



ESRI DEVELOPER SUMMIT 2023

ArcGIS Maps SDK for JavaScript: Programming Patterns and API Fundamentals

Lauren Boyd and Sage Wall

Presentation Resources:

<https://links.esri.com/DS23-programming-patterns>

```
const layer = view.map.allLayers.get(0);  
view.whenLayerView(layer)  
.then(layerView => console.log(`  
// if there were problems with  
// the layer, they would appear here`))  
.catch(console.error);
```

```
const view = new MapView({  
  container: "viewDiv",  
  map: map,  
  environment: {  
    lightings: {  
      directShadowsEnabled: true  
    }  
  }  
});
```

Accessing the SDK

Sage Wall

```
const layer = view.map.allLayers.get(0);
view.whenLayerView(layer)
  .then(layerView => console.log(layerView))
  // if there were problems with the layer
  .catch(console.error);
```

```
const view = new View({
  container: "view",
  map: map,
  environment: {
    lightings: {
      directShadowsEnabled: true
    }
  }
});
```

Get Started

The screenshot shows the ArcGIS Developers website with a dark blue header. The header includes the ArcGIS Developers logo, navigation links for Documentation, Features, Pricing, and Support, a search bar, and a sign-in button.

The main content area has a purple header bar with the text "ArcGIS Maps SDK for JavaScript" on the left and "Overview", "Sample Code", "API Reference", "Showcase", and "Blogs" on the right.

The main content starts with a "Find page..." search bar and a sidebar menu containing:

- Overview
- Key features
- Get started (selected)
- Install and set up
- Release notes
- FAQ
- Community
- Tutorials
- Core concepts
- Visualization
- Building your UI
- Working with ArcGIS Online and Enterprise
- Developer tooling
- Migrating from 3.x

The main article is titled "Get started" and lists three steps:

- 1 | [Install and set up](#) the API
- 2 | [Get an API key](#) (access services)
- 3 | Start the [Display a map \(2D\)](#) tutorial

On the right side, there is a "On this page" sidebar with the same three items. Below it is a "Was this page helpful?" section with "Yes" and "No" buttons. At the bottom of the article, there is a code block:

```
<link rel="stylesheet" href="https://js.arcgis.com/4.25/esri/themes/light/main.css">
<script src="https://js.arcgis.com/4.25/"></script>
```

Asynchronous Module Definition (AMD)



AMD Modules via ArcGIS CDN

```
<link rel="stylesheet" href="https://js.arcgis.com/4.26/esri/themes/light/main.css">
<script src="https://js.arcgis.com/4.26/"></script>
```

Loading Classes via AMD



Array of ordered namespace
strings for each class

Class loaded into
arguments of callback

```
1 require(["esri/Map", "esri/views/MapView"], (Map, MapView) => {  
2   // The application logic using `Map` and `MapView` goes here  
3   const map = new Map();  
4   ...  
5 });
```

ECMAScript Modules (ESM)

ES modules via NPM

```
npm install @arcgis/core
```



```
import Map from "@arcgis/core/Map.js";
import MapView from "@arcgis/core/views/MapView.js";

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

Constructors, properties and methods

Classes have a constructor; property values can be set by passing parameters to the constructor

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

Then the class's properties and methods are available



```
// sets the basemap property to "streets-vector"  
map.basemap = "streets-vector";
```



```
// destroys the map and any associated resources  
map.destroy();
```

Autocasting



Without Autocasting

```
import Color from "@arcgis/core/Color.js";
import Graphic from "@arcgis/core/Graphic.js";
import SimpleLineSymbol from "@arcgis/core/symbols/SimpleLineSymbol.js";
import SimpleFillSymbol from "@arcgis/core/symbols/SimpleFillSymbol.js";

const graphic = new Graphic({
  symbol: new SimpleFillSymbol({
    color: new Color([173, 216, 230, 0.2]),
    outline: new SimpleLineSymbol({
      color: new Color([255, 255, 255]),
      width: 1
    })
  })
});
```

symbol [Symbol](#) [autocast](#)



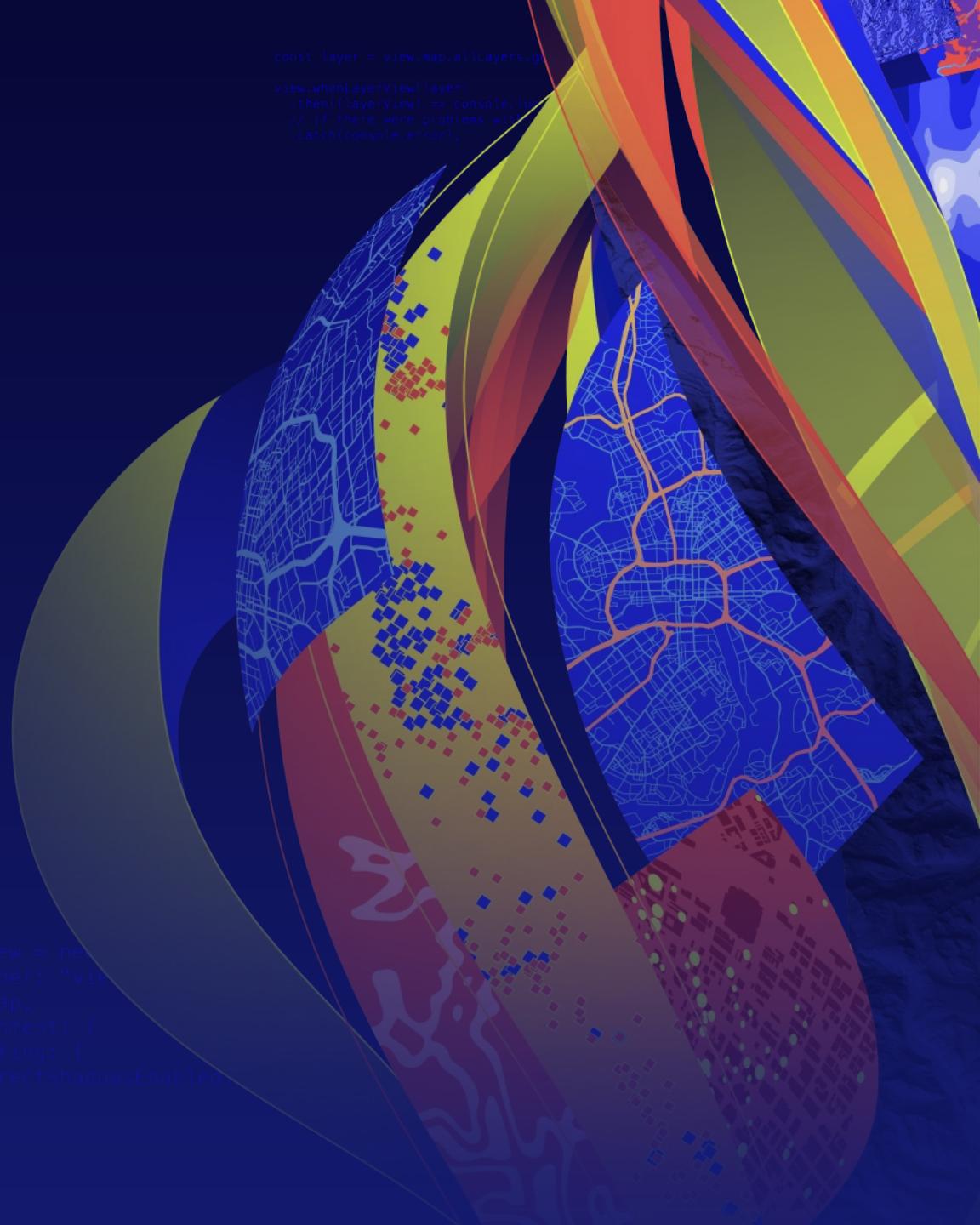
With Autocasting

