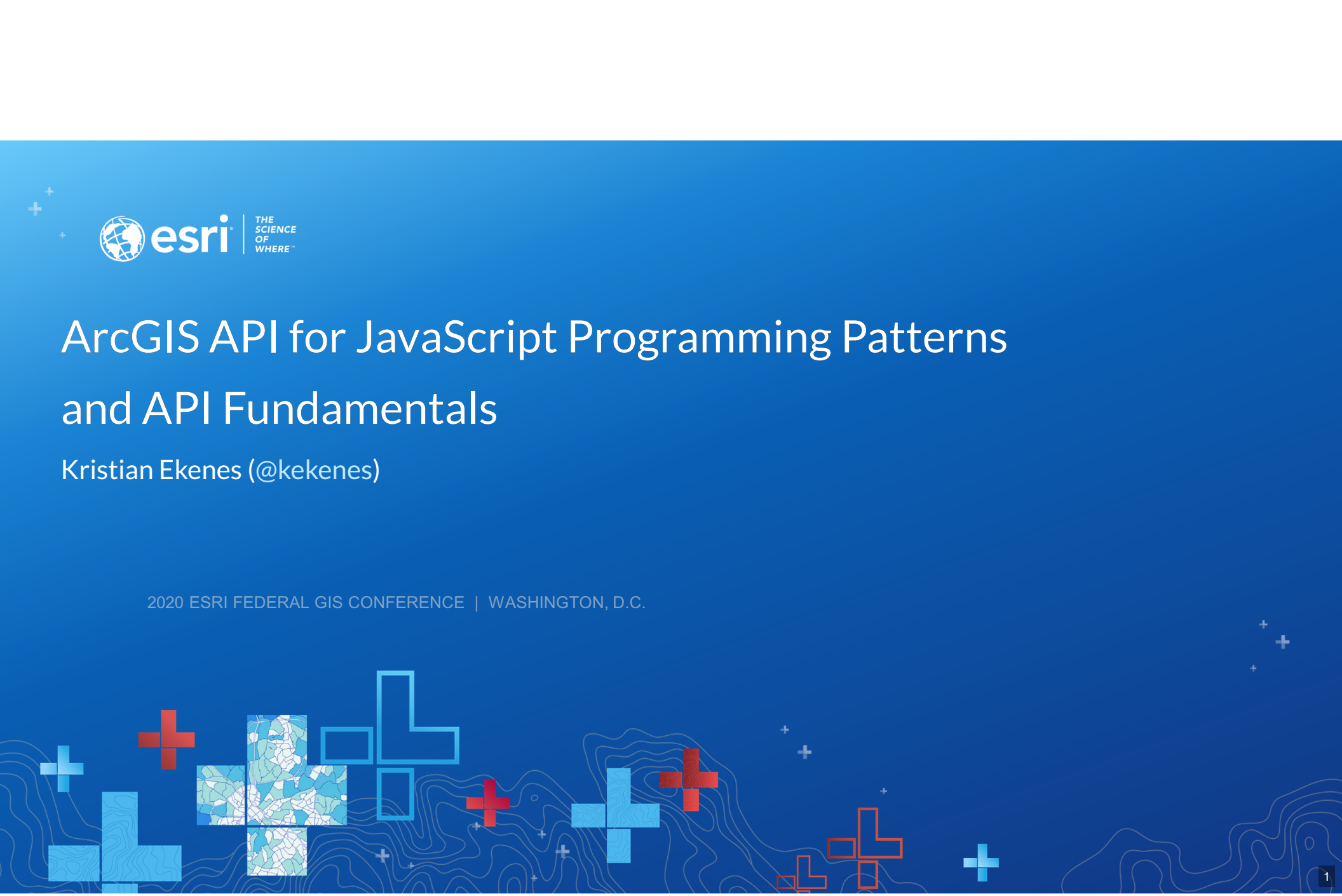

esri
THE SCIENCE OF WHERE
ArcGIS API for JavaScript Programming Patterns and API Fundamentals
Kristian Ekenes (@kekenes)
2020 ESRI FEDERAL GIS CONFERENCE | WASHINGTON, D.C.
1

