



```
new Map({  
    viewer: "viewD",  
    map,  
    zoom: 10,  
    center: [-73.95, 40.75]  
});
```

ESRI EUROPEAN DEVELOPER SUMMIT 2023

ArcGIS Experience Builder for Developers

Christine Wiltawsky

Workshop Agenda (1)

- Introduction to ArcGIS Experience Builder
- Introduction to custom Widgets (TypeScript, REACT)
- Build a basic Widget
- Localization
- Settings: Connect Widget with a map

- LUNCH from 12:00 – 13:00

Workshop Agenda (2)

- Actions
- DataSources
- Templates
- Themes
- Components and Widget Styling (jimu-ui, Calcite)
- Mobile Layouts
- Deployment
- Extending OOTB Widgets

Quick Introduction

- Introduce yourself
- Your knowledge of configuration and customization of Experience Builder
- Your knowledge of configuration and customization of Web AppBuilder

Introduction to ArcGIS Experience Builder

Christine Wiltawsky

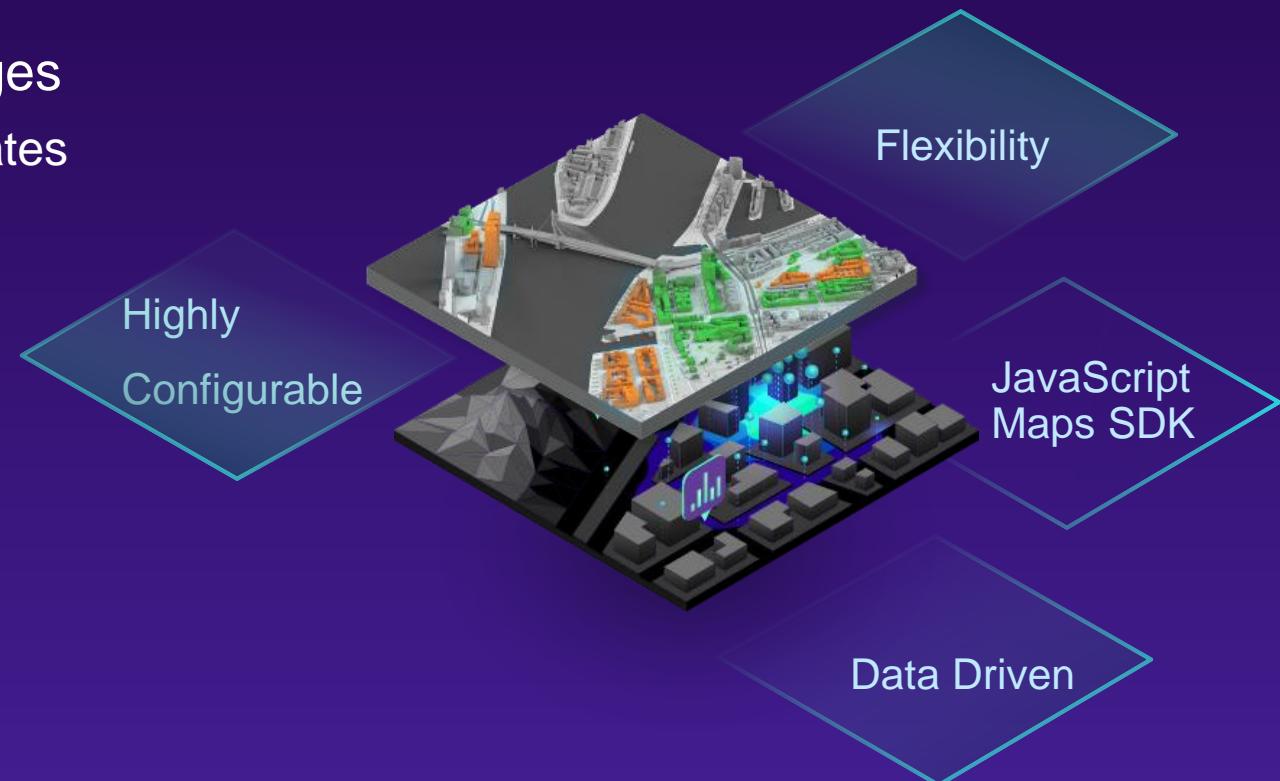
```
const routeParams = new RouteParam.  
stops: new FeatureSet({  
    features: view.graphics.toArray()  
}),  
returnDirections: true  
});
```

```
var mapView =  
    container: "v_...  
    map: map,  
    zoom: 10,  
    center: [-73.95, 40..  
]);
```

