined                    main.js:1
    (anonymous function) @ main.js:1
```



Add layers

- Various layer types
  1. Load module
  2. Create layers
  3. Set properties
  4. Add to map or scene

- Basic steps remain the same



```
require(["esri/layers/FeatureLayer"], function(FeatureLayer){
  // points to the states layer in a service storing U.S. censu
  var fl = new FeatureLayer({
    url: "https://sampleserver6.arcgisonline.com/arcgis/rest/se
  });
  map.add(fl);  // adds the layer to the map
});
```

Properties

- No need for a bunch of get/set statements

```
const map = new Map();
map.basemap = "streets";
const view = new MapView();
view.center = [-100, 40];
view.zoom = 6;
```

- Properties can be set in constructor

```
const map = new Map({
  basemap: "streets"
});
const view = new MapView({
  map: map,
  center: [-100, 40],
  zoom: 6
});
```

Watch for property changes

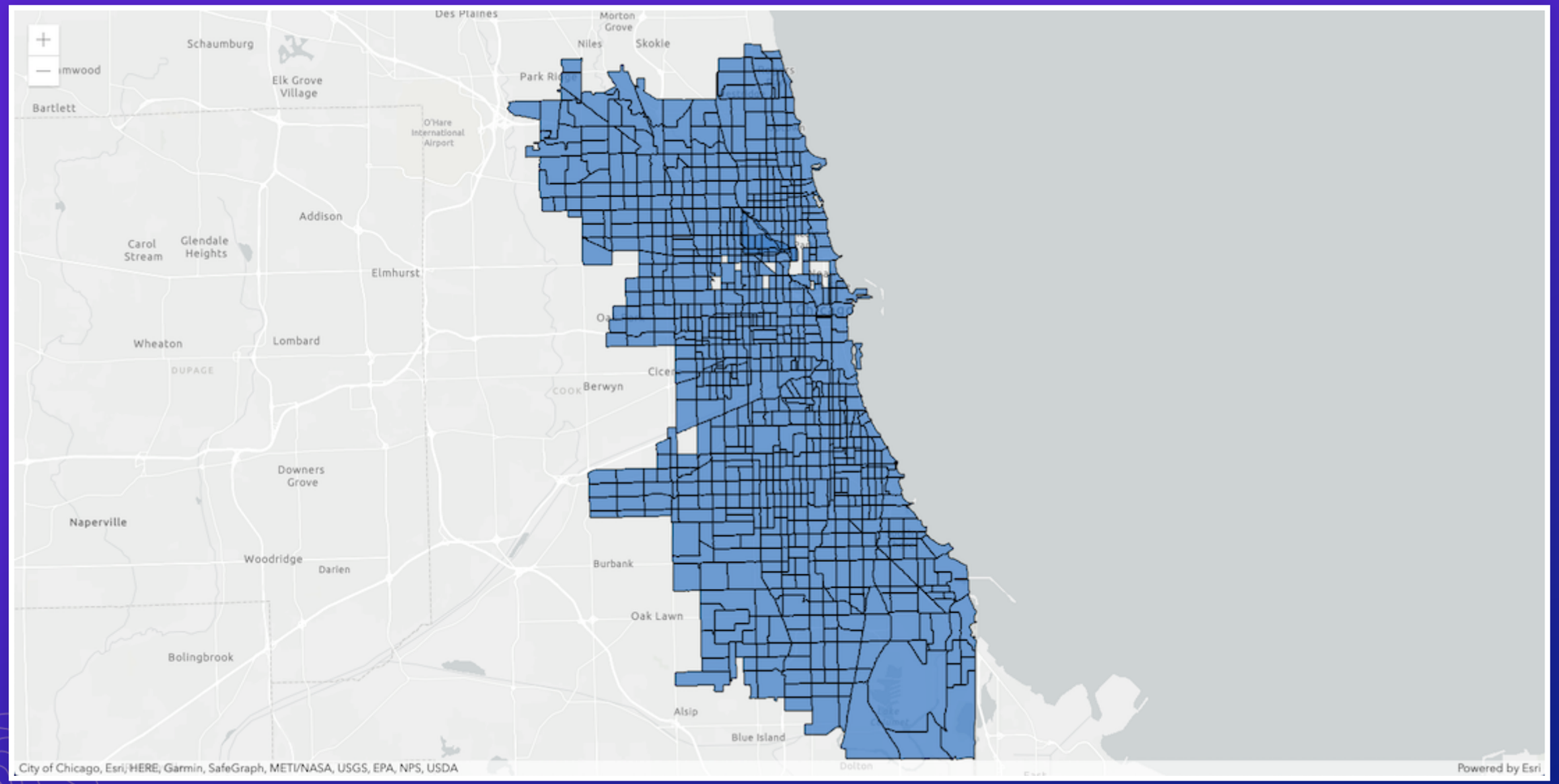
- Watch for changes

```
layer.watch("loadStatus", function(status) {// do something});
```

- Can also use esri/core/watchUtils utility methods

- See this in action with the Watch for Changes sample

Demo: Add layer to sample app



Renderers

- Define a set of symbols to use for the layer
- Sets the rules on how the symbols are used