What you get with the JS API

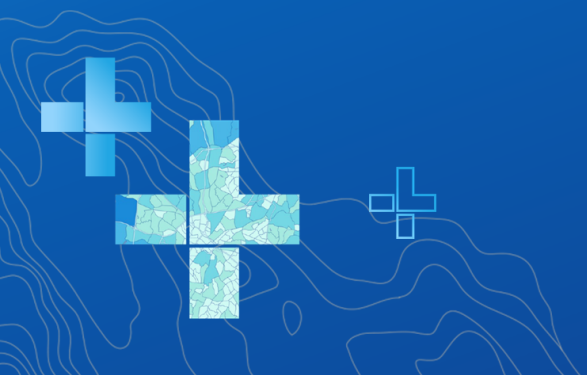
- Simplified and consistent web mapping API
- Wraps and consumes ArcGIS Services, but...
- You can use other data types as well
  - GeoJSON, CSV, OGC, Feature Collections
- Write apps in ES6 or TypeScript
- Modern browser support (IE11+)
- Supported in 30+ locales

What you get with the JS API

- Technical support
- GeoNet Community
- Excellent documentation and samples
  - Links from the sandbox
- Blogs, videos, and other resources

Fundamentals

...and some patterns



Map and View

Map and View

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

// 2d
const mView = new MapView({
  map: map,
  container: "viewDiv"
});
//3d
const sView = new SceneView({
  map: map,
```

Basemaps and Ground

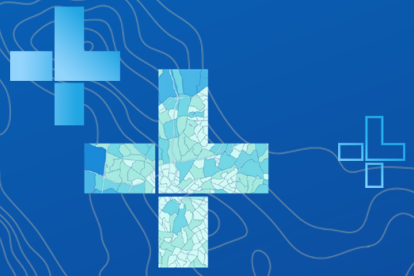
- Convenience Strings

```
const map = new Map({
  /*
   streets, satellite, hybrid, terrain, topo, gray,
   dark-gray, oceans, national-geographic, osm,
   dark-gray-vector, gray-vector, streets-vector, topo-vector,
   streets-night-vector, streets-relief-vector, streets-navigation-vector
   */
  basemap: "streets"

  /*
   world-elevation
   */
```

Layer and LayerView

20+ layer types!

- FeatureLayer
- MapImageLayer
- CSVLayer
- GeoJSONLayer
- WMSLayer, WMTSLayer
- TileLayer
- ...



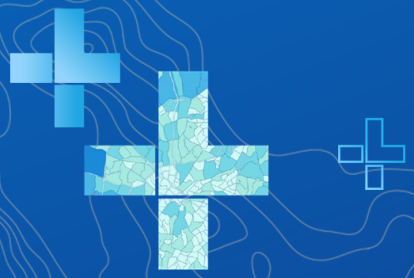
FeatureLayer

```
const layer = new FeatureLayer({
  url: "https://services.arcgis.com/V6ZHFr6zdgNZuVG0/arcgis/rest/services/Enriched_United_Sta
});

const map = new Map({
  basemap: "topo",
  layers: [ layer ]
});

const mView = new MapView({
  map: map,
  container: "viewDiv"
```

Example

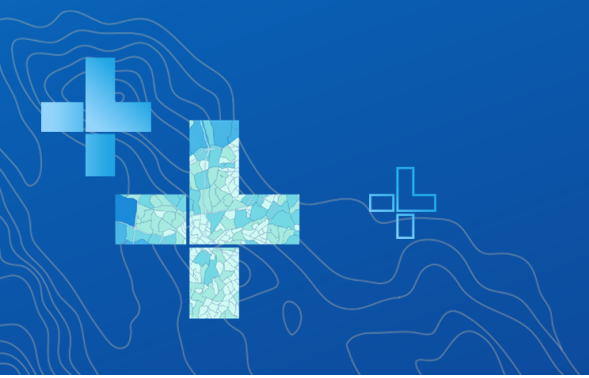
Popup Templates

