Widgets are so passe are you ready for components?

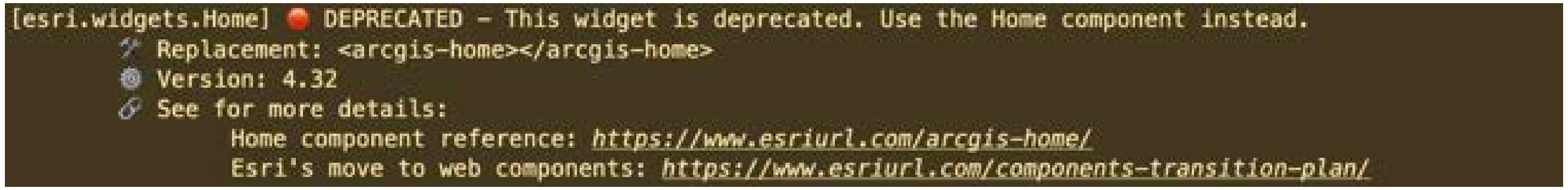
Texas GIS Forum – November 5th, 2025

Why this Topic?

- For many years ESRI has used JavaScript-based widgets to bundle up functionality to work with its technologies and ArcGIS Maps SDK for JavaScript.

Why - continued

- Over a decade in fact that they’ve been doing this pattern. It started with the first framework they adopted to do custom ‘thin JavaScript clients’ for the web – Dojo.



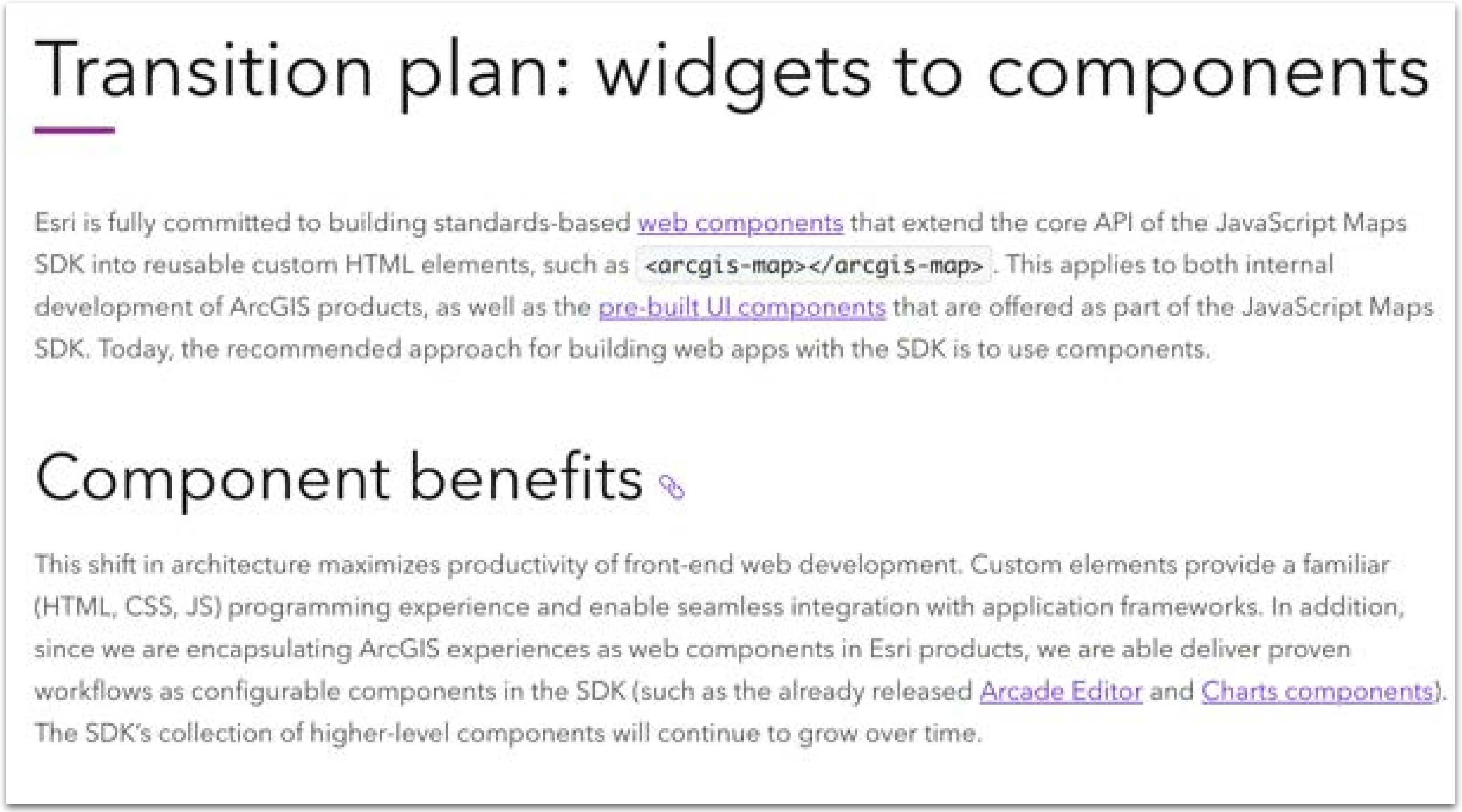
```
[esri.widgets.Home] 🔴 DEPRECATED - This widget is deprecated. Use the Home component instead.
        🛠️ Replacement: <arcgis-home></arcgis-home>
        ⚙️ Version: 4.32
        🔗 See for more details:
                Home component reference: https://www.esriurl.com/arcgis-home/
                Esri's move to web components: https://www.esriurl.com/components-transition-plan/
```

- This last spring while teaching a Web GIS Scripting course at ACC, I started getting warnings like these

Status Quo Shake Up!

Why - continued

- The link in that message led me to this post on ESRI's site...



Transition plan: widgets to components

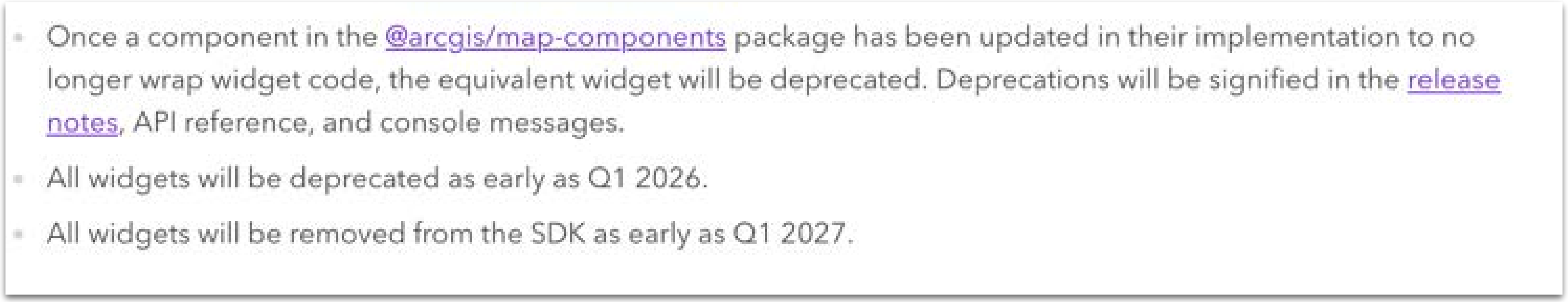
Esri is fully committed to building standards-based web components that extend the core API of the JavaScript Maps SDK into reusable custom HTML elements, such as `<arcgis-map></arcgis-map>`. This applies to both internal development of ArcGIS products, as well as the pre-built UI components that are offered as part of the JavaScript Maps SDK. Today, the recommended approach for building web apps with the SDK is to use components.

Component benefits

This shift in architecture maximizes productivity of front-end web development. Custom elements provide a familiar (HTML, CSS, JS) programming experience and enable seamless integration with application frameworks. In addition, since we are encapsulating ArcGIS experiences as web components in Esri products, we are able deliver proven workflows as configurable components in the SDK (such as the already released Arcade Editor and Charts components). The SDK's collection of higher-level components will continue to grow over time.

When!!!!???

- Another part of that post had this for a timeline



- Once a component in the @arcgis/map-components package has been updated in their implementation to no longer wrap widget code, the equivalent widget will be deprecated. Deprecations will be signified in the release notes, API reference, and console messages.
- All widgets will be deprecated as early as Q1 2026.
- All widgets will be removed from the SDK as early as Q1 2027.

Q1 2026

- Yes. We are ALREADY in Q1 2026 (started September 2025)

Announcement === Big Deal!!

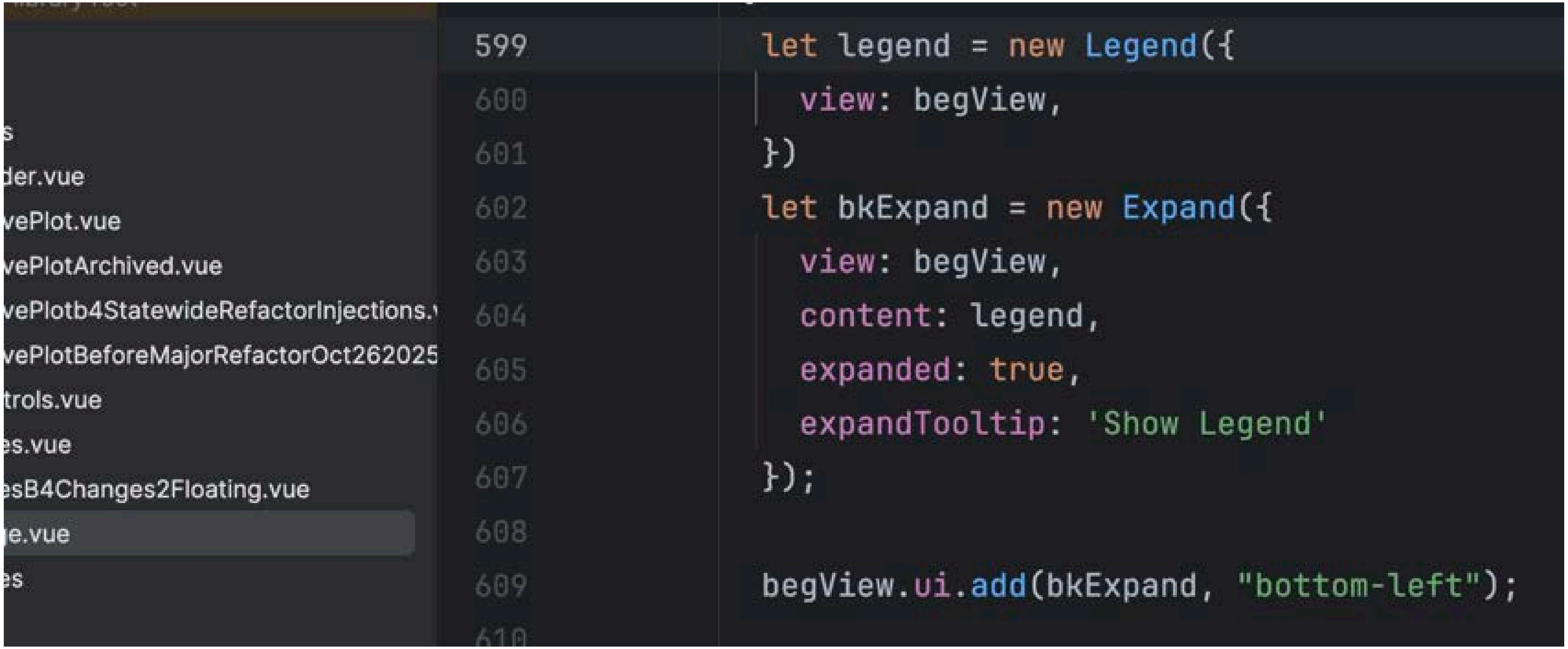
- This was a big deal!
- A change in the overall pattern or philosophy of how ESRI was going to provide bundled functionality by building custom html-based components

Broader Implications

- I teach and have taught GIS folks in workforce GIS since 2010 at ACC – and the patterns and ideas.
- HYPERBOLE ALERT - I realized that component-based access was in some ways going to shake the foundations of how folks get into GIS web scripting. Component-based is by nature a more complex coding pattern than widgets.