What Is ArcGIS Experience Builder?

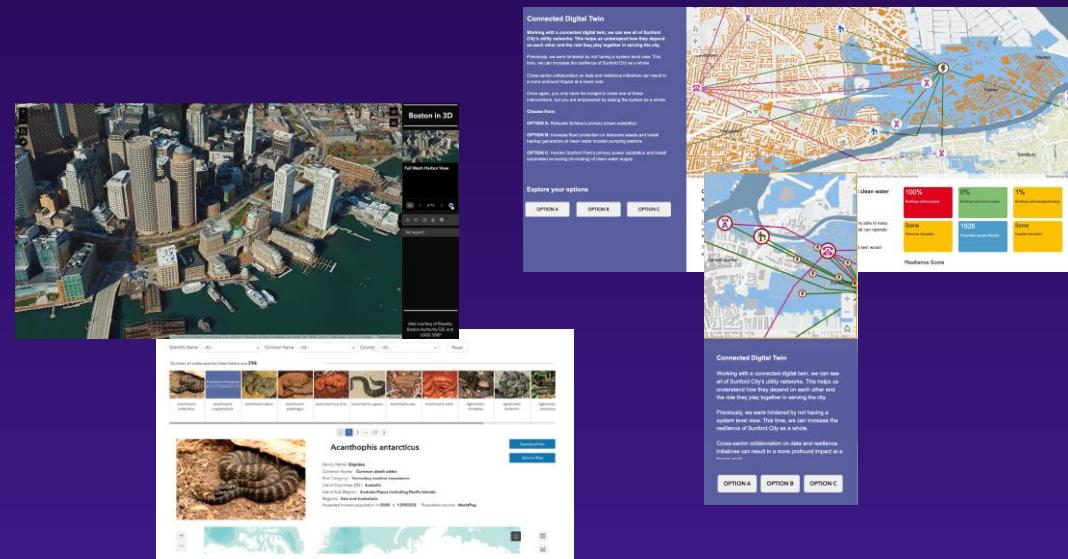
Create web apps you envision

- Design and build web apps and pages
 - Explore design decisions with templates
 - Access ready-to-use widgets
 - Build custom tools
- Data driven
- Modern interface



ArcGIS Experience Builder Key Features

- Flexible design
- Mobile optimization
- Interconnection between widgets
- 2D and 3D in one app
- Integration with other ArcGIS apps
- Extensibility