```
layer.popupTemplate = {
  title: "{NAME} County",
  content: "{POP2020} people live here.",
  fieldInfos: [{
    fieldName: "POP2020",
    format: {
      places: 0,
      digitSeparator: true
    }
  }]
}
```



Popup Templates

- Define fields, charts, custom html content
- Using a function
- Promises
- Arcade



Renderers

```
layer.renderer = {
  type: "simple",
  visualVariables: [{
    type: "size",
    field: "POP2020",
    minSize: 6,
    maxSize: 50,
    minDataValue: 10000,
    maxDataValue: 1000000
  }]
};
```

Go to the Visualization with the ArcGIS API for JavaScript session later this week!

LayerView

- Renders the Layer
- Provides access to features on the client
- When is it done?

LayerView

LayerView - Ready

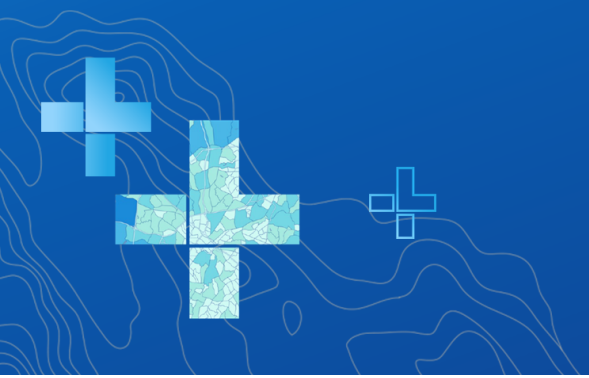
A PEN BY odoe

Run Pen

LayerView vs. Layer (Query)

The same methods execute queries client-side (layer view) and server side (layer)

```
*Layer/LayerView.queryFeatures()
*Layer/LayerView.queryFeatureCount()
*Layer/LayerView.queryExtent()
*Layer/LayerView.queryObjectIds()
```



LayerView vs. Layer (Query)

Same methods execute queries client-side (layer view) and server side (layer)

```
const layer = new FeatureLayer({ url: "https://..." });
const view = new MapView({
  map: new Map({
    layers: [ layer ]
  })
});

const query = layer.createQuery();
query.where = "Population > 40000";

// layer.queryFeatures queries the server
const response = await layer.queryFeatures(query);
```

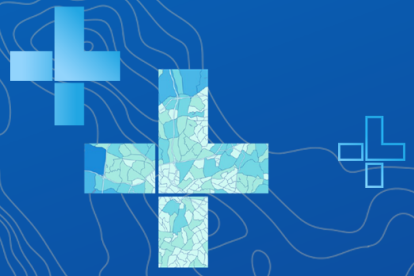
LayerView vs. Layer (Query)

Same methods execute queries client-side (layer view) and server side (layer)

```
const layer = new FeatureLayer({ url: "https://..." });
const view = new MapView({
  map: new Map({
    layers: [ layer ]
  })
});

const query = layer.createQuery();
query.where = "Population > 40000";

// layerView.queryFeatures queries the server
const layerView = await view.whenLayerView(layer) as FeatureLayerView;
```

createQuery

- When you can do `layer.createQuery()`
  - `query` object will already have the layers filters and layer definitions
  - more consistent
- Use `new Query()` when you don't want predefined filters to be applied



createQuery

createQuery

A PEN BY odoe

Run Pen

Basemaps and Ground

```
const map = new Map({
  basemap: {
    // Layers drawn at the bottom
    baseLayers: [
      new TileLayer({ url: baselayer })
    ],
    // Layers drawn on top
    referenceLayers: [
      new TileLayer({ url: refUrl })
    ],
  },
  ground: {
```

Basemap and Ground

VT Basemaps

A PEN BY odoe

Run Pen

Collections

- `esri/core/Collection`

Working with properties

Getting and setting

```
layer.opacity = 0.5;
layer.title = "My test layer";

// setting multiple values
layer.set({
  opacity: 0.5,
  title: "My test layer"
});

// accessing the value of a deep property
console.log(view.map.basemap.title);
view.set("map.basemap.title", "new title");
```

Working with properties

Watching (no events!)