Let’s Compare

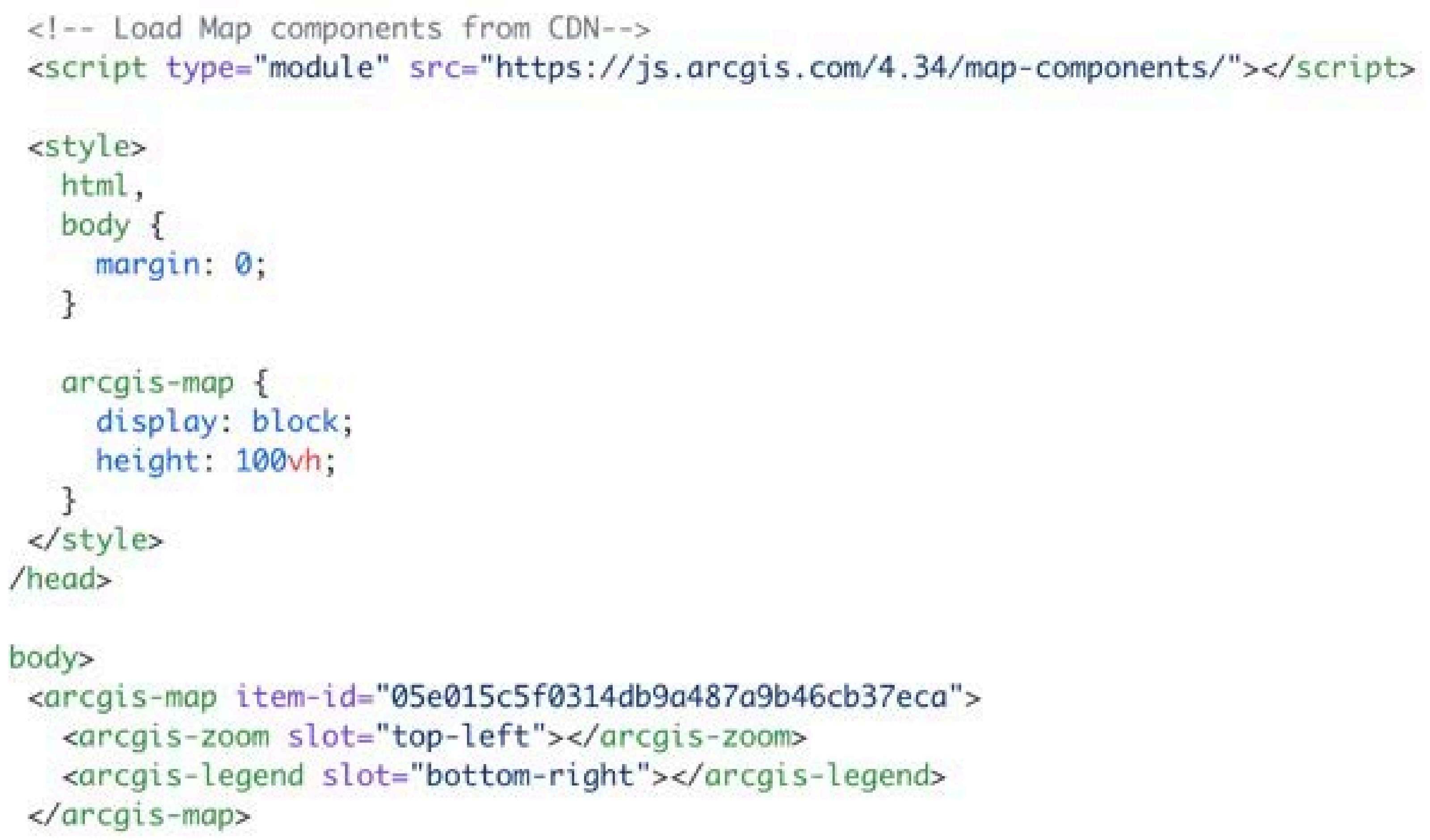
Old Widget method



```
let legend = new Legend({
  view: begView,
})
let bkExpand = new Expand({
  view: begView,
  content: legend,
  expanded: true,
  expandTooltip: 'Show Legend'
});

begView.ui.add(bkExpand, "bottom-left");
```

Let’s Compare

New Component Method



```
 <!-- Load Map components from CDN-->
 <script type="module" src="https://js.arcgis.com/4.34/map-components/"></script>

 <style>
   html,
   body {
     margin: 0;
   }

   arcgis-map {
     display: block;
     height: 100vh;
   }
 </style>
/head>

body>
 <arcgis-map item-id="05e015c5f0314db9a487a9b46cb37eca">
   <arcgis-zoom slot="top-left"></arcgis-zoom>
   <arcgis-legend slot="bottom-right"></arcgis-legend>
 </arcgis-map>
```

Wait a minute…

That’s not really so bad is it?

I am kidding here…

I am showing you how components are sold – “easy peasy”

Just add a tag and go!

Ok That’s It

Do you think that’s it?

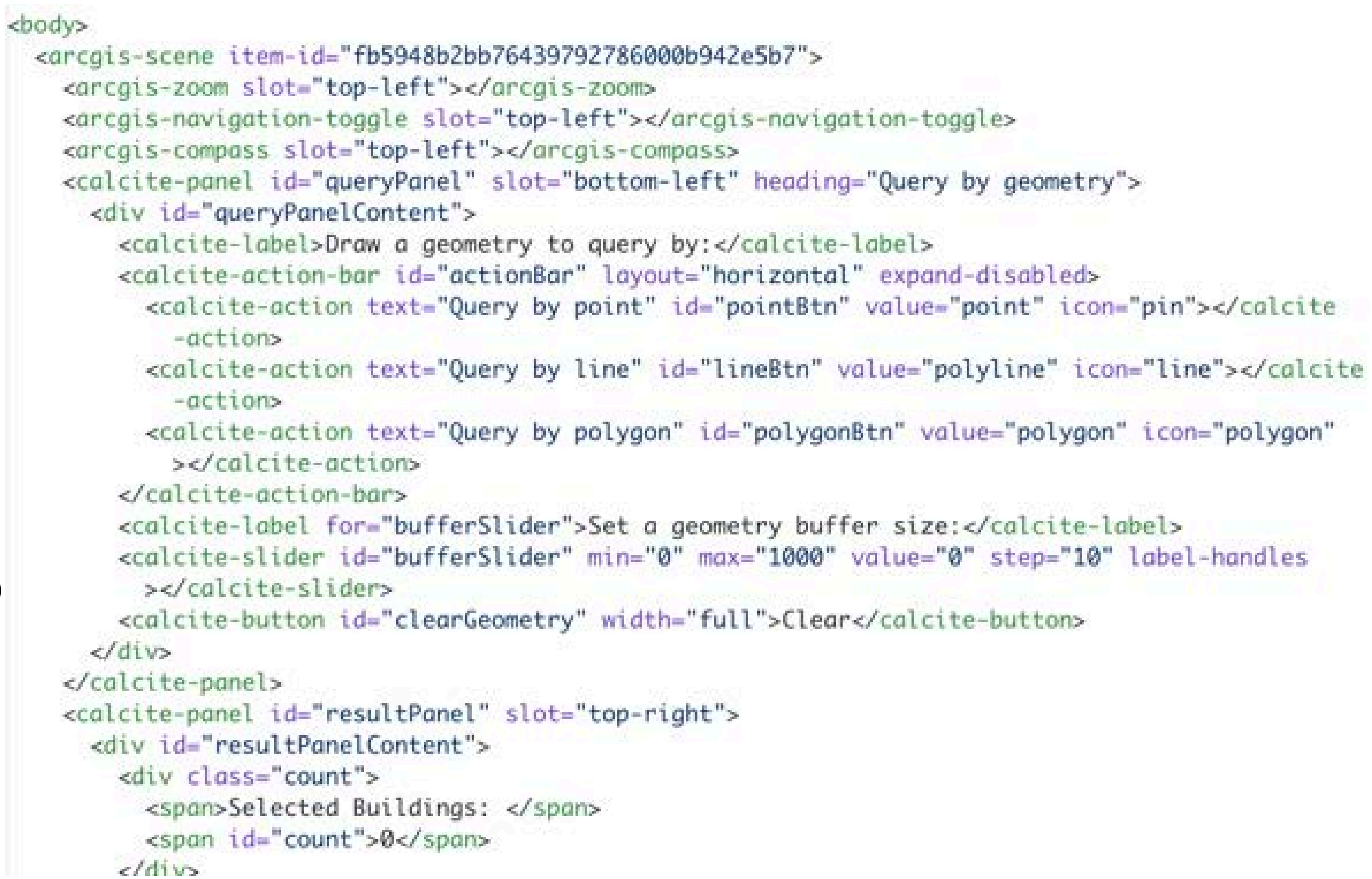
Reality Check!

New Component Method

Of course not…

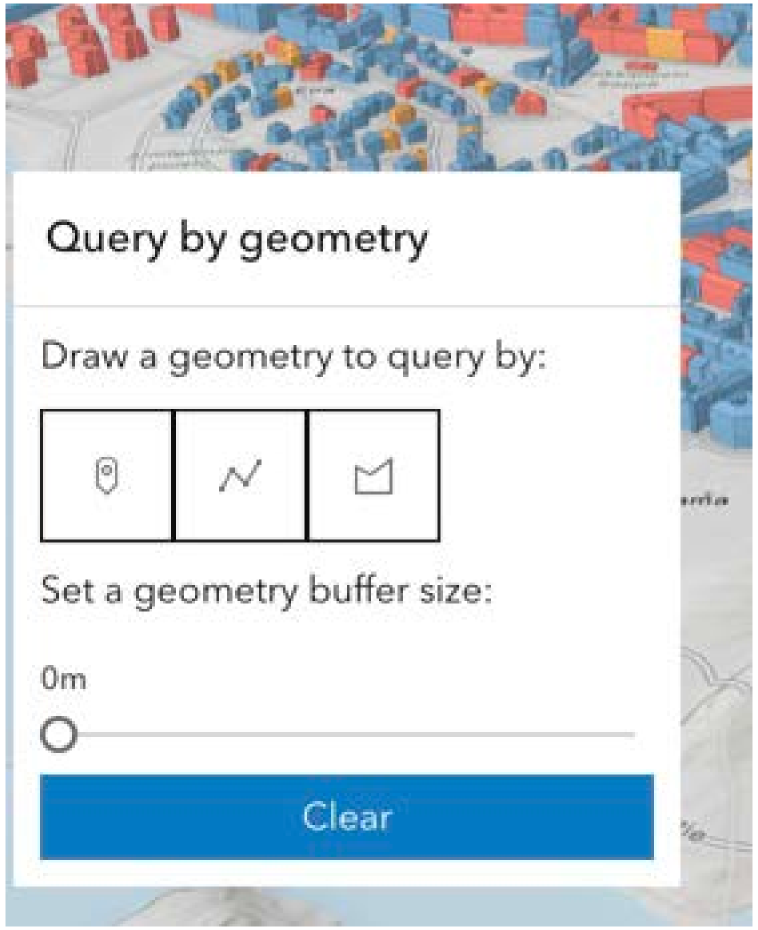
because doing something more complex becomes much more difficult.