```
import Graphic from "@arcgis/core/Graphic.js";

const graphic = new Graphic({
  symbol: {
    type: "simple-fill", // autocasts as SimpleFillSymbol
    color: [173, 216, 230, 0.2],
    outline: {
      color: [255, 255, 255],
      width: 1
    }
  }
});
```

Maps and Views

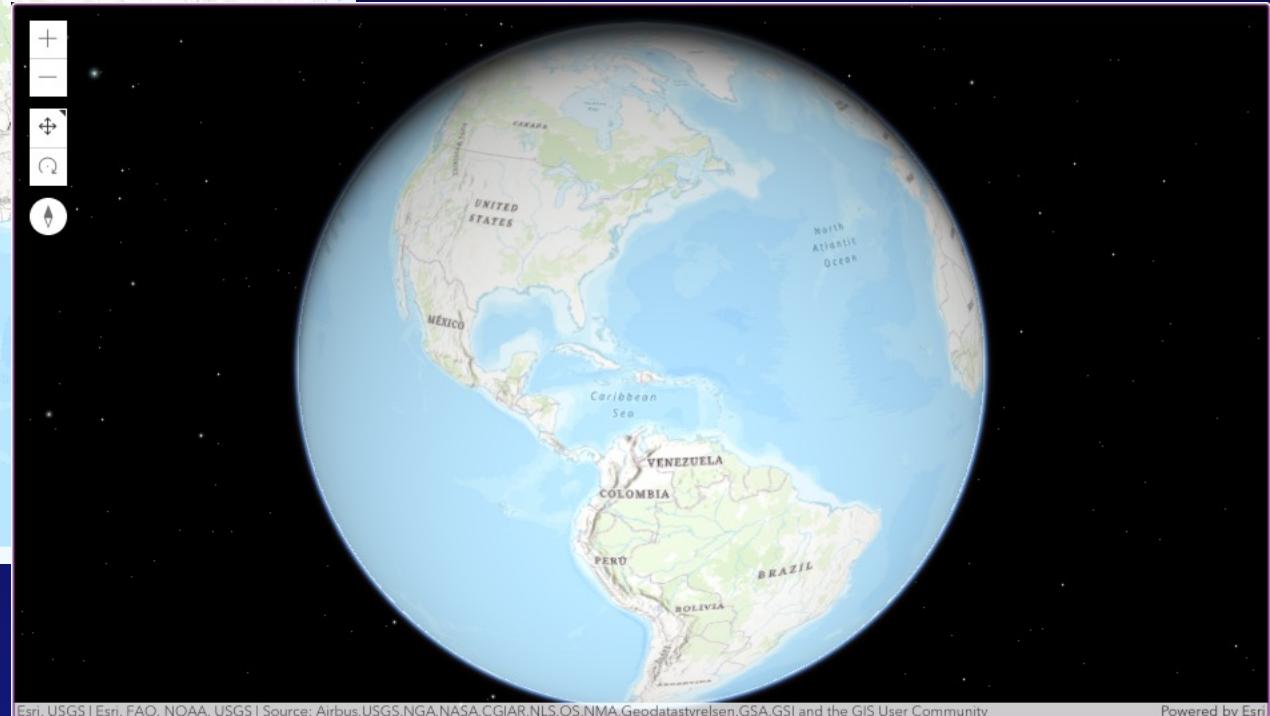
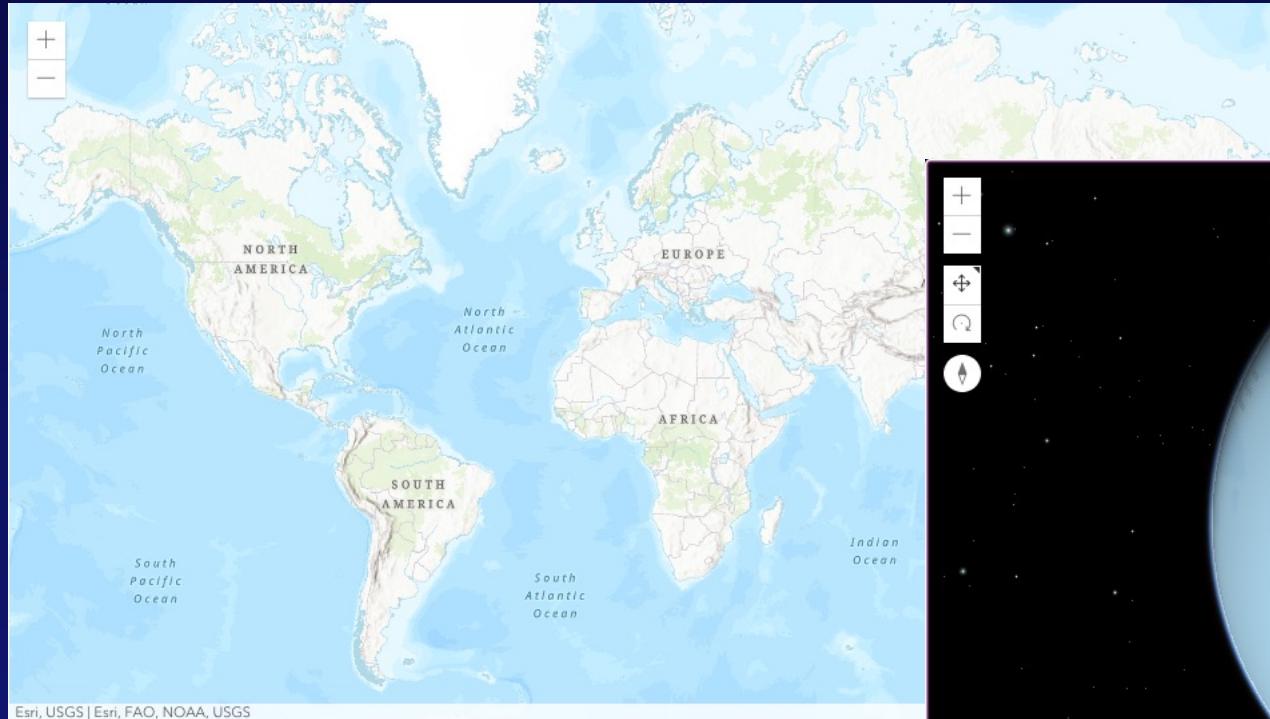
Lauren Boyd



```
const view = new View({  
  container: "view",  
  map: map,  
  environment: {  
    lightings: {  
      directShadowsEnabled: true  
    }  
  }  
})
```

```
const layer = view.map.allLayers.get(0);  
view.whenLayerView(layer)  
.then((layerView) => console.log("Layer loaded"));  
// if there were problems with loading:  
.catch(console.error);
```

MapView & SceneView



Powered by Esri

Map and Views

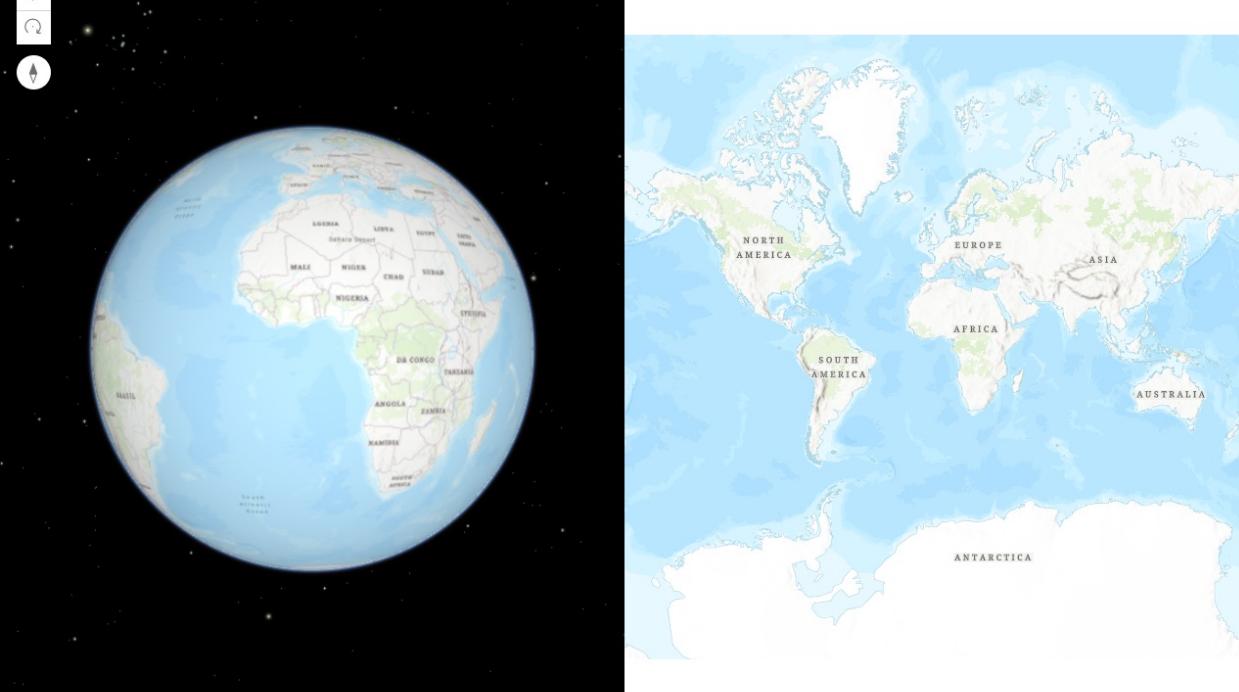
```
1 const map = new Map({
2   basemap: "topo-vector",
3   // World elevation service
4   ground: "world-elevation"
5 });
6
7 const mapView = newMapView({
8   map: map,
9   container: "viewDiv"
10 });
11
12 const sceneView = newSceneView({
13   map: map,
14   container: "viewDiv"
15 });
```

Basemaps

- Basemaps for developers
 - For use with API keys
- Basemaps for ArcGIS Organization users
 - For use with ArcGIS Online/ArcGIS Enterprise users