```
mapView.watch("scale", (newValue, oldValue, property, target) => {
  console.log(`scale changed: ${newValue}`);
});


mapView.watch("map.basemap.title", (newValue, oldValue, property, target) => {
  console.log(`new basemap title: ${newValue}`);
});


mapView.watch("ready, stationary", (newValue, oldValue, property, target) => {
  console.log(`property ${property}: ${newValue}`);
```

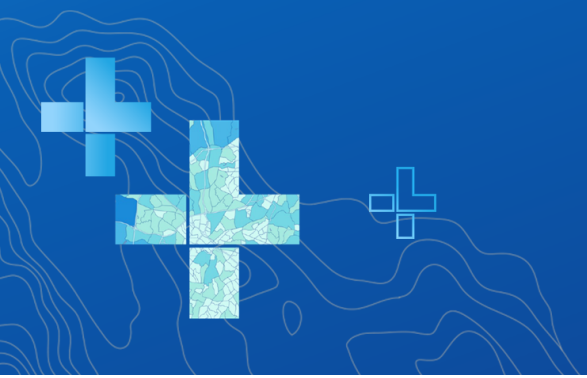
watchUtils

Working with properties

Autocasting and single constructor

```
// 4.x
layer.renderer = {
  type: "simple",
  symbol: {
    type: "simple-marker",
    style: "square",
    color: "red",
    size: 10,
    outline: {
      color: "rgba(255, 255, 255, 0.5)"
      width: 4
    }
```

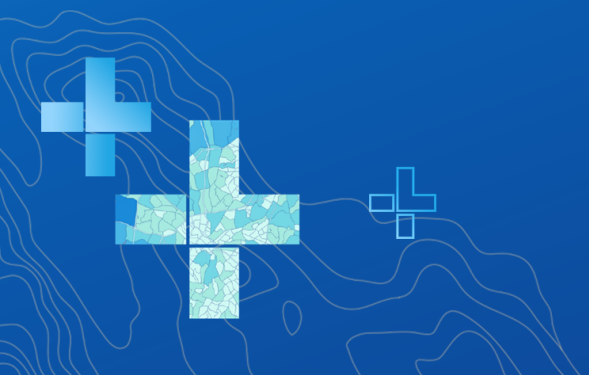
Promises



Promises

- All asynchronous methods return a promise
- No more events
- The basic pattern looks like this:

```
layer.queryFeatures(query).then(handleResult).catch(handleError);
```



Promises with async/await

```
const doQuery = async (query) => {
  const results = await layer.queryFeatures(query);
  const transformedResults = results.map(transformData);
  return transformedResults;
}
```

Promises

- Load resources
- Asychronously initialized `Layer`, `WebMap`, `WebScene`, `View`

```
const map = new Map({...})

const view = new SceneView({
  map: map,
  //...
});

view.when().then(() => {
  // the view is ready to go
});
```

Promises

```
view.when().then(() => {
  return view.whenLayerView(map.findLayerById("awesomeLayer"));
})
.then(layerView => {
  return watchUtils.whenFalseOnce(layerView, "updating");
})
.then(result => {
  const layerView = result.target;
  return layerView.queryFeatures();
})
.then(doSomethingWithFeatures)
.catch(errorHandler);
```

API sample

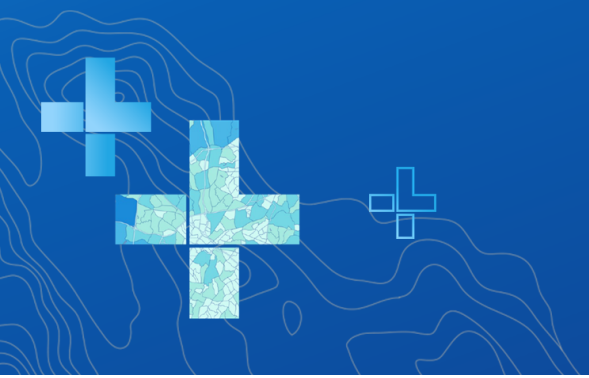
async/await