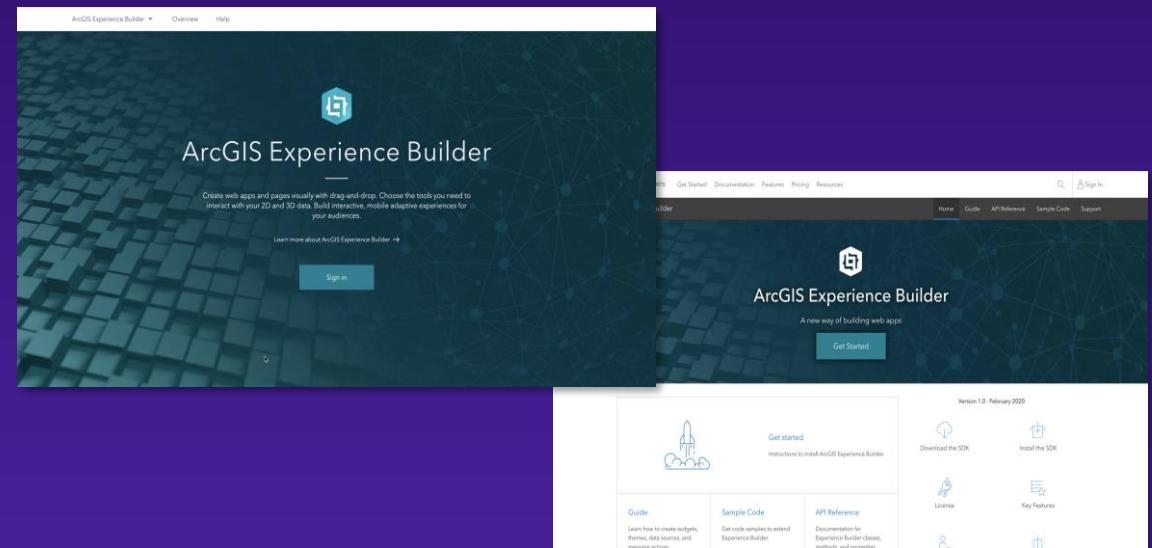
Available in ArcGIS Online, ArcGIS Enterprise, and developer edition

ArcGIS Experience Builder Offering

- In ArcGIS Online and ArcGIS Enterprise
- ArcGIS Online URL: <https://experience.arcgis.com/>
 - ArcGIS Enterprise 10.8.1 and up



- Developer Edition
 - Download and install locally



Versioning and Releases

Introduction to ArcGIS Experience Builder

- Rolling release with ArcGIS Online updates
- Fixed release packaged with each version of ArcGIS Enterprise
- Fixed releases individually available with ArcGIS Experience Builder Developer Edition

exbVersion	JSAPI versions	ArcGIS Enterprise	Developer edition	ArcGIS Online
1.12	4.27		1.12	July 2023
1.11	4.26		1.11	March 2023
1.10	4.25	11.1	1.10	November 2022
1.9	4.24		1.9	June 2022
1.8	4.23	11.0	1.8	March 2022
1.7	4.22		1.7	January 2022
1.6	4.21		1.6	October 2021
1.5	4.20	10.9.1	1.5	June 2021
1.4	4.19		1.4	April 2021
1.3	4.18	10.9	1.3	December 2020
1.2	4.17		1.2	October 2020
1.1	4.16		1.1	June 2020
1.0	4.15	10.8.1	1.0	February 2020