Check out this pattern to include some graphics tools
to map app



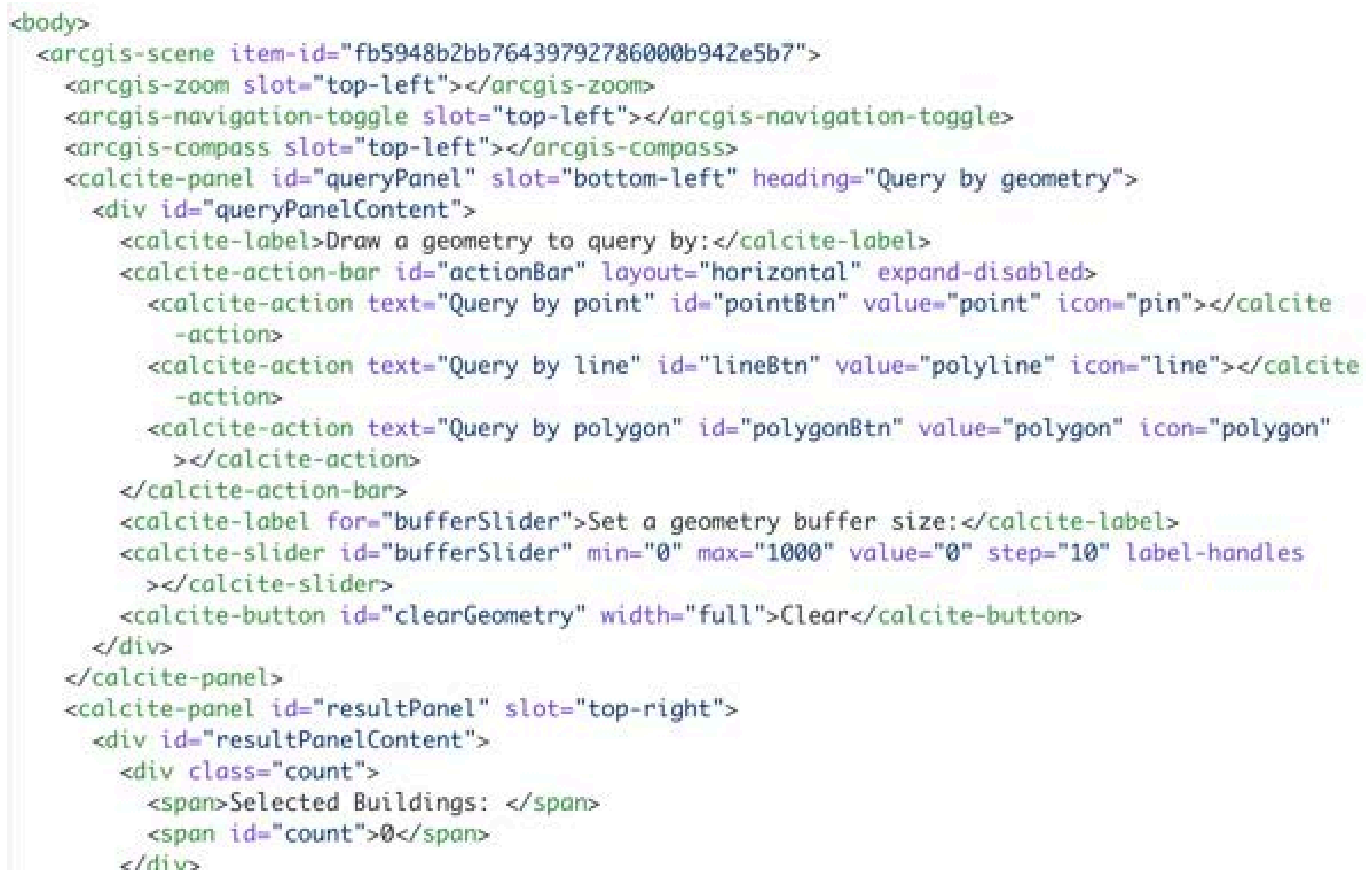
```
<body>
  <arcgis-scene item-id="fb5948b2bb76439792786000b942e5b7">
    <arcgis-zoom slot="top-left"></arcgis-zoom>
    <arcgis-navigation-toggle slot="top-left"></arcgis-navigation-toggle>
    <arcgis-compass slot="top-left"></arcgis-compass>
    <calcite-panel id="queryPanel" slot="bottom-left" heading="Query by geometry">
      <div id="queryPanelContent">
        <calcite-label>Draw a geometry to query by:</calcite-label>
        <calcite-action-bar id="actionBar" layout="horizontal" expand-disabled>
          <calcite-action text="Query by point" id="pointBtn" value="point" icon="pin"></calcite
            -action>
          <calcite-action text="Query by line" id="lineBtn" value="polyline" icon="line"></calcite
            -action>
          <calcite-action text="Query by polygon" id="polygonBtn" value="polygon" icon="polygon"
            ></calcite-action>
        </calcite-action-bar>
        <calcite-label for="bufferSlider">Set a geometry buffer size:</calcite-label>
        <calcite-slider id="bufferSlider" min="0" max="1000" value="0" step="10" label-handles
          ></calcite-slider>
        <calcite-button id="clearGeometry" width="full">Clear</calcite-button>
      </div>
    </calcite-panel>
    <calcite-panel id="resultPanel" slot="top-right">
      <div id="resultPanelContent">
        <div class="count">
          <span>Selected Buildings: </span>
          <span id="count">0</span>
        </div>
```

Gives You This

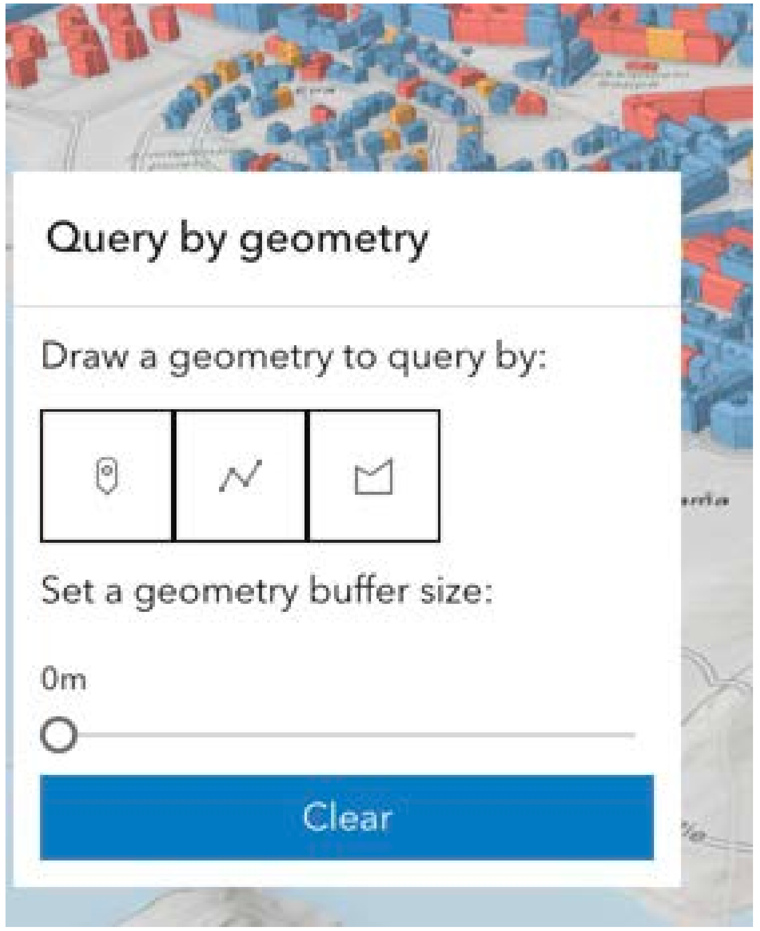


All Together

New Component Method



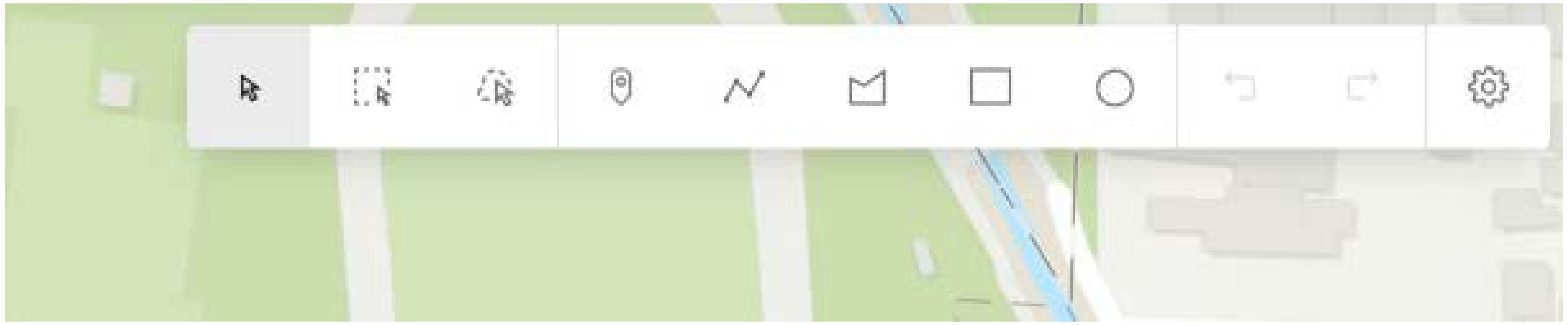
```
<body>
  <arcgis-scene item-id="fb5948b2bb76439792786000b942e5b7">
    <arcgis-zoom slot="top-left"></arcgis-zoom>
    <arcgis-navigation-toggle slot="top-left"></arcgis-navigation-toggle>
    <arcgis-compass slot="top-left"></arcgis-compass>
    <calcite-panel id="queryPanel" slot="bottom-left" heading="Query by geometry">
      <div id="queryPanelContent">
        <calcite-label>Draw a geometry to query by:</calcite-label>
        <calcite-action-bar id="actionBar" layout="horizontal" expand-disabled>
          <calcite-action text="Query by point" id="pointBtn" value="point" icon="pin"></calcite
            -action>
          <calcite-action text="Query by line" id="lineBtn" value="polyline" icon="line"></calcite
            -action>
          <calcite-action text="Query by polygon" id="polygonBtn" value="polygon" icon="polygon"
            ></calcite-action>
        </calcite-action-bar>
        <calcite-label for="bufferSlider">Set a geometry buffer size:</calcite-label>
        <calcite-slider id="bufferSlider" min="0" max="1000" value="0" step="10" label-handles
          ></calcite-slider>
        <calcite-button id="clearGeometry" width="full">Clear</calcite-button>
      </div>
    </calcite-panel>
    <calcite-panel id="resultPanel" slot="top-right">
      <div id="resultPanelContent">
        <div class="count">
          <span>Selected Buildings: </span>
          <span id="count">0</span>
        </div>
```



