ArcGIS API for JavaScript: What's New

Julie Powell | Noah Sager

SEE WHAT OTHERS CAN'T

Welcome!

Release notes for 4.12

Time

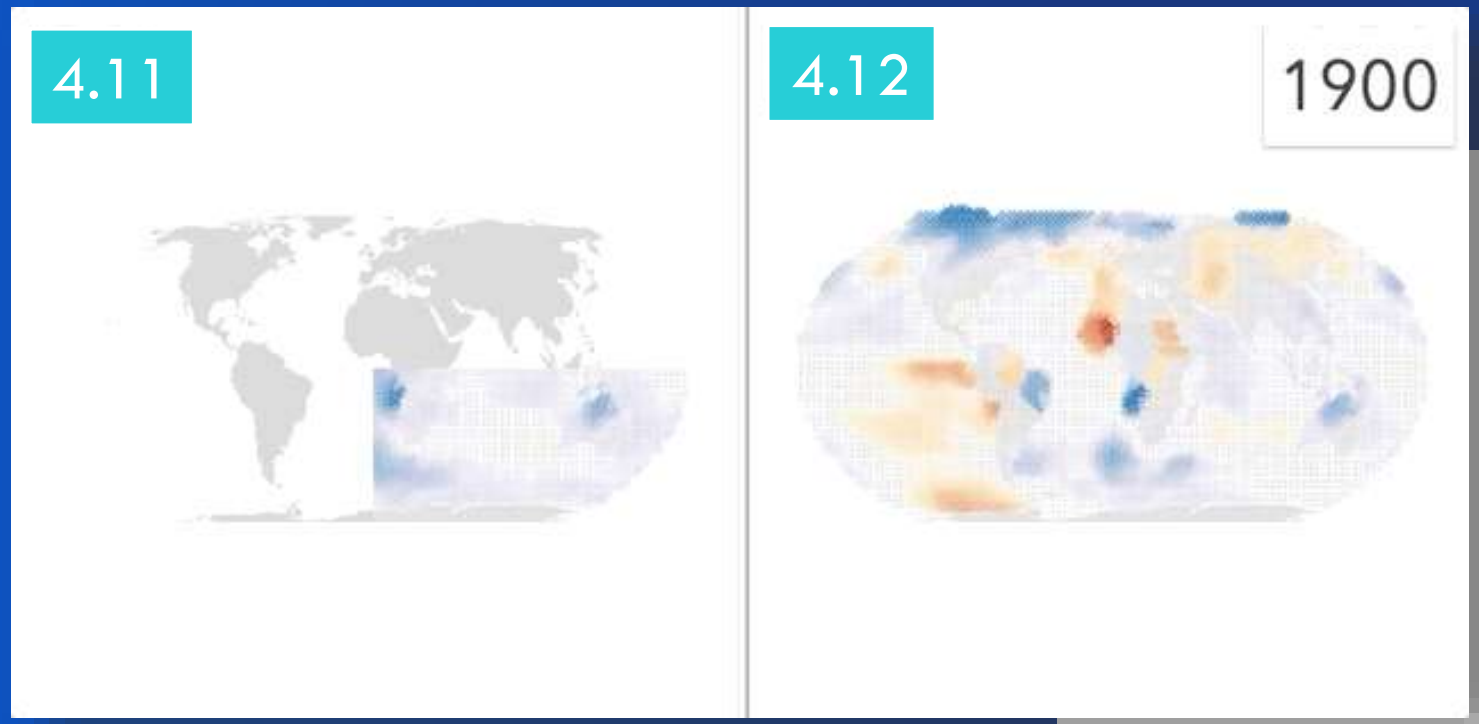
We increased our capabilities to visualize temporal data in both 2D MapViews and 3D SceneViews. We added the TimeInterval class to describe a length of time in different temporal units, which is referenced by time-aware layers and the TimeSlider widget.

The beta version of the TimeSlider widget simplifies time manipulation in your application, and can be configured to update the View's timeExtent, which means all time aware layers will update their contents to conform to this change. You can also use TimeSlider widget to visualize temporal data on the client-side by setting filters or effects on FeatureLayerView, CSVLayerView, and GeoJSONLayerView.

We will continue to add more support for time-awareness. This includes, but is not limited to, adding timeExtent properties on layers that store temporal data, allowing layers to follow their own timeline without having to follow View.timeExtent, and continued improvements and enhancements to the TimeSlider widget.

PERFORMANCE

- Faster loading. >100kb less JS
- Vector tile optimizations
- Faster rendering of line features
- Highly performant FeatureLayers through Feature Tiles
- Fast renderer updates → no flashing!



4.11

4.12

FEATURE [TILE] LAYERS

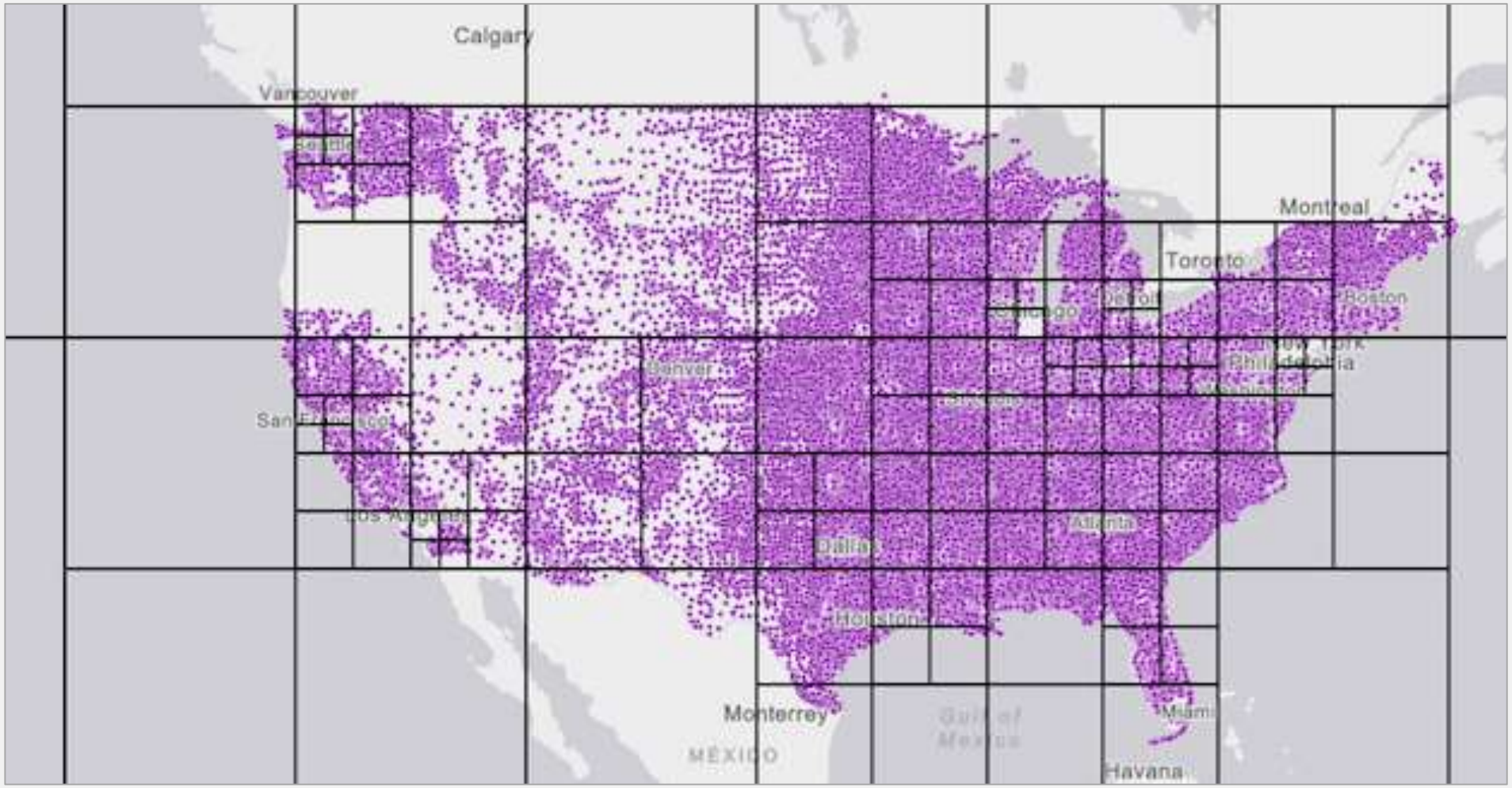
Maximizing performance: a look under the hood

1. Query in an efficient way
2. Minimize size of data delivered to browser
3. Fast rendering

FEATURE [TILE] LAYERS

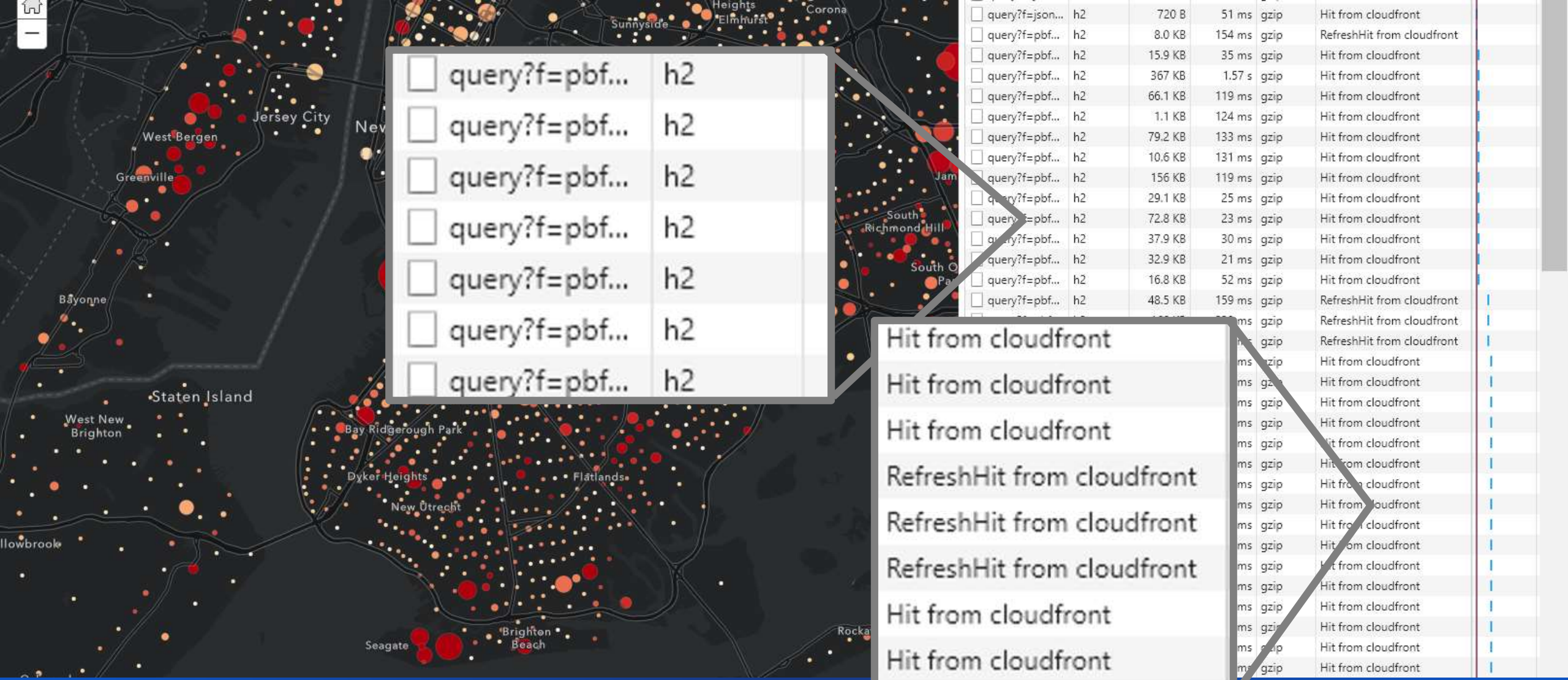
Maximizing performance: a look under the hood

1. Query in an efficient way -> feature tiles & caching
2. Minimize size of data delivered to browser-> *binary* format (PBF) & brotli compression
3. Fast rendering -> WebGL (all layers)



FEATURE FETCH STRATEGY

- Feature tile queries
- Progressive feature tile subdivisions
- Smaller tiles in feature dense areas