```
1 const map = new Map({  
2   /*  
3    For ArcGIS Organizational Accounts:  
4    "satellite", "hybrid", "terrain", "oceans", "osm",  
5    "dark-gray-vector", "gray-vector", "streets-vector",  
6    "topo-vector", "streets-night-vector", "streets-relief-vector",  
7    "streets-navigation-vector"  
8   For use with API keys:  
9   "arcgis-imagery", "arcgis-imagery-standard", "arcgis-imagery-labels",  
10  "arcgis-light-gray", "arcgis-dark-gray", "arcgis-navigation",  
11  "arcgis-navigation-night", "arcgis-streets", "arcgis-streets-night",  
12  "arcgis-streets-relief", "arcgis-topographic", "arcgis-oceans",  
13  "osm-standard", "osm-standard-relief", "osm-streets",  
14  "osm-streets-relief", "osm-light-gray", "osm-dark-gray",  
15  "arcgis-terrain", "arcgis-community", "arcgischarted-territory",  
16  "arcgis-colored-pencil", "arcgis-nova", "arcgis-modern-antique",  
17  "arcgis-midcentury", "arcgis-newspaper", "arcgis-hillshade-light",  
18  "arcgis-hillshade-dark", "arcgis-human-geography", "  
19  arcgis-human-geography-dark"  
20  */  
21  basemap: "topo-vector",  
22  // World elevation service  
23  ground: "world-elevation"  
24 })
```

```
const view = new SceneView({  
  container: "viewDiv",  
  map: map,  
  environment: {  
    lighting: {  
      directShadowsEnabled: true  
    }  
  }  
})
```



Maps and Views

[ASM Demo](#)
[ESM Demo](#)

```
myView.layers.getItem(index)  
  .then(function(layer){  
    console.log(layerView);  
  })  
// If you try to do something with the layerView, you'll get an error here.
```

Layers and visualization

Lauren Boyd

```
const layer = view.map.allLayers.get(0);
view.whenLayerView(layer)
  .then(layerView => console.log(`Layer ${layer.name} loaded`))
  // if there were problems with loading
  .catch(console.error);
```

```
const view = new View({
  container: "view",
  map: map,
  environment: {
    lightings: {
      directShadowsEnabled: true
    }
  }
});
```



Layers

Create a layer from a URL or portal item



```
const layer = new FeatureLayer({
  url: "https://services3.arcgis.com/GVgbJbqm8hXASVYi/arcgis/rest/services/Trailheads/FeatureServer/0",
  // portalItem: {
  //   id: "883cedb8c9fe4524b64d47666ed234a7",
  //   portal: "https://www.arcgis.com"           // Default: The ArcGIS Online Portal
  // }
});

map.add(layer);
```

Different layer types depending on your data source – see the [Layer documentation](#) for the full list

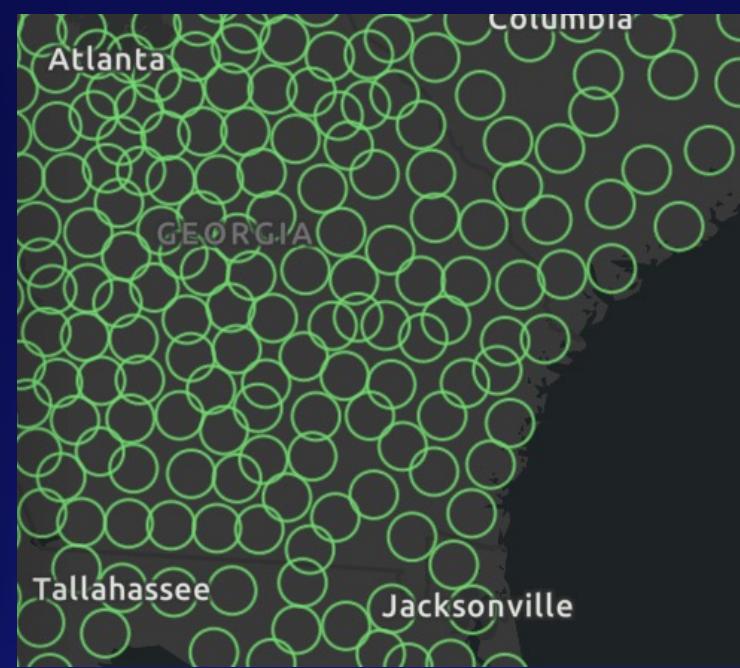
Visualization

Visualization Style	Renderer
Location Style	Simple Renderer
Data Driven Style	Unique Types , Class Breaks , Visual Variables , Time , Multivariate , Predominance , Dot Density , Relationship , Smart Mapping , Pie Charts
High Density Data	Clustering , Binning , Heatmap , Opacity , Bloom , Aggregation , Thinning , Visible Scale Range
3D Visualization	Global and local scenes , terrain rendering , cities in 3D , Visualizing points with 3D symbols

Renderers

- Renderers define how to visually represent features in the layer

```
1 const layer = new FeatureLayer({  
2   url: "SERVICE URL",  
3   renderer: {  
4     type: "simple", // autocasts as new SimpleRenderer()  
5     symbol: {  
6       // define symbol properties  
7       type: "simple-marker", // autocasts as new SimpleMarkerSymbol()  
8       outline: {  
9         color: [0, 246, 84, 1]  
10      },  
11      size: 14,  
12      color: [255, 255, 255, 0]  
13    }  
14  }  
15 })
```



Smart Mapping

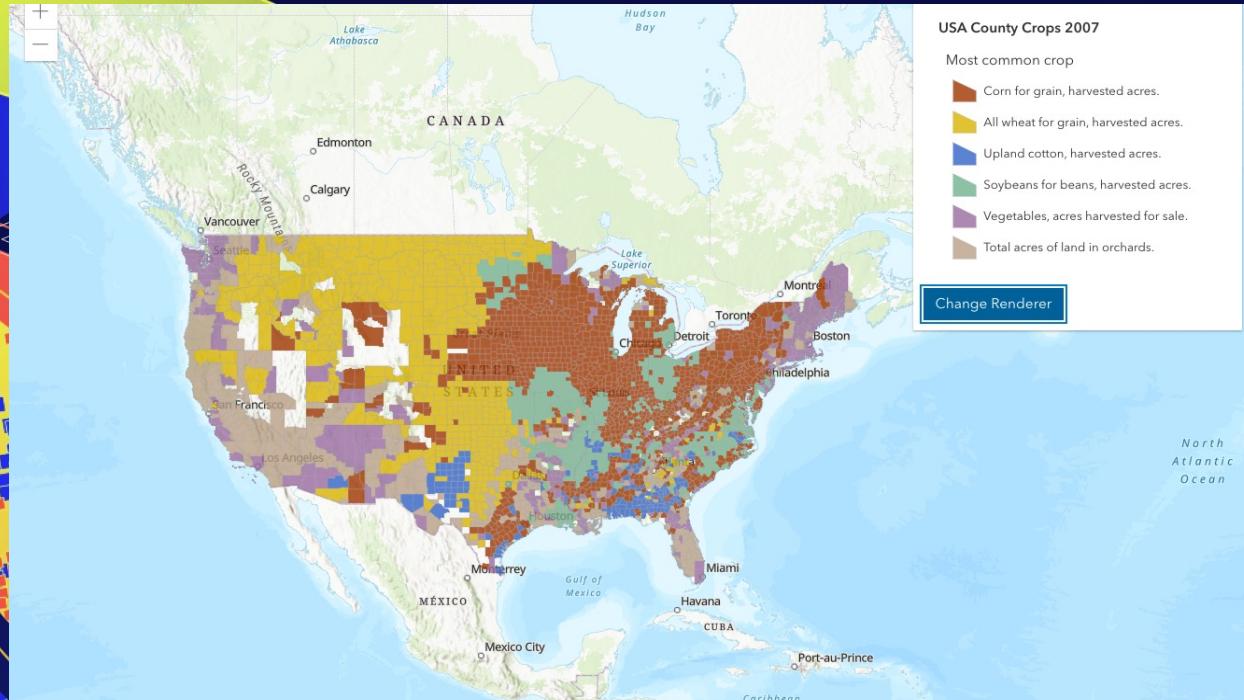
- Easy way to generate visualizations
- Focus on:
 - One visualization method
 - Subset of attributes
 - Specific Area



```
1 let layer = new FeatureLayer({
2   url:
    "https://services.arcgis.com/V6ZHFr6zdgNZuVG0/arcgis/rest/services
     /counties_politics_poverty/FeatureServer/0"
3 });
4
5 // simple visualization to indicate features with a single symbol
6 let params = {
7   layer: layer,
8   view: view
9 };
10
11 // when the promise resolves, apply the renderer to the layer
12 locationRendererCreator.createRenderer(params)
13   .then(function(response){
14     layer.renderer = response.renderer;
15   });

```

