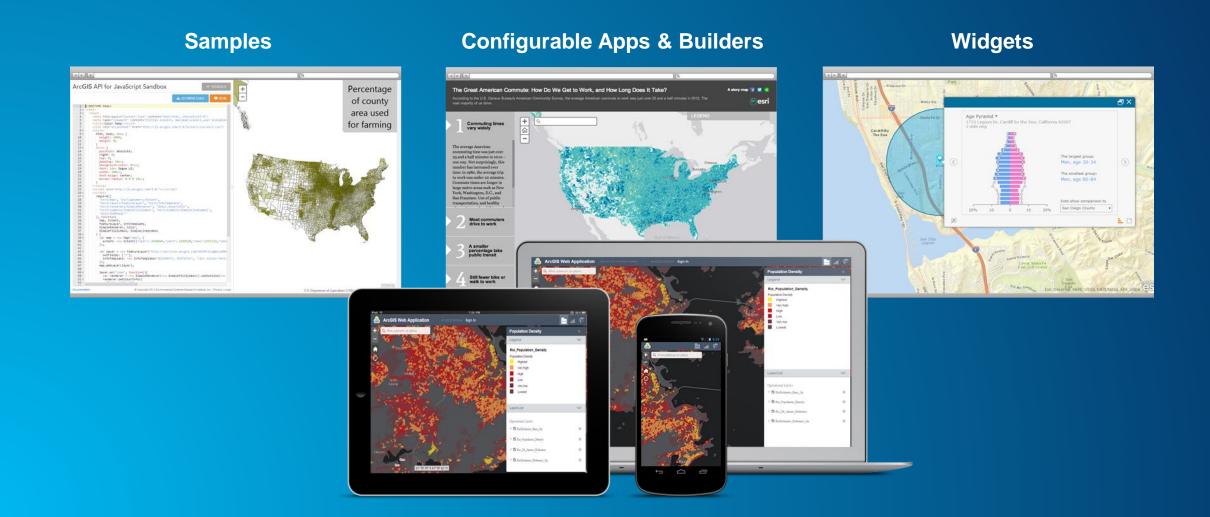
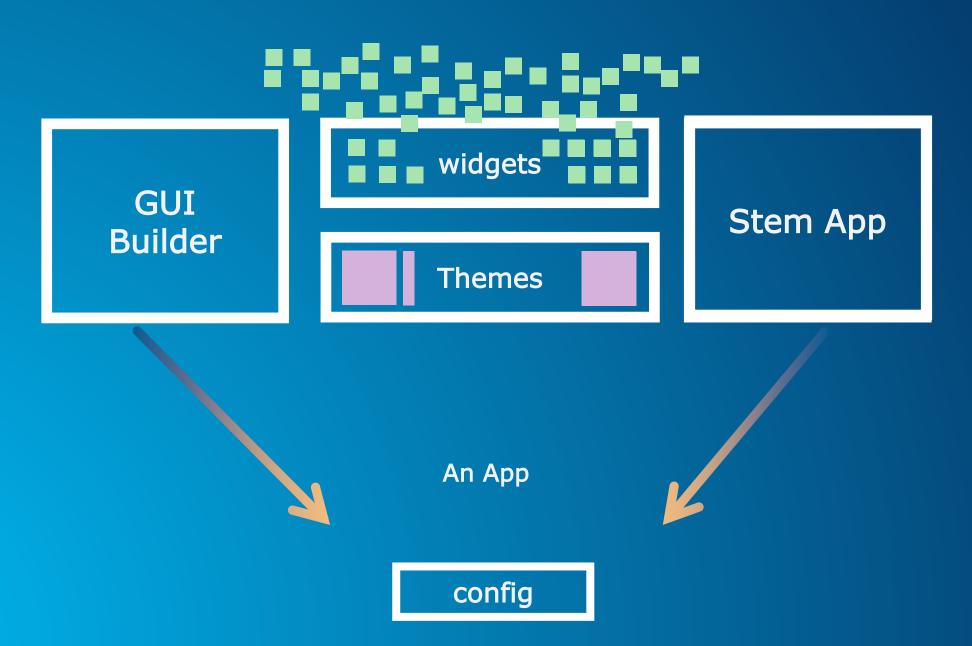
Introduction