HIGH PERFORMANCE FEATURE LAYERS

Efficient querying with feature tiles & caching

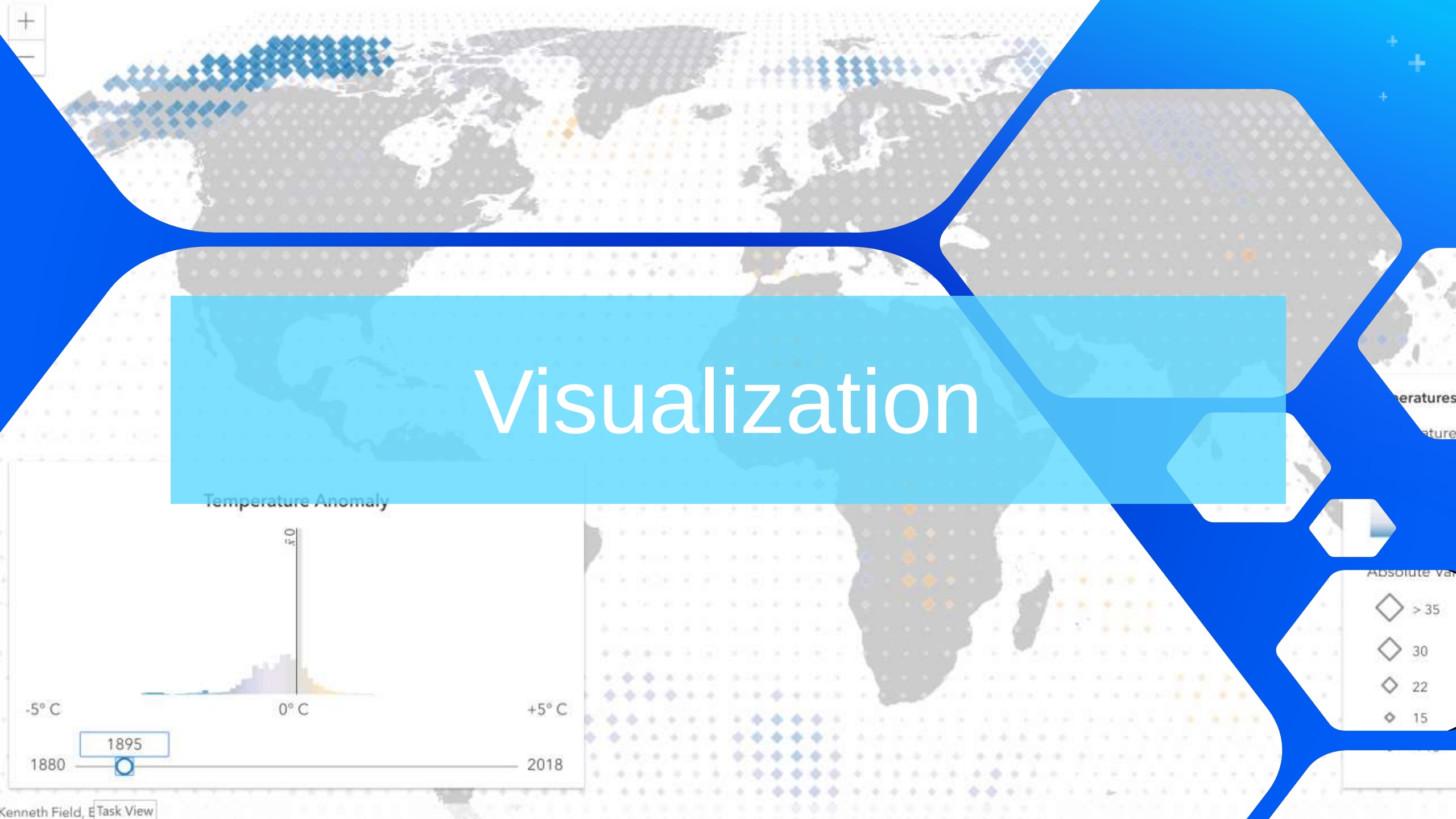
PBF

WebGL

PERFORMANCE

Improved integrated mesh performance
Fast feature layers

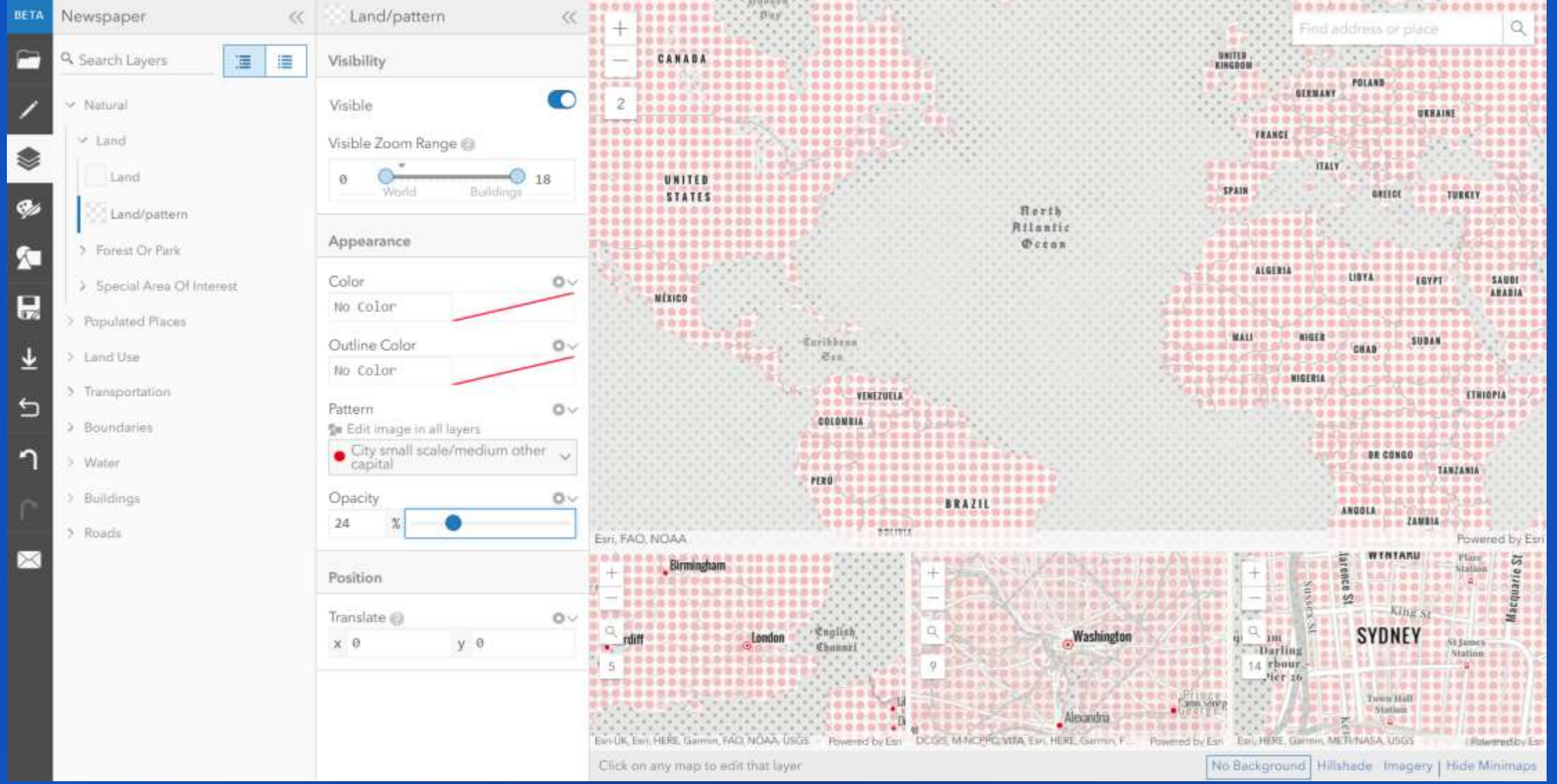
Visualization



VECTOR TILE LAYERS

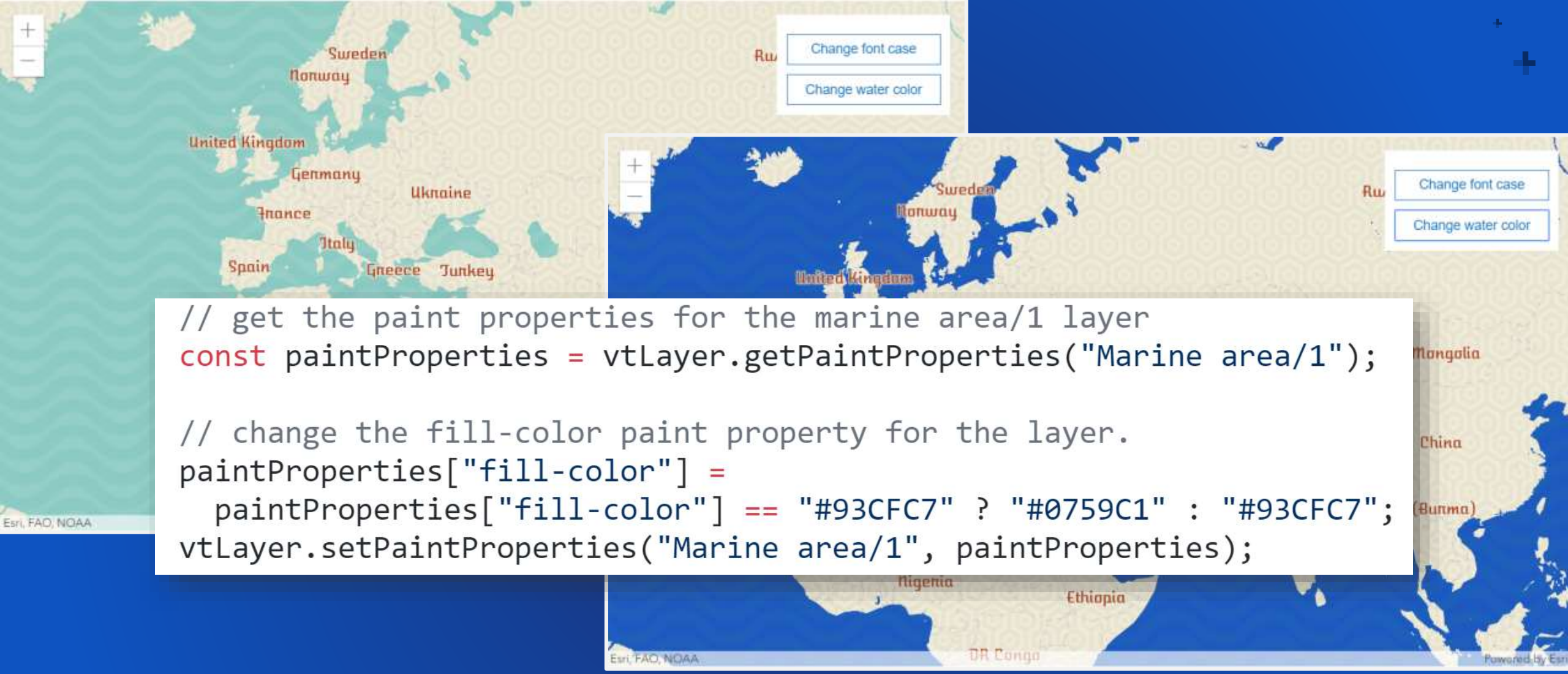
Style esri's basemaps

Or create your own



VECTOR TILE LAYERS

Style the map ahead of time with the Vector Tile Style Editor



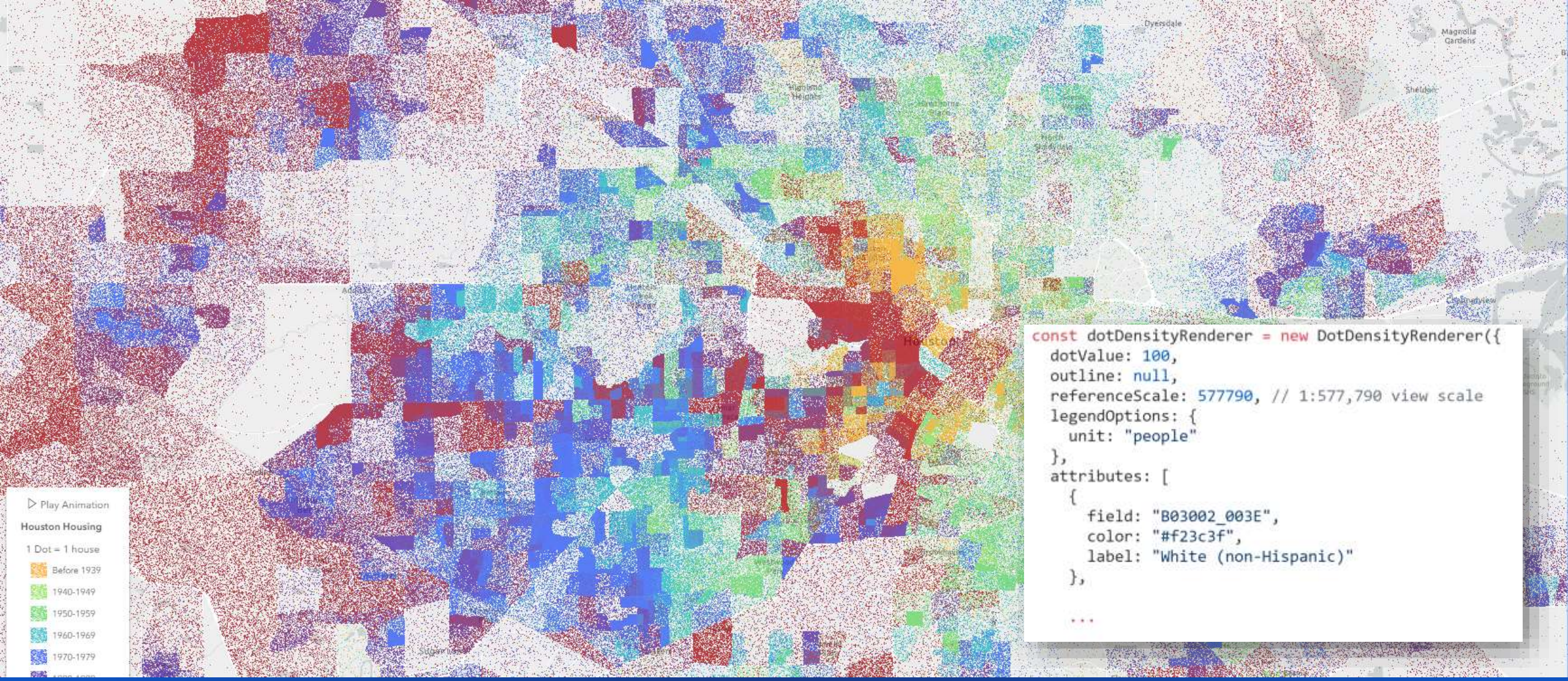

```
// get the paint properties for the marine area/1 layer
const paintProperties = vtLayer.getPaintProperties("Marine area/1");

// change the fill-color paint property for the layer.
paintProperties["fill-color"] =
  paintProperties["fill-color"] == "#93CFC7" ? "#0759C1" : "#93CFC7";
vtLayer.setPaintProperties("Marine area/1", paintProperties);
```

VECTOR TILE LAYERS

Load the style of your choice, or
Style the layer in code.
Option to update it without a reload (4.10)

DATA-DRIVEN STYLING

Where?
What?
When?
How much?