```
    const view = new SceneView({  
      container: "viewDiv",  
      map: map,  
      environment: {  
        lighting: {  
          directShadowsEnabled: true  
        }  
      }  
    })
```



Smartmapping

Demo

LayerViews and querying

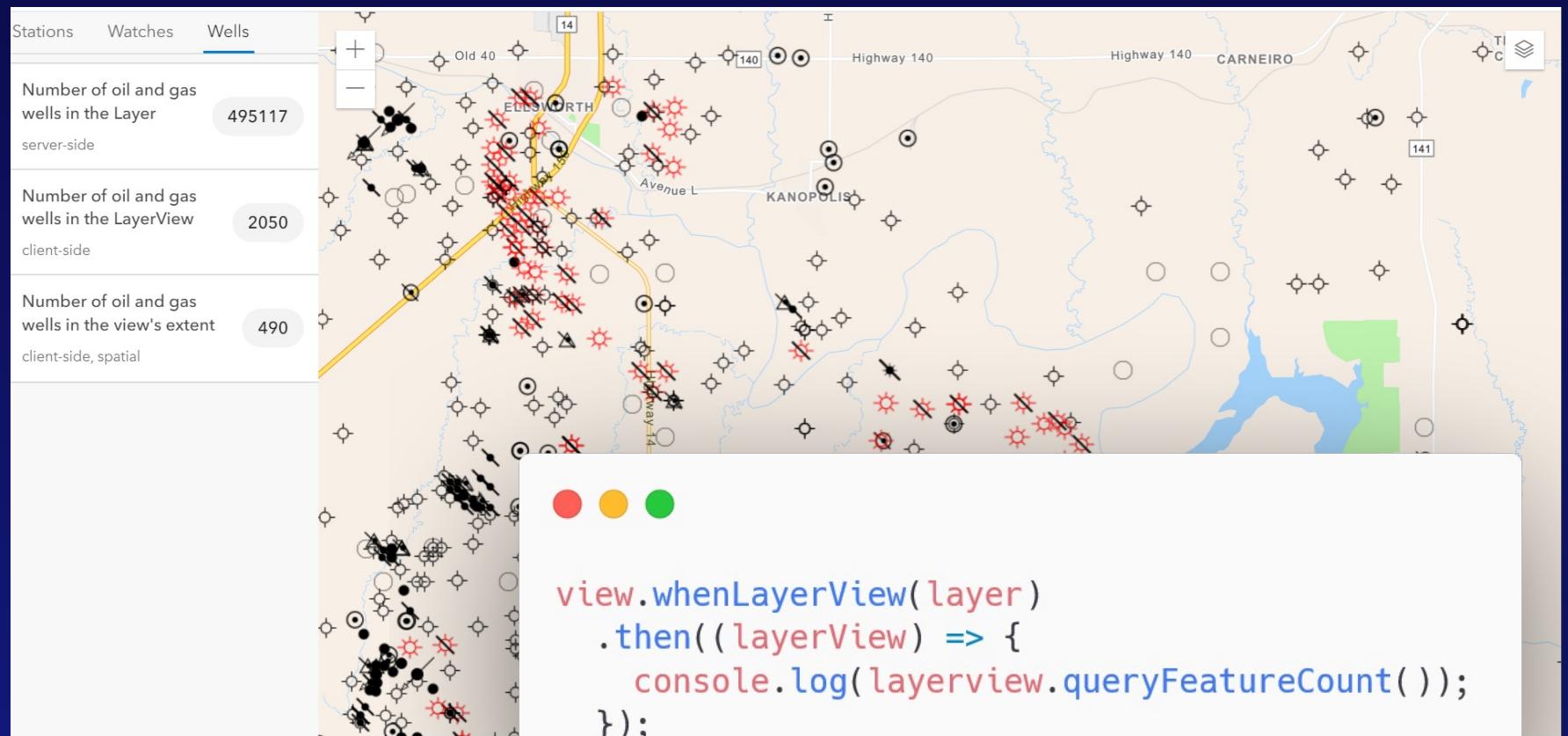
Sage Wall

```
const view = new  
  container: "view",  
  map: map,  
  environment: {  
    lightings: {  
      directShadowsEnabled: true  
    }  
  }
```

```
const layer = view.map.allLayers.get(0);  
  
view.whenLayerView(layer)  
.then((layerView) => console.log(layerView))  
// if there were problems with the layer  
.catch(console.error);
```

LayerView

- Renders features in the view
- Client-side queries, filtering, and effects



Creating and using a query

- Using **createQuery()**
 - Maintains filters and definition expressions
- **new Query()**
 - Clean query object



```
const query = featureLayer.createQuery();
query.where = "Name = 'Sage'";

featureLayer.queryFeatures(query)
  .then((results) => {
    console.log(results)
 });
```

Query Methods

`queryFeatures()`

- Queries features and returns a FeatureSet

`queryExtent()`

- Queries features and returns an extent

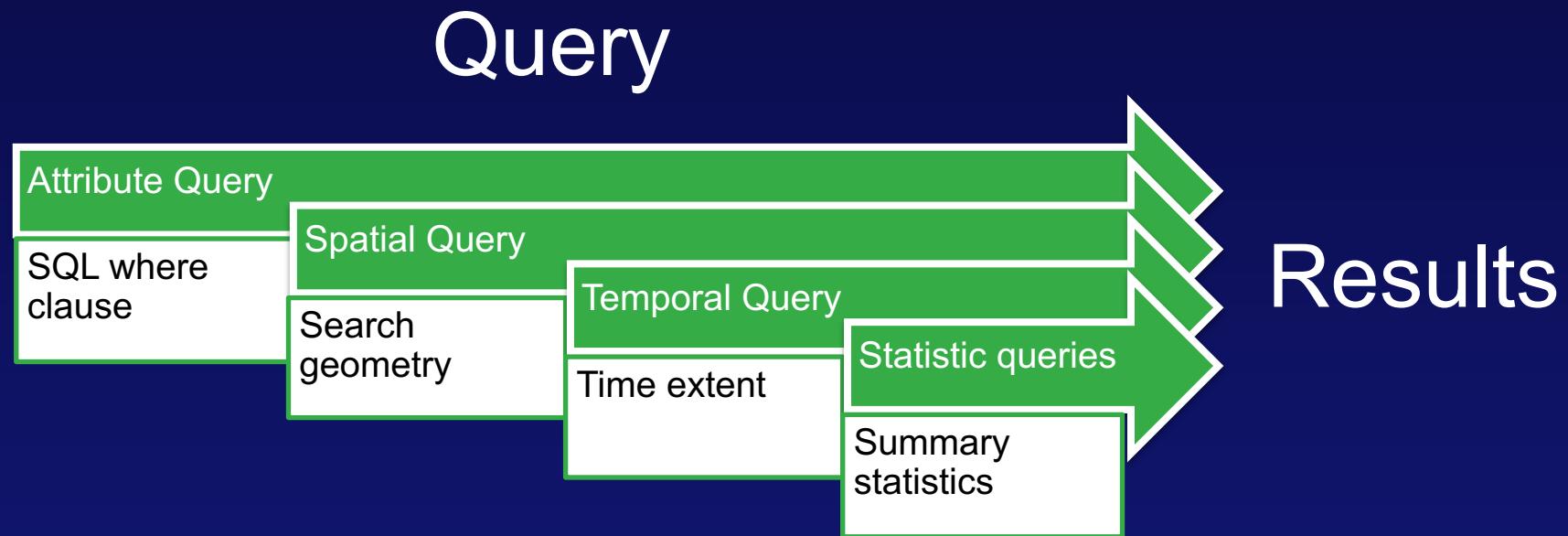
`queryFeatureCount()`

- Queries features and returns a count

`queryObjectIds()`

- Queries features and returns an array of objectIds

Query Types



Should I use a layer query or a layer view query?

What's Important

Performance

- LayerView Query

Geographic precision

- Layer Query

Query every feature

- Layer Query

Tips when using client-side queries on layer views

```
● ● ●

// add fields to the layer's outFields properties
const stationsLayer = new FeatureLayer({
  outFields: ["OBJECTID", "COUNTRY", "TEMP", "WIND_DIRECT", "WIND_SPEED"],
  url
});

// wait for the layer view to be loaded in the view
const stationsLayerView = await view.whenLayerView(stationsLayer);

// wait for the layer view to stop updating
await reactiveUtils.whenOnce(
  () =>
    stationsLayerView.updating === false
);

// query the number of features in the stations layer view
const stationsLayerViewCount = await stationsLayerView.queryFeatureCount();
```

