

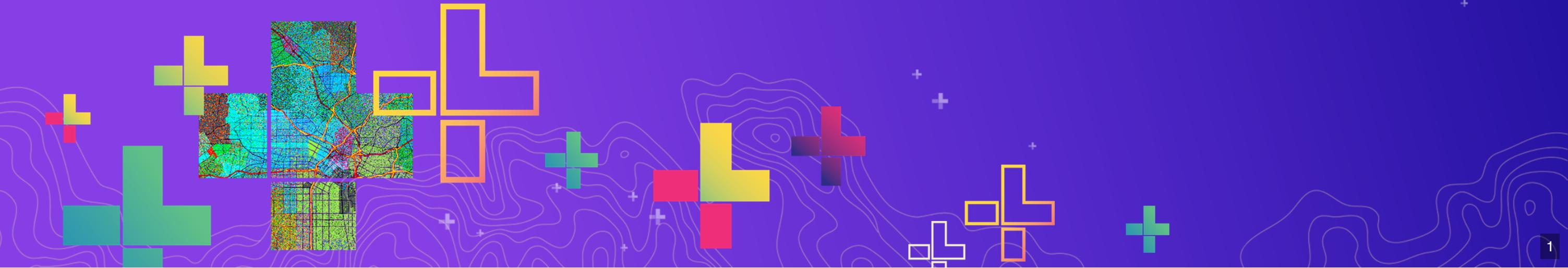


Getting Started with Web Development and the ArcGIS API for JavaScript

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

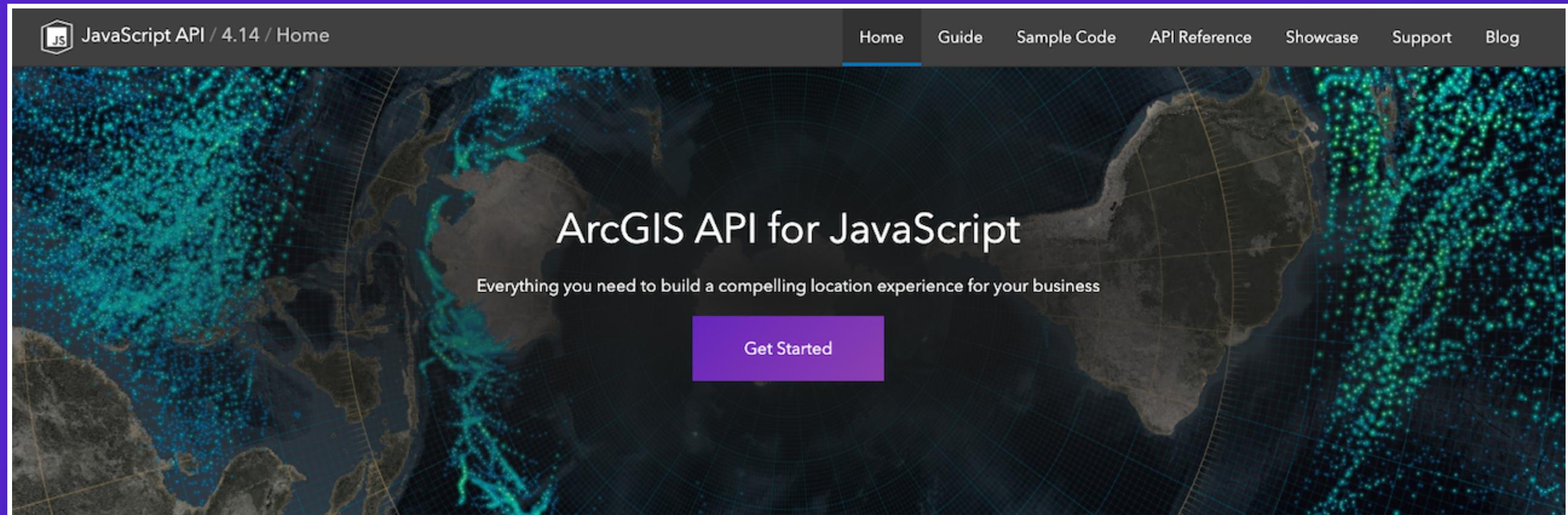


Presentations accessible via GitHub

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



Where do I begin?



The screenshot shows the ArcGIS API for JavaScript homepage. The top navigation bar includes links for Home, Guide, Sample Code, API Reference, Showcase, Support, and Blog. The main heading "ArcGIS API for JavaScript" is displayed over a world map background with glowing green and blue data points. A sub-headline reads "Everything you need to build a compelling location experience for your business". A prominent purple "Get Started" button is centered below the headline. Below the main section, there are two cards: "Tutorials" featuring a rocket launch icon and "Guide" featuring a lightbulb icon. A note at the bottom right indicates "Version 4.14 · December 2019 · Looking for v3.31?".

JavaScript API / 4.14 / Home

Home Guide Sample Code API Reference Showcase Support Blog

ArcGIS API for JavaScript

Everything you need to build a compelling location experience for your business

Get Started

Tutorials

Use tutorials to start building an app with the ArcGIS API for JavaScript.

Guide

Learn how to do mapping, geocoding, routing, and other spatial analytics.

Version 4.14 · December 2019 · Looking for v3.31?

</>

Get the API

What's new

Which version of the API is best?

The screenshot shows a web browser displaying the [ArcGIS JavaScript API / 4.14 / Guide](#). The top navigation bar includes links for Home, Guide (which is highlighted), Sample Code, API Reference, and Showcase. The main content area has a purple header with the text "Choose between version 3.x and 4.x". On the left, there's a sidebar with a navigation menu:

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

At the bottom of the sidebar, it says "2.x to 4.x functionality matrix". The main content area contains the following text and table:

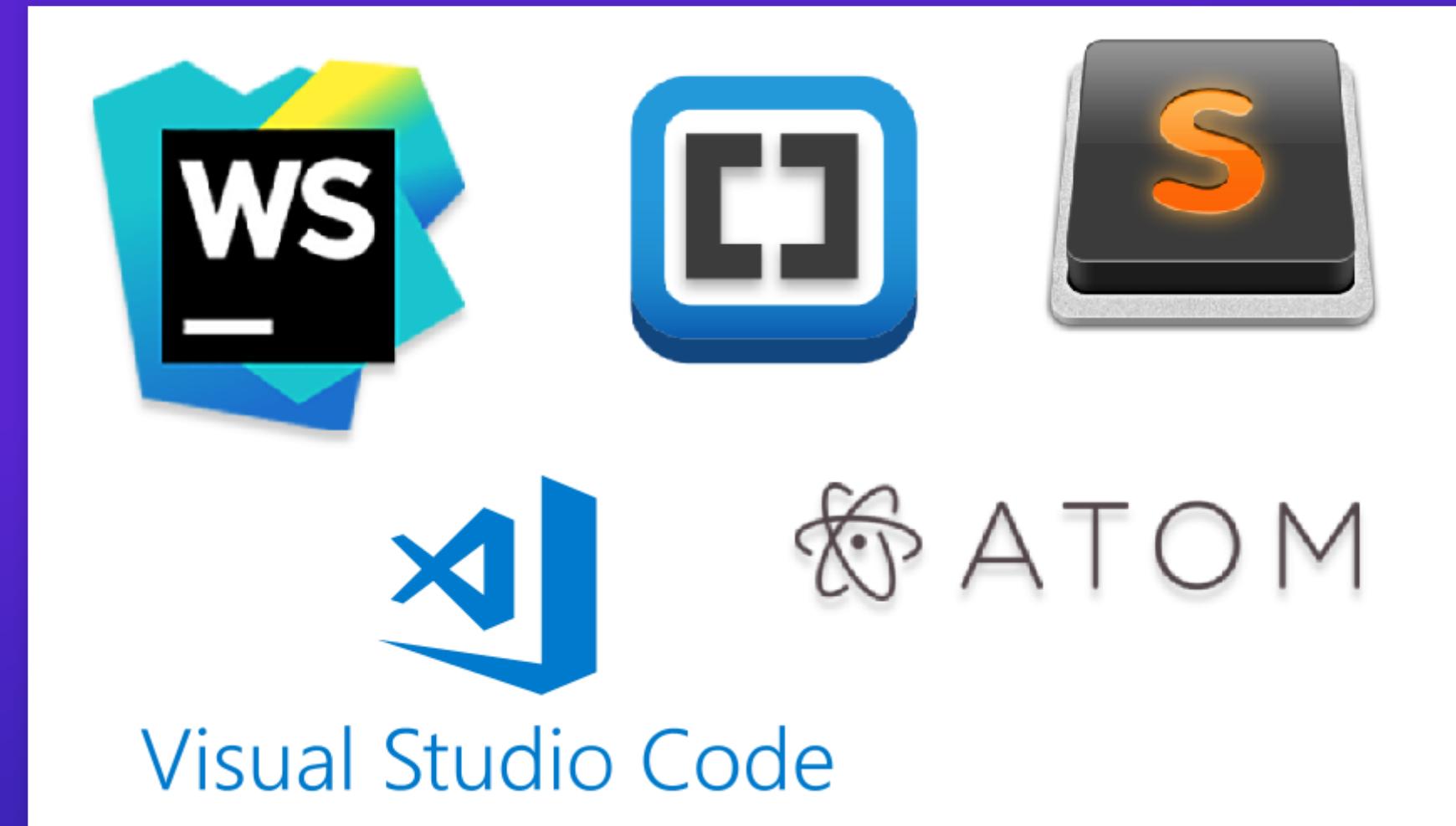
Choose between version 3.31 and 4.14

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.

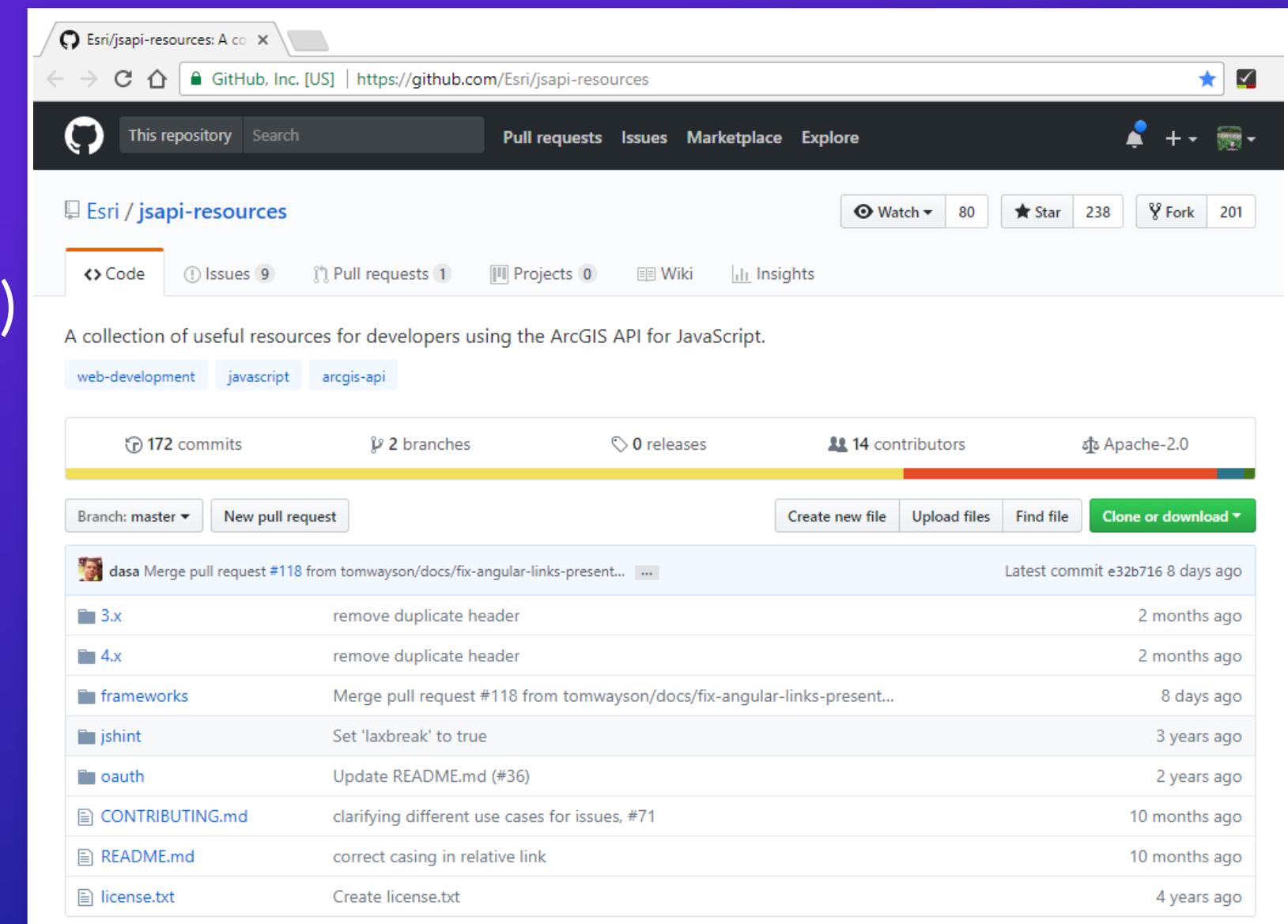
Capability	3.31	4.14
3D	Not available	Released
2D	Released	Released (partial support)
Vector Tile Layer	Released	Released
Raster Tile Layer	Released	Released
Imagery Layer	Released	Released
Map Image Layer	Released	Released

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



A screenshot of a Mac OS X desktop window. The window title bar has three colored buttons (red, yellow, green) on the left and a close button (white square with a black 'X') on the right. The main content area contains the following HTML code:

```
<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?