- Basic coding pattern

```
const layerRenderer = new UniqueValueRenderer(); // Set the renderer
const featurelayer = new FeatureLayer({
  url: "featurelayer url",
  renderer: layerRenderer // pass in renderer to featurelayer using
default properties
})
```

Symbols

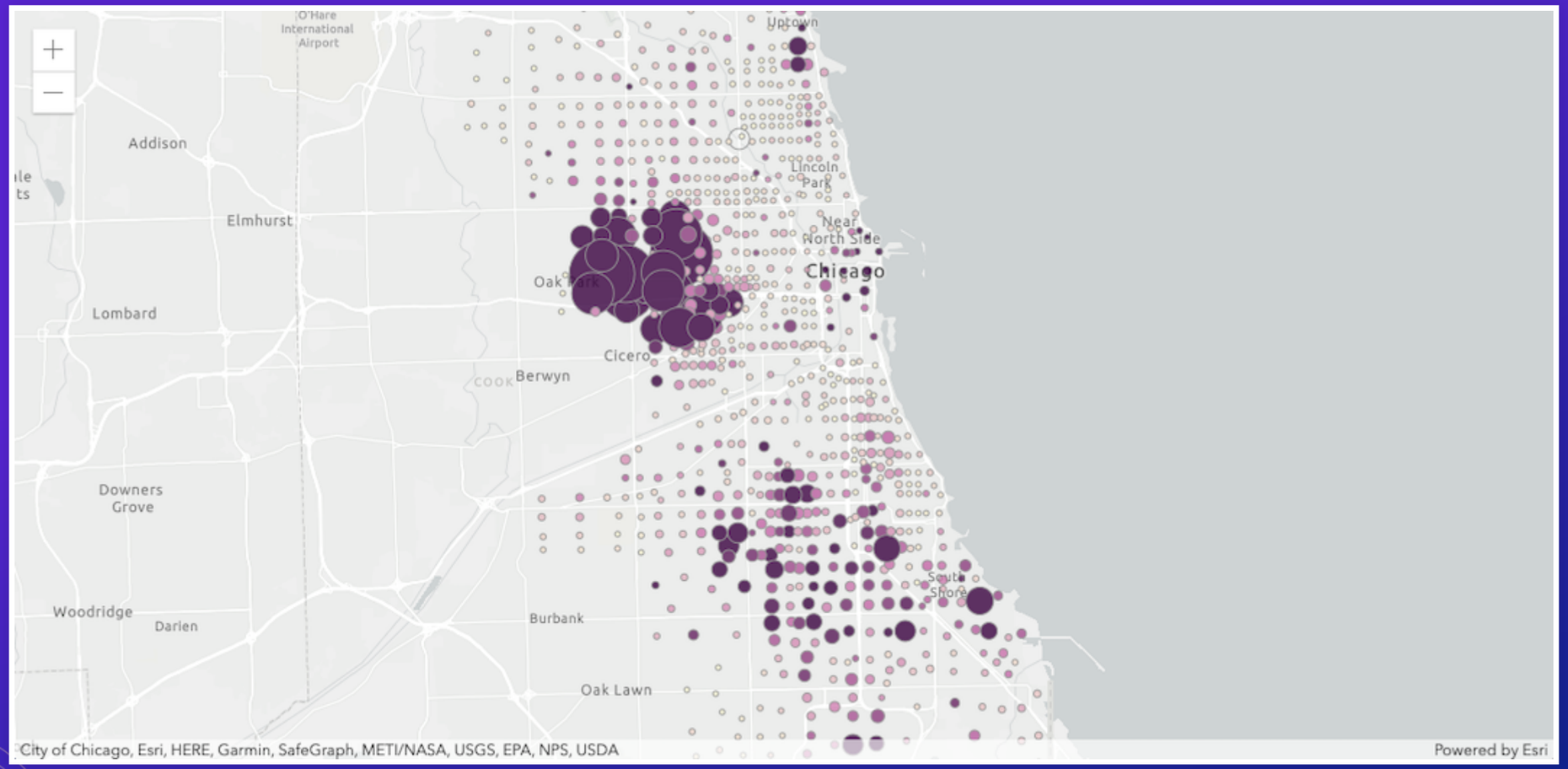
- Renderers use symbology, e.g. points, lines, polygons
- Can set symbology via a renderer's visual variables

```
const sizeVisualVariable = {
  type: 'size',
  field: 'WIND_SPEED',
  minDataValue: 0,
  maxDataValue: 60,
  minSize: 8,
  maxSize: 40
}
// Apply visual variable to renderer
renderer.visualVariables = [sizeVisualVariable]