```
    inst view = new SceneView({  
        container: "viewDiv",  
        map: map,  
        environment: {  
            lighting: {  
                directShadowsEnabled: true  
            }  
        }  
    })
```

The figure shows a map of Colorado, USA, with several data layers and statistics displayed on the left side.

Map Statistics:

- Number of stations within 50 miles of mouse click: client-side, spatial, statistics
- Average wind speed within 50 miles from mouse click: client-side, spatial, statistics
- Number of stations in the Layer: 5892 server-side
- Number of stations in the LayerView: 5892 client-side
- Number of stations in the view's extent: 68 client-side, spatial

Map Labels:

- Medicine Bow National Forest
- Laramie
- Cheyenne
- Sidney
- Routt National Forest
- Roosevelt National Forest
- Pawnee National Grasslands
- South Platte River
- Craig
- Longmont
- Boulder
- Denver
- Fort Collins
- Greenley
- Greeley
- Fort Morgan
- Highway 13
- Highway 40
- Highway 70
- Highway 76
- Highway 285
- Highway 25
- Highway 24
- Highway 287
- Gypsum
- White River National Forest
- Aspen
- Castle Rock
- Colorado Springs
- San Isabel National Forest
- Gunnison
- Highway 50

Code Snippet (Bottom Right):

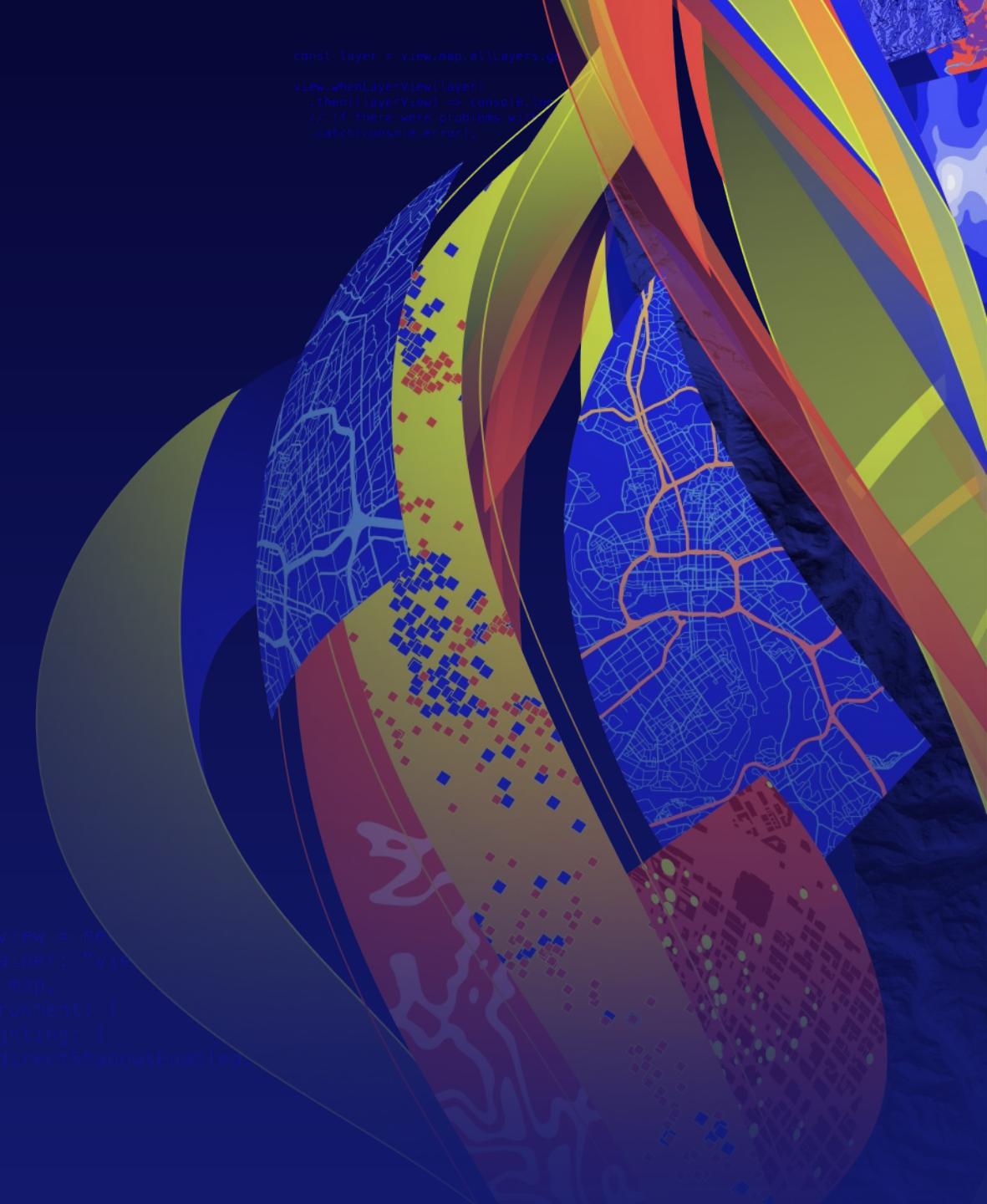
```
var layerView = map.layers.getItem(index);
layerView.click(function(e) {
    console.log(layerView);
});
```

Note: If you mess with the layerView, you'll get an error here.

Querying and LayerView

Event handling

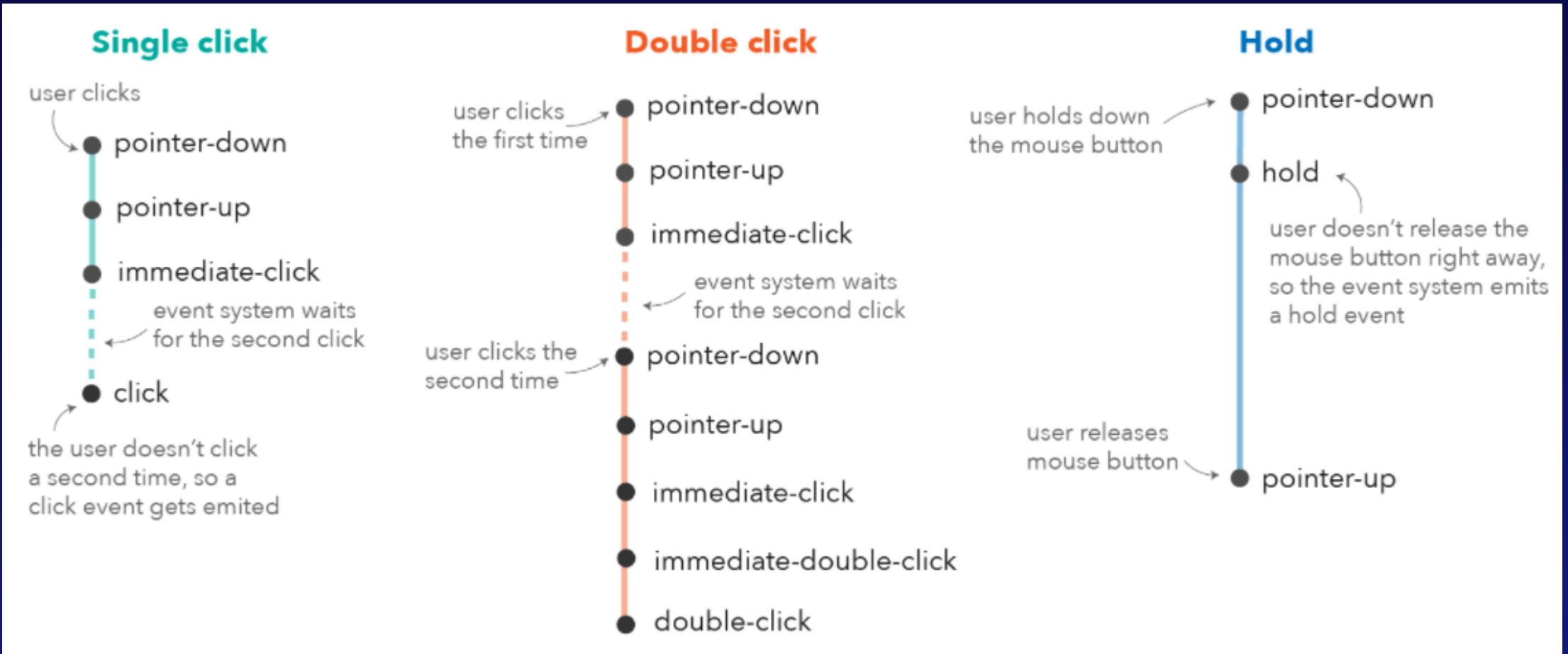
Sage Wall



```
const layer = view.map.allLayers.get(0);
view.whenLayerView(layer)
  .then((layerView) => console.log(`Layer loaded`))
  // if there were problems with loading
  .catch(console.error);
```