```
<!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/main.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>
```

```
✓ Step1_Map
  ✓ css
    # main.css
  ✓ js
    JS main.js
  <> index.html
```

MapView

Visualize data within Map or Scene

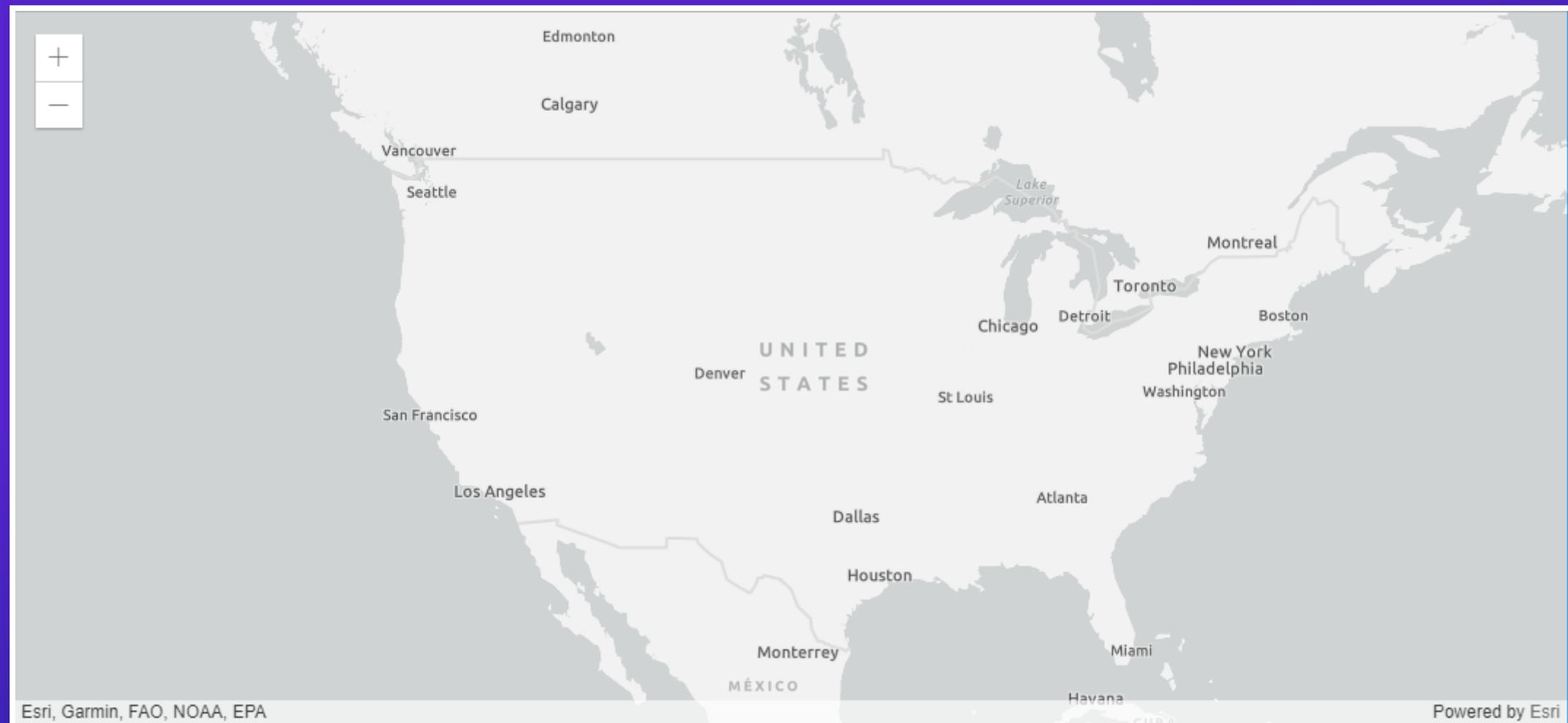
```
const map = new Map({  
  basemap: "gray-vector"  
});
```

```
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  
      }  
    }  
  }  
});
```

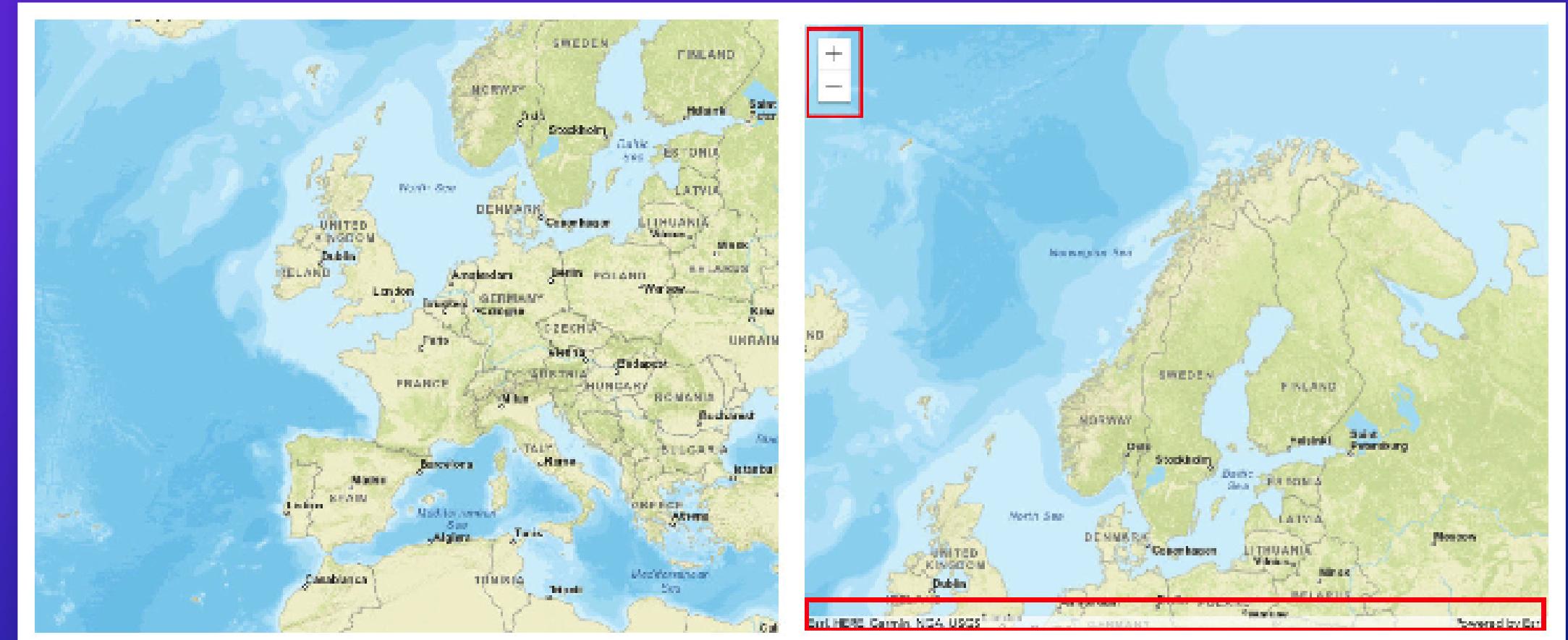


Demo: Make a map



Common Gotchas

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



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. census data  
    var f1 = new FeatureLayer({  
        url: "https://sampleserver6.arcgisonline.com/arcgis/rest/se  
    });  
    map.add(f1); // adds the Layer to the map  
});
```