Web AppBuilder for ArcGIS

Building Web Apps for Your Organization Using the ArcGIS API for JavaScript



App Builder Concept



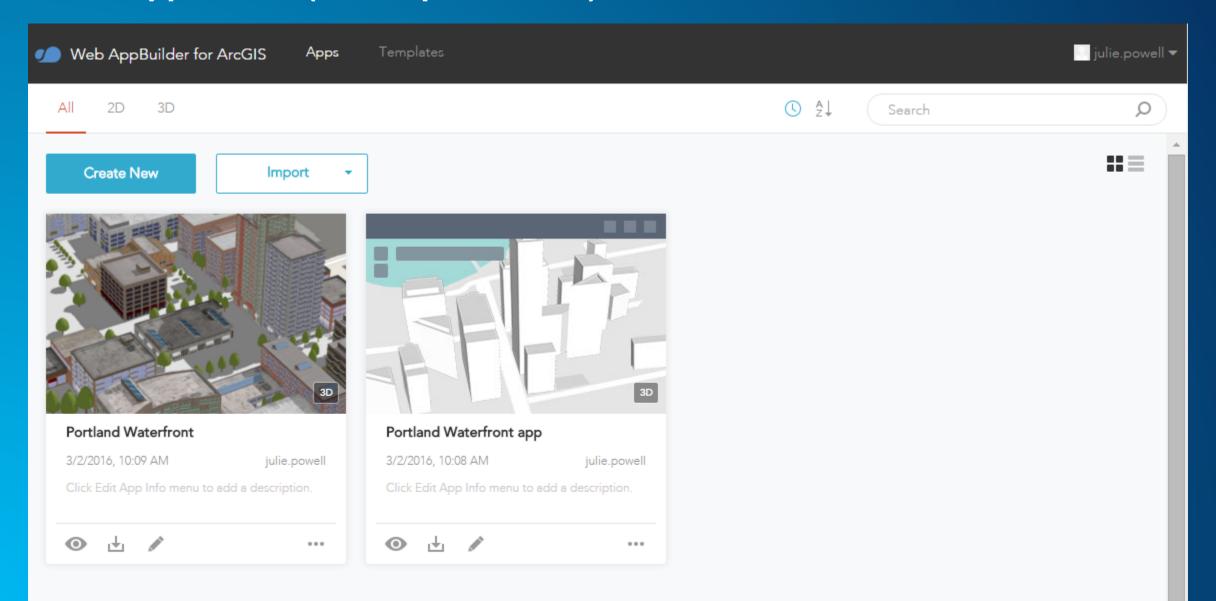
Widget

- Execution at run time
- Configure-in, not cut/paste
- Self sufficient and distributable
- Need container, no coding block
- Has programing framework of container

Theme

- Applied at run time
- Configure-in, not modify css
- Need container
- Self sufficient and distributable
- Has programing framework of container

Web AppBuilder (Developer Edition)



Widgets

Building Blocks of Apps

Break the code into files

MyWidget.css

```
html, body, #map{
height: 100%;
margin: 0;
padding: 0;
}
```

MyWidget.js

```
define(['dojo/_base/declare', 'jimu/BaseWidget'],
function(declare, BaseWidget){
  var clazz = declare([BaseWidget],{
  });
  return clazz;
});
```

MyWidget.html

Tutorial:

https://developers.arcgis.com/javascript/jshelp/intro_custom_dijit.html

```
!DOCTYPE html>
<html>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <!--The viewport meta tag is used to improve the presentation and behavior of the samples
    on iOS devices-->
   <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no">
   <title>Class Breaks Renderer</title>
    (link rel="stylesheet" href="http://js.arcgis.com/3.13/esri/css/esri.css">
     html, body, #map{
      height: 100%;
       margin: 0;
       padding: 0;
   <script>
        "esri/map", "esri/layers/FeatureLayer",
        "esri/InfoTemplate", "esri/symbols/SimpleFillSymbol",
        "esri/renderers/ClassBreaksRenderer",
        "esri/Color", "dojo/dom-style", "dojo/domReady!"
       Map, FeatureLayer,
       InfoTemplate, SimpleFillSymbol,
       ClassBreaksRenderer,
       Color, domStyle
       map = new Map("map", {
        basemap: "streets",
         center: [-98.215, 38.382],
         zoom: 7,
         slider: false
       var symbol = new SimpleFillSymbol();
       symbol.setColor(new Color([150, 150, 150, 0.5]));
        // Add five breaks to the renderer.
       // If you have ESRI's ArcMap available, this can be a good way to determine beak values.
       // You can also copy the RGB values from the color schemes ArcMap applies, or use colors
       // from a site like www.colorbrewer.org
       // alternatively, ArcGIS Server's generate renderer task could be used
       var renderer = new ClassBreaksRenderer(symbol, "POP07_SQMI");
       renderer.addBreak(0, 25, new SimpleFillSymbol().setColor(new Color([56, 168, 👢 0.5])));
       renderer.addBreak(25, 75, new SimpleFillSymbol().setColor(new Color([139, 209 0, 0.5])));
       renderer.addBreak(75, 175, new SimpleFillSymbol().setColor(new Color([255, 25, 0, 0.5])));
       renderer.addBreak(175, 400, new SimpleFillSymbol().setColor(new Color([255, 18, 0, 0.5])));
       renderer.addBreak(400. Infinity, new SimpleFillSymbol().setColor(new Color([2008, 0, 0, 0.5])));
       var infoTemplate = new InfoTemplate("${NAME}", "${*}");
       var featureLayer = new FeatureLayer("http://sampleserver1.arcgisonline.com/Ar SIS/rest/services/Demographics
         mode: FeatureLayer.MODE_SNAPSHOT,
         outFields: ["*"],
           nfoTemplate: infoTemplate
                                onExpression("STATE_NAME = 'Kansas'");
  (body>
   <div id="map"></div>
```