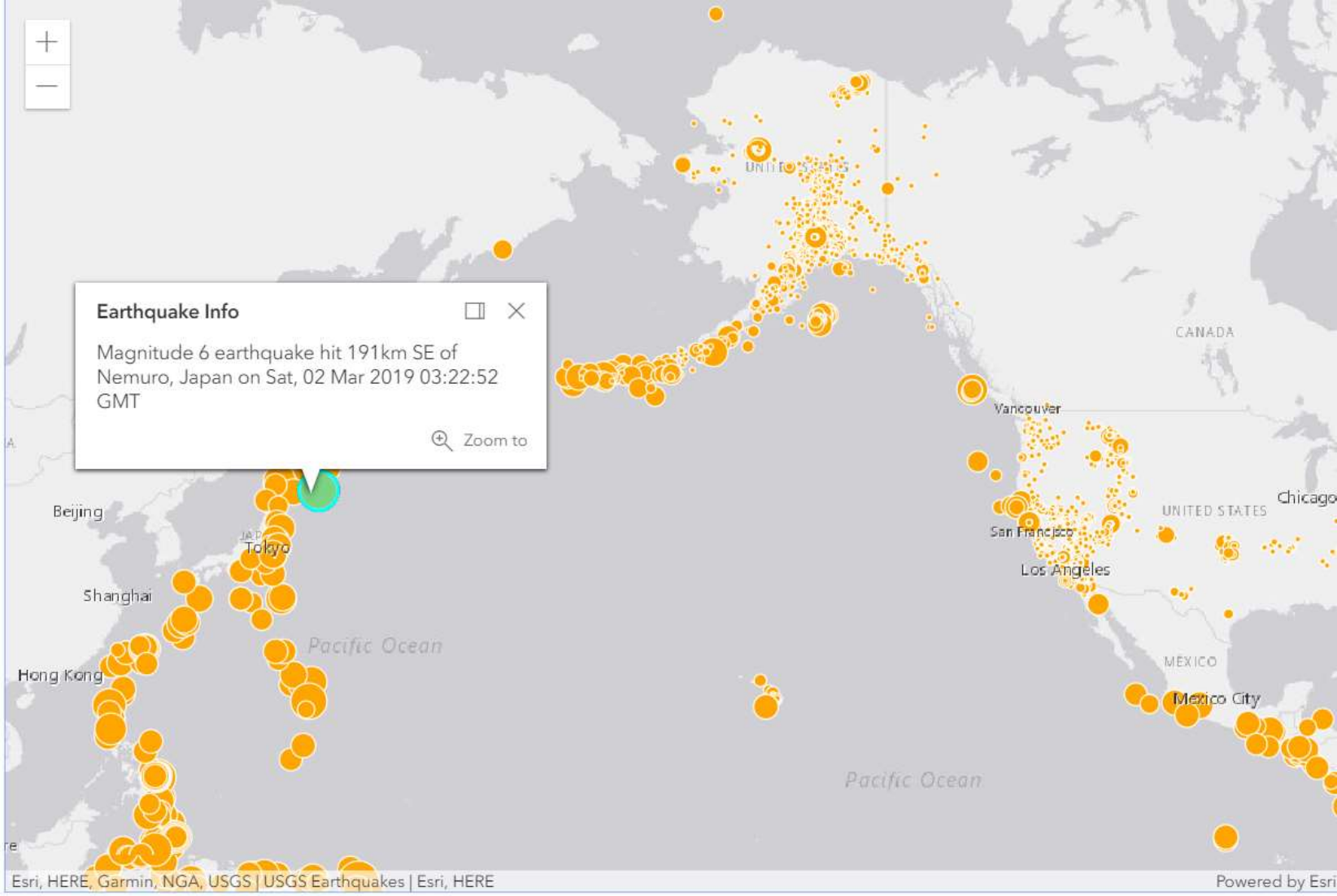

```
const dotDensityRenderer = new DotDensityRenderer({
  dotValue: 100,
  outline: null,
  referenceScale: 577790, // 1:577,790 view scale
  legendOptions: {
    unit: "people"
  },
  attributes: [
    {
      field: "B03002_003E",
      color: "#f23c3f",
      label: "White (non-Hispanic)"
    },

    ...
```

DOT DENSITY

Randomly drawn dots to represent a field value.

Configure how much each dot represents

```
const template = {
  title: "Earthquake Info",
  content: "Magnitude {mag} {type} hit {place} on
    {time:DateString}"
};

const renderer = {
  type: "simple",
  field: "mag",
  symbol: {
    type: "simple-marker",
    color: "orange",
    outline: {
      color: "white"
    }
  },
  visualVariables: [
    {
      type: "size",
      field: "mag",
      stops: [
        {
          value: 2.5,
          size: "4px"
        },
        {
          value: 8,
          size: "40px"
        }
      ]
    }
  ]
};

const geojsonLayer = new GeoJSONLayer({
  url: url,
  copyright: "USGS Earthquakes",
  popupTemplate: template,
  renderer: renderer //optional
});

const map = new Map({
```