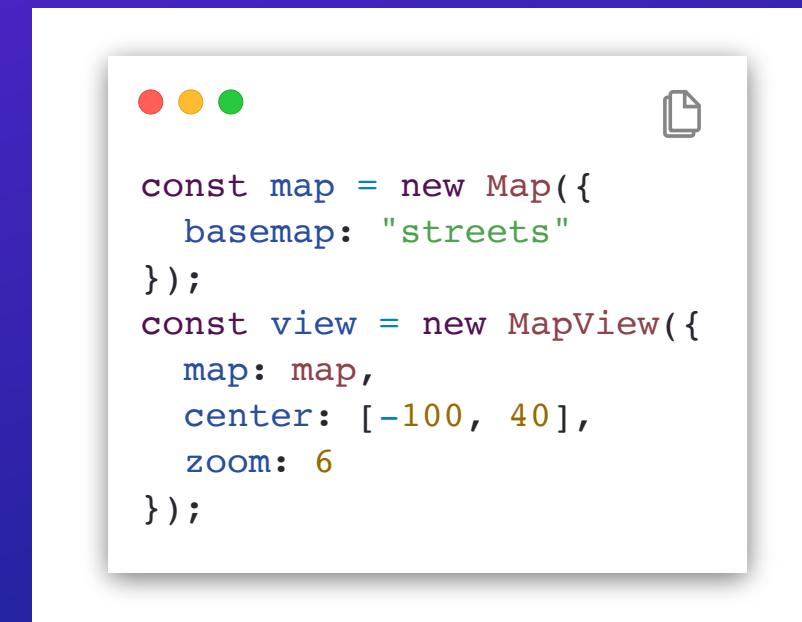
Properties

- 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

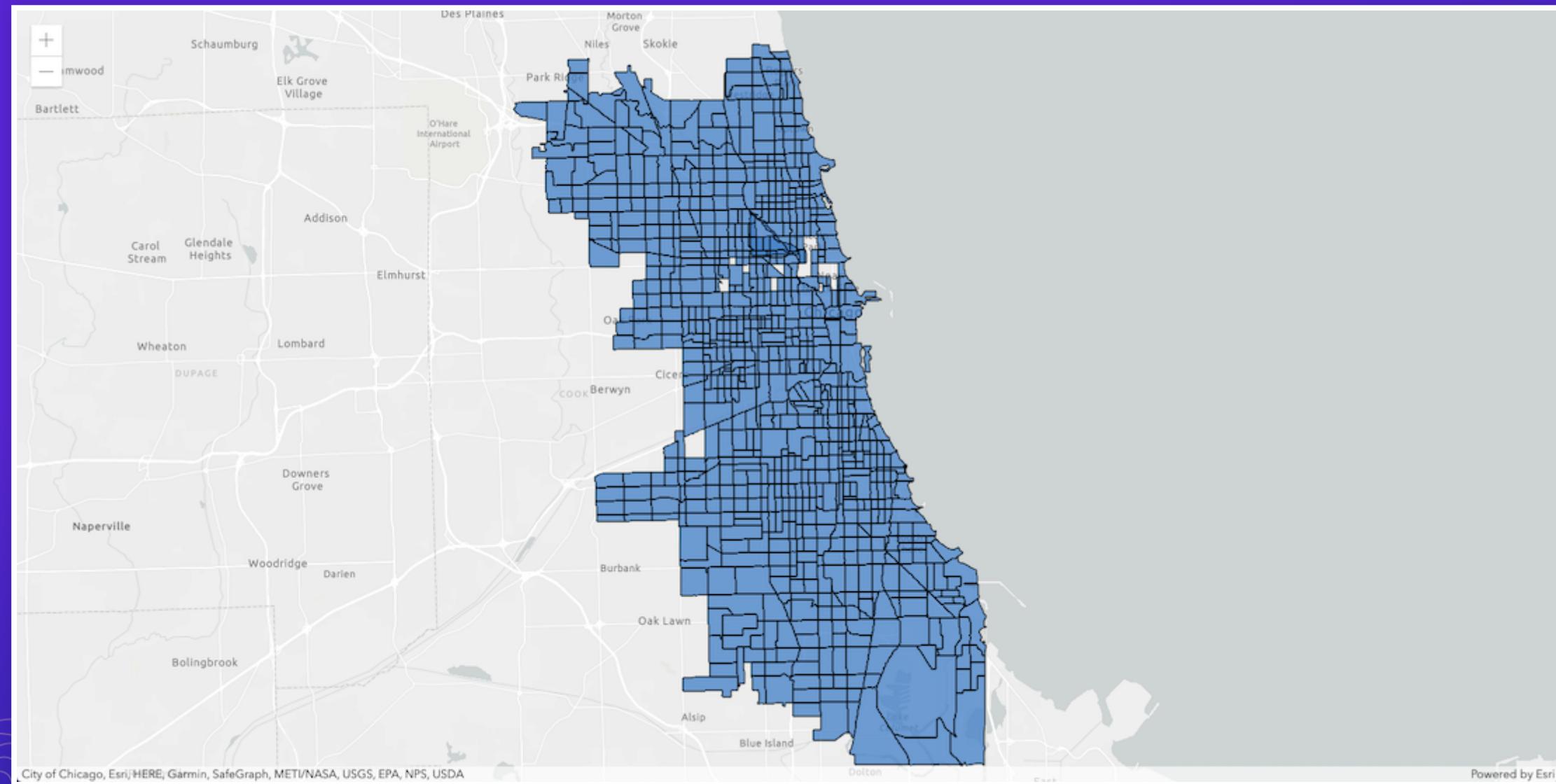


```
layer.watch("loadStatus", function(status) { // do something});
```