// Create the layer
const featureLayer = new FeatureLayer({
  url: '<URL to feature layer',
  outFields: ['*'],
  renderer: renderer // Add the renderer to the feature layer
})
```

Autocasting

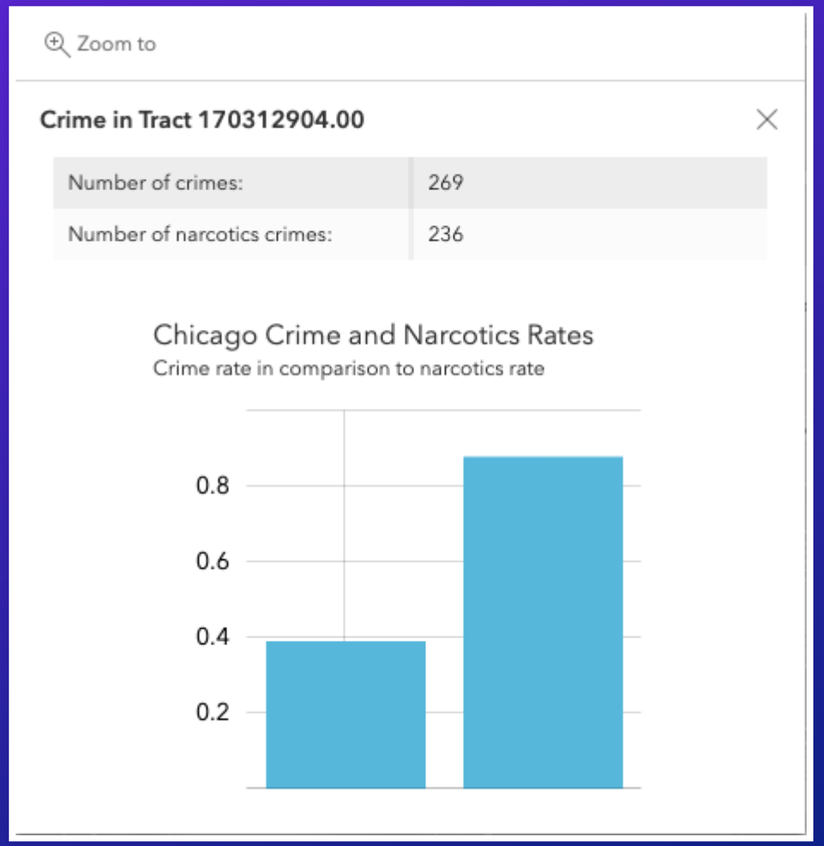
- No need to **Require()** the module
- Look for the `autocast` label in the API Reference
- Create a layer from portal item sample shows autocasting in action
- Read more about Autocasting in the Guide

Demo: Update a feature layer's renderer



Map interaction using popups

- Responds to mouse clicks
- Provides info on:
  - feature attributes
  - location
  - search results