```
const view = new View({
  container: "view",
  map: map,
  environment: {
    lightings: {
      directShadowsEnabled: true
    }
  }
});
```

Events



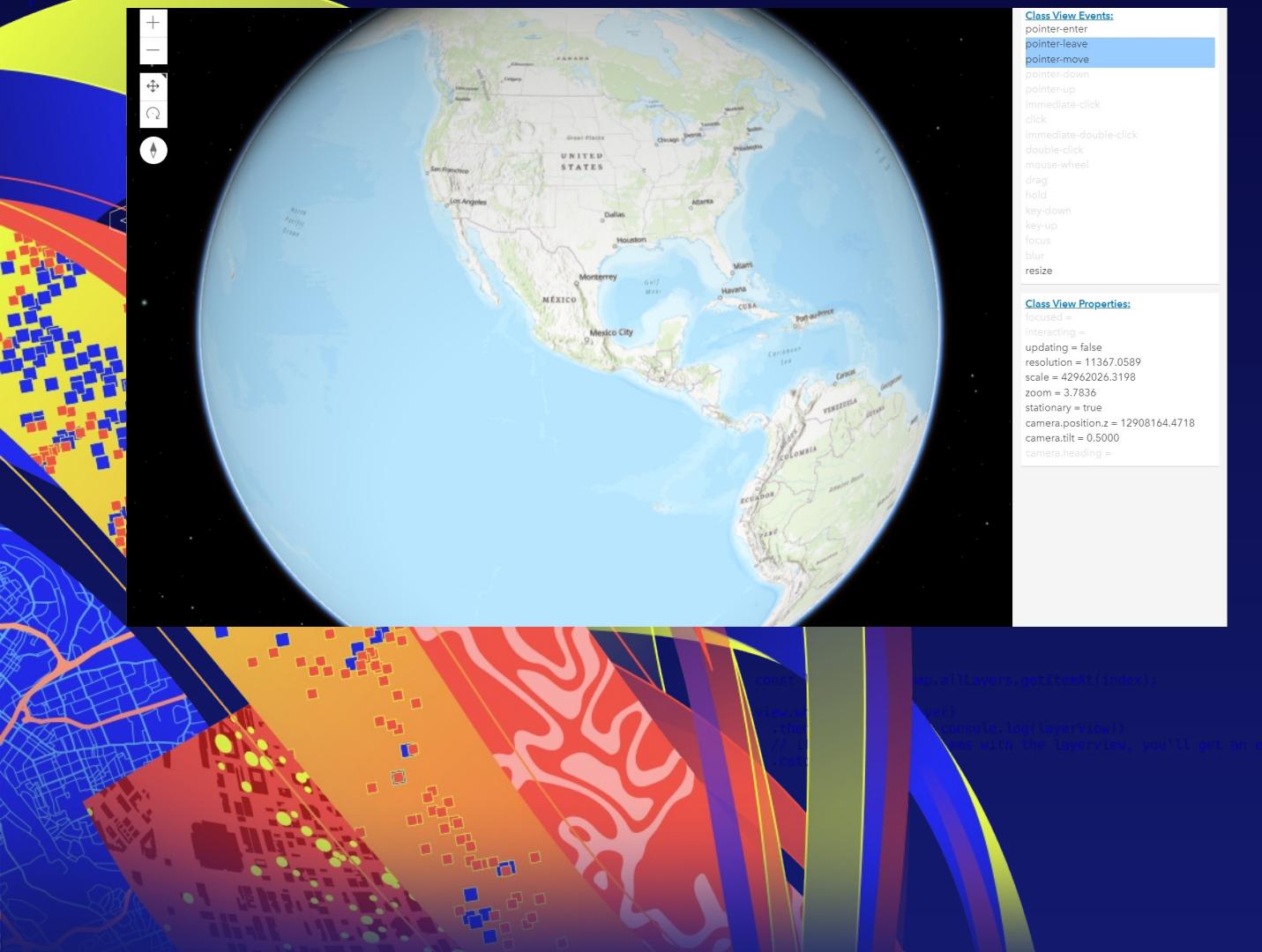
Event Handling



```
// when view emits a click event
view.on("click", (event) => {
  view.graphics.removeAll();
  addGraphic(event);
});

// add a graphic to the view at the click event's location
const addGraphic = (screenPoint) => {
  const point = view.toMap(screenPoint);
  const newClickGraphic = clickGraphic.clone();
  newClickGraphic.geometry = point;
  view.graphics.add(newClickGraphic);
};
```

```
const view = new SceneView({  
  container: "viewDiv",  
  map: map,  
  environment: {  
    lighting: {  
      directShadowsEnabled: true  
    }  
  }  
})
```



Event handling

Demo

Programming patterns

Lauren Boyd

```
const layer = view.map.allLayers.get(0);
view.whenLayerView(layer)
  .then(layerView => console.log(`Layer ${layer.name} loaded`))
  // if there were problems with loading
  .catch(console.error);
```

```
const view = new View({
  container: "view",
  map: map,
  environment: {
    lightings: {
      directShadowsEnabled: true
    }
  }
});
```

Promises

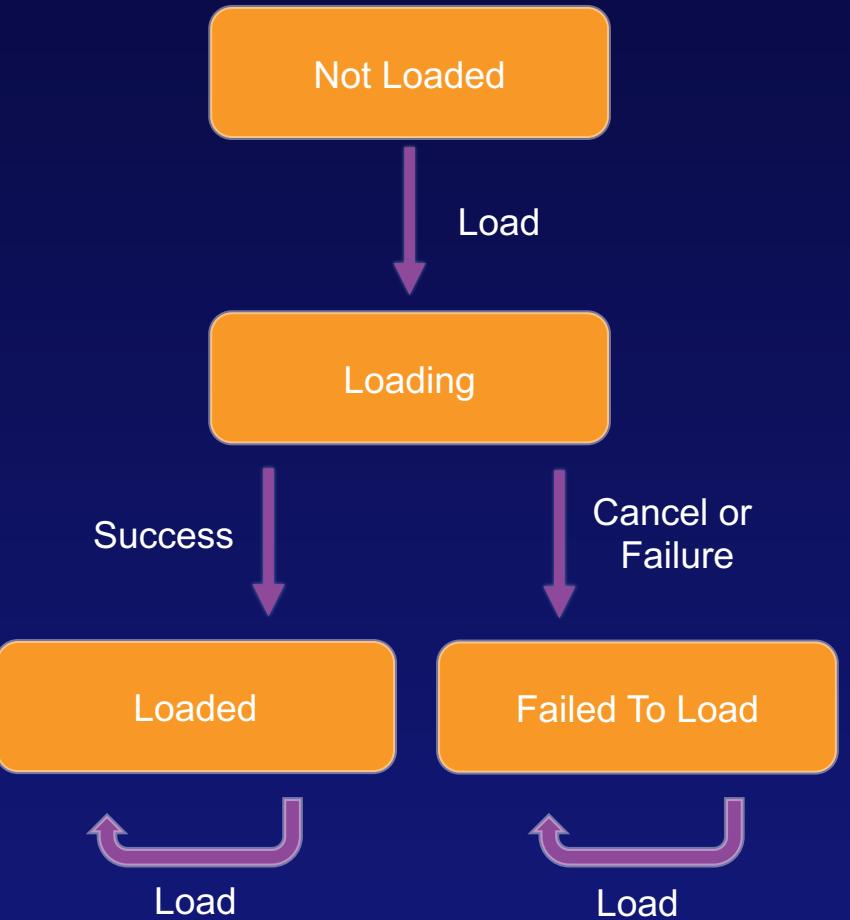
Promise = representation of a future value returned from an asynchronous task



```
1 // Create a feature layer from a url
2 let layer = new FeatureLayer(url);
3
4 map.add(layer);
5
6 view.whenLayerView(layer)
7   .then(function(layerView) {
8     // Do something with the layerview
9   })
10  .catch(function(error) {
11    // Error occurred creating the layerview
12  });
```

Loadable

```
1 const view = new MapView({  
2   container: "viewDiv"  
3 });  
4  
5 const portal = new Portal({  
6   url: "https://myportal/"  
7 });  
8  
9 const webmap = new WebMap({  
10   portalItem: {  
11     portal: portal,  
12     id: "f2e9b762544945f390ca4ac3671cfa72"  
13   }  
14 });  
15  
16 webmap.load()  
17   .then(() => { view.map = webmap; })  
18   .catch((error) => {  
19     console.error("The resource failed to load: ", error);  
20   });
```



Reacting to Changes

- Watch when values change
- React when expressions become truthy

```
1 // React when an expression becomes truthy
2 reactiveUtils.when(
3   () => view.scale > 1000,
4   () => {
5     console.log(`Scale is greater than 1000`);
6   }
7 );
8
9 // Observe for when a boolean property becomes true
10 // Equivalent to watchUtils.whenTrue()
11 reactiveUtils.when(
12   () => view?.stationary === true,
13   async () => {
14     console.log("User is no longer interacting with the map");
15     await drawBuffer();
16   }
17 );
```

Reacting to Changes

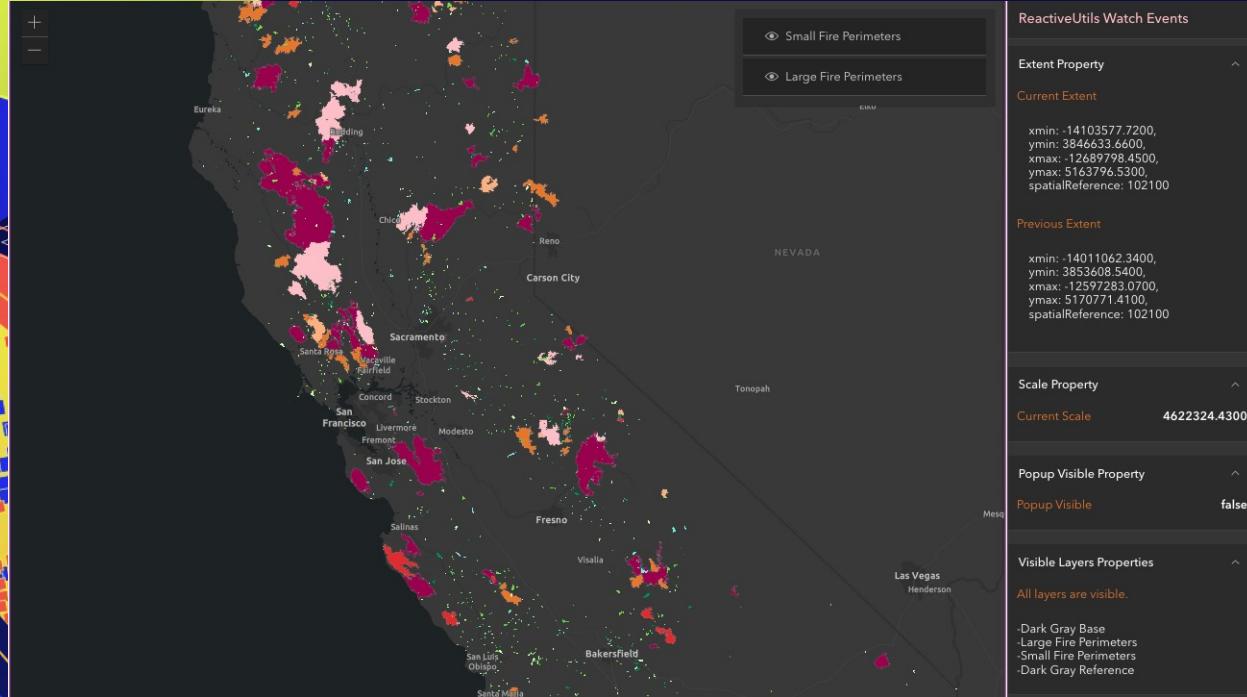