- <https://developers.arcgis.com/experience-builder/guide/release-versions/>

ArcGIS Experience Builder Licensing

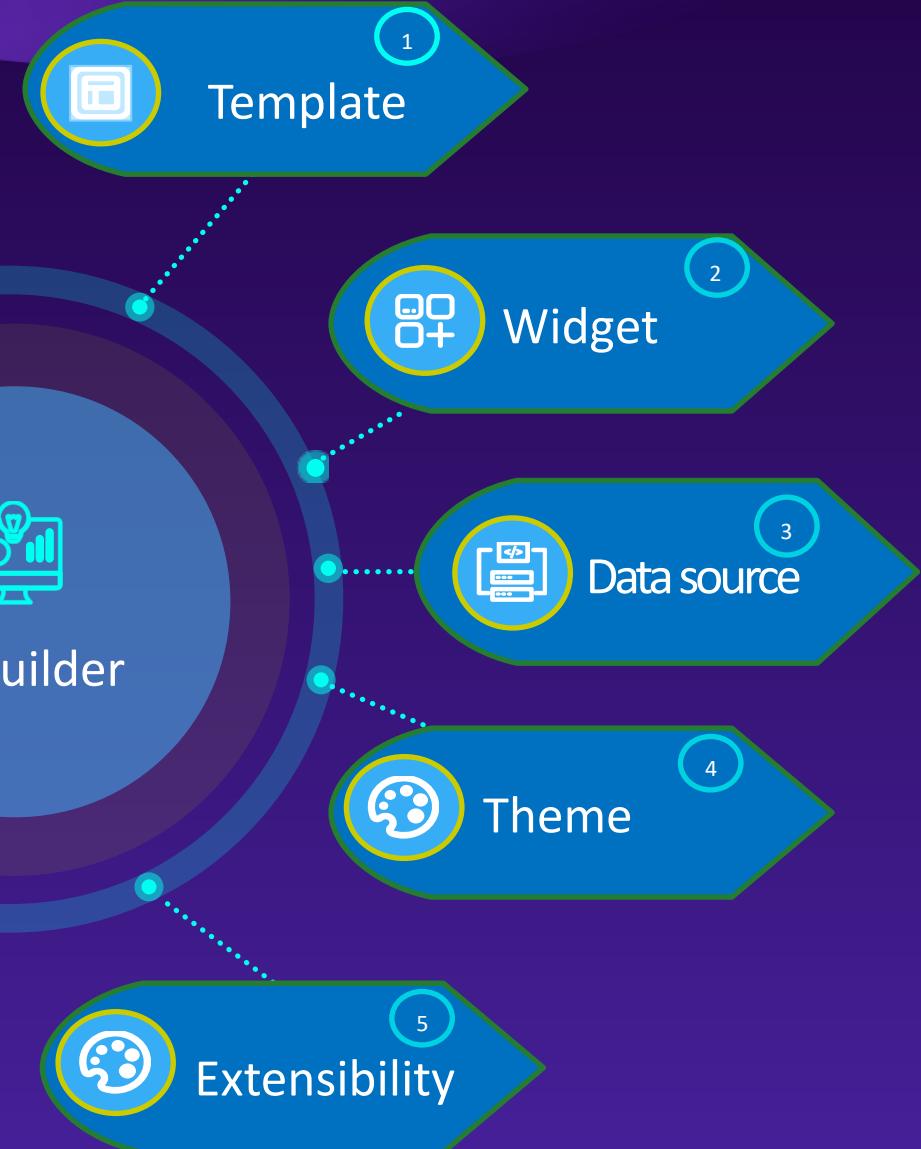


- Essential Apps Bundle
 - Must have a Creator or GIS Professional user type to create experiences
- ArcGIS Developer subscription
 - For developers who are not part of an ArcGIS organization to use Developer Edition