- Customizable



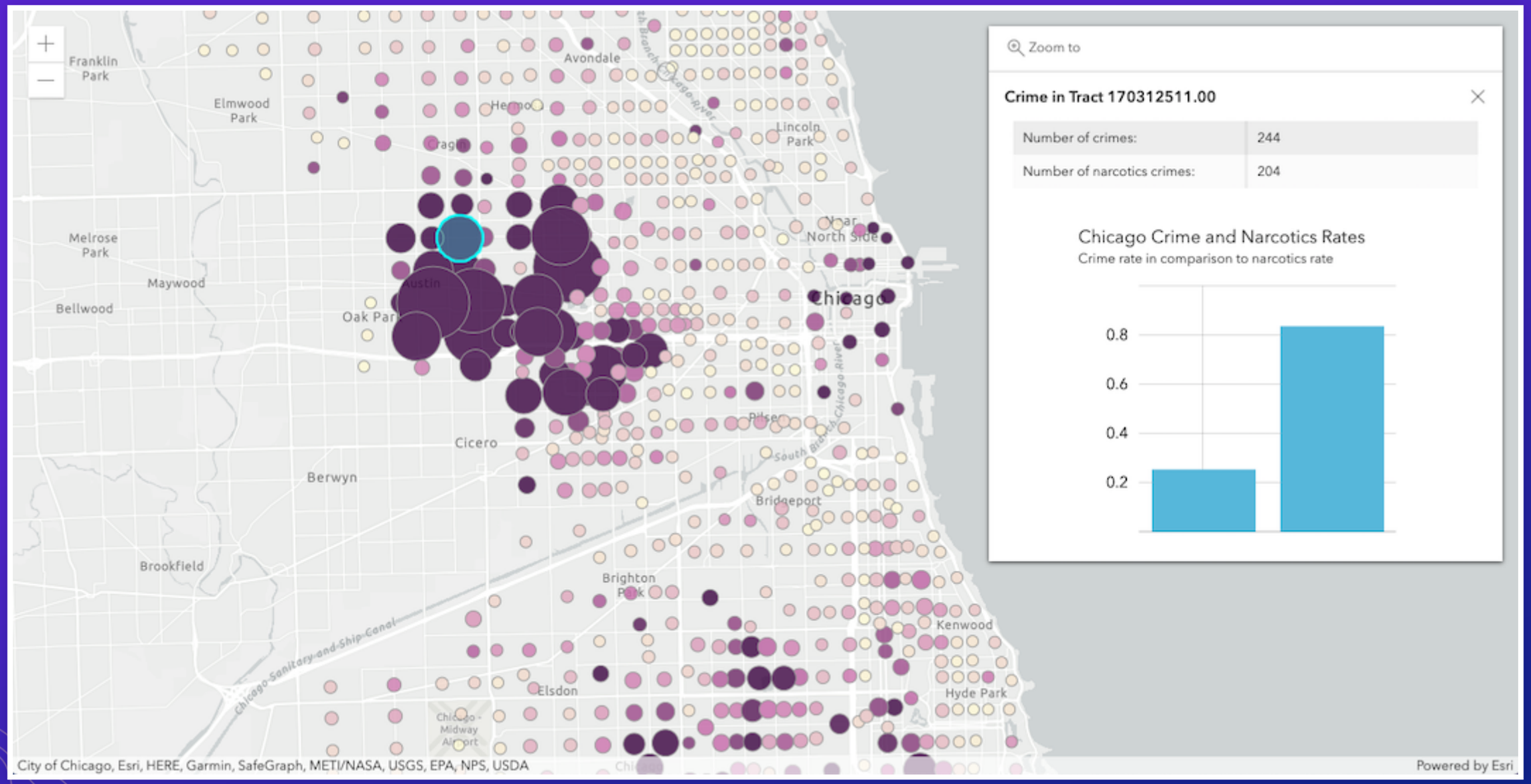
PopupTemplate

- View has a default instance of popup
- FeatureLayer has associated popupTemplate property
- Position the popup using `dockOptions`

```
const popupTemplate = new PopupTemplate({
  title: "Title of the popup",
  content: [{
    // Set the content here
  }]
});
```

```
const featurelayer = new FeatureLayer({
  url: "url to the feature layer",
  outFields: ["*"],
  popupTemplate: popupTemplate,
  renderer: renderer
});
```

Demo: Add a popup to the map

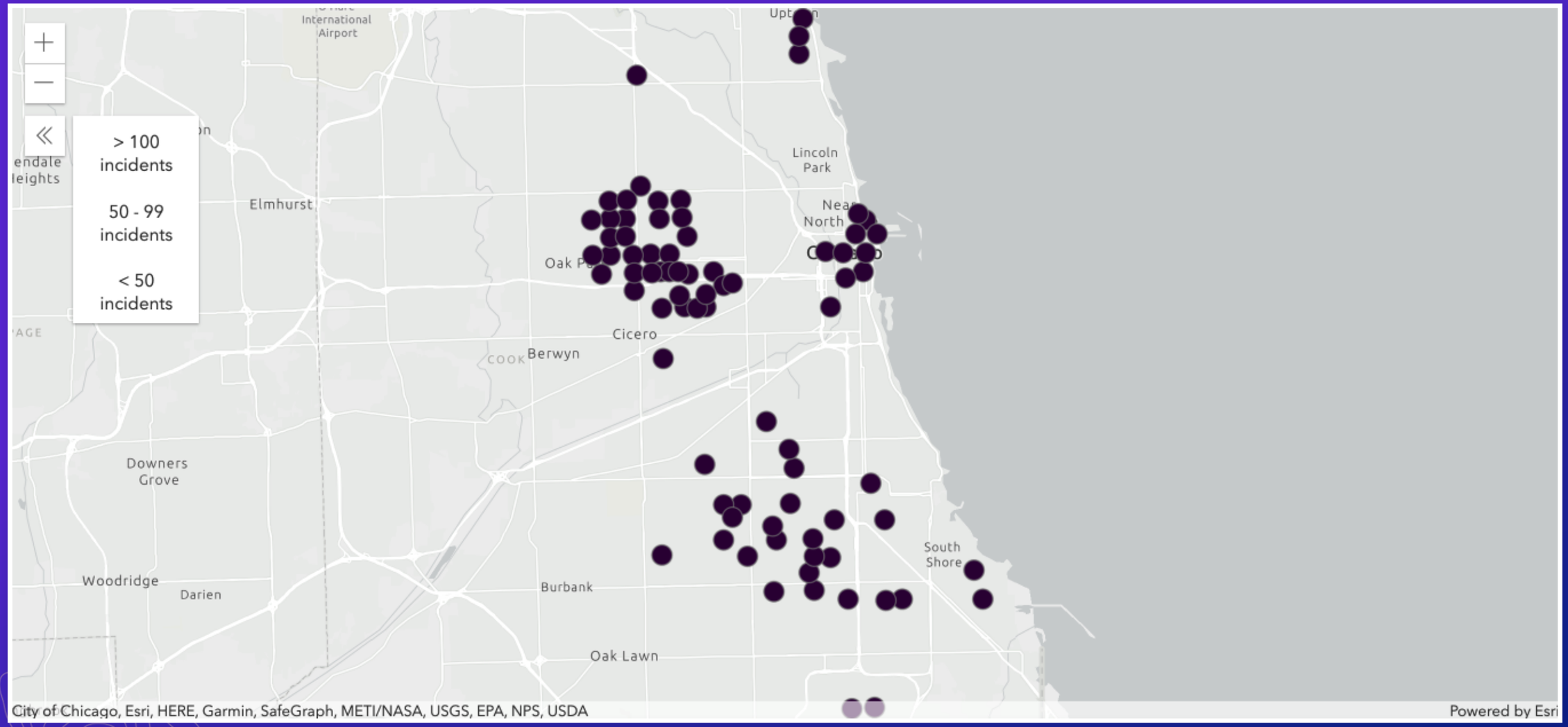


Filtering data

- `FeatureFilter`
- `FeatureLayerView`