```
1 // React to multiple property changes
2 reactiveUtils.watch(
3   () => [view.stationary, view.scale],
4   ([stationary, scale]) => {
5     console.log(`View is stationary: ${stationary} and scale is ${scale}`);
6   });
7
8 // React to Collection changes
9 reactiveUtils.watch(
10  () => view.map.layers((layer) => layer.title),
11  (titles) => {
12    console.log(`Layer titles changed! New titles: ${titles}`);
13  });

```

- Combine values from multiple sources
- Watch for changes within a Collection

```
    inst view = new SceneView({  
        container: "viewDiv",  
        map: map,  
        environment: {  
            lighting: {  
                directShadowsEnabled: true  
            }  
        }  
    })
```



reactiveUtils

Widgets

Lauren Boyd

```
const view = new  
  container: "view",  
  map: map,  
  environment: {  
    lightings: {  
      directShadowsEnabled: true  
    }  
  }
```

```
const layer = view.map.allLayers.get(0);  
view.whenLayerView(layer)  
.then((layerView) => console.log(layerView))  
// if there were problems with the layer  
.catch(console.error);
```

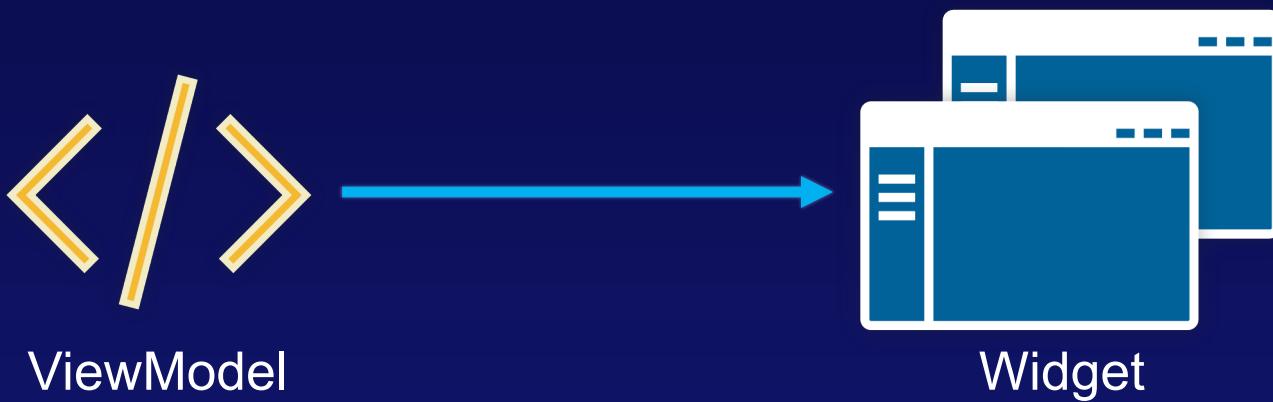
Widgets

- Professionally Designed and tested building blocks
- Responsive, Accessible, and Localized
- Out of the box

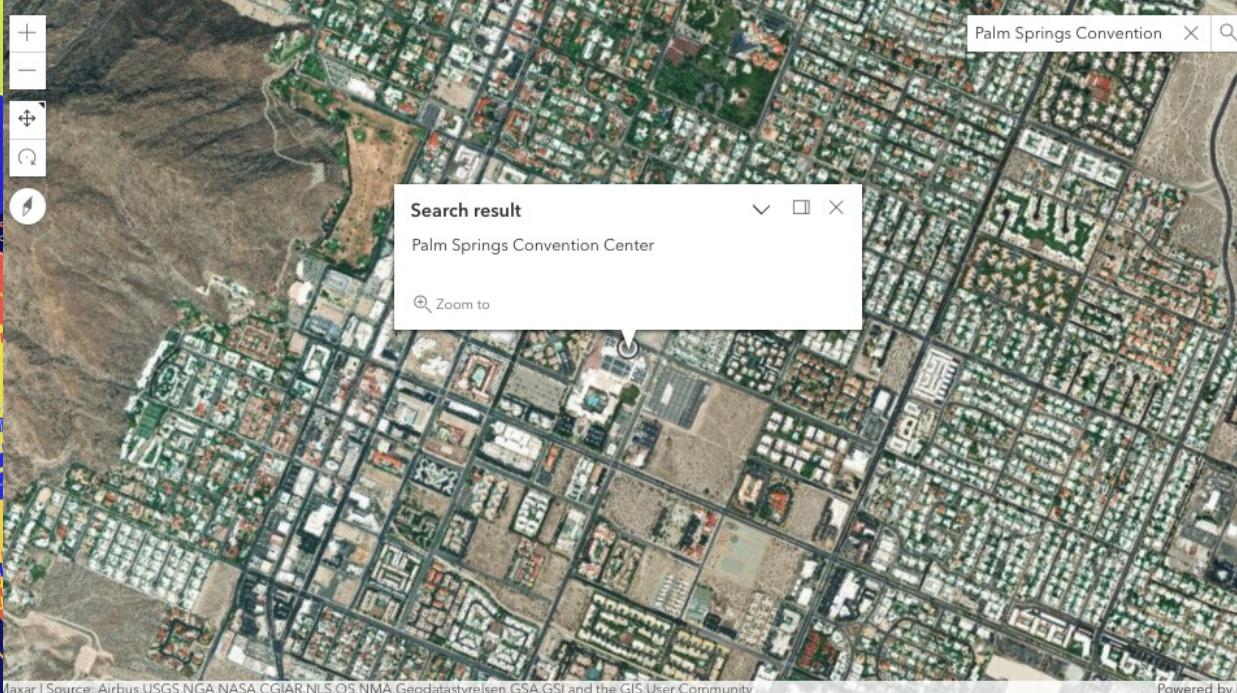


```
const widget = new SomeWidget({  
  view,  
  ...otherOptions  
})
```

ViewModels



```
const view = new SceneView({  
  container: "viewDiv",  
  map: map,  
  environment: {  
    lighting: {  
      directShadowsEnabled: true  
    }  
  }  
})
```



Search

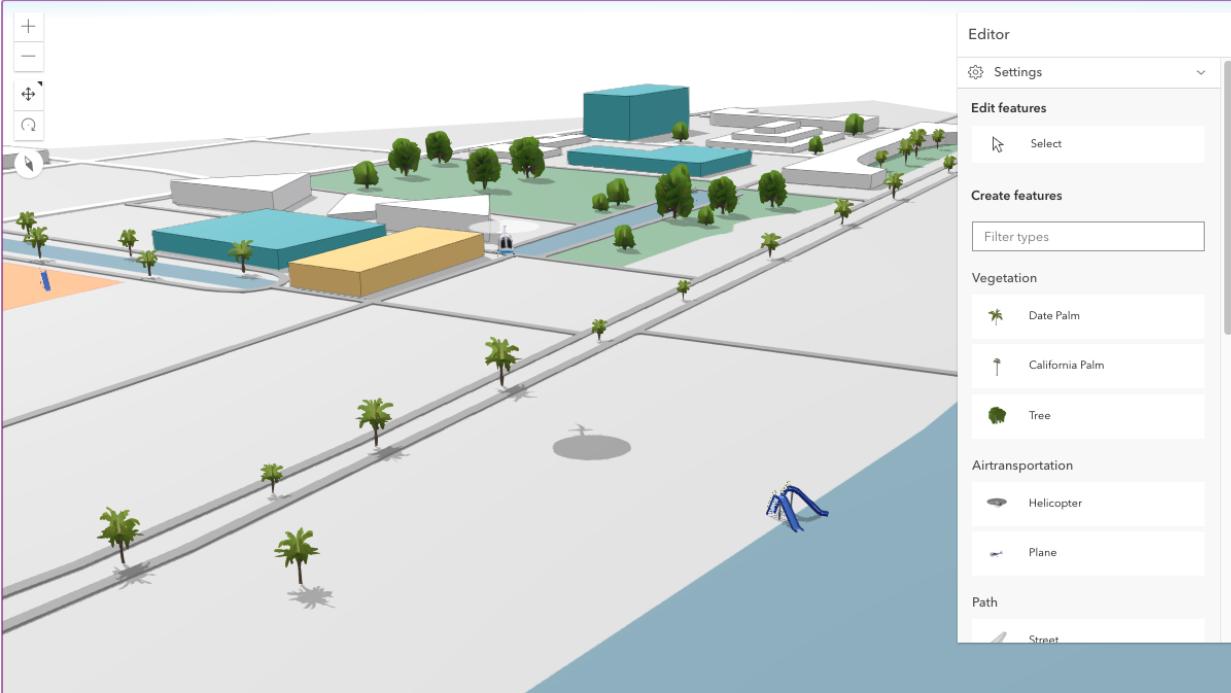
Demo

Maxar | Source: Airbus, USGS, NGA, NASA, CGIAR, NLS, OS, NIMA, Geodatistyrelsen, GSA, GSI and the GIS User Community

Powered by Esri