An
experience

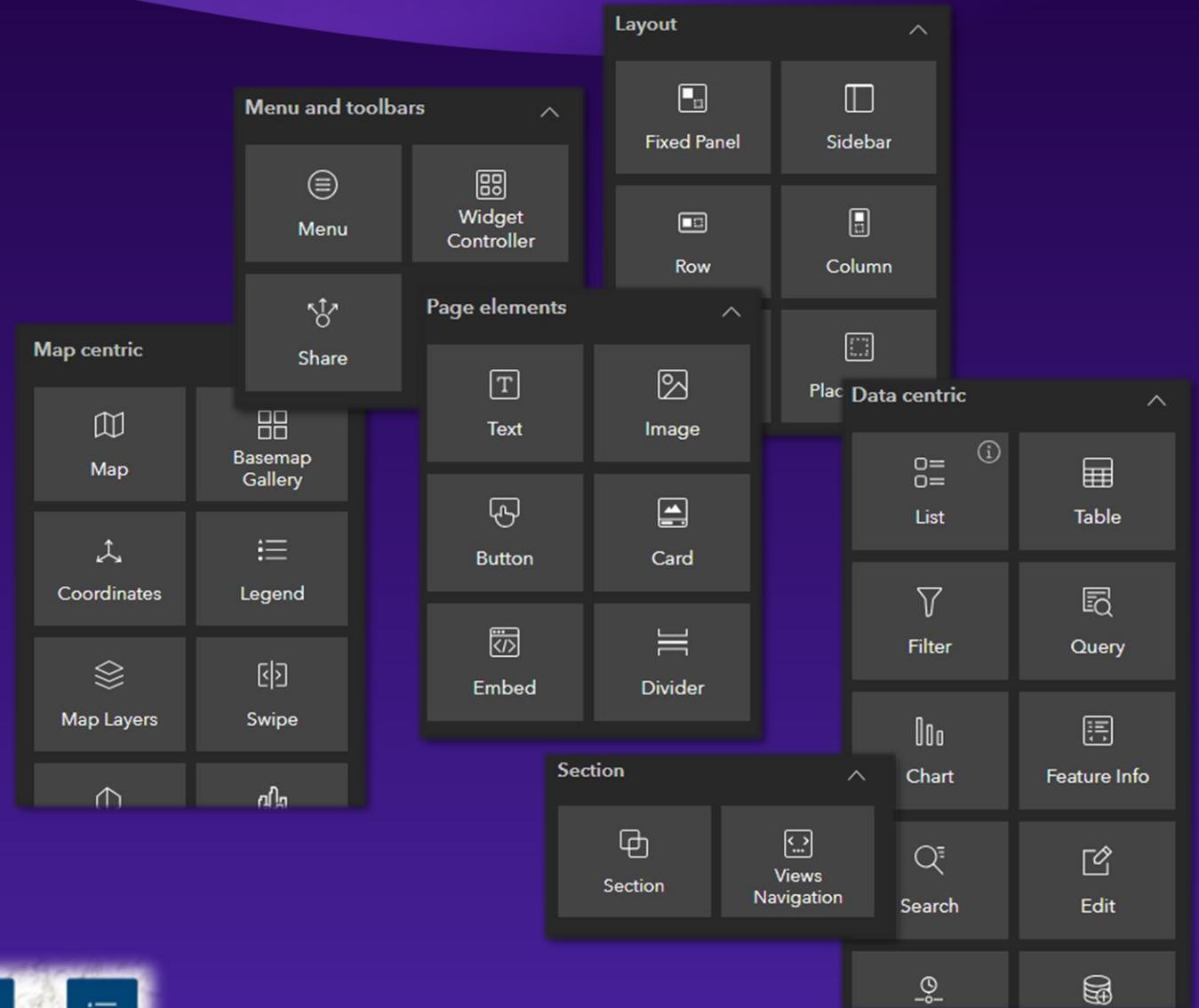
Design/
config

GUI builder



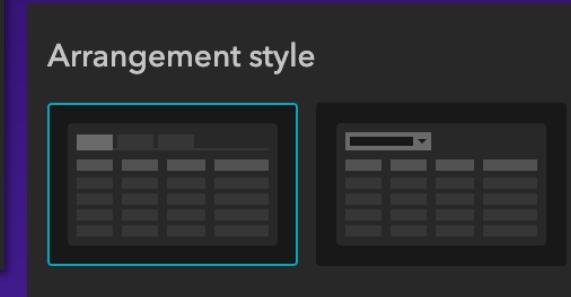
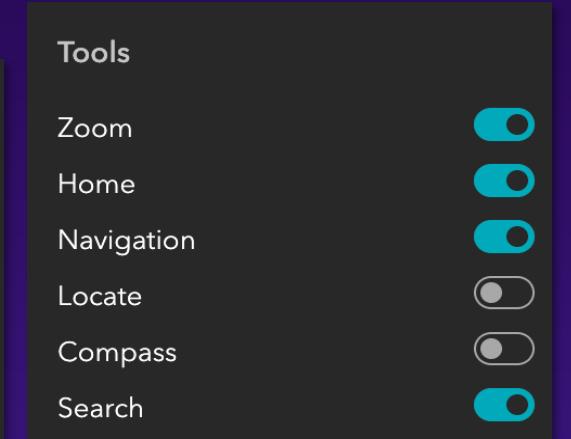
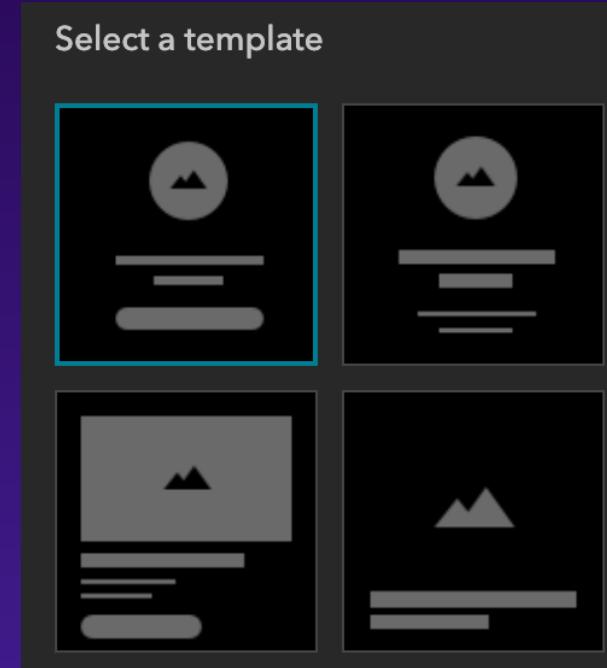
Types of Widgets