- All data is filtered on the client = better performance

```
switch (selectedCrimeAmount) {
  case '100':
    crimeLayerView.filter = {
      where: "CrimeCnt >= '" + selectedCrimeAmount + "'"
    }
    break
  case '50-99':
    crimeLayerView.filter = {
      where: '(CrimeCnt >= 50)' + 'AND' + '(CrimeCnt <= 99)'
    }
    break
  case '49':
    crimeLayerView.filter = {
      where: "CrimeCnt <= '" + selectedCrimeAmount + "'"
    }
}
```

Demo: Filter features within a layer

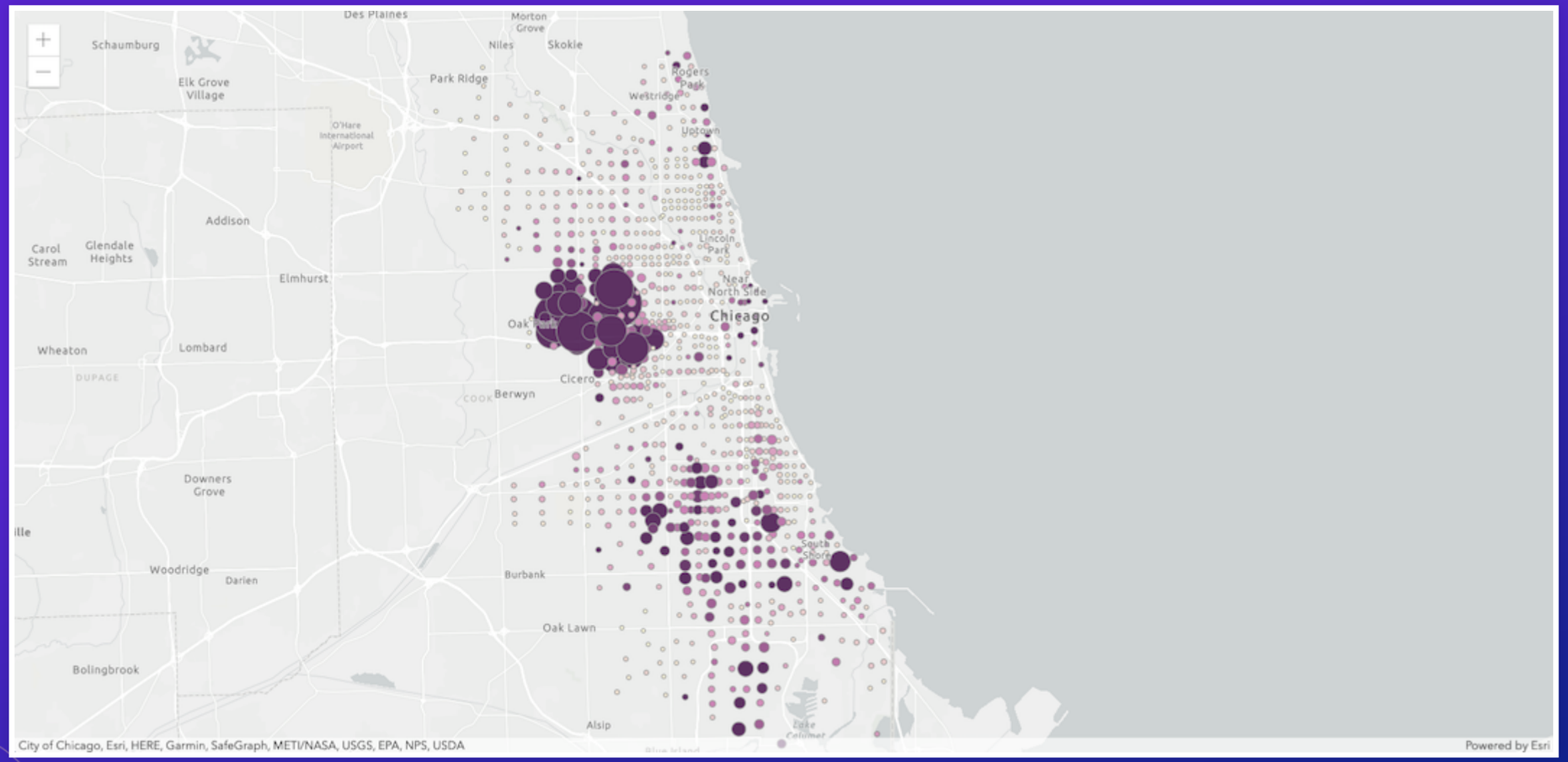


Using web maps

- Reduces coding effort
- Retains all customizations with rendering, popups, etc.