And for comparison - Widget

Widget Method

```
    const graphicsLayer = new GraphicsLayer();

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

    const view = new MapView({
      container: "viewDiv",
      map: map,
      zoom: 18,
      center: [139.5716, 35.6964],
    });

    view.when(() => {
      const sketch = new Sketch({
        layer: graphicsLayer,
        view: view,
        // graphic will be selected as soon as it is created
        creationMode: "update",
      });

      view.ui.add(sketch, "top-right");
    });
  </script>
</head>

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



The two philosophies are vastly different!!!

- There is a large increase in the *wordiness* of the component method – or you might say its syntax is *crunchier*.
- There are benefits to this though you have finer grain control but if you don’t want it

OR

- You aren’t ready to deal with that level of detail it can be overwhelming

Let’s take a stroll

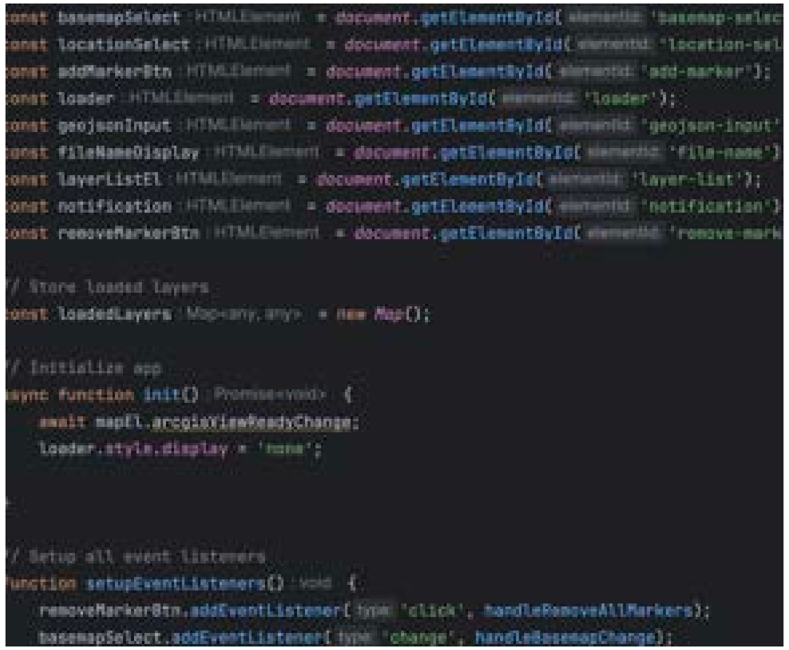
- Let’s walk through some of the patterns and see if there are some better pathways or at least more intuitive ways to get started using the new component views

Ready?!?!

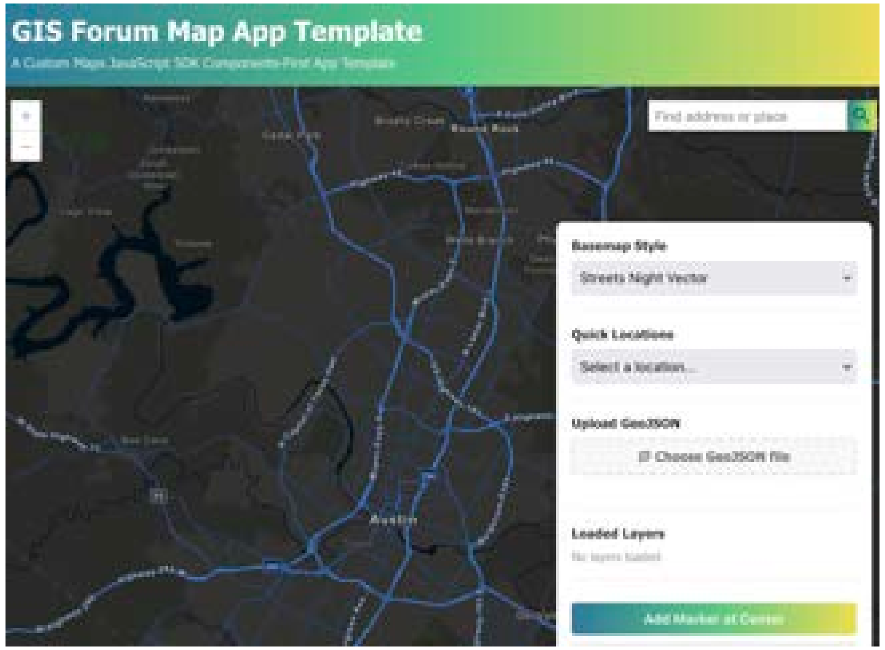
- Any Questions so far?

Once more unto the Breach then, Dear GISers!

- The comparison code I will be displaying/showcasing is a template that I am making public up on github https://github.com/brentporter/ComponentESRIStarter



```
onst basemapSelect :HTMLElement  = document.getElementById( elementId: 'basemap-selec
onst locationSelect :HTMLElement  = document.getElementById( elementId: 'location-sel
onst addMarkerBtn :HTMLElement  = document.getElementById( elementId: 'add-marker');
onst loader :HTMLElement  = document.getElementById( elementId: 'loader');
onst geojsonInput :HTMLElement  = document.getElementById( elementId: 'geojson-input'
onst fileNameDisplay :HTMLElement  = document.getElementById( elementId: 'file-name')
onst layerListEl :HTMLElement  = document.getElementById( elementId: 'layer-list');
onst notification :HTMLElement  = document.getElementById( elementId: 'notification')
onst removeMarkerBtn :HTMLElement  = document.getElementById( elementId: 'remove-mark

/ Store loaded layers
onst loadedLayers :Map<any, any>  = new Map();

/ Initialize app
sync function init() :Promise<void>  {
   await mapEl.arcgisViewReadyChange;
   loader.style.display = 'none';


/ Setup all event listeners
unction setupEventListeners() :void  {
   removeMarkerBtn.addEventListener( type: 'click', handleRemoveAllMarkers);
   basemapSelect.addEventListener( type: 'change', handleBasemapChange);
```

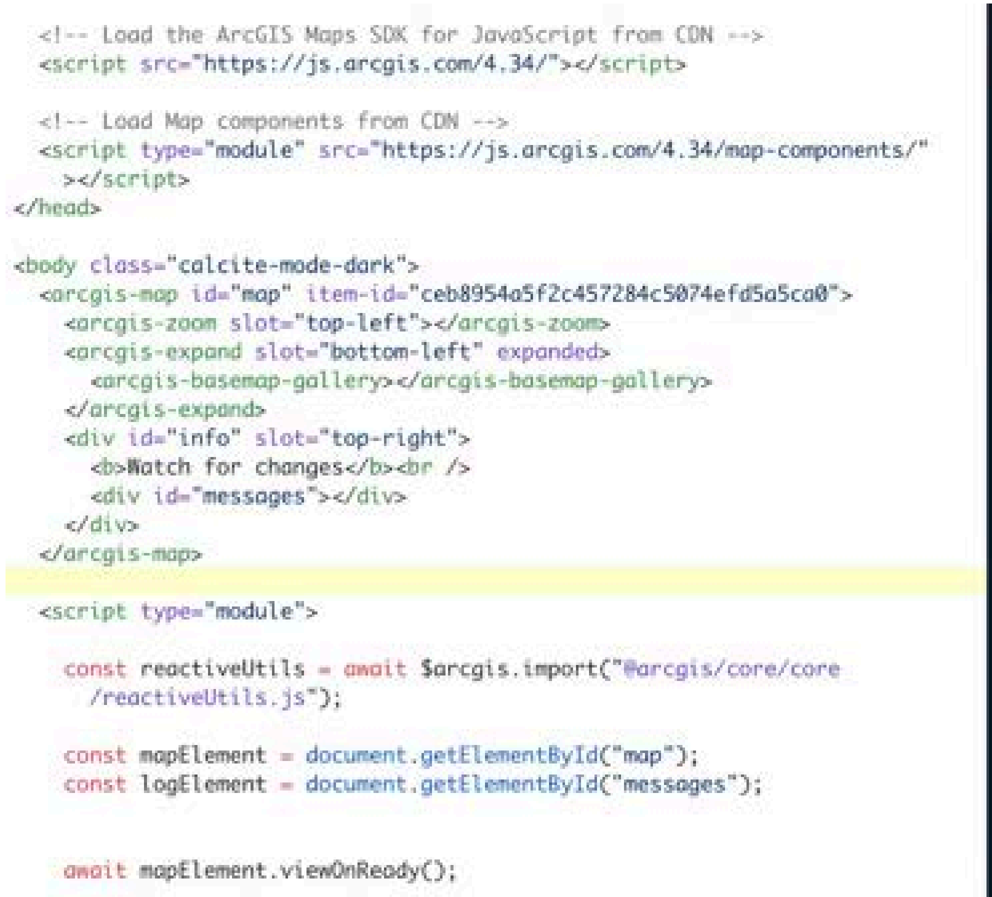


GIS Forum Map App Template

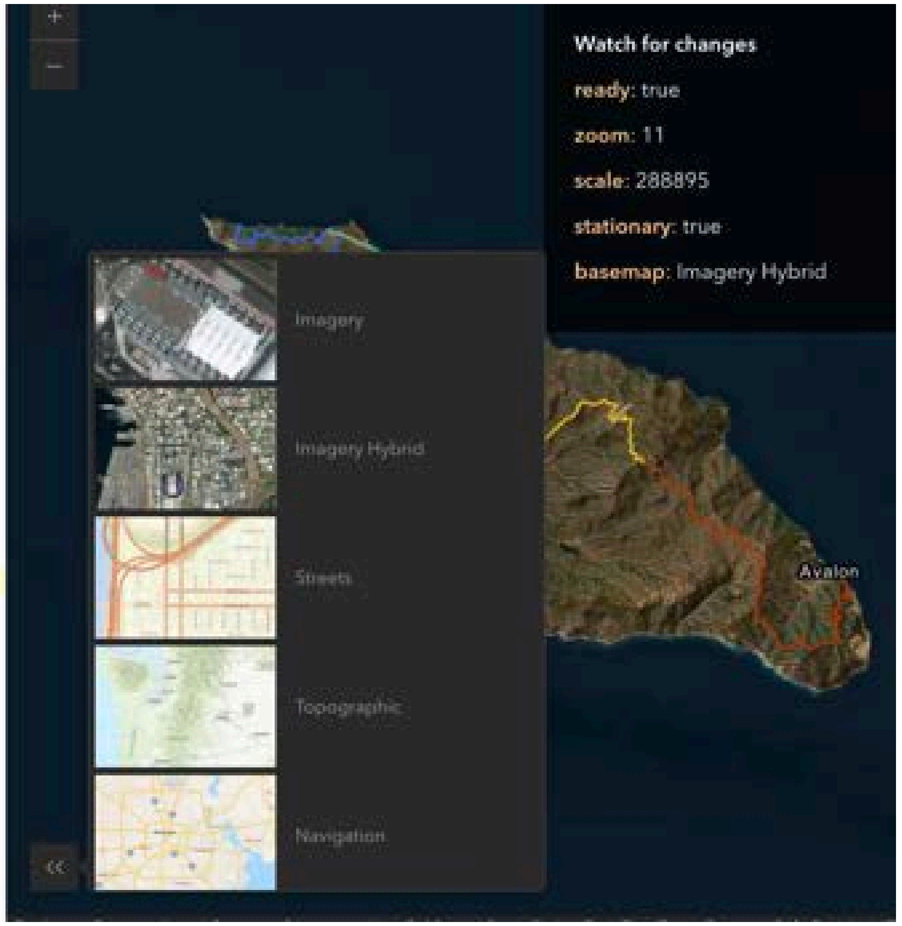
A Custom Maps JavaScript SDK Components-First App Template

Default ESRI Starter Code

- Sample App from Maps SDK Site using Components



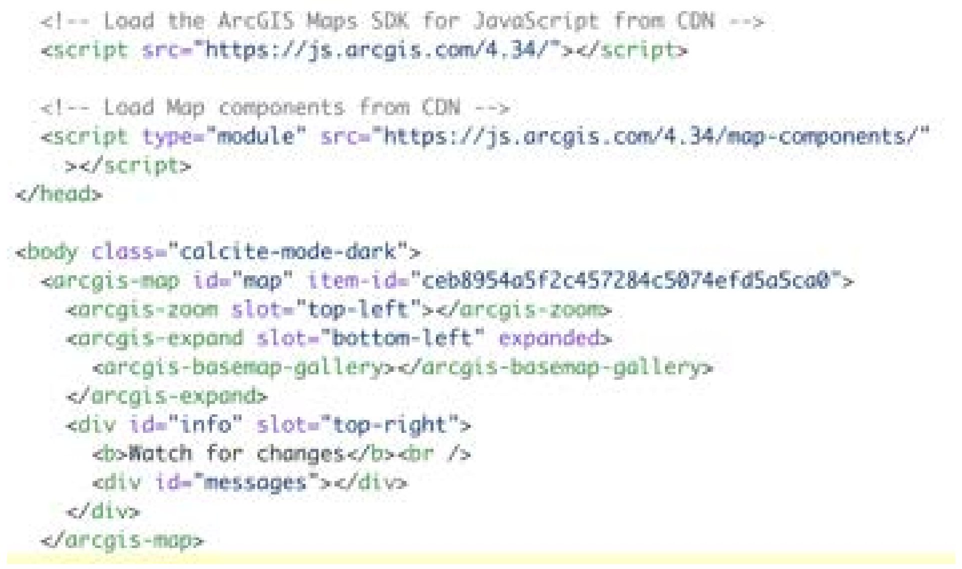
```
  <!-- Load the ArcGIS Maps SDK for JavaScript from CDN -->
  <script src="https://js.arcgis.com/4.34/"></script>