- 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
})
```

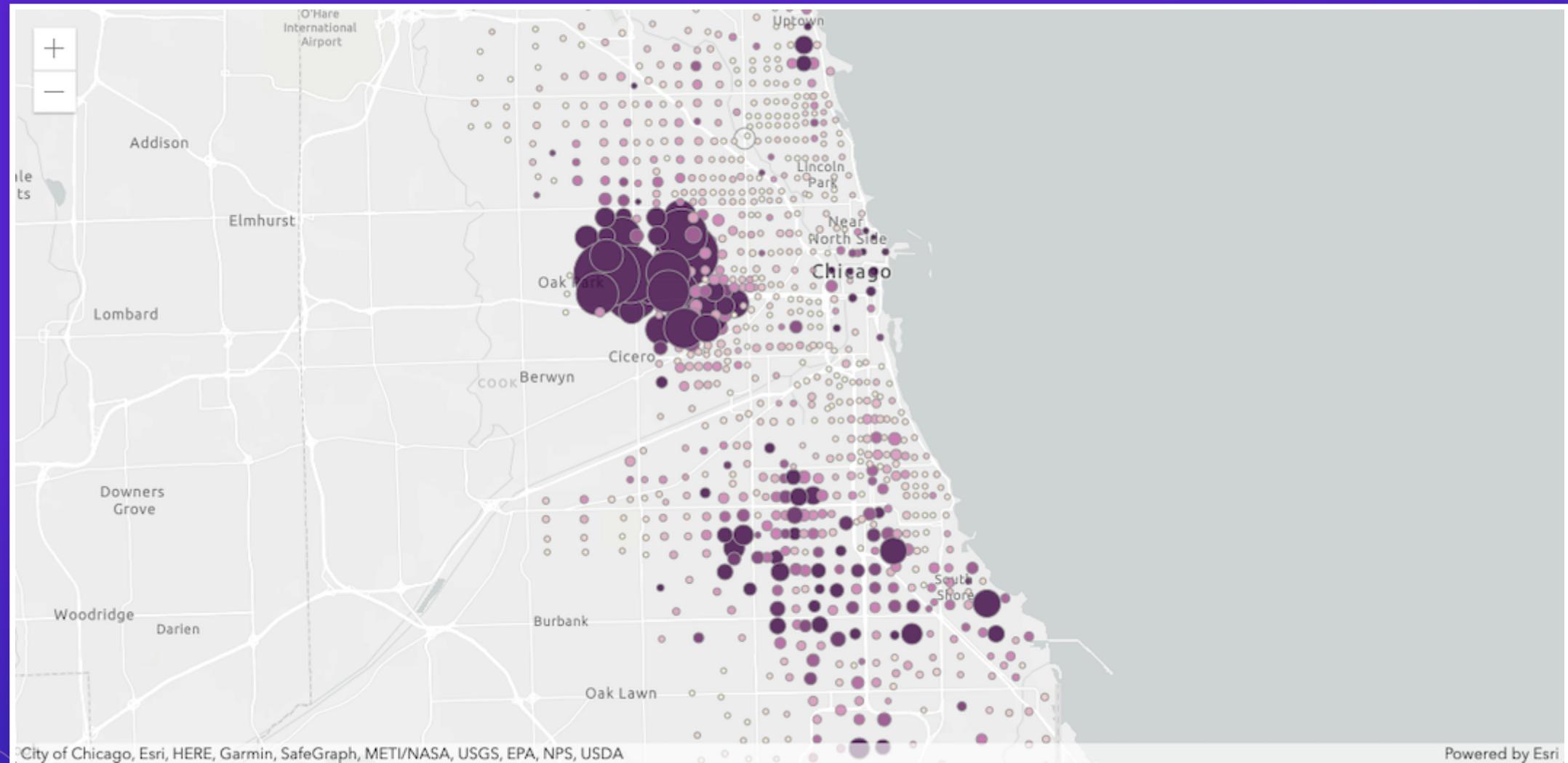


Autocasting

- 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

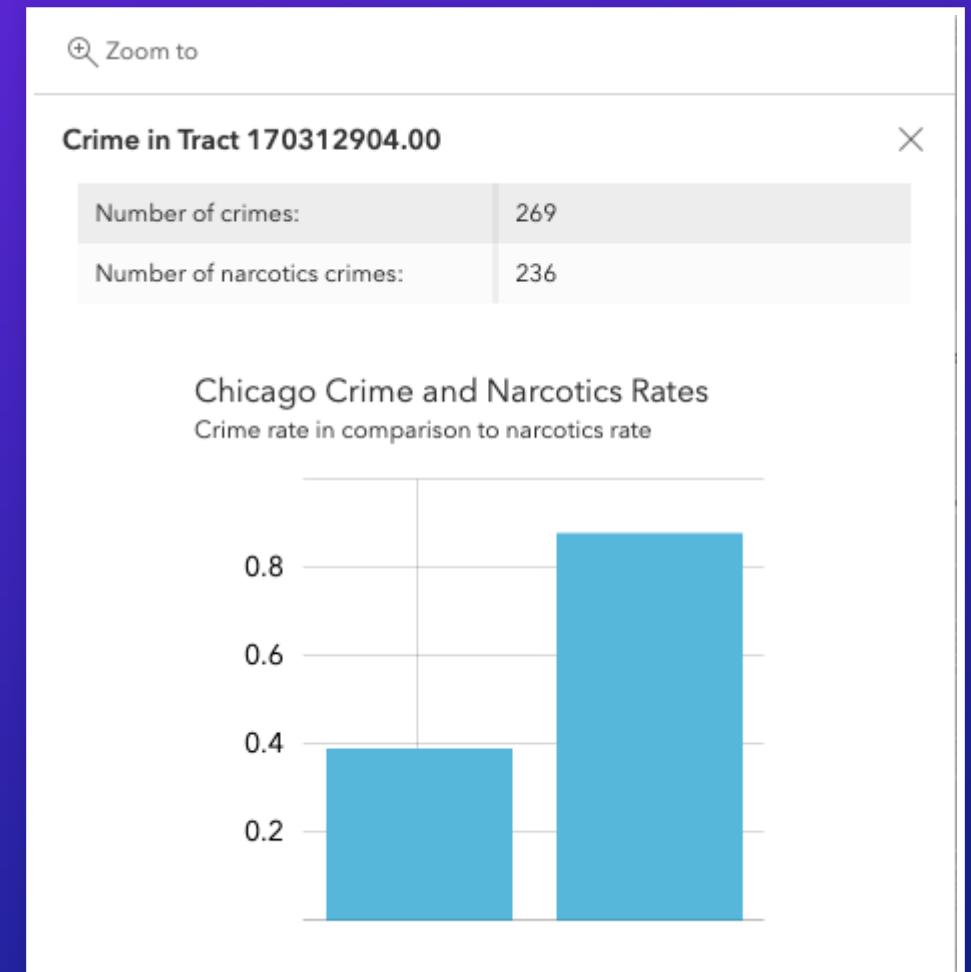


Demo: Update a feature layer's renderer



Map interaction using popups

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