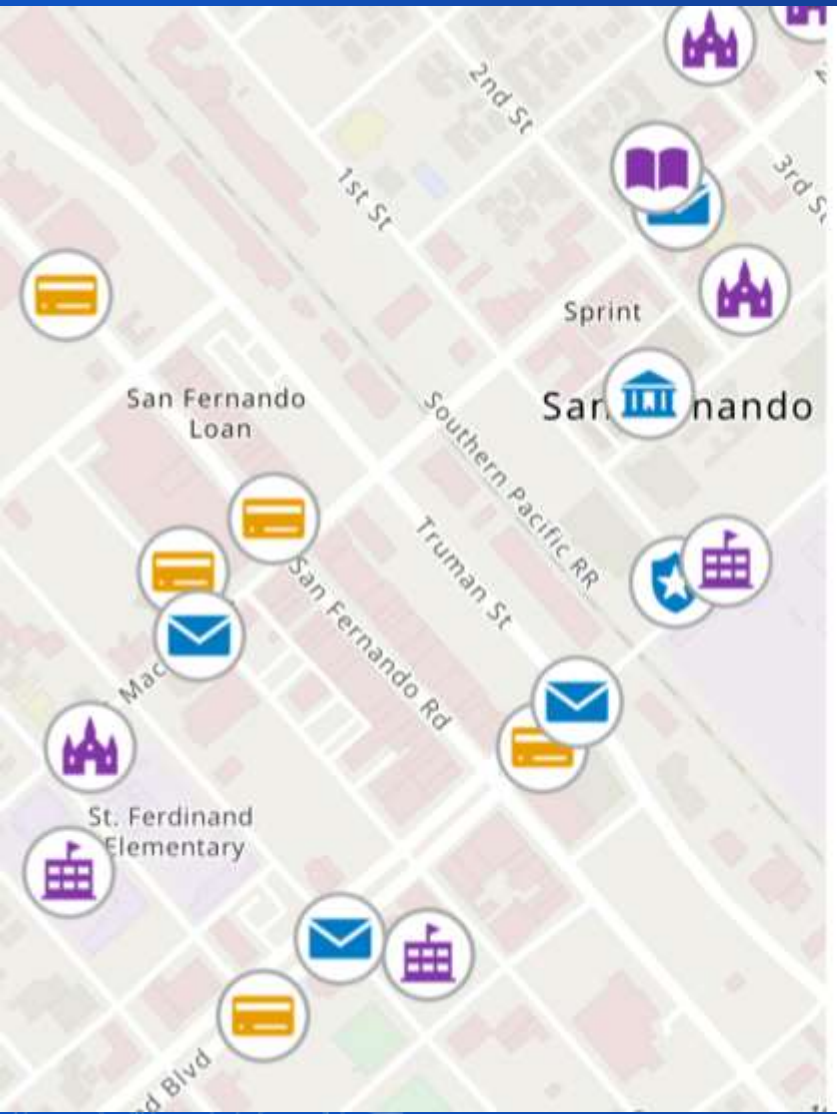


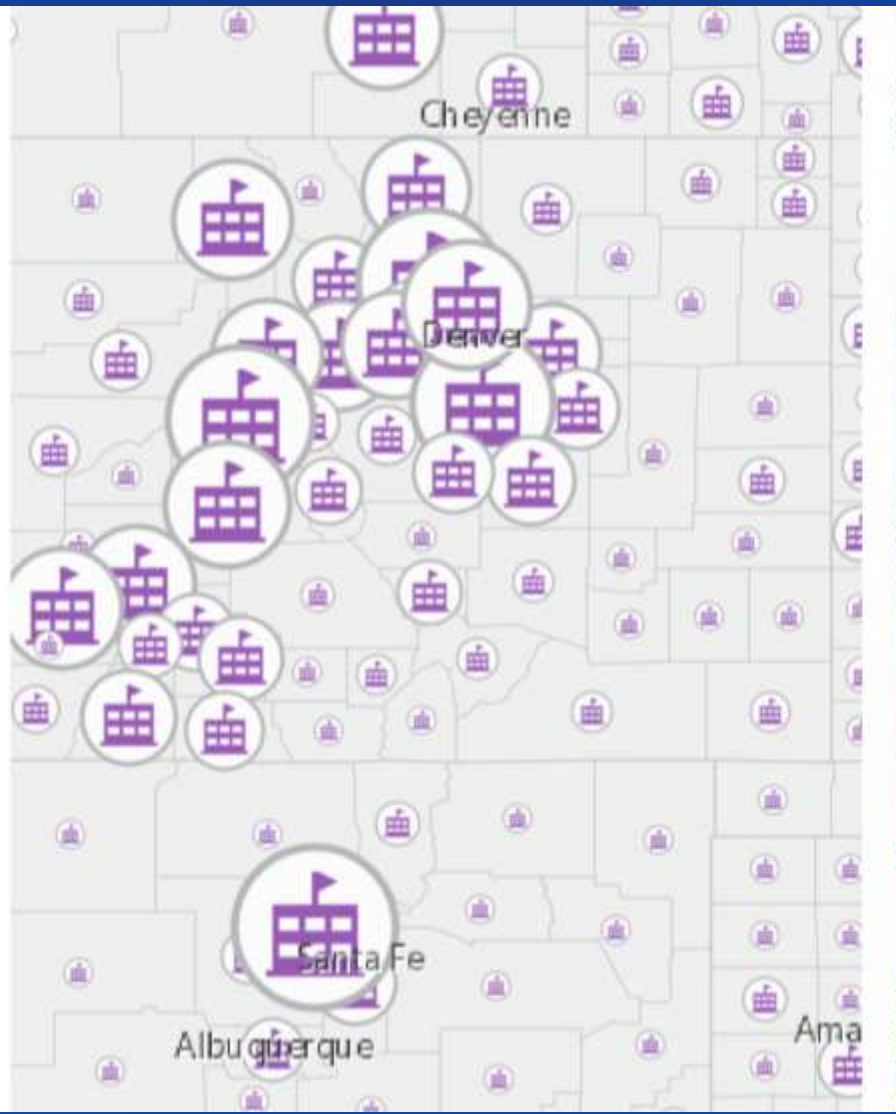
GEOJSON LAYER

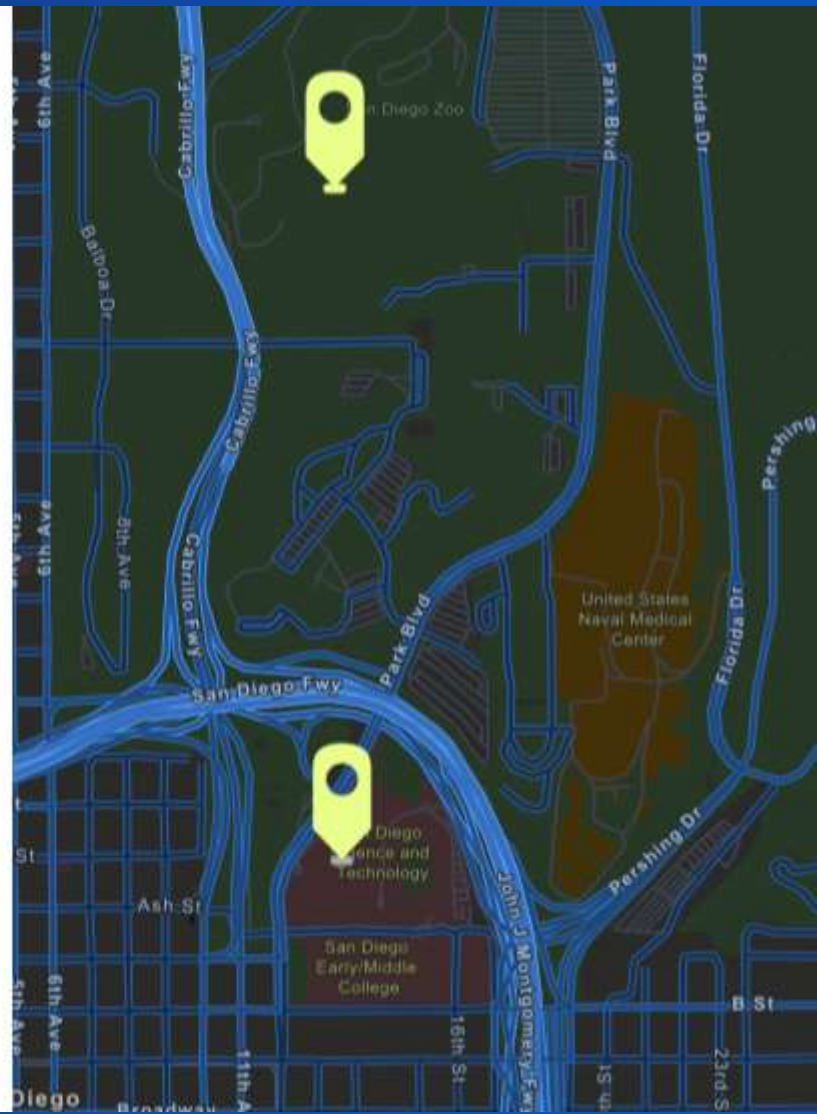
Style & interact like a feature layer

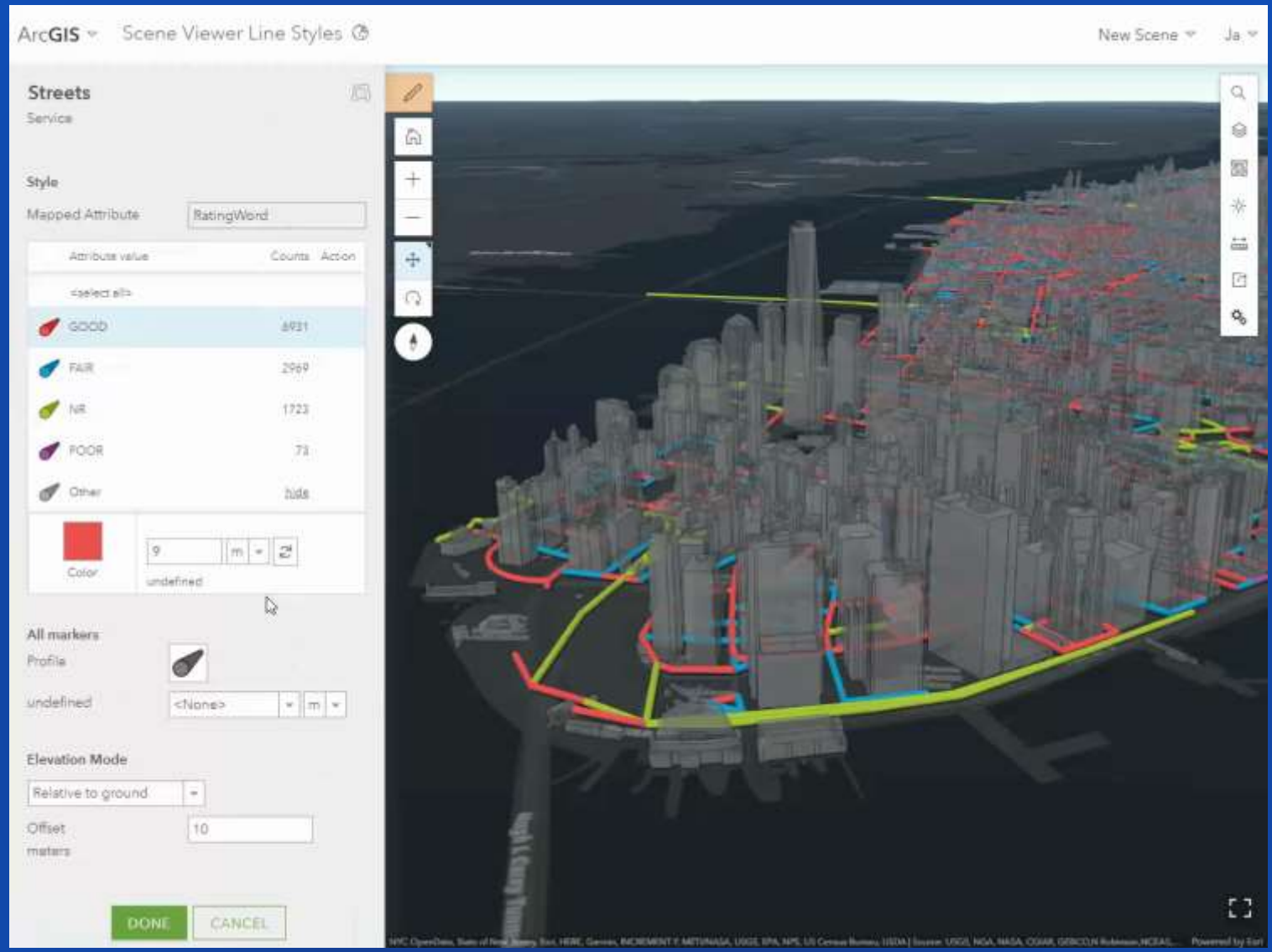
VECTOR MARKER SYMBOLS

More than 100 new 2D web style symbols
CIM symbols







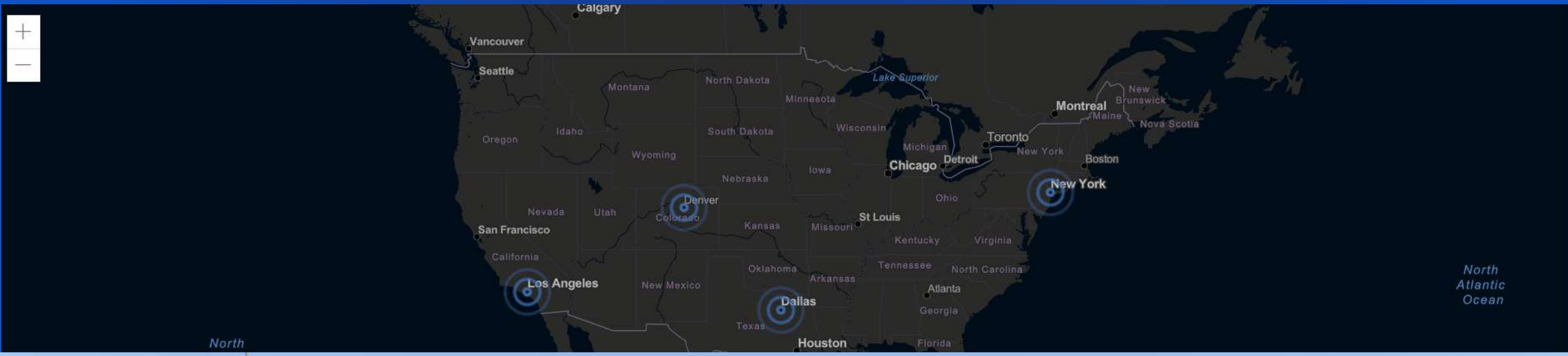


SMART MAPPING

Parity with 3.x, plus more. i.e.:

- Scale-driven outline thickness
- Icon sizing based on scale

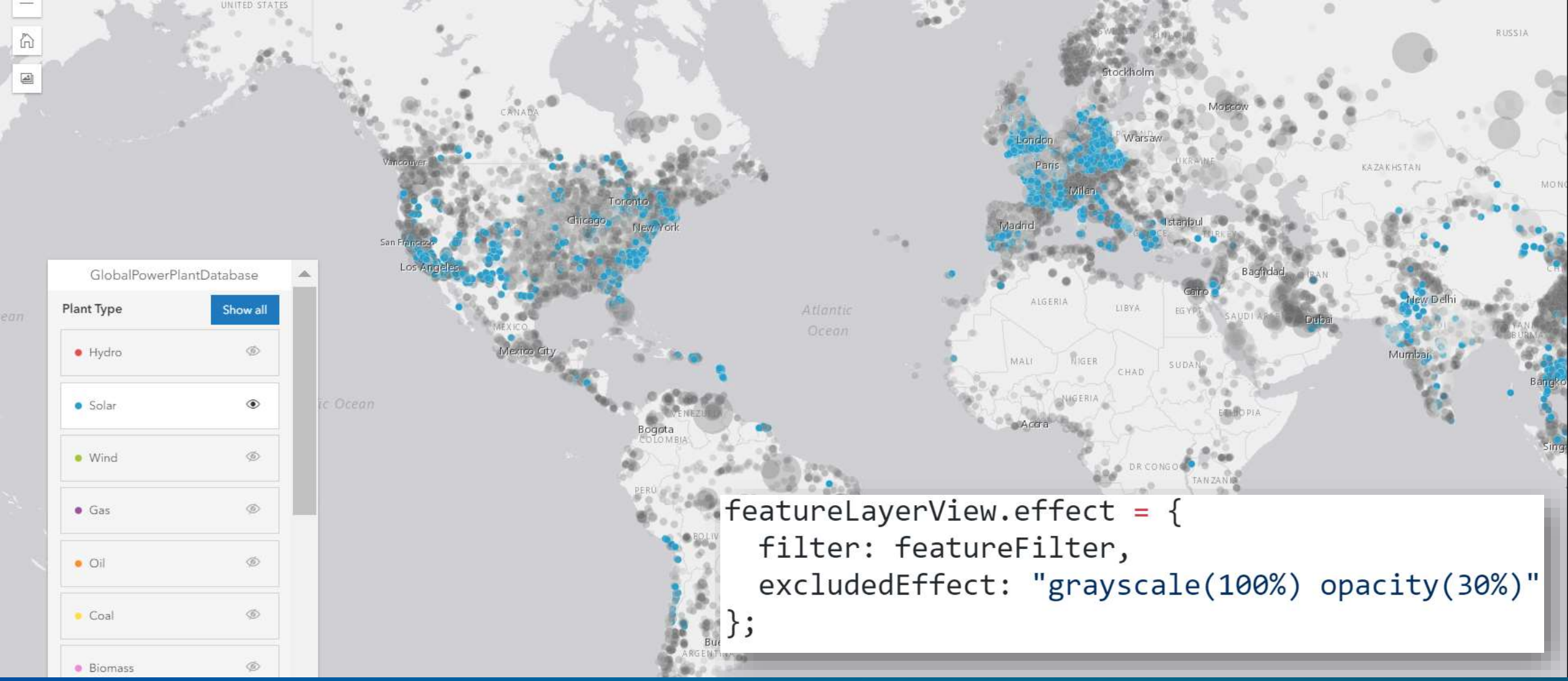
Create custom WebGL layer views



INTERACTIVITY

Client-side

- Querying
- Filtering
- Statistics
- Geometric operations

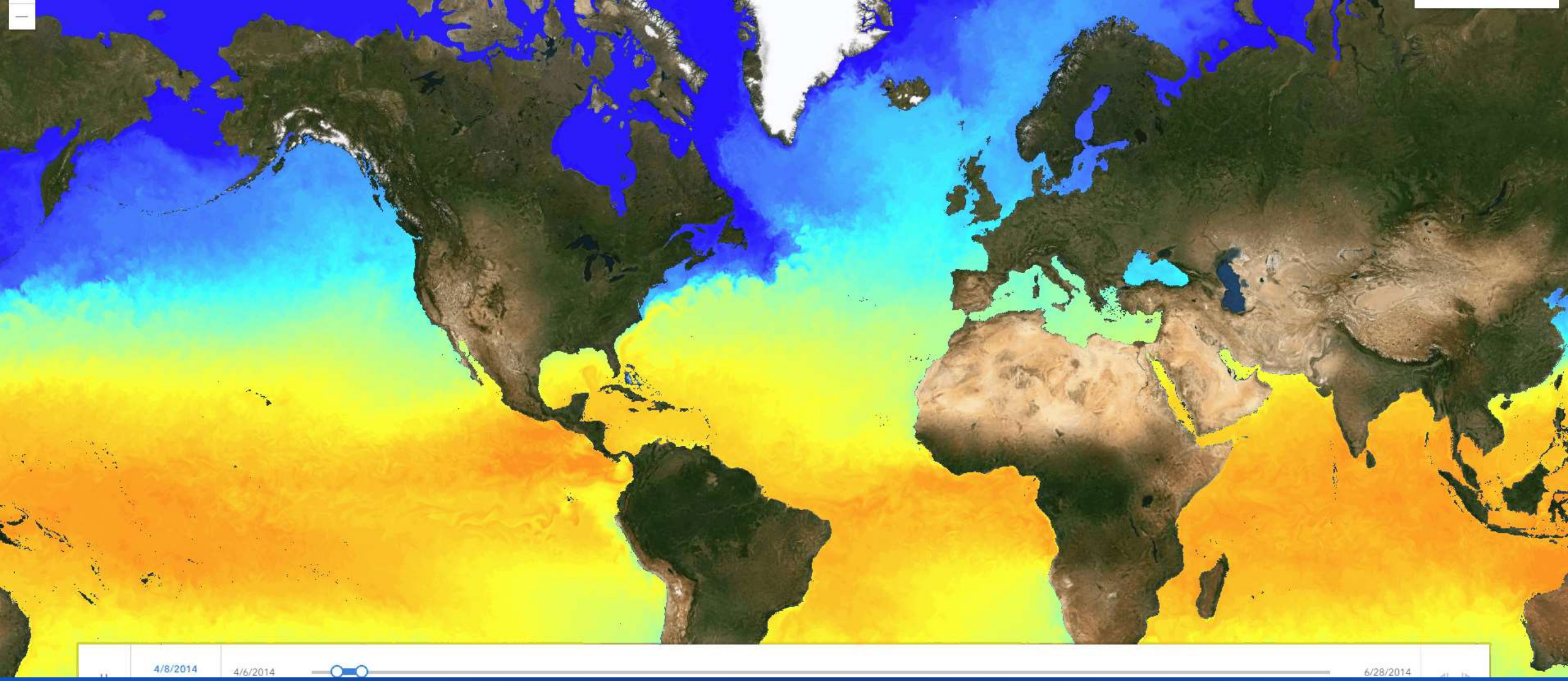


FILTERING

Client-side

Decide how to style feature within filter, and outside filter.