```
const map = new WebMap({
  portalItem: {
    id: "f9a9a7e3857d4d51b2c801cf8c399add" // Remember portalItem is autocasted
  }
});
```

Demo: Add a web map to an application



Widgets

- Encapsulates functionality
- Similar coding pattern across all widgets

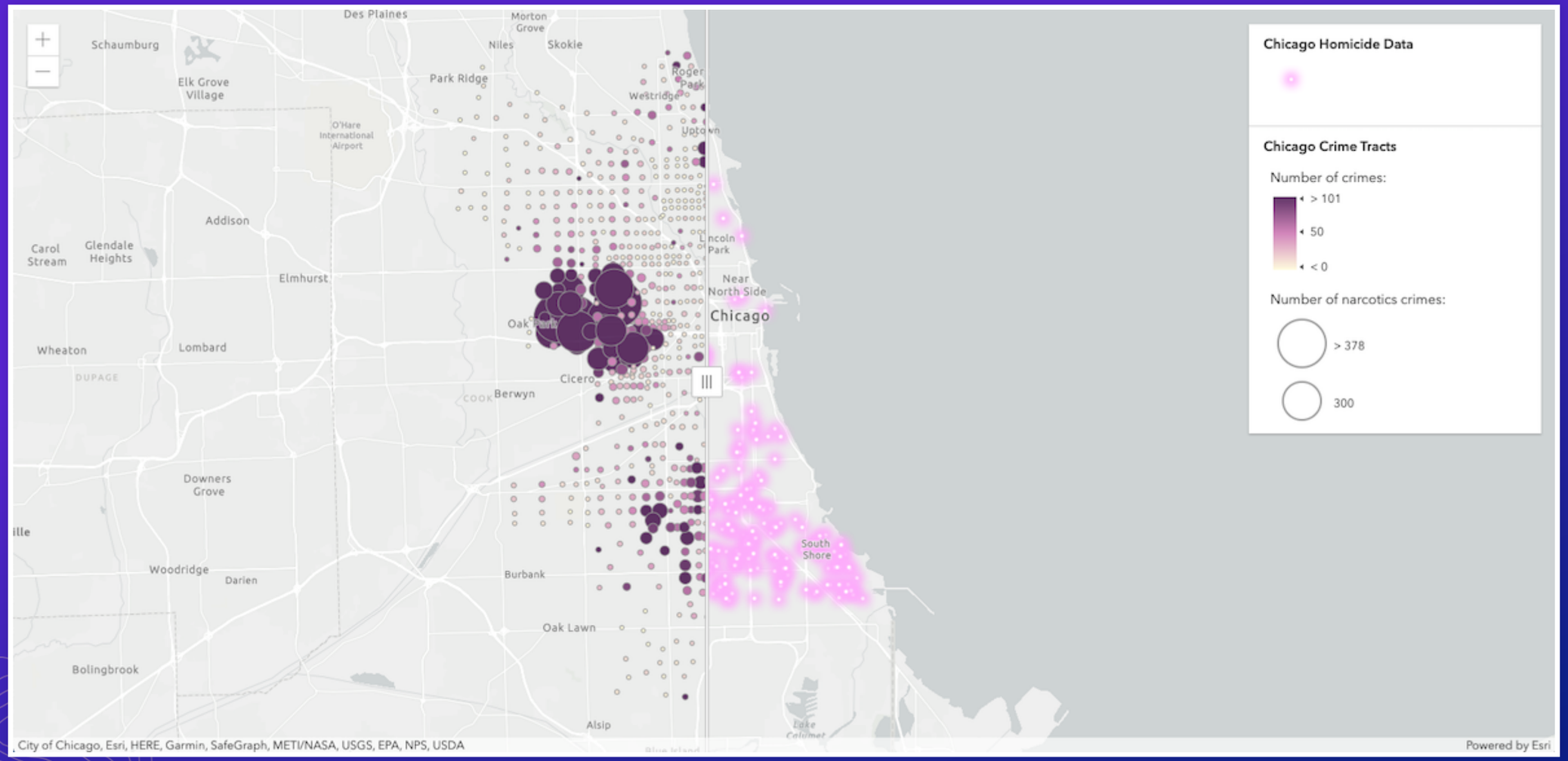
```
view.when(function){
  const featurelayer = map.layers.getItemAt(1);
  // 1. Create the widget
  const legend = new Legend({
    // 2. Specify properties for widget
    view: view,
    layerInfos: [{
        layer: featurelayer,
        title: "Name"
    }]
});
  // 3. Add widget to the view UI
  view.ui.add(legend, "top-right");
});
```

View UI

- Position widgets
  - Add
  - Move
  - Remove

```
view.ui.add(legend, "bottom-left");
view.ui.add(swipe);
```

Demo: Add widgets to the application



Where can I get more info?

- SDK Documentation
- Esri-related training and webinars
- JavaScript online training, free and not-so-free
- User forums, e.g. GeoNet, StackExchange, Spatial Community in Slack, etc.