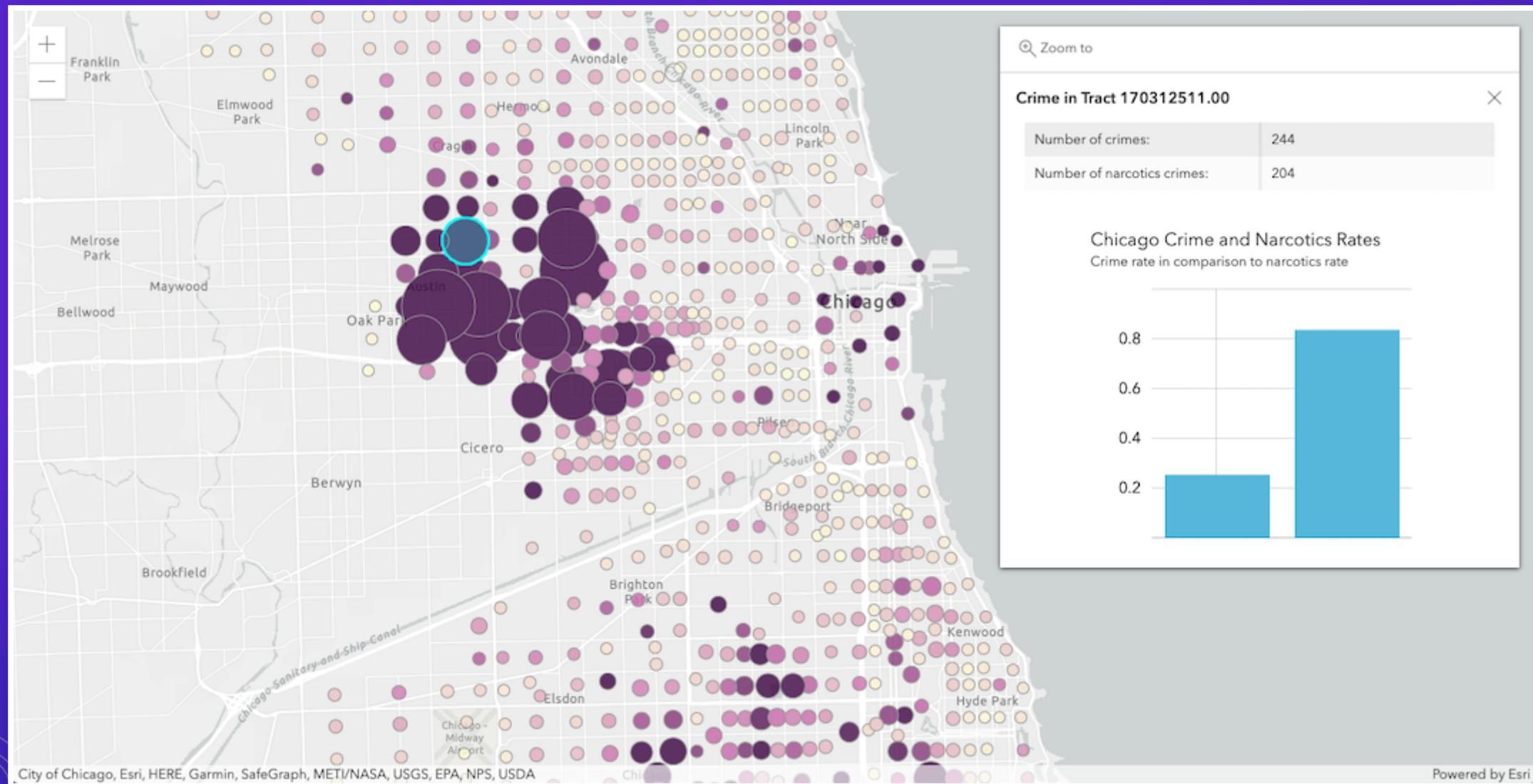
PopupTemplate

- 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

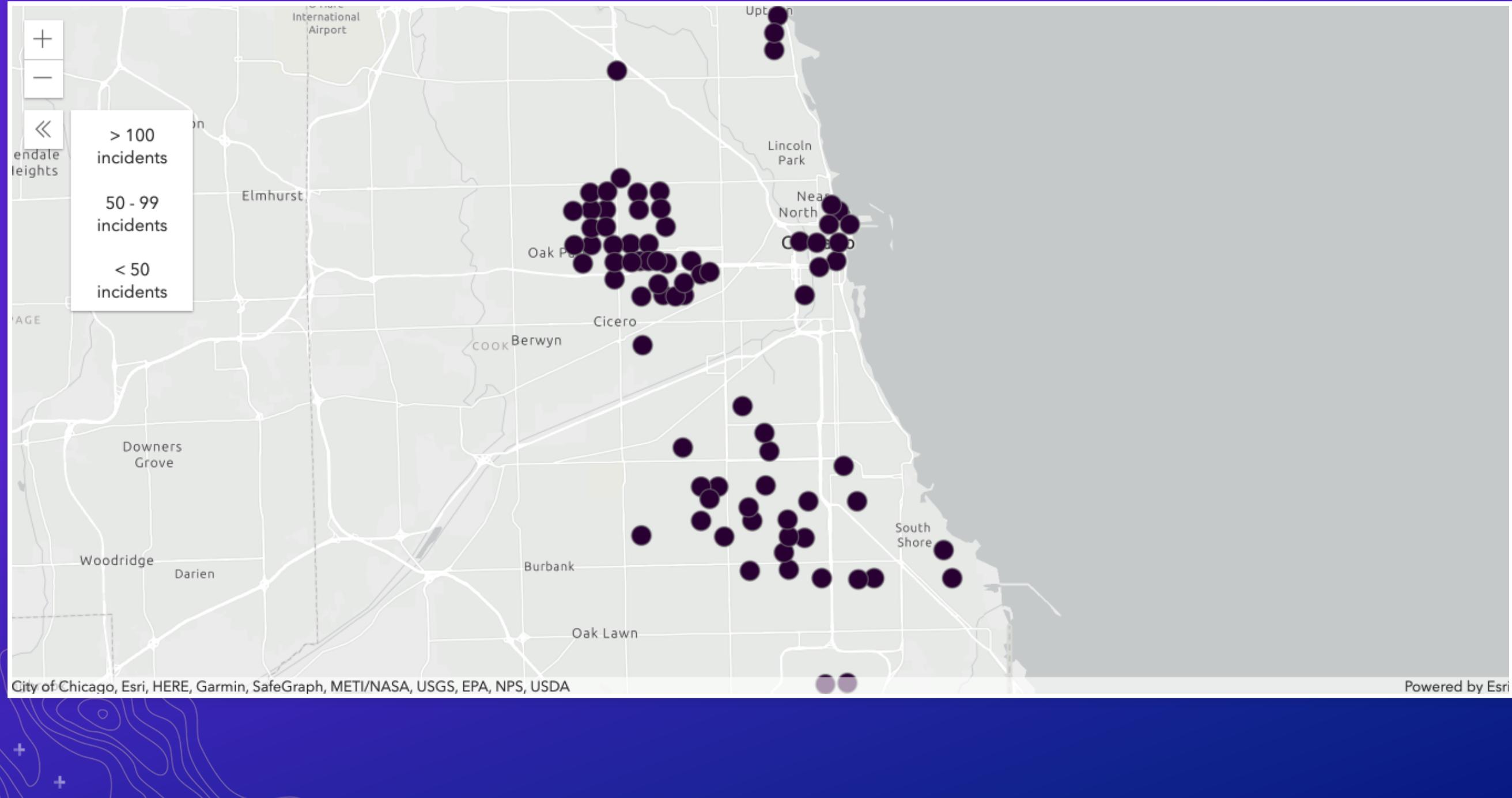


Filtering data

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

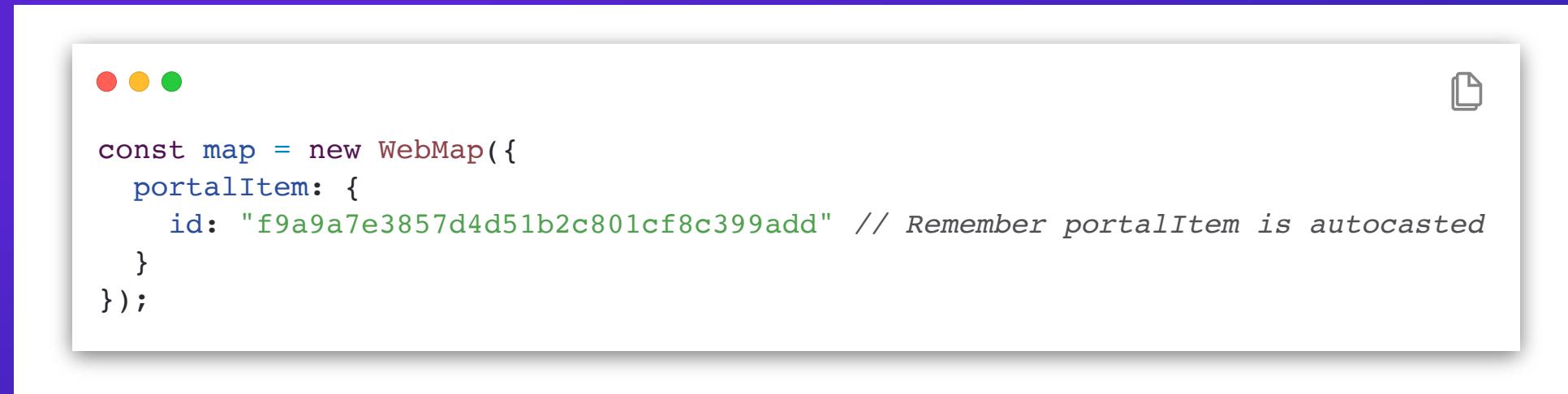
```
switch (selectedCrimeAmount) {  
    case '100':  
        crimeLayerView.filter = {  
            where: "CrimeCnt >= '" + selectedCrimeAmount + "'"  
        }  
        break  
    case '50-99':  
        crimeLayerView.filter = {  
            where: '(CrimeCnt >= 50)' + 'AND' + '(CrimeCnt <=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