```
const layers = view.map.layers;  
const layerView = layers.getLayerView();  
const index = 0;  
const layer = layers.getLayer(index);  
layerView.setLayer(layer);  
console.log(layerView);  
// If you try to interact with the layerView, you'll get an error here.
```

```
const view = new SceneView({  
  container: "viewDiv",  
  map: map,  
  environment: {  
    lighting: {  
      directShadowsEnabled: true  
    }  
  }  
})
```



Editor

Demo

Where can I learn more?

- [SDK Documentation](#)
- [Programming Patterns Guide](#)
- [Esri Blogs](#)
- [Esri Community](#)

Presentation Resources:

<https://links.esri.com/DS23-programming-patterns>

```
const layer = view.map.allLayers.get(0);  
view.whenLayerView(layer)  
.then(layerView => console.log(`  
// if there were problems with  
// the layer, they would appear here`))  
.catch(console.error);
```

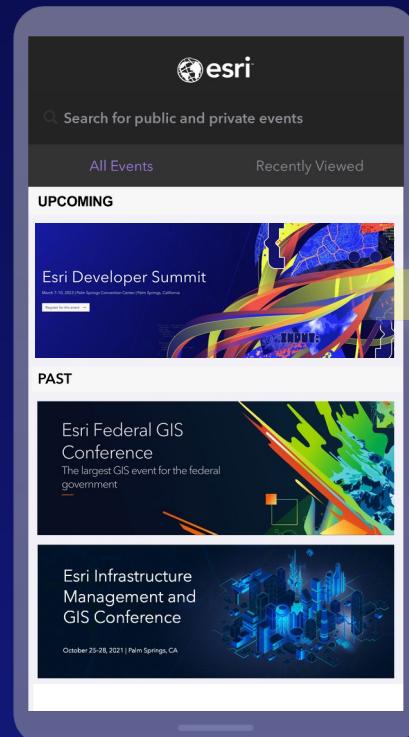
```
const view = new MapView({  
  container: "viewDiv",  
  map: map,  
  environment: {  
    lightings: {  
      directShadowsEnabled: true  
    }  
  }  
});
```

Upcoming Sessions

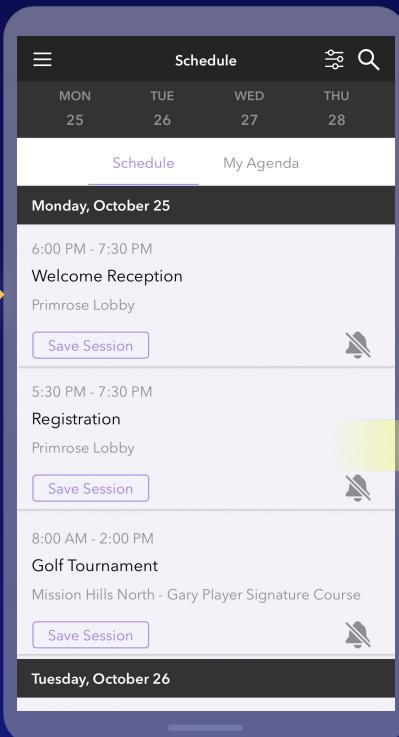
Session	Day/Time	Location
ArcGIS Maps SDK for JavaScript: Key Highlights of the Last Year	Tuesday, 5:30PM – 6:30PM	Primrose A PSCC
ArcGIS Maps SDK for JavaScript: Data Visualization	Wednesday, 10:30PM – 11:30PM	Catalina/Madera Renaissance Hotel
ArcGIS Online: Web Mapping with Arcade Expressions	Wednesday, 2:30PM – 3:30PM	Mojave Learning Center Renaissance Hotel
ArcGIS Maps SDK for JavaScript: Using the FeatureLayer	Wednesday, 4:00PM-5:00PM	Catalina/Madera Renaissance Hotel
ArcGIS Maps SDK for JavaScript: Bring in Data from Anywhere	Thursday, 10:30AM – 11:30AM	Mesquite C PSCC
ArcGIS Maps SDK for JavaScript: Tips and Tricks for Developing and Debugging Apps	Thursday, 1:00PM – 1:30PM	Demo Theater 3: Mesquite D-E PSCC
ArcGIS Maps SDK for JavaScript: Using Arcade with Your Apps	Thursday, 2:30PM – 3:30PM	Smoketree A-E PSCC
Q&A with the ArcGIS Maps SDK for JavaScript Team	Friday, 8:30AM – 9:30AM	Pasadena/Sierra/Ventura
ArcGIS Maps SDK for JavaScript: the Road Ahead	Friday, 10:00AM – 11:00AM	Catalina/Madera

Please Share Your Feedback in the App

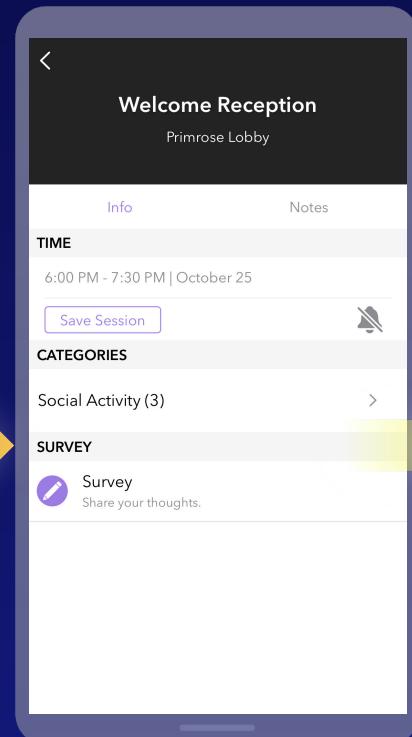
Download the Esri Events app and find your event



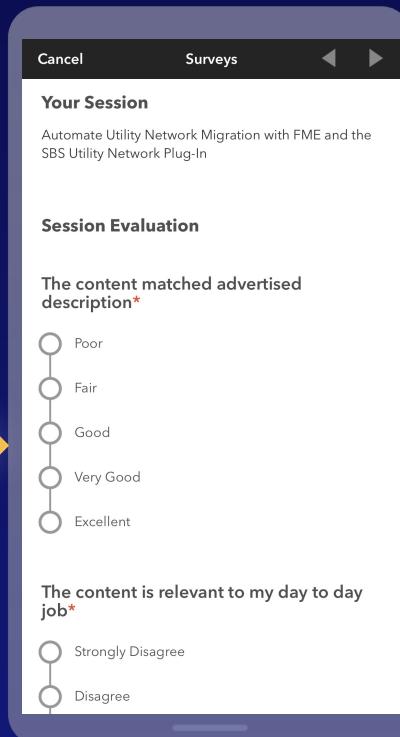
Select the session you attended



Scroll down to "Survey"



Log in to access the survey



Connect With Us On Social

And Join the Conversation Using #EsriDevSummit



twitter.com/EsriDevs #esridevsummit, #esridevsummit23



twitter.com/EsriDevEvents



youtube.com/@EsriDevs



links.esri.com/DevVideos



github.com/Esri



github.com/EsriDevEvents



links.esri.com/EsriDevCommunity

```
const layer = view.map.allLayers.get(0);  
view.whenLayerView(layer)  
.then(layerView => console.log(`  
// if there were problems with  
// this layer, they would appear here`))  
.catch(console.error);
```

```
const view = new MapView({  
  container: "viewDiv",  
  map: map,  
  environment: {  
    lightings: {  
      directShadowsEnabled: true  
    }  
  }  
});
```



esri®

THE
SCIENCE
OF
WHERE®

Copyright © 2022 Esri. All rights reserved.

```
const layer = view.map.addLayer(  
  view.whenLayerView(layer)   
    .then(layerView => const  
      {  
        if (there were problems)  
          catch(console.error);  
      }  
    );  
  );
```

E/SCRIPT>

```
const view = new SceneView({  
  container: "viewDiv",  
  map: map,  
  environment: {  
    lighting: {  
      directional:  
    },  
  },  
});
```

LIVE
BY
THE
CODE