  <!-- Load Map components from CDN -->
  <script type="module" src="https://js.arcgis.com/4.34/map-components/"
    ></script>
</head>

<body class="calcite-mode-dark">
  <arcgis-map id="map" item-id="ceb8954a5f2c457284c5074efd5a5ca0">
    <arcgis-zoom slot="top-left"></arcgis-zoom>
    <arcgis-expand slot="bottom-left" expanded>
      <arcgis-basemap-gallery></arcgis-basemap-gallery>
    </arcgis-expand>
    <div id="info" slot="top-right">
      <b>Watch for changes</b><br />
      <div id="messages"></div>
    </div>
  </arcgis-map>

  <script type="module">

    const reactiveUtils = await $arcgis.import("@arcgis/core/core
      /reactiveUtils.js");

    const mapElement = document.getElementById("map");
    const logElement = document.getElementById("messages");


    await mapElement.viewOnReady();
```



Watch for changes
ready: true
zoom: 11
scale: 288895
stationary: true
basemap: Imagery Hybrid

First Peeve

- It is very difficult to find samples of ESRI’s components that DON’T use Portal IDs…



```
  <!-- Load the ArcGIS Maps SDK for JavaScript from CDN -->
  <script src="https://js.arcgis.com/4.34/"></script>

  <!-- Load Map components from CDN -->
  <script type="module" src="https://js.arcgis.com/4.34/map-components/"
    ></script>
</head>

<body class="calcite-mode-dark">
  <arcgis-map id="map" item-id="ceb8954a5f2c457284c5074efd5a5ca0">
    <arcgis-zoom slot="top-left"></arcgis-zoom>
    <arcgis-expand slot="bottom-left" expanded>
      <arcgis-basemap-gallery></arcgis-basemap-gallery>
    </arcgis-expand>
    <div id="info" slot="top-right">
      <b>Watch for changes</b><br />
      <div id="messages"></div>
    </div>
  </arcgis-map>
```

Why you do this to me Dimi?

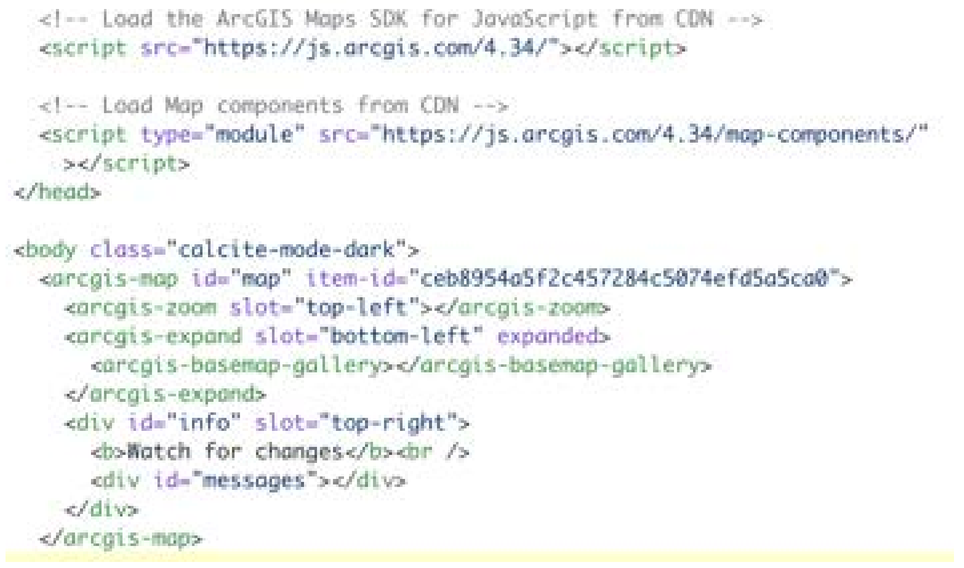
- Any Movie Buffs know what that’s from?

The Exorcist

This is how it makes me feel when I see portal ids in the sample code!

Why doesn’t Brent like Portal IDs

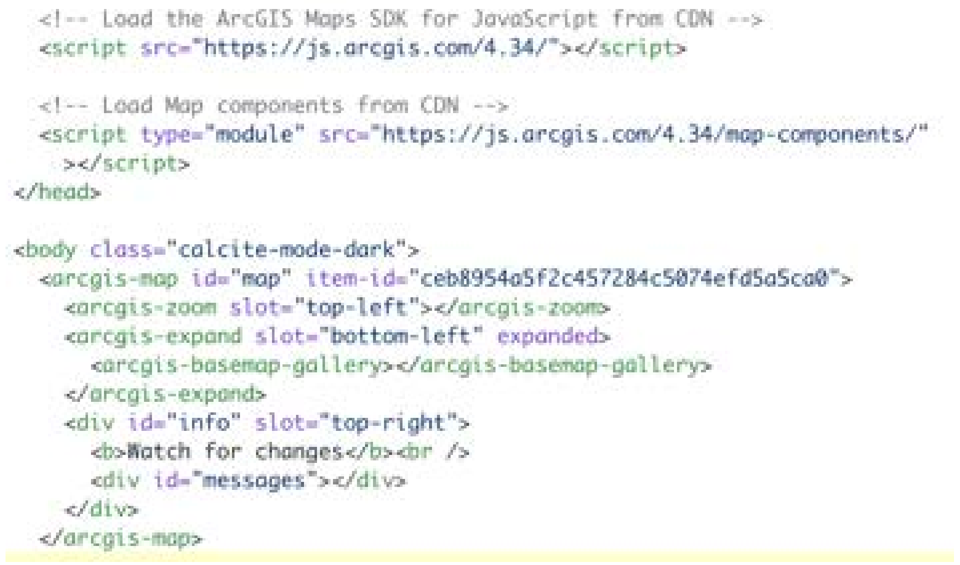
- Portal IDs have NO context!
- Does ceb8954a5f2c457284c5074efd5a5ca0 mean anything to you?
- It doesn’t mean anything to me either.



```
  <!-- Load the ArcGIS Maps SDK for JavaScript from CDN -->
  <script src="https://js.arcgis.com/4.34/"></script>

  <!-- Load Map components from CDN -->
  <script type="module" src="https://js.arcgis.com/4.34/map-components/"
    ></script>
</head>

<body class="calcite-mode-dark">
  <arcgis-map id="map" item-id="ceb8954a5f2c457284c5074efd5a5ca0">
    <arcgis-zoom slot="top-left"></arcgis-zoom>
    <arcgis-expand slot="bottom-left" expanded>
      <arcgis-basemap-gallery></arcgis-basemap-gallery>
    </arcgis-expand>
    <div id="info" slot="top-right">
      <b>Watch for changes</b><br />
      <div id="messages"></div>
    </div>
  </arcgis-map>
```

Why doesn’t Brent like Portal IDs

- Conspiracy Theory Alert!!!
- I think ESRI using portal AND portal ids are just a way to surreptiously push us all into using ArcGIS.com for our mapping platform.
- In Time… they say.

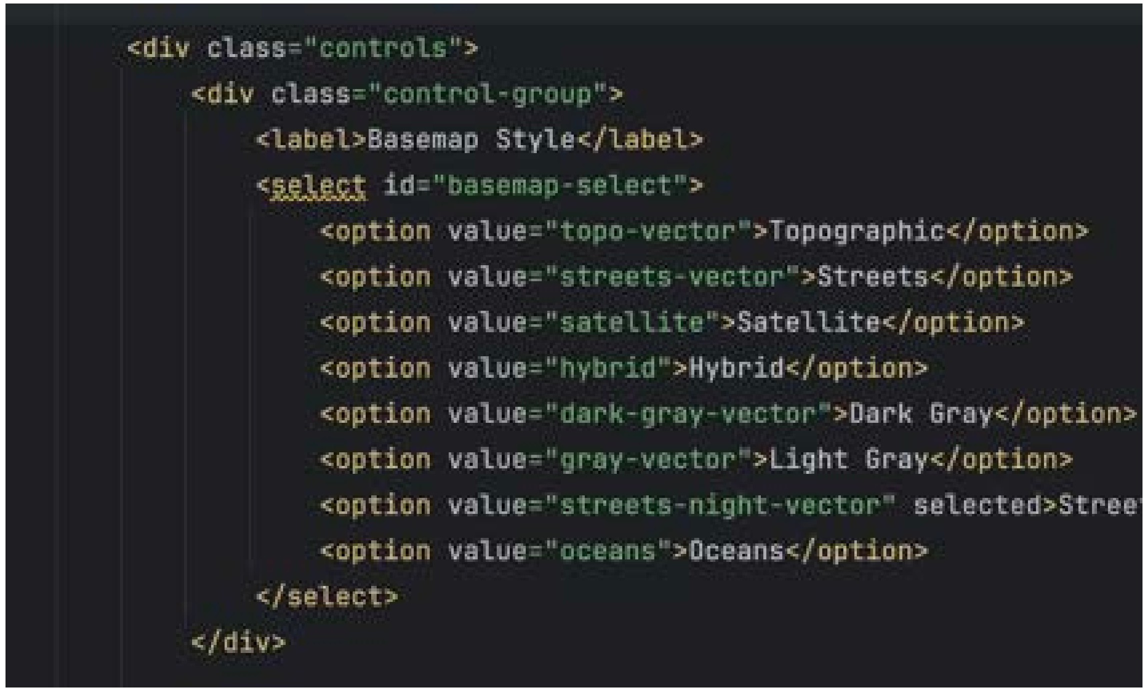