Inheriting from BaseWidget

A widget derived from the BaseWidget class

Dijit lifecycle

- postCreate
- startup



Widget events

- onOpen, onActive
- onClose, onDeActive



BaseWidget

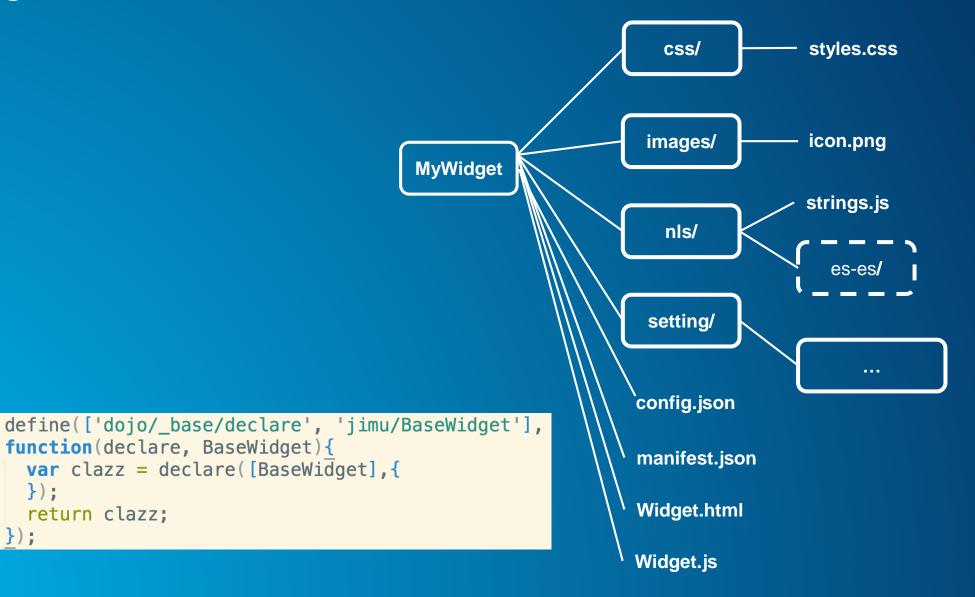
- App properties (name, icon, localization)
- App config data
- Widget's config data
- Map object
- Widget state (open, closed, active...)
- Events (open/signIn)
- Widget communication

Your job?

- Widget UI (HTML/template)
- Widget config file (JSON)
- Widget styles (CSS)
- Localization
- Your unique business logic / worklows (JavaScript)

Widget Files

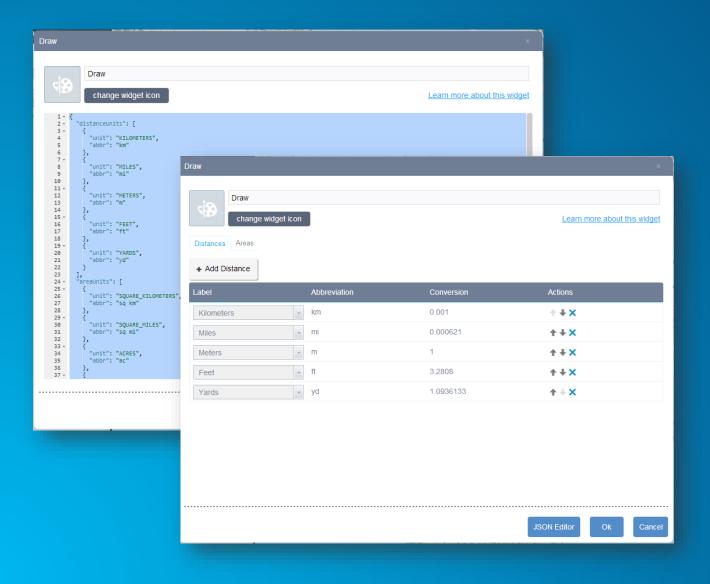
});



Getting Started...

- 1. <u>Download</u> developer edition
- 2. <u>Connect</u> to organization or portal
- 3. Copy widget template
- 4. Run the builder
- 5. Create an app with your widget
- 6. Build your widget in the app

Configure your custom widget inside the builder



Building a UI for the user

- Setting.js
 - Config info
 - getConfig, setConfig
- Setting.html
- Usual localization pattern
- CSS