```
const init = async (doSomethingWithFeatures) => {
  await view.when();
  const layerView = await view.whenLayerView(map.findLayerById("awesomeLayer"));
  const { target as layerView } = await watchUtils.whenFalseOnce(layerView, "updating");
  const features = await layerView.queryFeatures();
  doSomethingWithFeatures(features);
};

try {
  init();
}
catch(error) {
```

Zoom or Scale

```
const view = new MapView({
  container: "viewDiv",
  map: map,
  center: [-116.5, 33.80],
  zoom: 14 // what does that really mean?
});
```

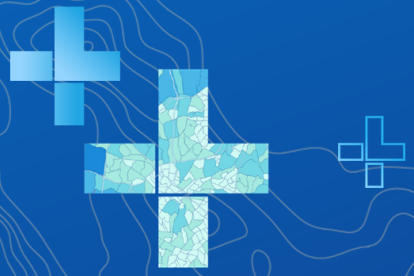
- Zoom = LOD (Level of Detail)
- Not all LODs are created equal



Zoom is not Scale

```
const view = new MapView({
  container: "viewDiv",
  map: map,
  center: [-116.5, 33.80],
  scale: 50000 // I know what that means!
});
```

- Scale is portable
- Scale has meaning
- We still snap to closest LOD/zoom, unless you disable it

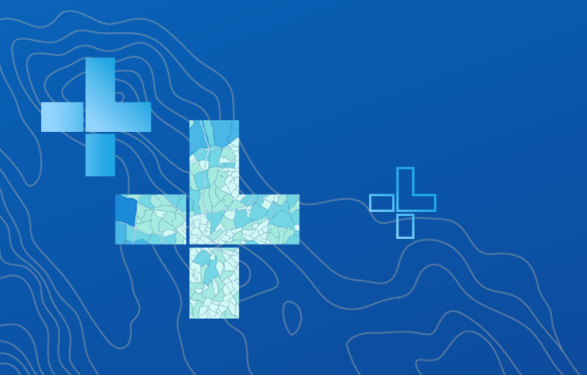


goTo() with View

- Sets the view to a given target.
  - Navigate to a geometry/feature/location
- API Sample

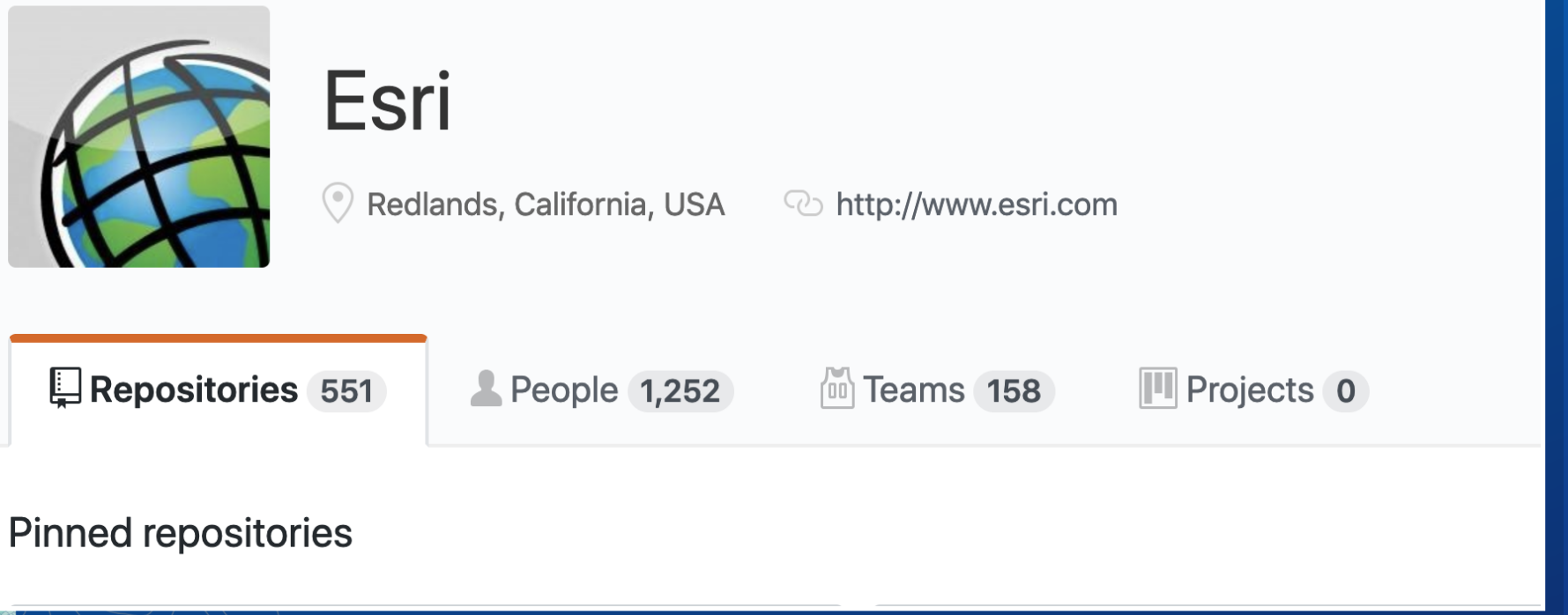
Patterns

Things you can do to reduce the amount of code you write



How the ArcGIS JS API helps you

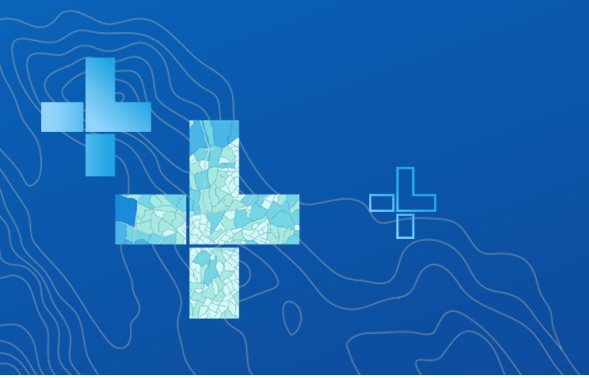
Open source app starters and templates



WebMap is still a Map