Time

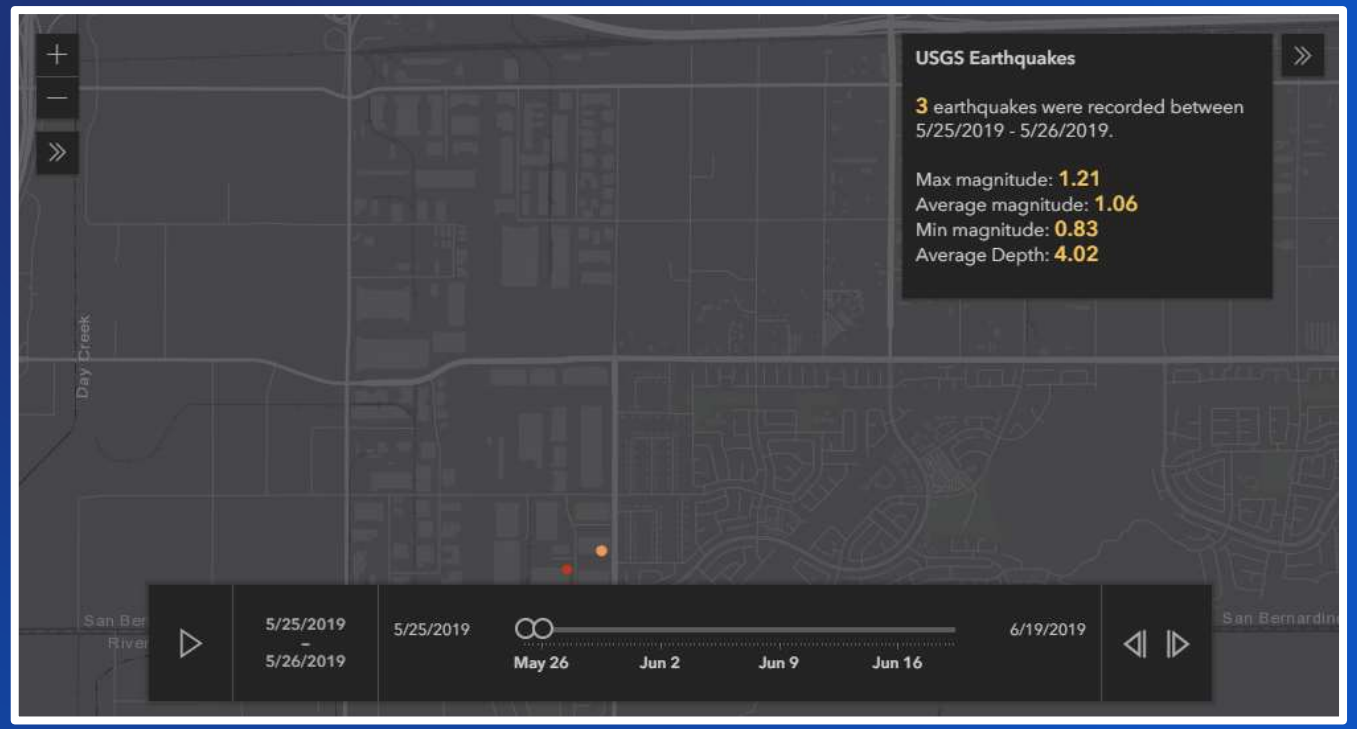


TIME

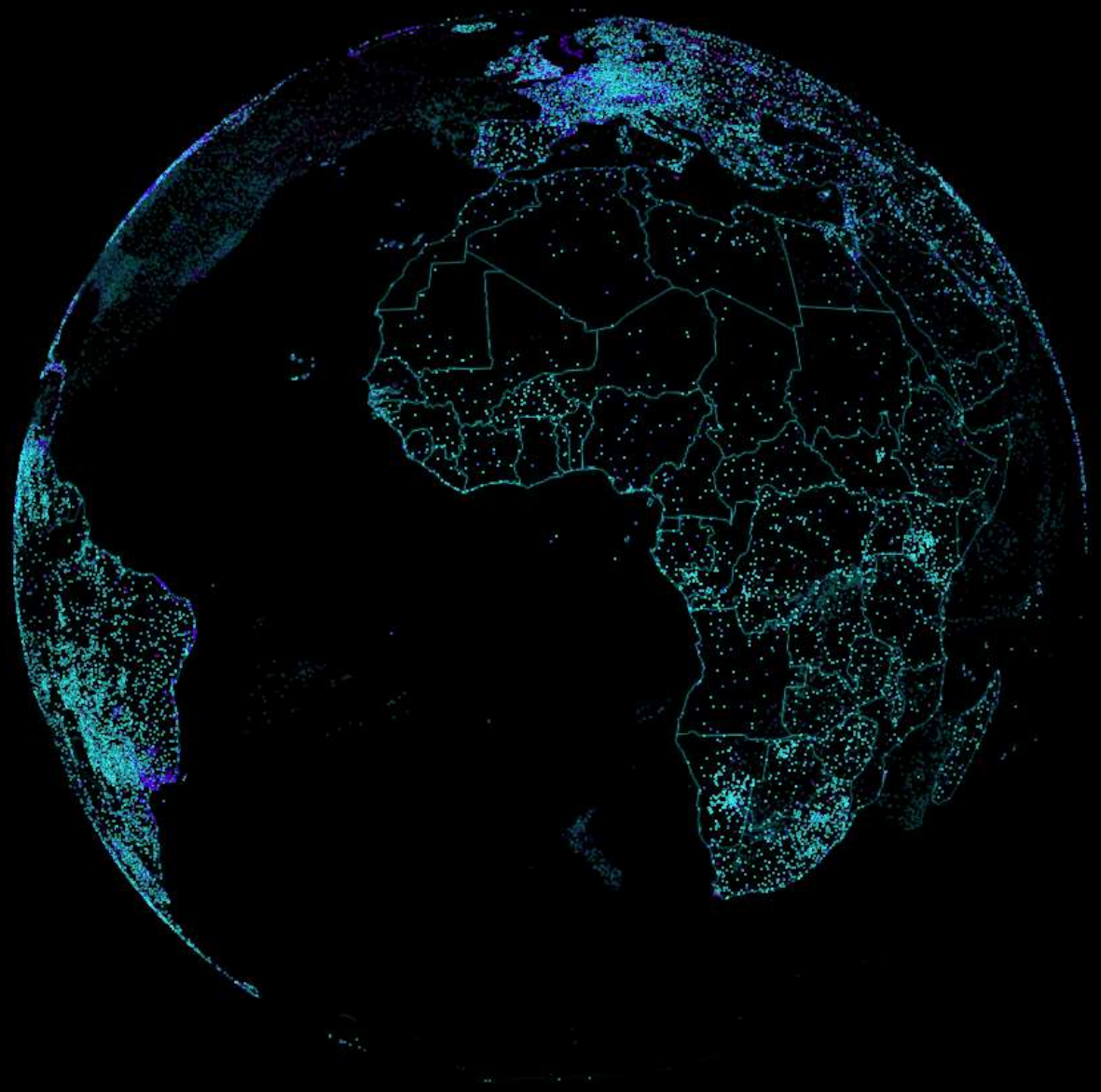
Time aware layers and views
Time slider widget