- Map centric
- Data centric
- Page elements
- Menu and toolbars
- Layout
- Section



Widget Content

- Data sources
- Options
- Tools
- Modes
- Templates



Widget Style

Size & Position

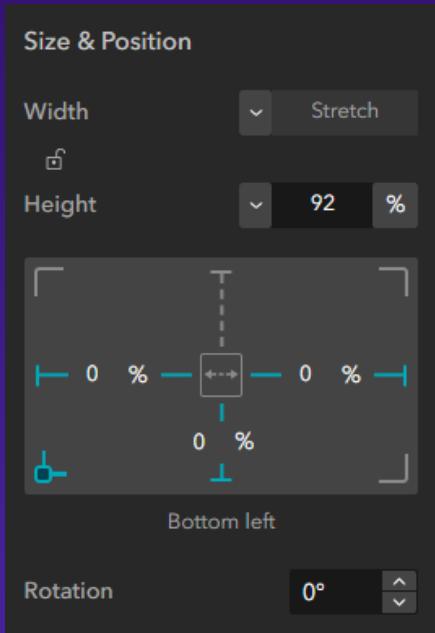
Size & Position

Width: Stretch

Height: 92 %

Bottom left

Rotation: 0°



Background, Border & Box shadow

Background

Fill:

Image: [Browse](#)

Position: Fill

Border

Style: Dashed

Width: 0 px

Border radius: 0 px

Box shadow

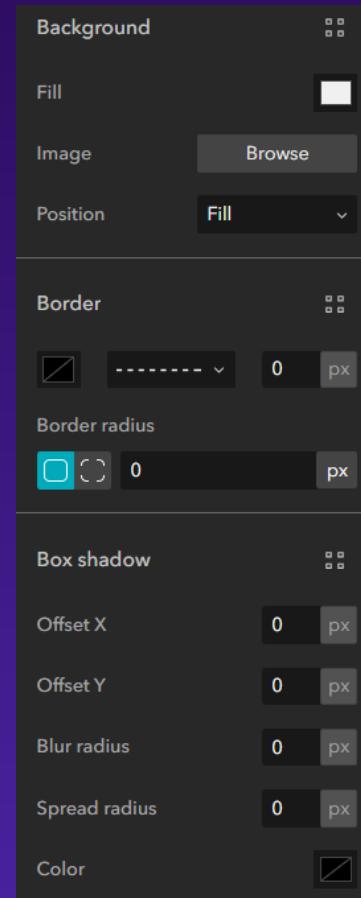
Offset X: 0 px

Offset Y: 0 px

Blur radius: 0 px

Spread radius: 0 px

Color:



Animation

Animation settings

None

Fade in

Float in

Fly in

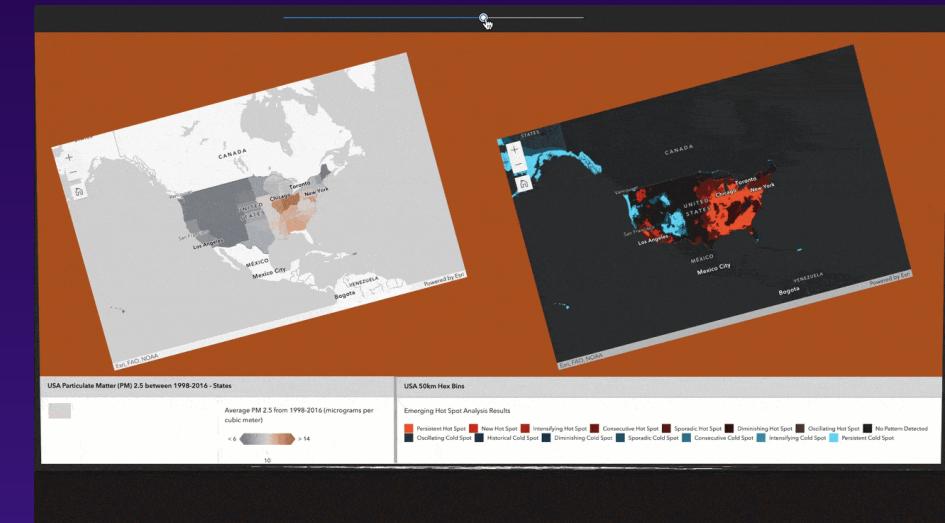
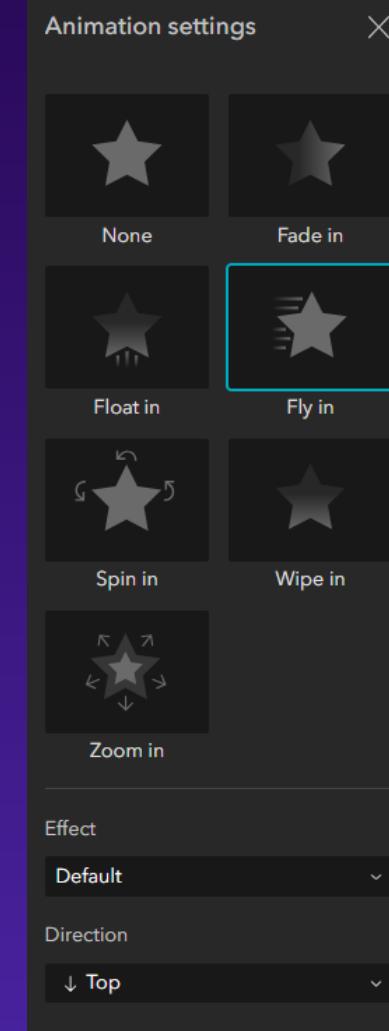
Spin in