```
const map = new WebMap({
  basemap: { ... },
  layers: [ ... ]
});
```

- Still acts like a regular `Map`
- Has some BIG advantages



WebMap is still a Map

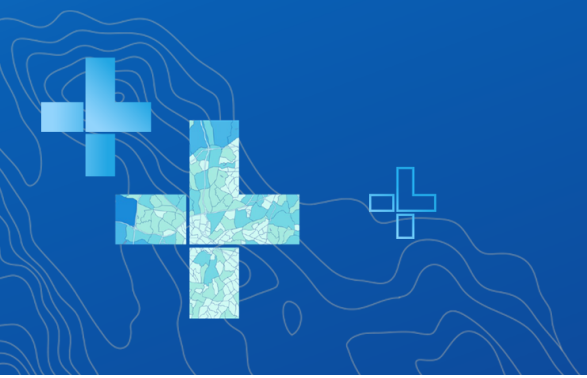
Local bookmarks

A PEN BY odoe

Run Pen

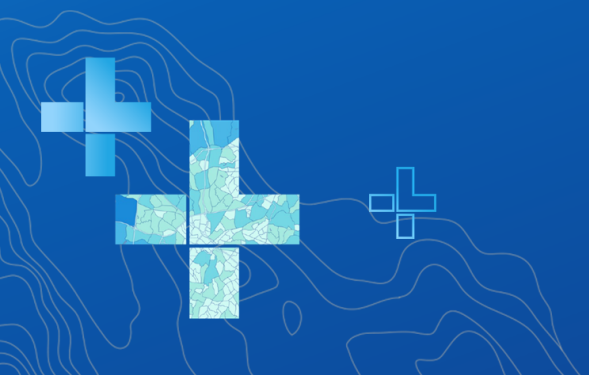
Widgets!

- We'll look at a few widgets



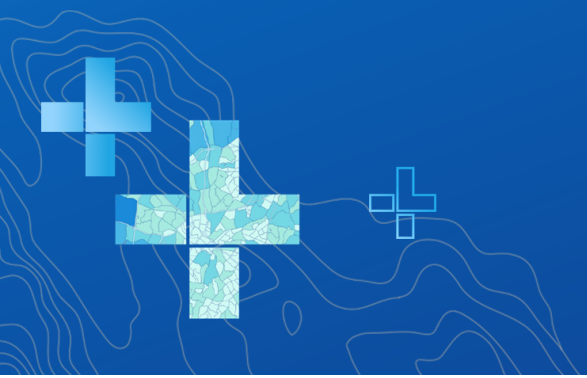
Widgets!

- We'll look at a few widgets
- ~50 Widgets out of the box



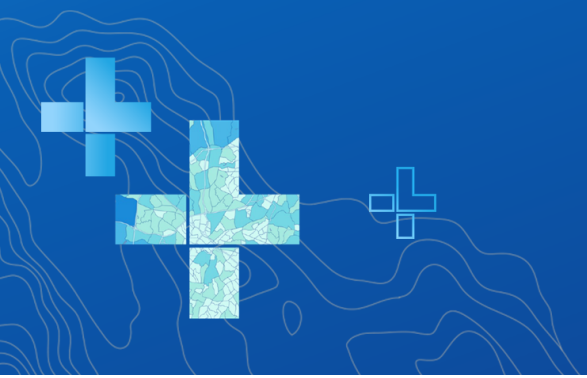
Widgets!

- We'll look at a few widgets
- ~50 Widgets out of the box
- Widgets help make great apps



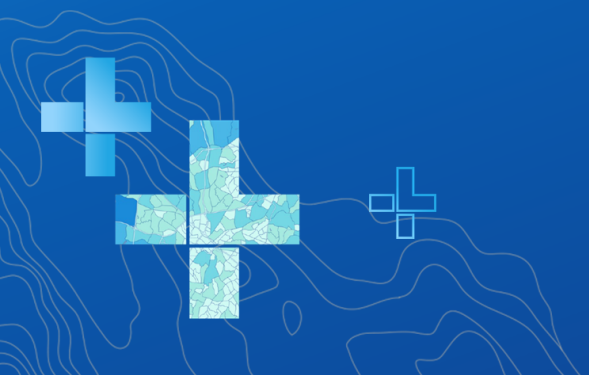
Widgets!

- We'll look at a few widgets
- ~50 Widgets out of the box
- Widgets help make great apps
- Less code for you to write



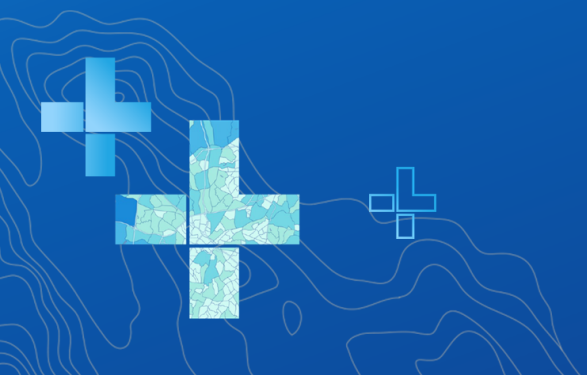