```
  <!-- Load the ArcGIS Maps SDK for JavaScript from CDN -->
  <script src="https://js.arcgis.com/4.34/"></script>

  <!-- Load Map components from CDN -->
  <script type="module" src="https://js.arcgis.com/4.34/map-components/"
    ></script>
</head>

<body class="calcite-mode-dark">
  <arcgis-map id="map" item-id="ceb8954a5f2c457284c5074efd5a5ca0">
    <arcgis-zoom slot="top-left"></arcgis-zoom>
    <arcgis-expand slot="bottom-left" expanded>
      <arcgis-basemap-gallery></arcgis-basemap-gallery>
    </arcgis-expand>
    <div id="info" slot="top-right">
      <b>Watch for changes</b><br />
      <div id="messages"></div>
    </div>
  </arcgis-map>
```

Ok Mr Tin Foil – What do we use instead??!!?

- Basemap IDs!

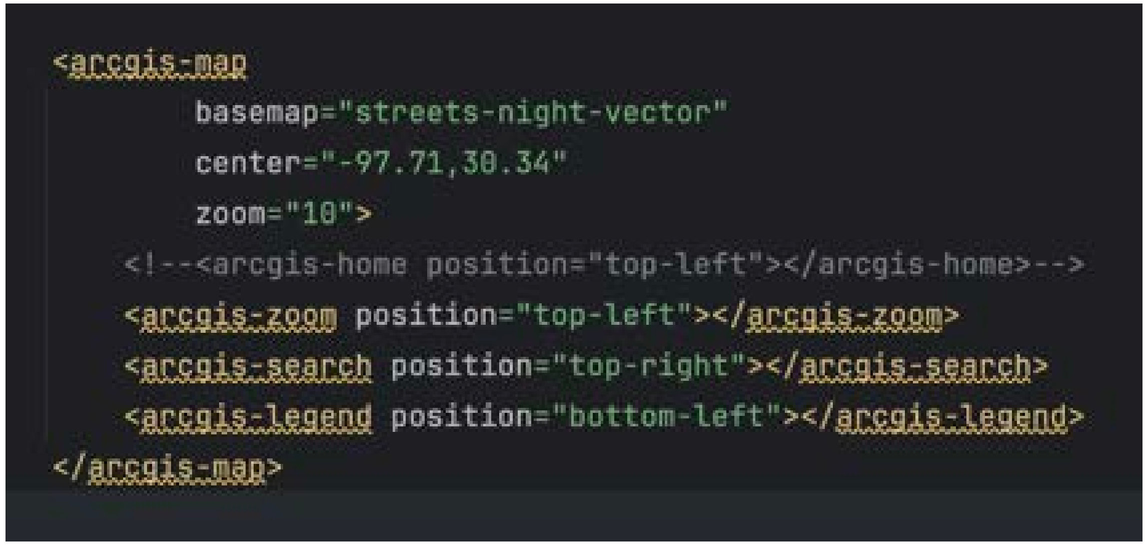
- These do have context! They make sense and are not as ephemeral as a generated id inside arcgis.com (or your own AGOO/portal)
- Examples
  - Topo-Vector
  - Satellite
  - More in the shot to the right
- PLAIN ENGLISH amiright?



```
<div class="controls">
    <div class="control-group">
        <label>Basemap Style</label>
        <select id="basemap-select">
            <option value="topo-vector">Topographic</option>
            <option value="streets-vector">Streets</option>
            <option value="satellite">Satellite</option>
            <option value="hybrid">Hybrid</option>
            <option value="dark-gray-vector">Dark Gray</option>
            <option value="gray-vector">Light Gray</option>
            <option value="streets-night-vector" selected>Stree
            <option value="oceans">Oceans</option>
        </select>
    </div>
```

These Values...

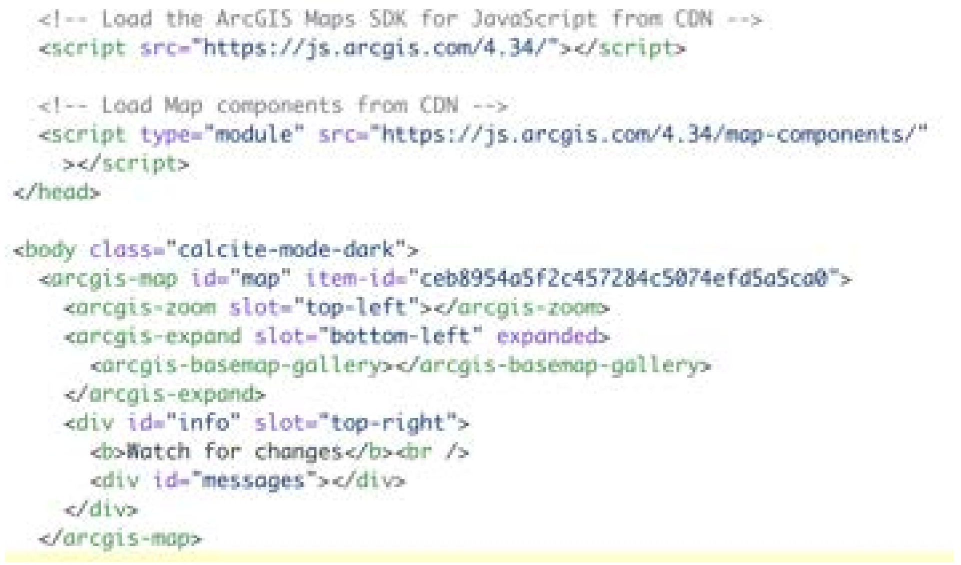
- Are in a select dropdown – a plain vanilla JS dropdown.
- And we can watch for them to change and swap it out in our map component!
- Much Better, imho



```
<arcgis-map
        basemap="streets-night-vector"
        center="-97.71,30.34"
        zoom="10">
    <!--<arcgis-home position="top-left"></arcgis-home>-->
    <arcgis-zoom position="top-left"></arcgis-zoom>
    <arcgis-search position="top-right"></arcgis-search>
    <arcgis-legend position="bottom-left"></arcgis-legend>
</arcgis-map>
```

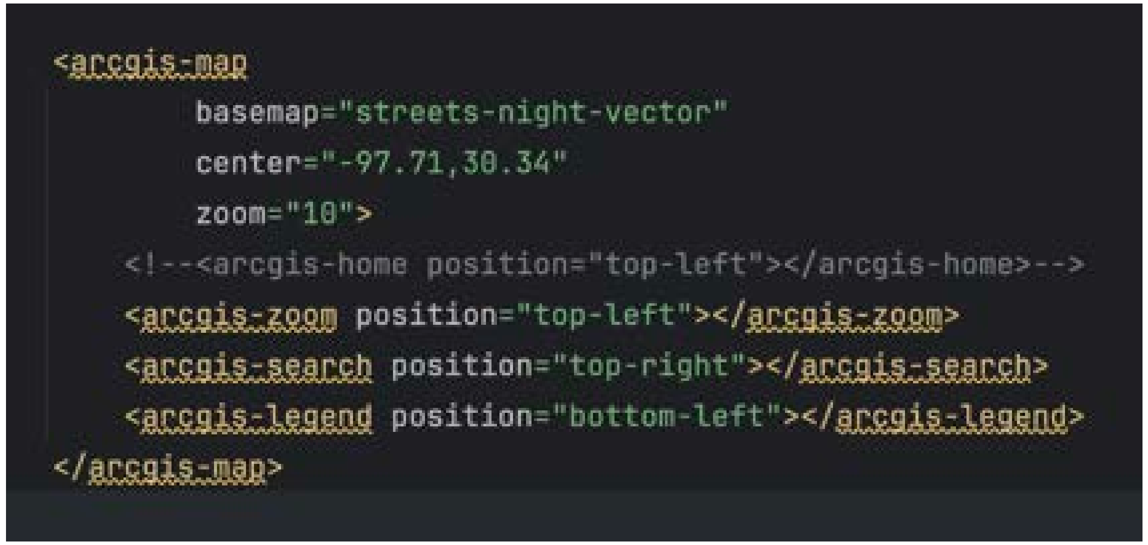
Compare

ESRI Portal ID



```html
  <!-- Load the ArcGIS Maps SDK for JavaScript from CDN -->
  <script src="https://js.arcgis.com/4.34/"></script>

  <!-- Load Map components from CDN -->
  <script type="module" src="https://js.arcgis.com/4.34/map-components/"
    ></script>
</head>

<body class="calcite-mode-dark">
  <arcgis-map id="map" item-id="ceb8954a5f2c457284c5074efd5a5ca0">
    <arcgis-zoom slot="top-left"></arcgis-zoom>
    <arcgis-expand slot="bottom-left" expanded>
      <arcgis-basemap-gallery></arcgis-basemap-gallery>
    </arcgis-expand>
    <div id="info" slot="top-right">
      <b>Watch for changes</b><br />
      <div id="messages"></div>
    </div>
  </arcgis-map>
```

ESRI Basemap ID