TIME

1. timeInfo on the layer
2. timeExtent on the view
3. TimeSlider widget

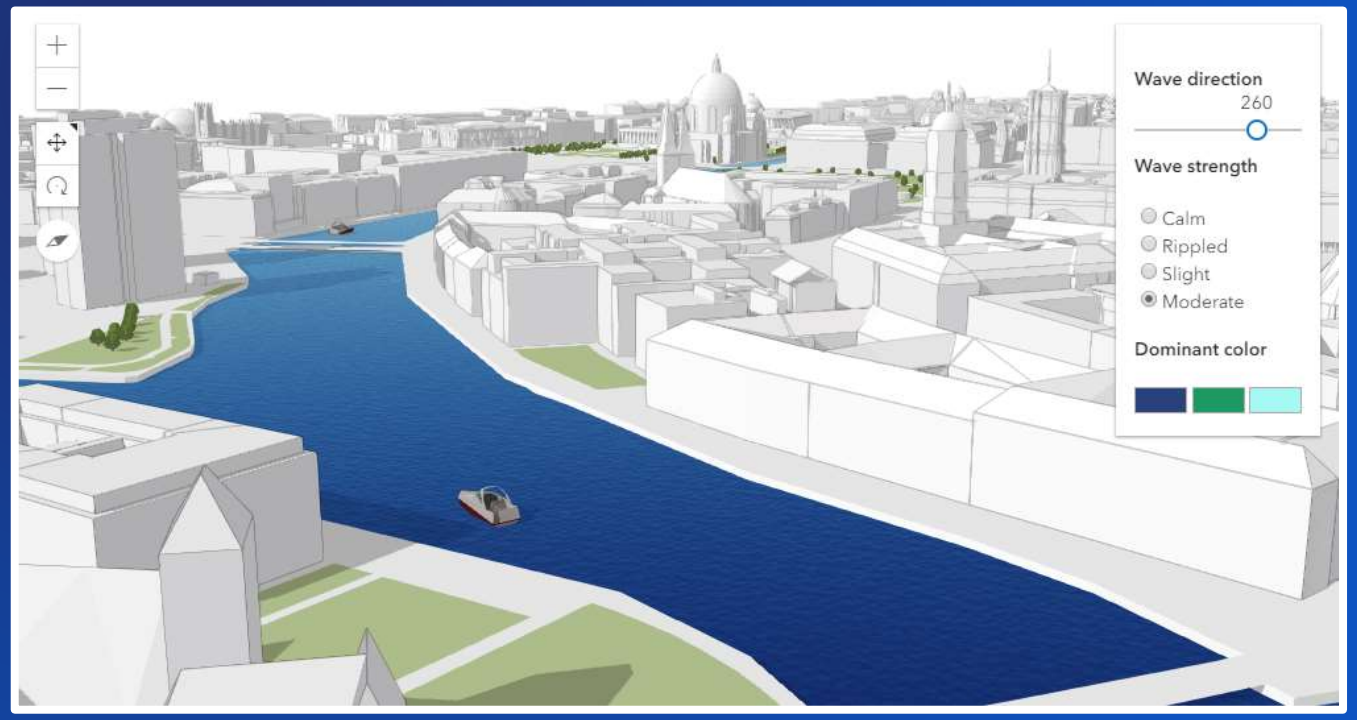


3D



REALISTIC RENDERING

1. **WaterSymbol3DLayer**
2. **Esri Web Style Symbols**
3. **glTF models**

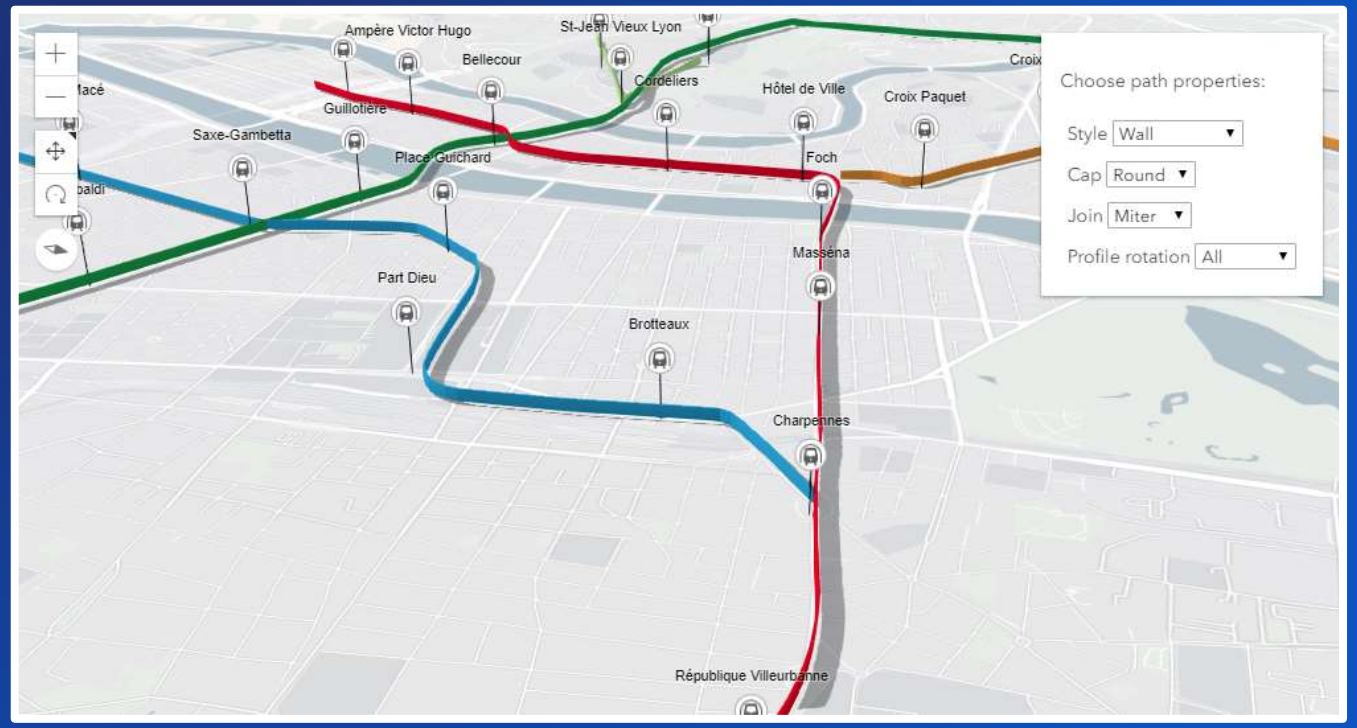


BUILDING SCENE LAYER

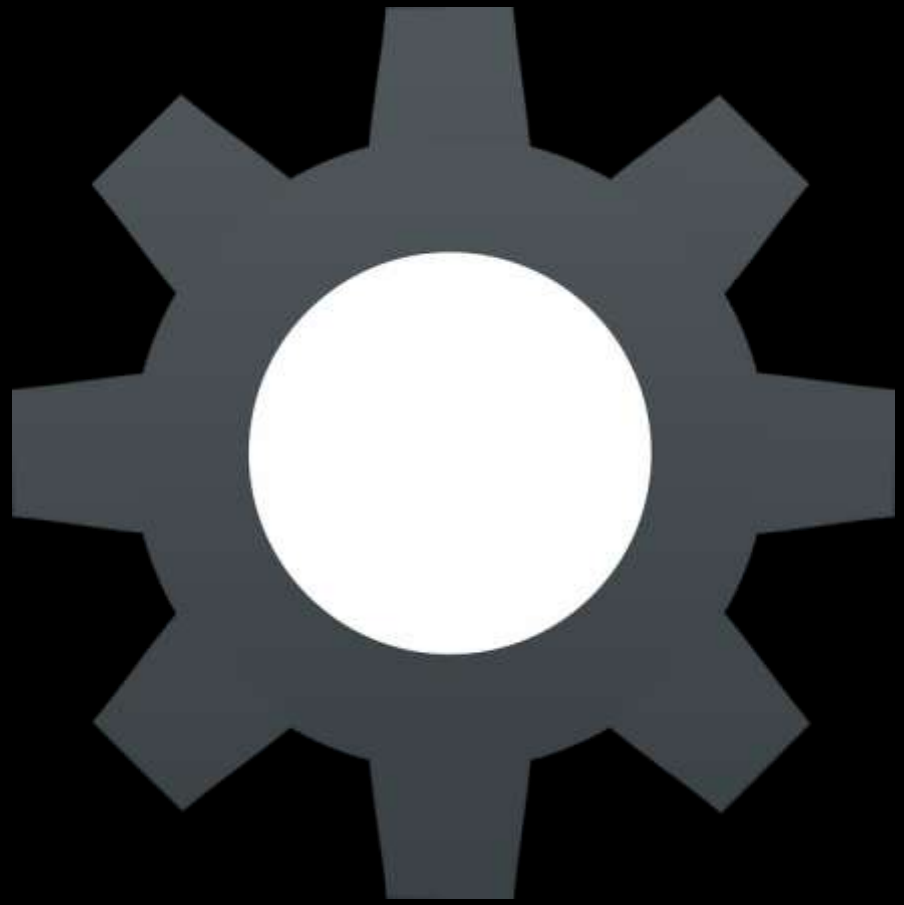
1. **Detailed exteriors / interiors**
2. **Slice widget**
3. **Filtering**

LINE SYMBOLS

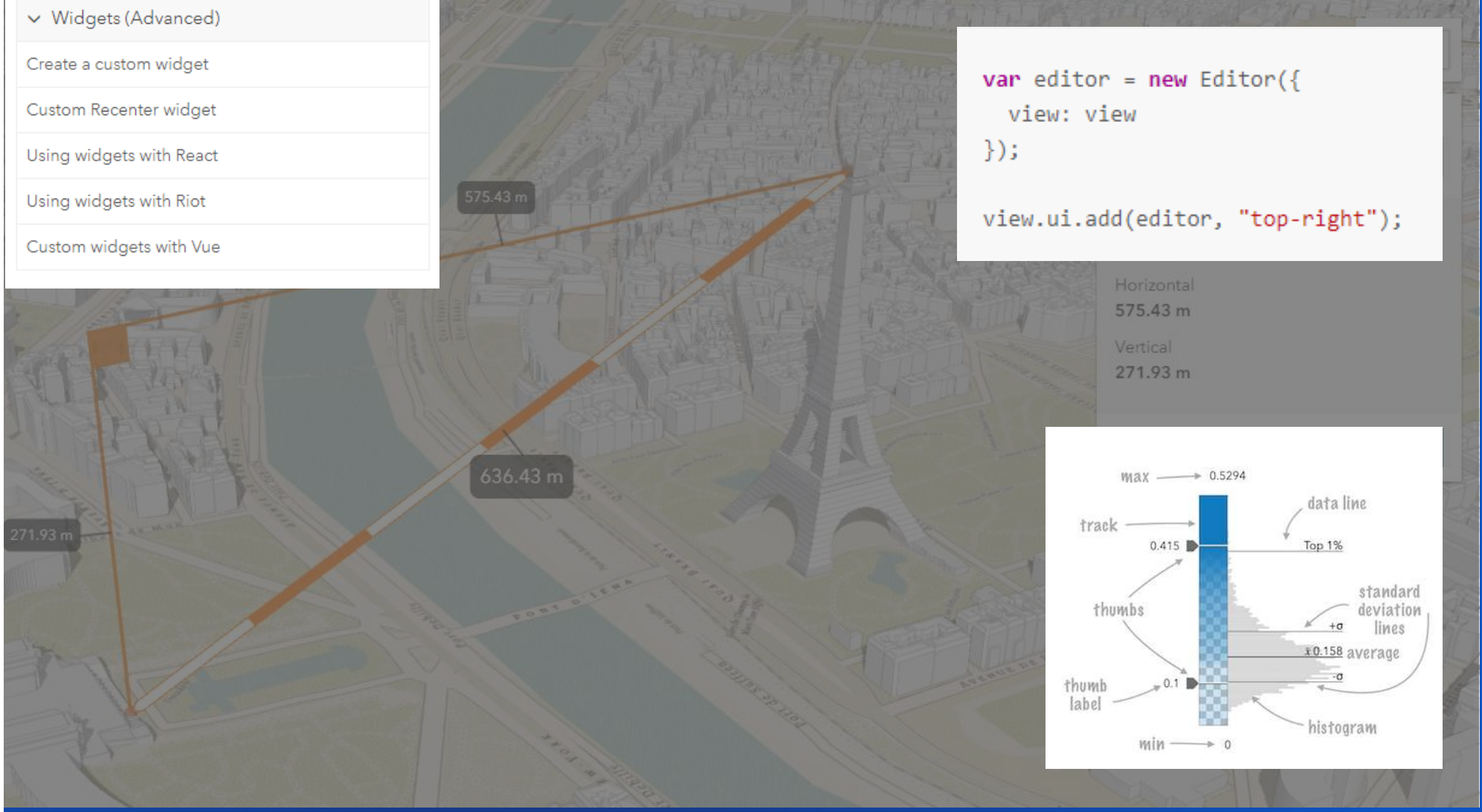
1. Round tube
2. Square tube
3. Wall
4. Strip



Widgets



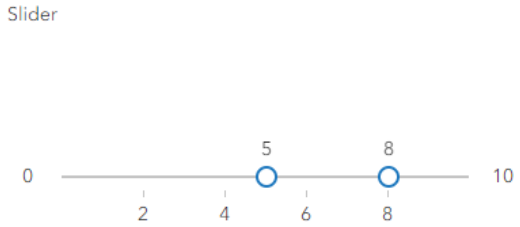
- Widgets
  - BasemapGallery widget
  - Bookmarks widget
  - CoordinateConversion widget
  - CoordinateConversion widget - custom formats
  - Directions widget
  - LayerList widget
  - LayerList widget with actions
  - Legend widget
  - Add a Legend to LayerList
  - Legend widget card style
  - Locate button
  - Measurement in 2D
  - Measurement in 3D
  - BuildingSceneLayer with Slice widget
  - Print widget
  - Track current location
  - Track widget simulation
  - Expand widget
  - Feature widget
  - Using the view's UI
  - Responsive widgets
  - Responsive apps using CSS
  - TimeSlider Widget
- Widgets (Advanced)
  - Create a custom widget
  - Custom Recenter widget
  - Using widgets with React
  - Using widgets with Riot
  - Custom widgets with Vue

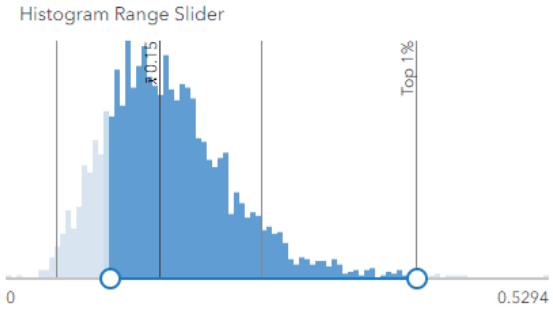


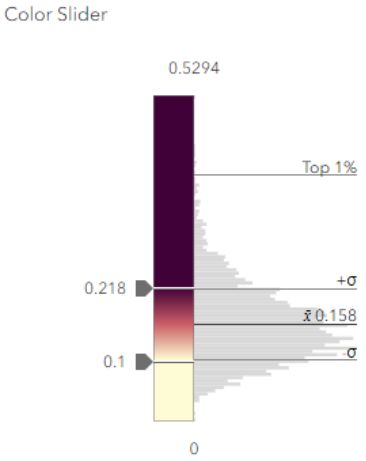

```
var editor = new Editor({
  view: view
});

view.ui.add(editor, "top-right");
```

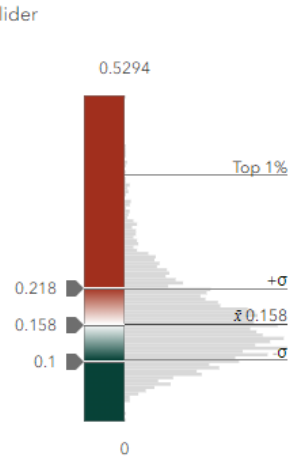

BUILD YOUR UI

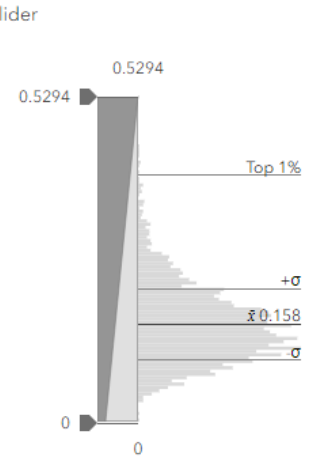
Collection of widgets
Customizable
Easy placement