Widgets!

- We'll look at a few widgets
- ~50 Widgets out of the box
- Widgets help make great apps
- Less code for you to write
- Designed with responsive apps in mind



Widgets - Expand

- Clickable button to open container
- Icons
- Group
- Mode



Widgets - Use Portal Content

- Search
- Basemap Gallery

Widgets - Architecture

View + View Model

SearchViewModel

Constructors | Properties | Methods | Type definitions | Events

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

Class: `esri/widgets/Search/SearchViewModel`

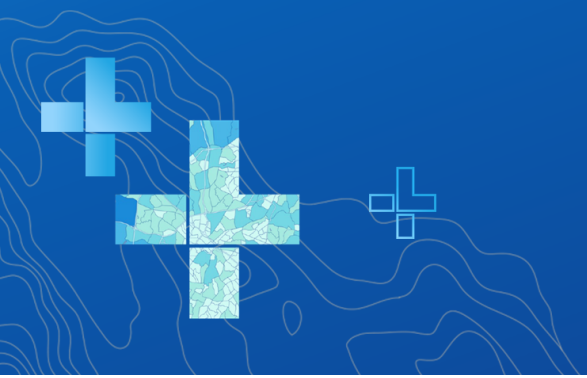
Inheritance: SearchViewModel → Accessor

Since: ArcGIS API for JavaScript 4.0

Provides the logic for the Search widget, which performs search operations on locator service(s) and/or map/feature service feature