```html
<arcgis-map
        basemap="streets-night-vector"
        center="-97.71,30.34"
        zoom="10">
    <!--<arcgis-home position="top-left"></arcgis-home>-->
    <arcgis-zoom position="top-left"></arcgis-zoom>
    <arcgis-search position="top-right"></arcgis-search>
    <arcgis-legend position="bottom-left"></arcgis-legend>
</arcgis-map>
```

What else? Two Roads* for Development

The ESRI samples do have lots of interesting functionality.

And the components method of doing things involves one of two things

1. Using a JavaScript Framework – react, vue or angular for example. Used in conjunction with the framework.
2. Plain vanilla samples like some of the snippets we've already been looking at.

Let's look at both

* Really 3 – but we will get to that.

First Road JavaScript Frameworks Integration

JavaScript Frameworks

- Pros (just a few – I promise I am not going to dive into a rabbit hole)
  - Lots of Energy and Excitement around the frameworks – interesting working with them (at least for the developers).
  - They scale well! People are doing incredible things with these frameworks (and others) that could use spatial integration
- Cons
  - Learning Curve can be quite high. Particularly if you are coming from a GIS background and not a developer background
  - The samples are ok that ESRI provides but not great particularly for a beginner.

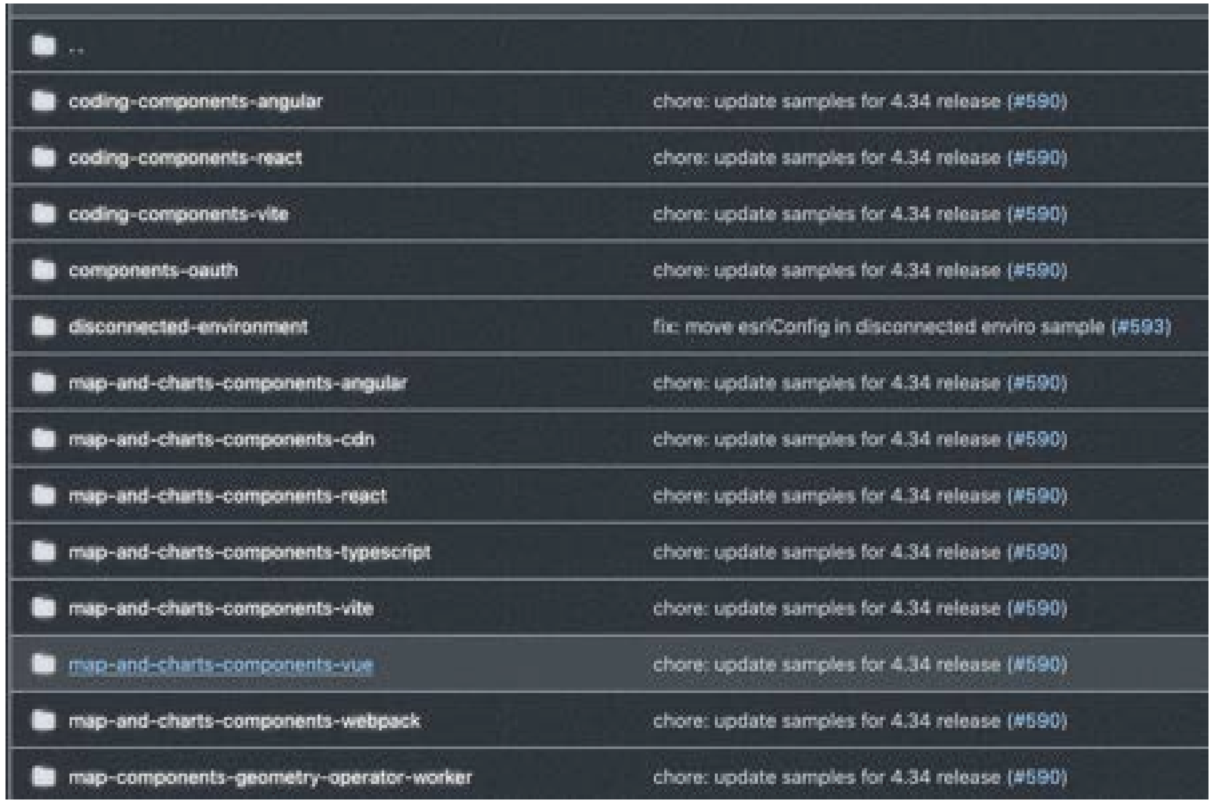
First Road JavaScript Frameworks Integration

JavaScript Frameworks

- ESRI's github for jsapi resources https://github.com/Esri/jsapi-resources

- And for components specifically

- https://github.com/Esri/jsapi-resources/tree/main/component-samples

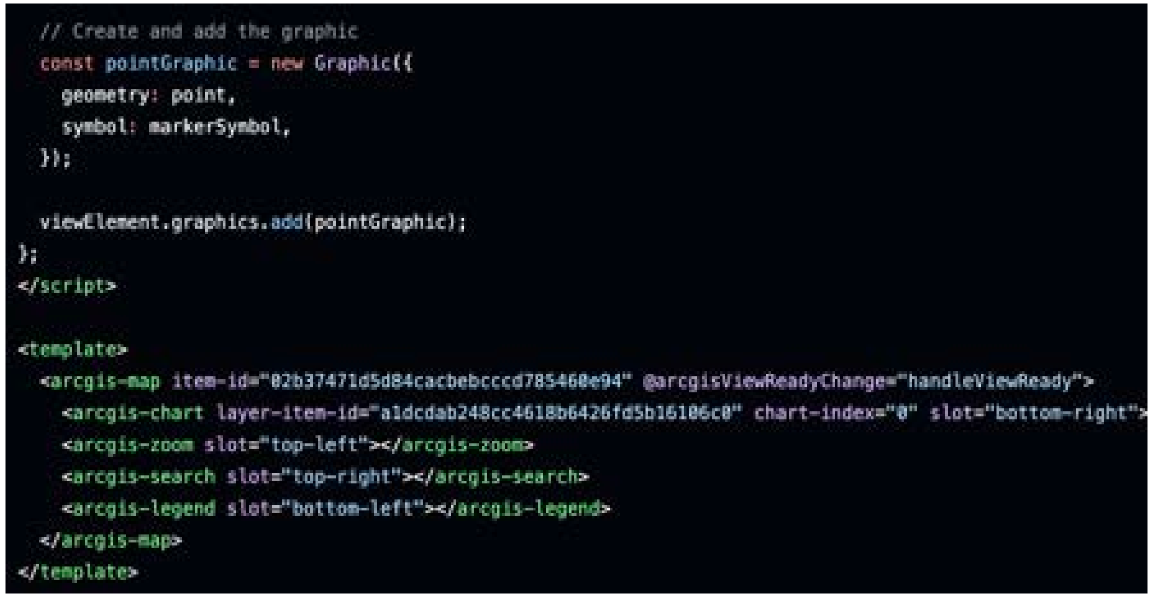


..	
coding-components-angular	chore: update samples for 4.34 release (#590)
coding-components-react	chore: update samples for 4.34 release (#590)
coding-components-vite	chore: update samples for 4.34 release (#590)
components-oauth	chore: update samples for 4.34 release (#590)
disconnected-environment	fix: move esriConfig in disconnected enviro sample (#593)
map-and-charts-components-angular	chore: update samples for 4.34 release (#590)
map-and-charts-components-cdn	chore: update samples for 4.34 release (#590)
map-and-charts-components-react	chore: update samples for 4.34 release (#590)
map-and-charts-components-typescript	chore: update samples for 4.34 release (#590)
map-and-charts-components-vite	chore: update samples for 4.34 release (#590)
map-and-charts-components-vue	chore: update samples for 4.34 release (#590)
map-and-charts-components-webpack	chore: update samples for 4.34 release (#590)
map-components-geometry-operator-worker	chore: update samples for 4.34 release (#590)

First Road JavaScript Frameworks Integration

My personal fav for the frameworks is VueJS
Let’s look at an example to get what I’m saying

- https://github.com/Esri/jsapi-resources/blob/main/component-samples/map-and-charts-components-vue/src/components/Map.vue
- This is a trivial example though.
- It adds a point on the map after it loads a portal item.
- They really need to give these samples the same treatment they use on the developers site!



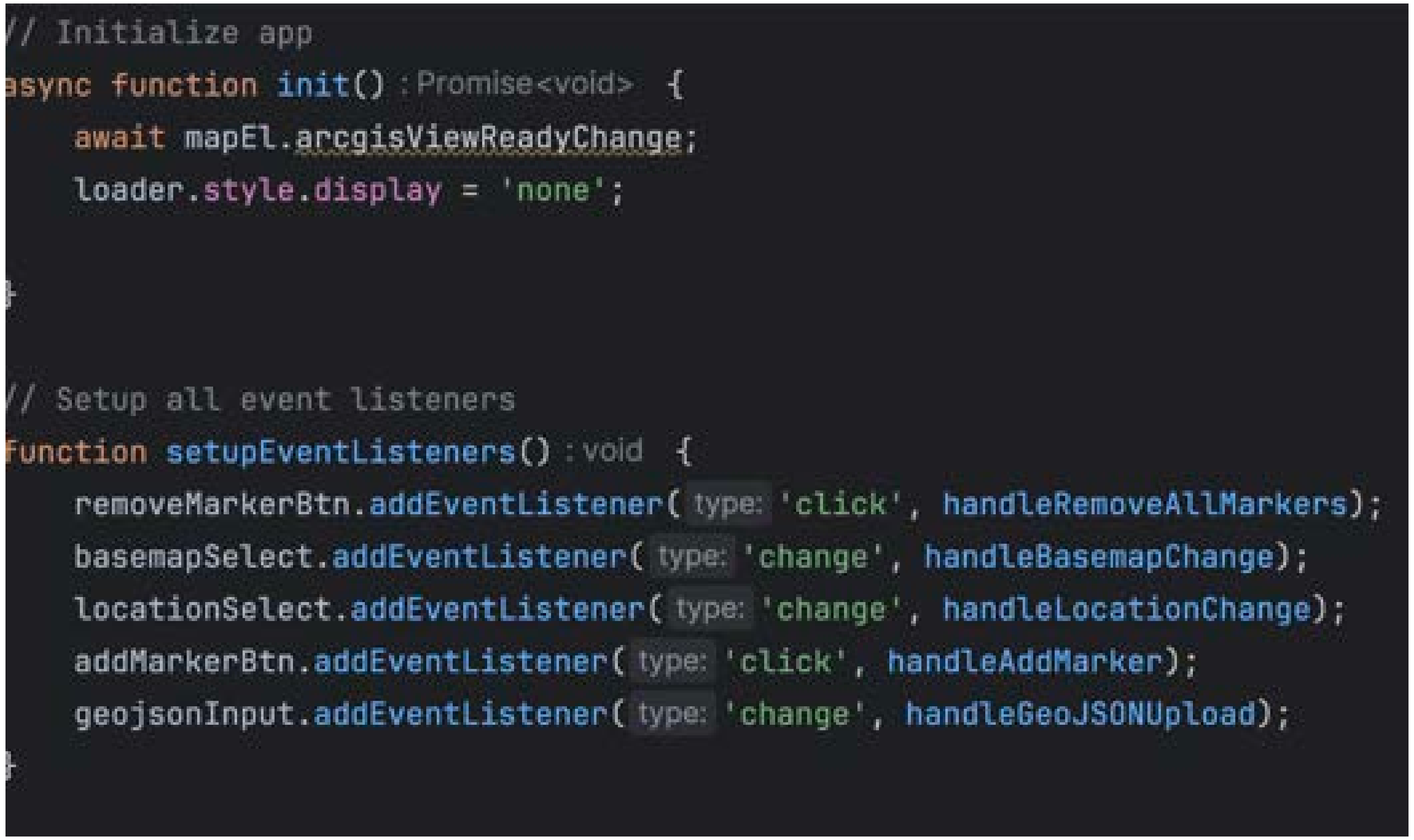
```
  // Create and add the graphic
  const pointGraphic = new Graphic({
    geometry: point,
    symbol: markerSymbol,
  });

  viewElement.graphics.add(pointGraphic);
};
</script>

<template>
  <arcgis-map item-id="02b37471d5d84cacbebcccd785460e94" @arcgisViewReadyChange="handleViewReady">
    <arcgis-chart layer-item-id="a1dcdab248cc4618b6426fd5b16106c0" chart-index="0" slot="bottom-right">
    <arcgis-zoom slot="top-left"></arcgis-zoom>
    <arcgis-search slot="top-right"></arcgis-search>
    <arcgis-legend slot="bottom-left"></arcgis-legend>
  </arcgis-map>
</template>
```

Second Road - Plain Vanilla JavaScript

- Pros (no rabbit's holes here either)
  - Lower bar for entry
  - Lots of examples on ESRI's SDK website – https://developers.arcgis.com/javascript/latest/
- Cons
  - If you're not careful your app can get complicated quickly.
  - And remember, if we are looking at this coming at this from someone with a GIS background, there can be a lot to manage!
  - Things like promises and async calls, etc.

Second Road - Plain Vanilla JavaScript

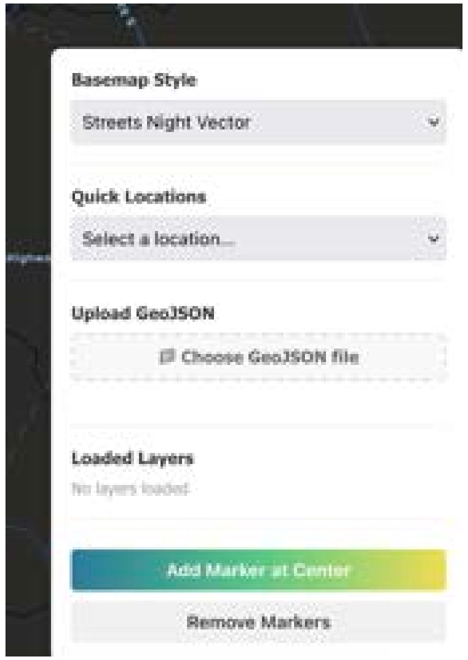
- The template I am providing via github is a nice starter plain vanilla web app.
- It uses an initializer to setup the core elements and some event registers for the custom controls that need them