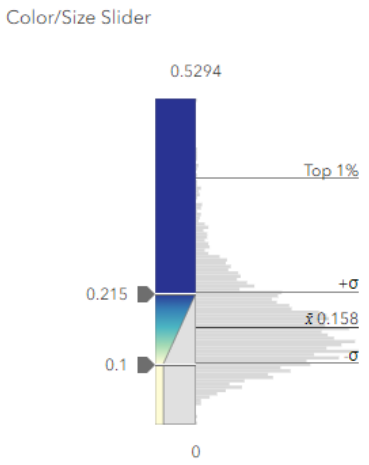


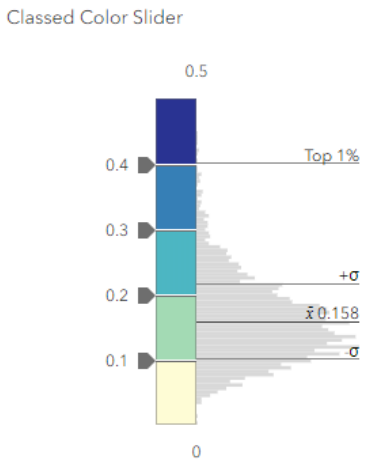


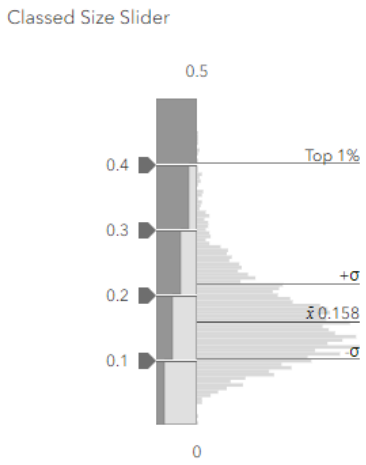


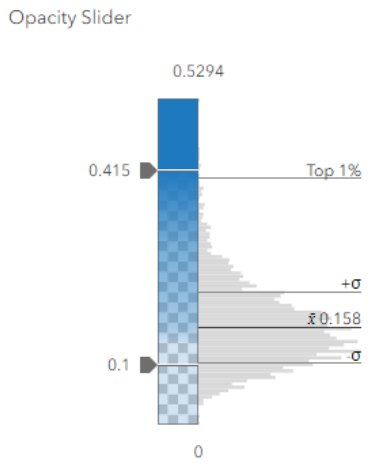


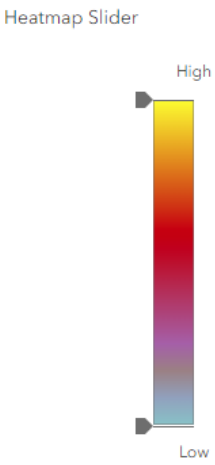










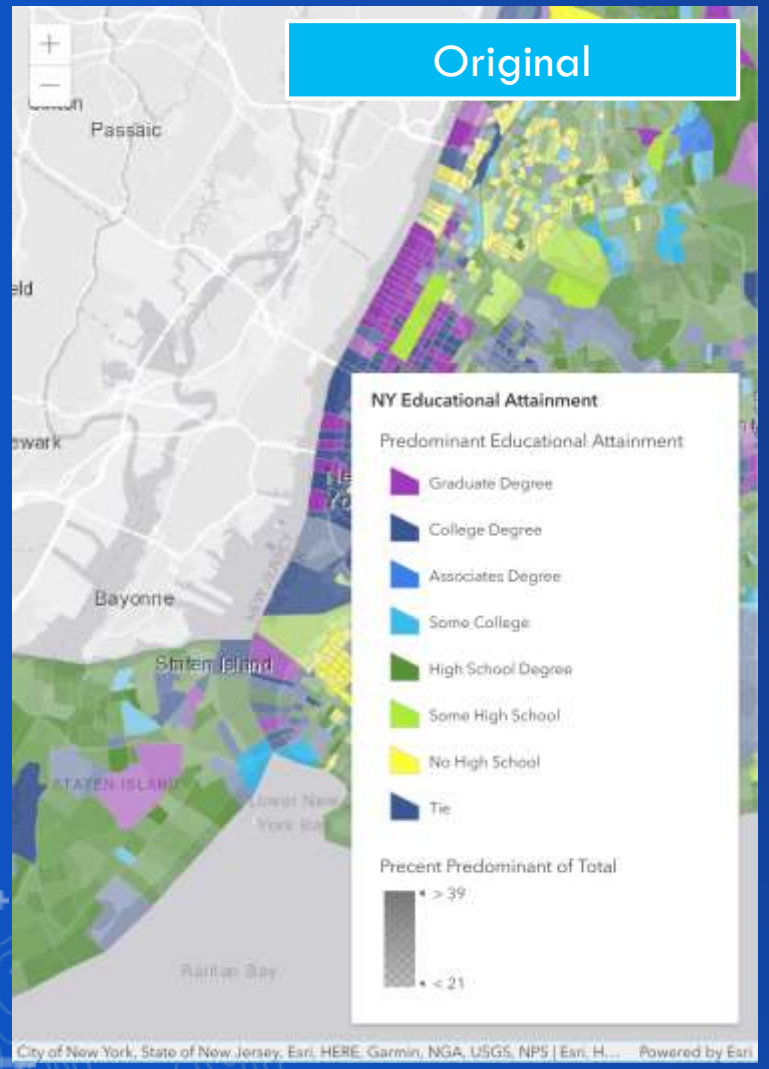


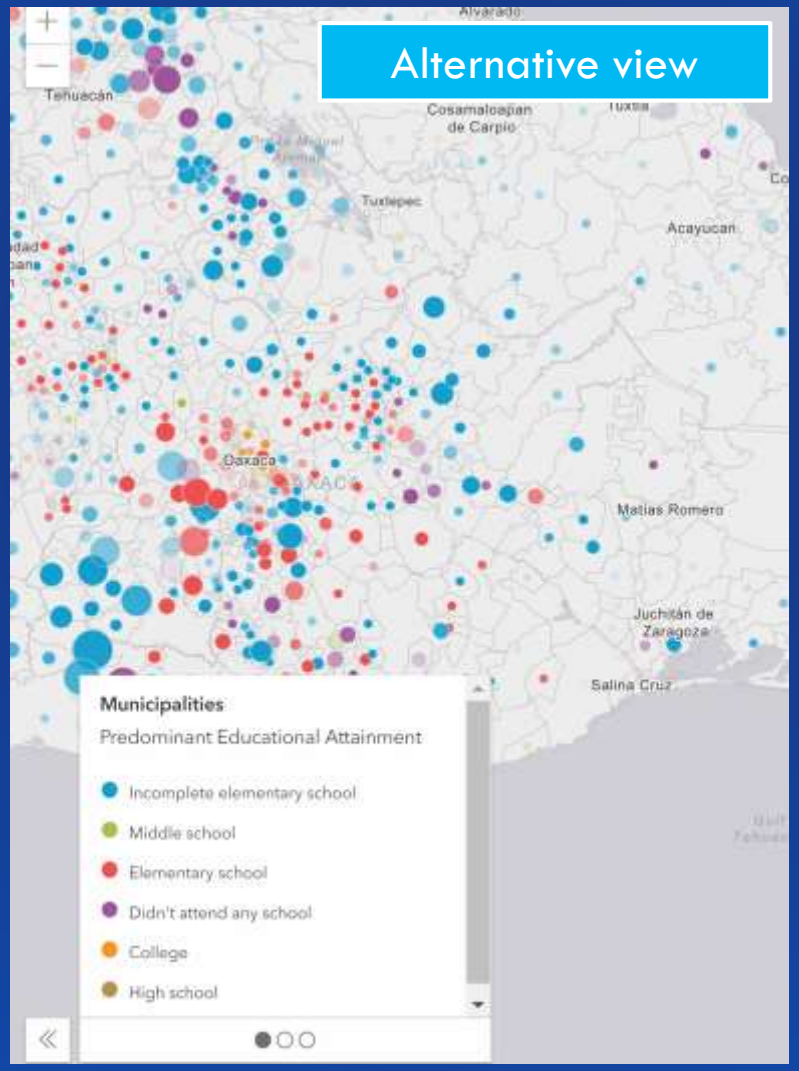
SLIDERS

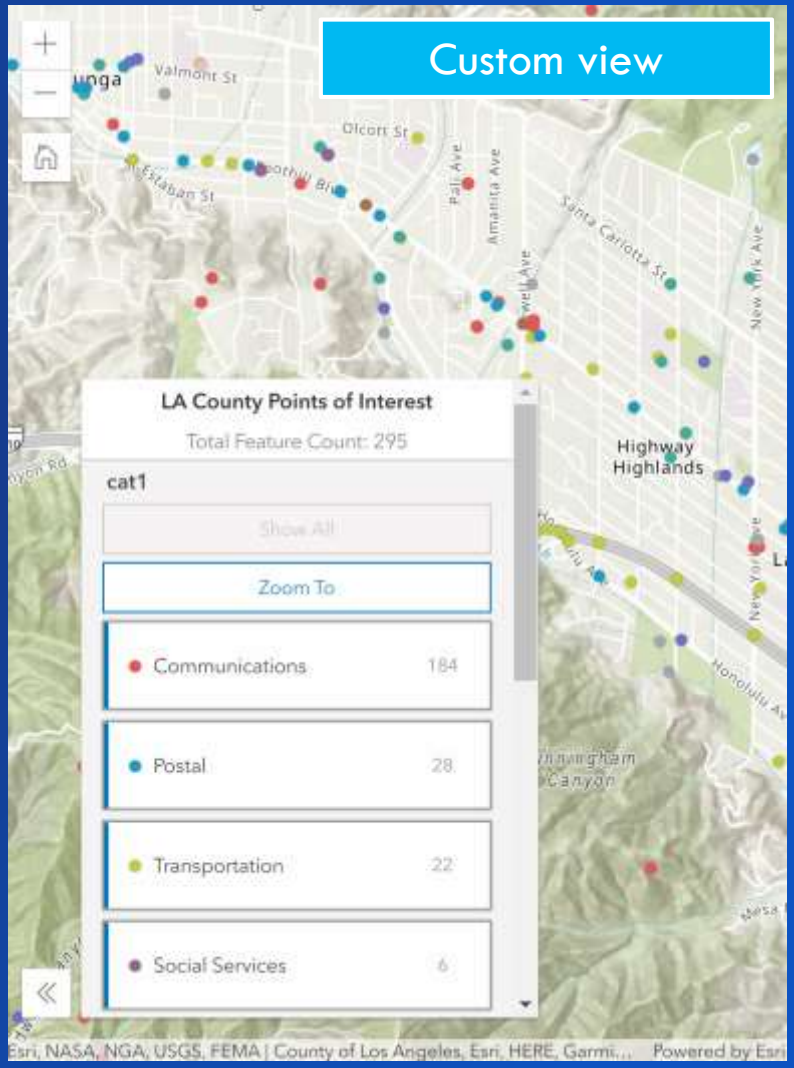
- Generic slider class
- Useful for any kind of numeric or range of values

WIDGETS

- Use OOB or customize/extend
- Widget view / view model architecture
- On deck: Layer swipe







Widget development

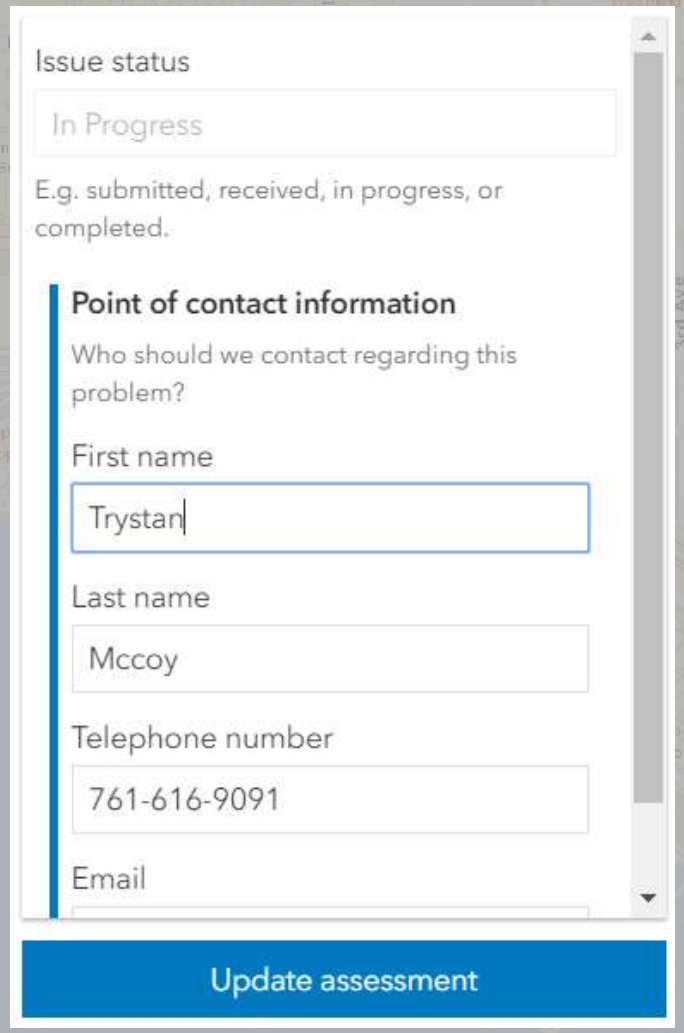
Widgets are reusable user-interface components and are key to providing a rich user experience. The ArcGIS for JavaScript API provides a set of ready-to-use widgets. Beginning with version 4.2, it also provides a foundation for you to create custom widgets.

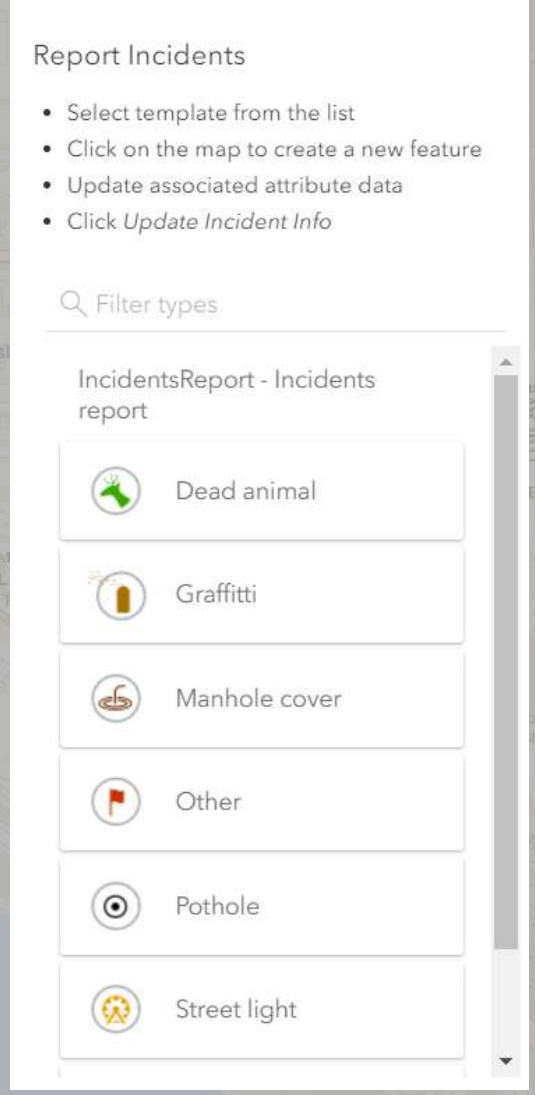
ViewModel pattern

There are two parts to working with the widget framework. These are: 1) the widget, and 2) the widget's ViewModel. The Widget (i.e. View), part is responsible for handling the User Interface (UI) of the widget, meaning how the widget displays and handles user interaction via the DOM. The ViewModel part is responsible for the underlying functionality of the widget, or rather, its business logic.

SKETCHING

- Draw graphics on the graphics layer
- Use the OOB widget





EDITING

Form-based editing
Feature templates
Create & update geometry

Editor | API Reference

Search API Reference

- esri
- esri/core
- esri/core/accessorSupport
- esri/core/workers
- esri/geometry
- esri/geometry/support
- esri/identity
- esri/layers
- esri/layers/buildingSublayers

Editor

Constructors | Properties | Methods | Type definitions

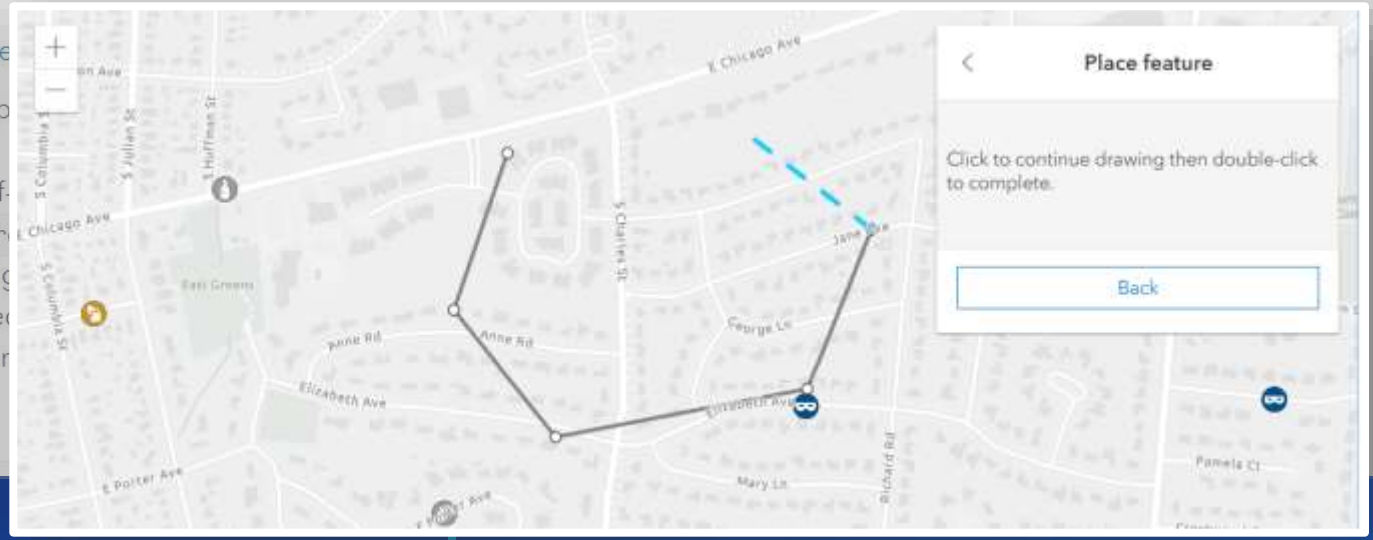
```
require(["esri/widgets/Editor"], function(Editor) { /* code goes here */ });
```

Class: esri/widgets/Editor

Inheritance: Editor → Widge

Since: ArcGIS API for JavaScrip

This widget provides an out-of- has two different Workflow, cr editable feature layer. The widg editable, the layers can be use editingConfig property. This pr layers.