layer(s). If using a locator with a geocoding service, the findAddressCandidates operation is used, whereas queries are used on feature layers.

View Models

- Custom View
- Use the view model
  - Additional Examples



Widgets - Styling

Available Themes

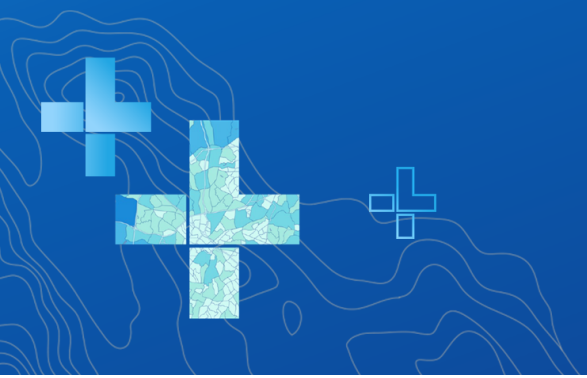
Theme Testing

A PEN BY odoe

Run Pen

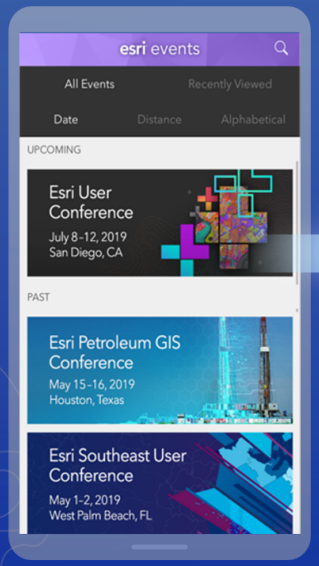
Widgets - Styling

- CSS Extension language
- SASS
- Theme Utility

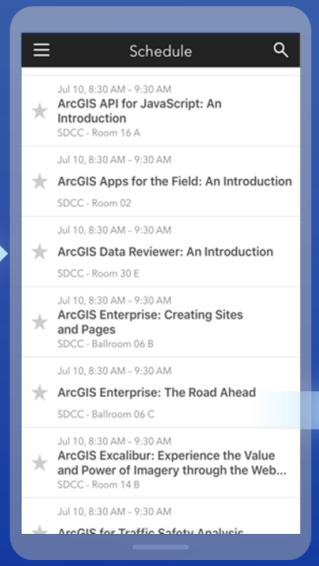


Please Share Your Feedback in the App

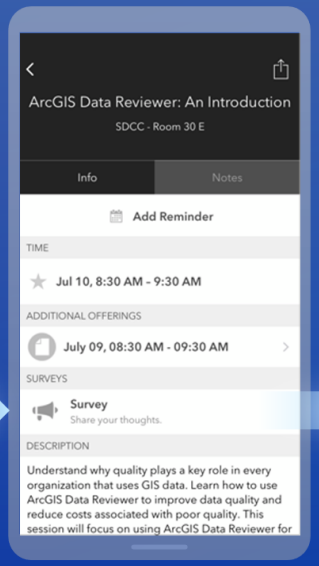
Download the Esri Events app and find your event



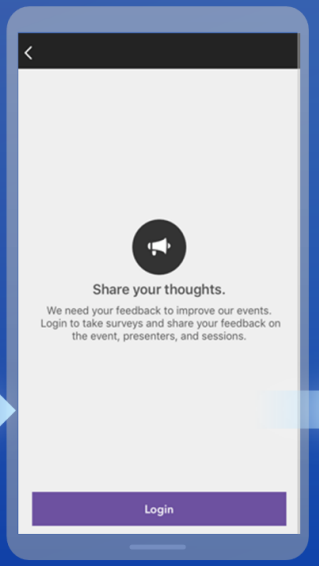
Select the session you attended



Scroll down to "Survey"



Log in to access the survey



Complete the survey and select "Submit"