```
// Initialize app
async function init() :Promise<void>  {
    await mapEl.arcgisViewReadyChange;
    loader.style.display = 'none';

}

// Setup all event listeners
function setupEventListeners() :void  {
    removeMarkerBtn.addEventListener( type: 'click', handleRemoveAllMarkers);
    basemapSelect.addEventListener( type: 'change', handleBasemapChange);
    locationSelect.addEventListener( type: 'change', handleLocationChange);
    addMarkerBtn.addEventListener( type: 'click', handleAddMarker);
    geojsonInput.addEventListener( type: 'change', handleGeoJSONUpload);
}
```

Second Road - Plain Vanilla JavaScript

- There are some handy functions for showing you how to add graphics to the map (and how to stylize them).



Basemap Style

Streets Night Vector

Quick Locations

Select a location...

Upload GeoJSON

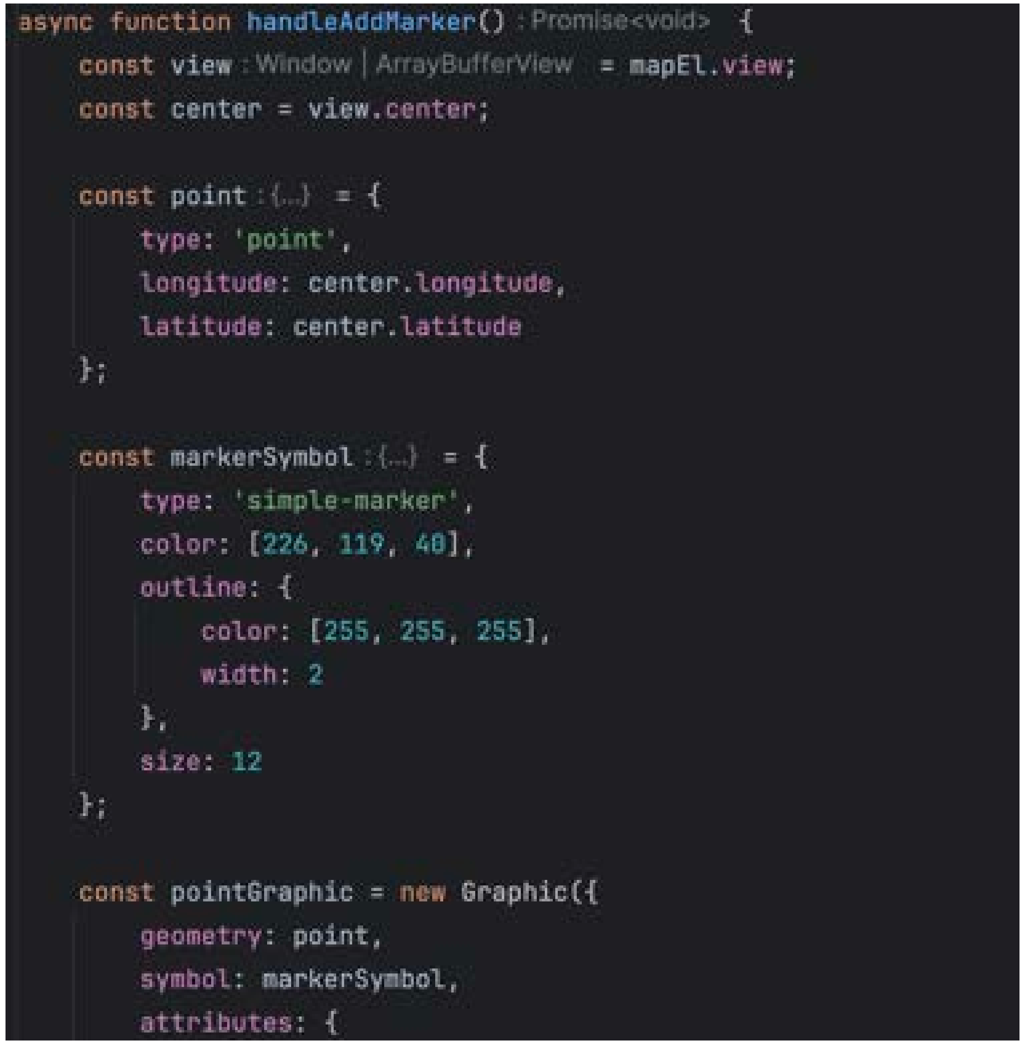
Choose GeoJSON file

Loaded Layers

No layers loaded

Add Marker at Center

Remove Markers



```
async function handleAddMarker() : Promise<void>  {
    const view : Window | ArrayBufferView  = mapEl.view;
    const center = view.center;

    const point : {...}  = {
        type: 'point',
        longitude: center.longitude,
        latitude: center.latitude
    };

    const markerSymbol : {...}  = {
        type: 'simple-marker',
        color: [226, 119, 40],
        outline: {
            color: [255, 255, 255],
            width: 2
        },
        size: 12
    };

    const pointGraphic = new Graphic({
        geometry: point,
        symbol: markerSymbol,
        attributes: {
```

My Choice for the GIS Professional building their way into Web GIS

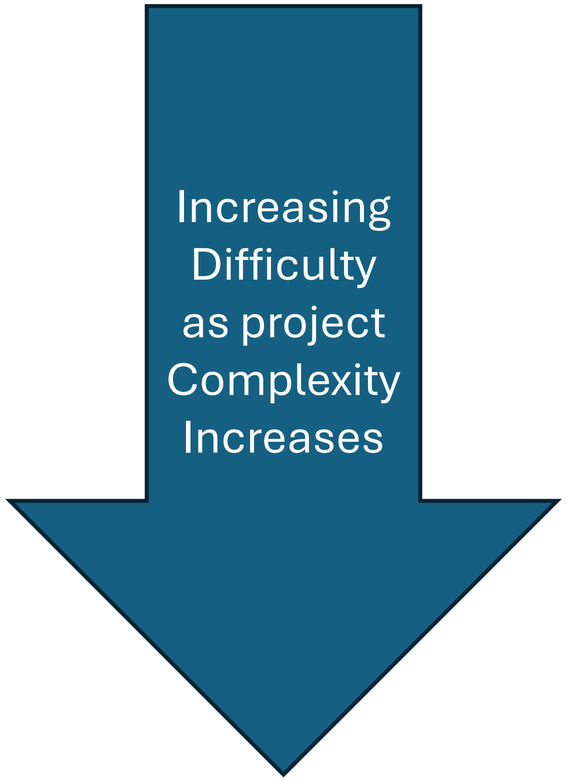
- Vanilla JavaScript!
- At least to start. As you become more comfortable and IF you decide you want to dive into rabbit holes, chose a framework and go for it!
- Let’s tool around this template a bit to show you what’s there!

Recap of Choices

1. Vanilla JavaScript w/some components & programmatic functionality/components

2. ESRI Components

3. ESRI Component Library Integration with JavaScript Frameworks

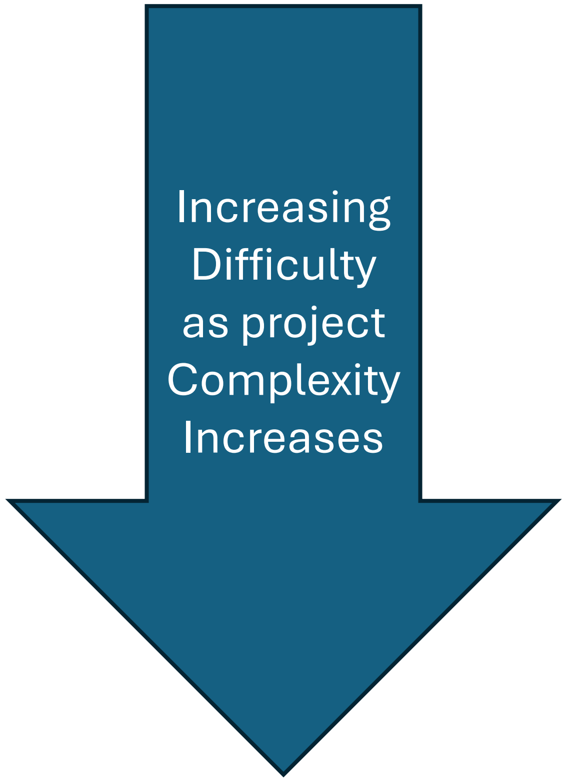


Recap of Choices – Some Pros of Each

1. Can completely (or nearly completely) decide on what to do and how you are doing it! Lots of flexibility

2. ESRI Components look really professional/ESRI-coded! Familiarity is useful when pushing out apps.

3. Scalability! And robust ecosystems for each of the Frameworks means that you lots of prebuilt/plugin-style choices for doing different things.

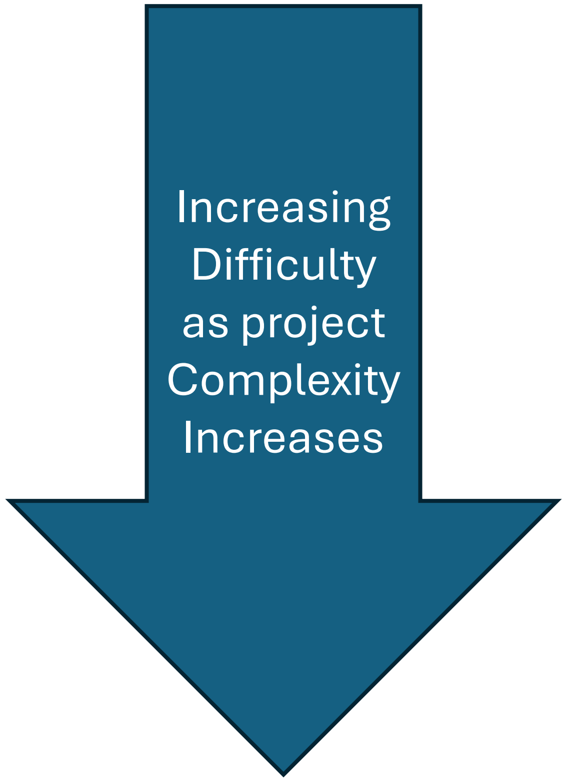


Recap of Choices – A Few Cons of Each

1. Really can become difficult as the complexity increases. It's a paradox really. It is easy to get started, hard once you are 'in the stew'.

2. Can be very crunchy! See slide earlier for reference. And you are kind of locked in for look-and-feel. Sometimes feels like you are wading through unnecessary code just for UI.

3. Frameworks are really the definition of a 2-edged sword! They have this incredible ecosystem but also your worldview on app dev has to sync with the framework otherwise you are swimming upstream!



Any Questions NOW? ☺

- Again, here is the link to the github repo
- https://github.com/brentporter/ComponentESRIStarter

- As was mentioned at the beginning I work at BEG here on the Pickle Campus and for Austin Community College if you want to get in touch with me.

- Thanks!