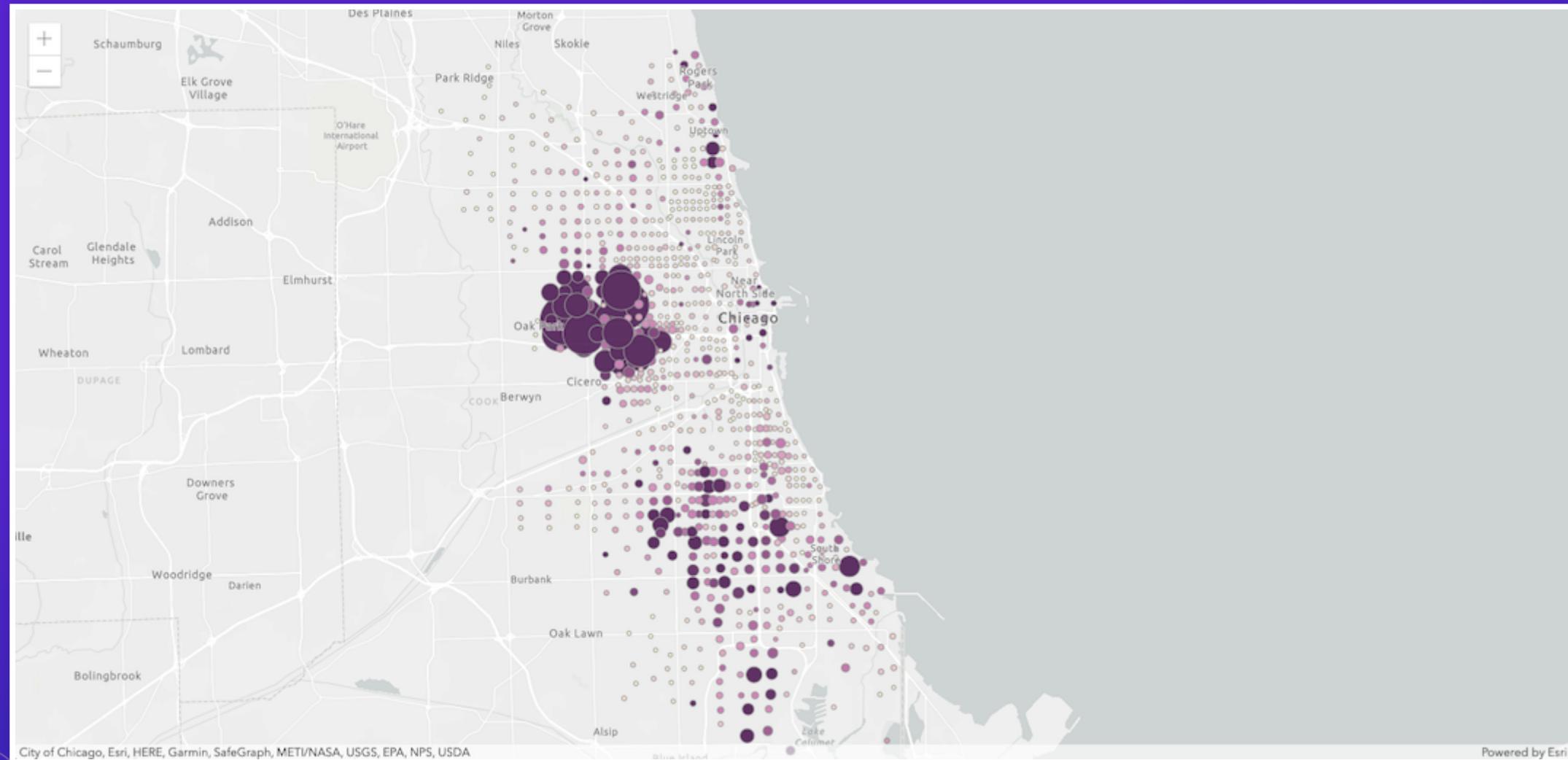


The screenshot shows a code editor window with a white background. In the top-left corner, there are three small colored circles (red, yellow, green). In the top-right corner, there is a small icon of a document with a pencil. The code itself is written in a monospaced font:

```
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

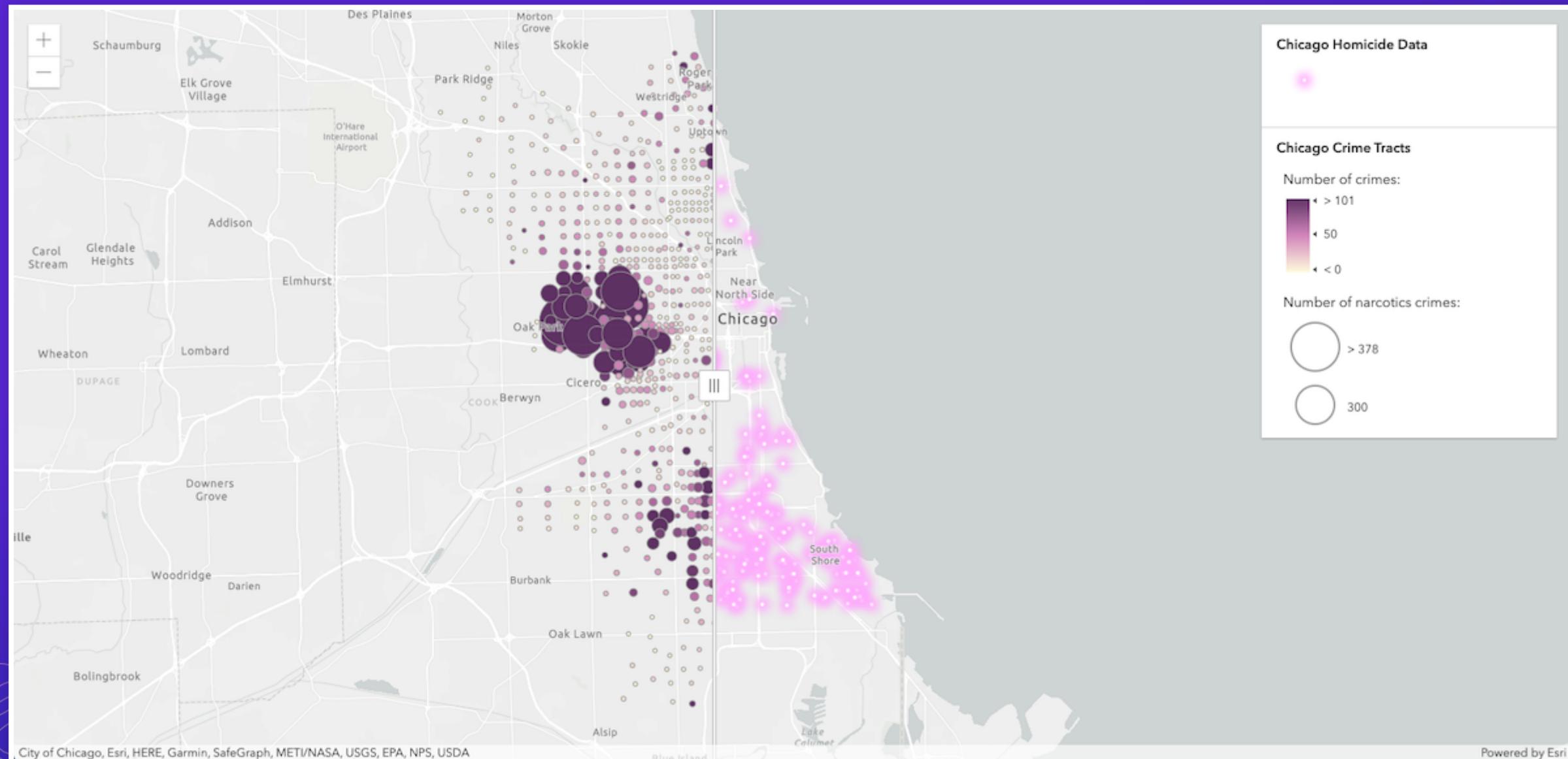
```
view.when(function){
  const featurelayer = map.layers.getItemAt(1);
  // 1. Create the widget
  const legend = new Legend({
    // 2. Specify properties for widget
    view: view,
    layerInfos: [
      {
        layer: featurelayer,
        title: "Name"
      }
    ],
    // 3. Add widget to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

View UI

- Position widgets
 - Add
 - Move
 - Remove



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.

The screenshot shows the 'Support' section of the ArcGIS JavaScript API documentation. At the top, there's a navigation bar with links to Home, Guide, Sample Code, API Reference, Showcase, Support (which is underlined), and Blog. The main content area has a blue header 'Support'. Below it, there's a section titled 'From our Blog' with three cards:

- MAPPING**: Using GeoJSON layers and more in the ArcGIS API for JavaScript by Julie Powell | February 13, 2020. Build web apps with striking visualizations and interactive workflows using a variety of data types with the ArcGIS API for JavaScript.
[Read this article](#)
- MAPPING**: How and why to adjust symbol size by scale in web maps by Kristian Ekenes | January 29, 2020. Though subtle, icon and outline sizes varied by scale will turn a mediocre visualization into a great one.
[Read this article →](#)
- 3D VISUALIZATION & ANALYTICS**: An in depth 3D globe of earthquakes by Raluca Nicola | January 26, 2020. Learn to create a 3D globe of earthquakes with exaggerated depth using ArcGIS API for JavaScript.
[Read this article](#)

On the right side, there's a 'Dedicated Support' section with a link to 'Esri Technical Support'. It says: 'Esri Technical Support is available using live chat, phone or email (with a paid subscription plan)'. Below that is an 'Additional Support' section with links to 'Videos', 'Share Your Ideas', 'Esri Training Courses', and 'Pluralsight'.