EDITOR WIDGET

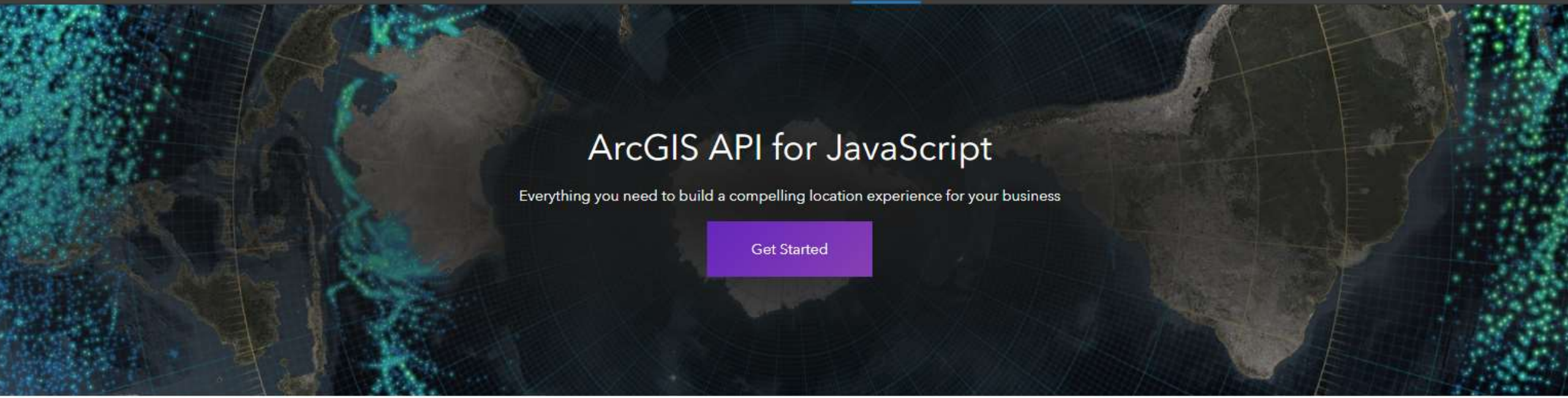
Widget that brings together the editing experience

Use client-side geometric operations

Buffer, cut, merge, validation workflows, etc…

Method Overview

Name	Return Type	Summary		Object
`buffer()`	`Polygon` | `Polygon[]`	Creates planar (or Euclidean) buffer polygons at a specified distance around the input geometries.	more details	geometryEngine
`clip()`	`Geometry`	Calculates the clipped geometry from a target geometry by an envelope.	more details	geometryEngine
`contains()`	`Boolean`	Indicates if one geometry contains another geometry.	more details	geometryEngine
`convexHull()`	`Geometry` | `Geometry[]`	Calculates the convex hull of the input geometry.	more details	geometryEngine
`crosses()`	`Boolean`	Indicates if one geometry crosses another geometry.	more details	geometryEngine
`cut()`	`Geometry[]`	Split the input Polyline or Polygon where it crosses a cutting Polyline.	more details	geometryEngine
`densify()`	`Geometry`	Densify geometries by plotting points between existing vertices.	more details	geometryEngine
`difference()`	`Geometry` | `Geometry[]`	Creates the difference of two geometries.	more details	geometryEngine
`disjoint()`	`Boolean`	Indicates if one geometry is disjoint (doesn't intersect in any way) with another geometry.	more details	geometryEngine



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.

Sample Code

Get code samples for mapping, visualization, and spatial analysis.

API Reference

Documentation for all ArcGIS API for JavaScript classes, methods, and properties.

Showcase

See how to combine functionality into interactive and compelling applications.

Version 4.12 · July 2019 · Looking for v3.29?

Get the API

What's new

Licensing

Tooling

Using Frameworks

The ArcGIS API for JavaScript has all the tools you would need to build fully scalable and effective applications. However, you may want to utilize another framework's specific capabilities, or leverage your in house expertise in a particular framework.

Frameworks and Libraries

There are many examples of integrating the ArcGIS API for JavaScript with popular frameworks such as React, Angular, Vue, Ember, and many others. Some frameworks and libraries are easier to integrate with than others, so below we introduce some tools and methods to assist you.

You can approach this framework integration in one of two ways.

1. Integrate a framework into your ArcGIS API for JavaScript application.
2. Integrate the ArcGIS API for JavaScript into an application built with a framework.

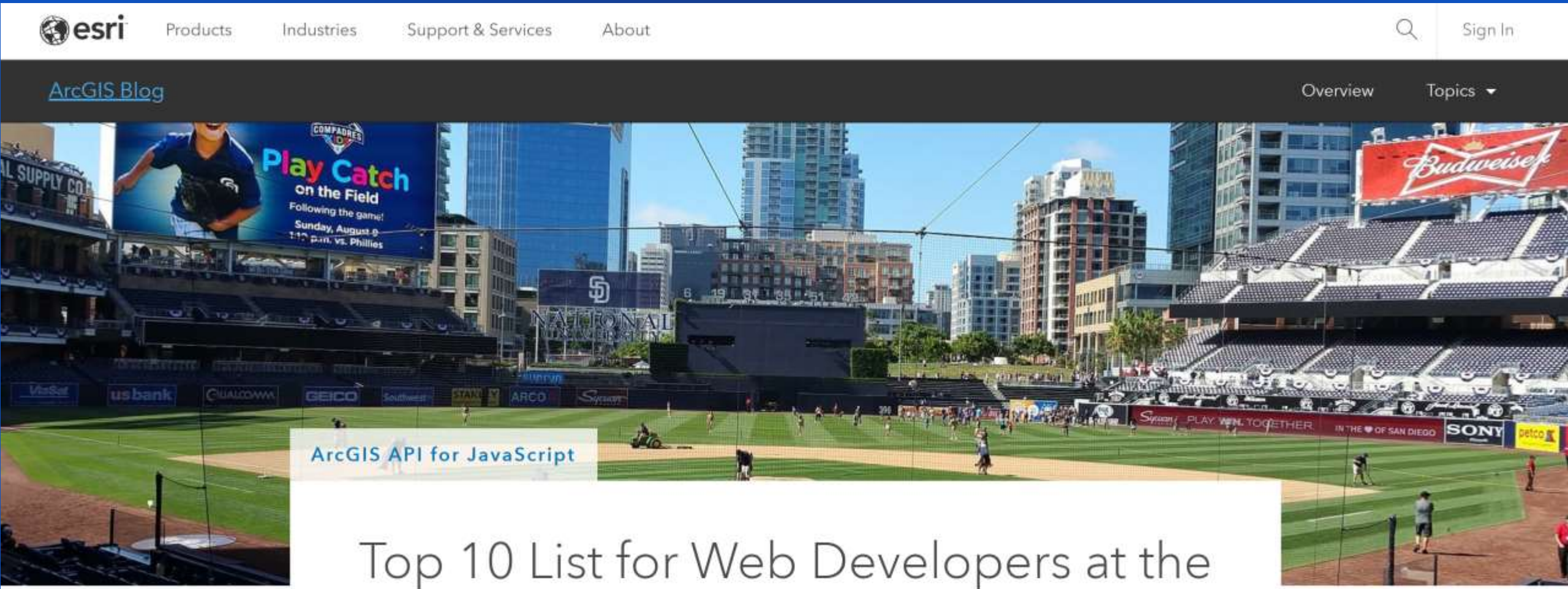
Map Centric Integration

In the first scenario, you may want to leverage a framework to help you build UI components to use with your application, but the map is still the main focus of your application and your development efforts. The ArcGIS API for JavaScript framework samples demonstrate how to take advantage of features such as view models to make it easy to use components from your framework of choice in an application that is built following the conventions of the ArcGIS API for JavaScript.

Examples:

Plan your week…

http://esriurl.com/uc2019webdev

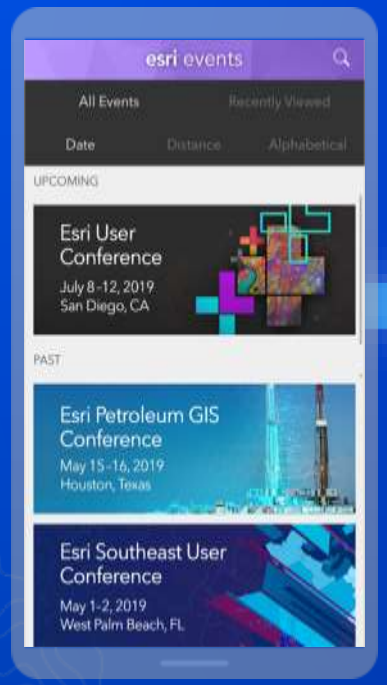


Share your apps and suggestions with us…

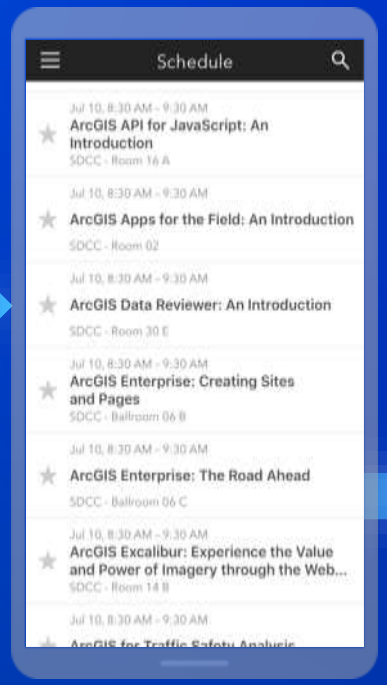
- Your apps!
- Your impressions on the latest API
- Ideas for next UC or Developer Summit related to web development

Please Share Your Feedback in the App

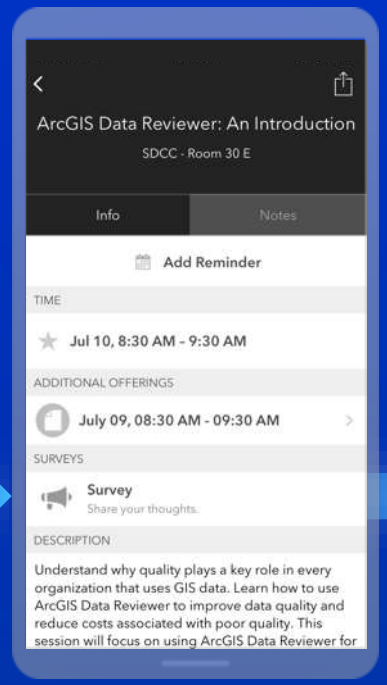
Download the Esri Events app and find your event



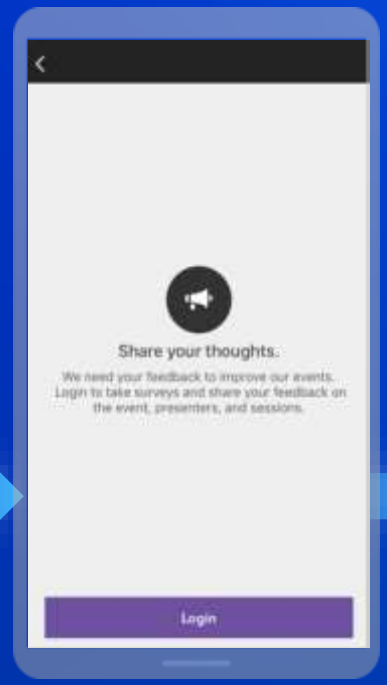
Select the session you attended



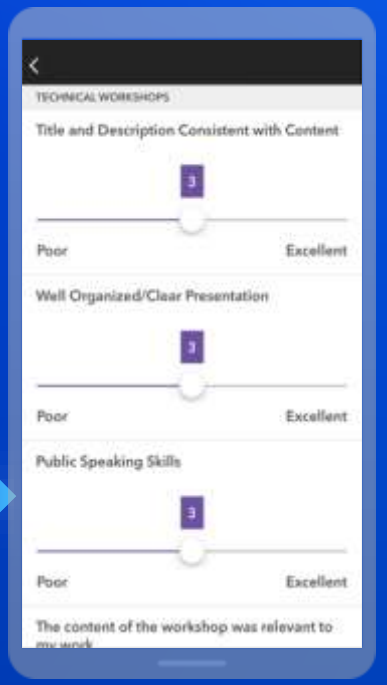
Scroll down to "Survey"



Log in to access the survey



Complete the survey and select "Submit"



Section Header

Section Subhead

Demo Title

Presenter(s)