Wipe in

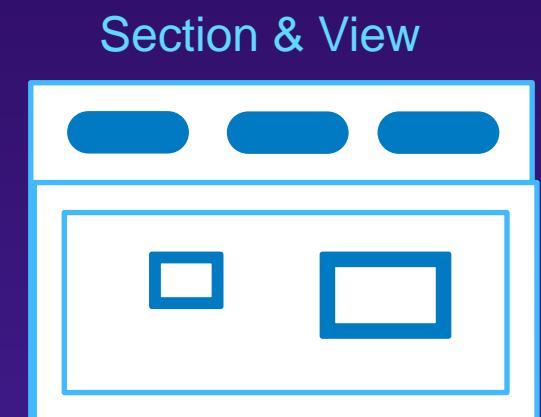
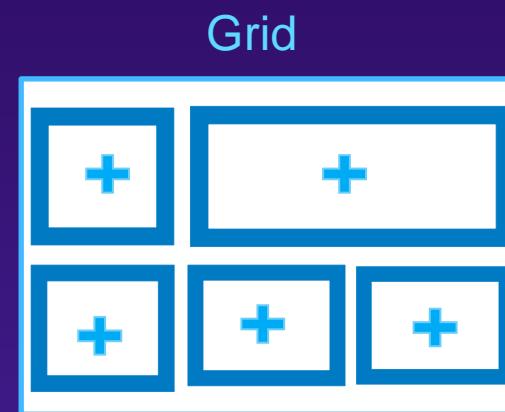
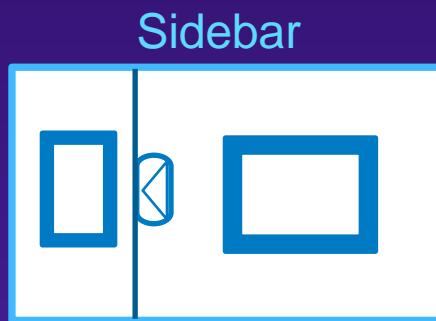
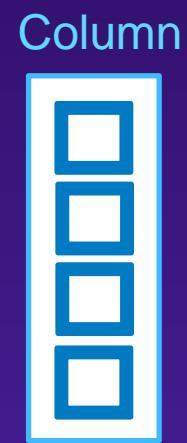
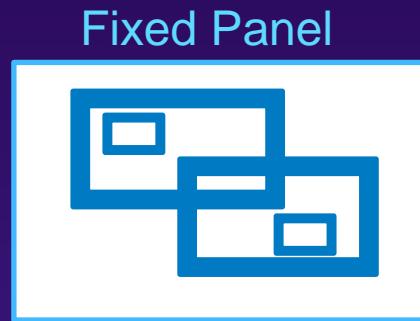
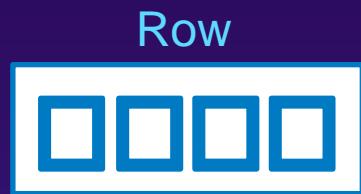
Zoom in

Effect: Default

Direction: Top

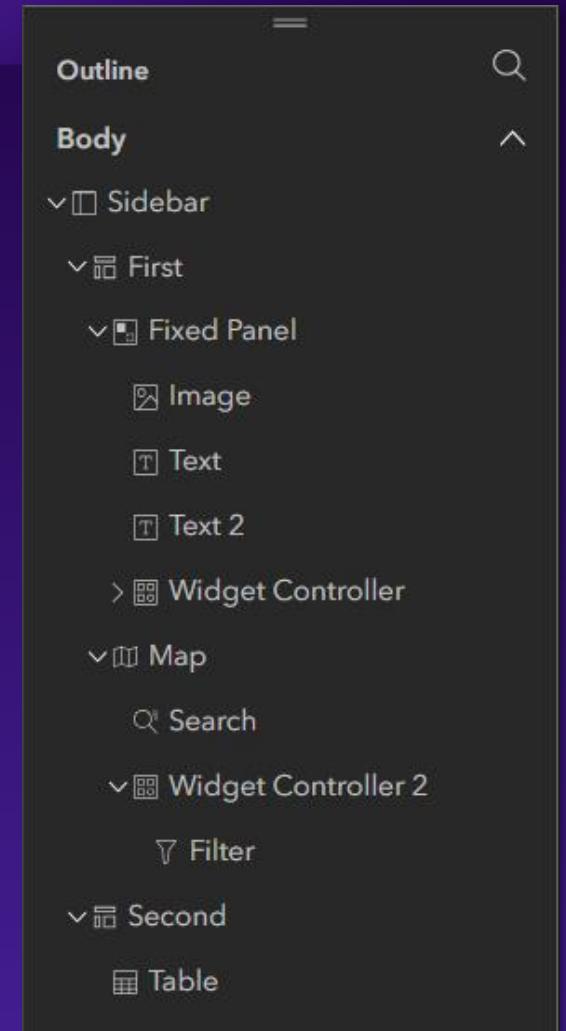
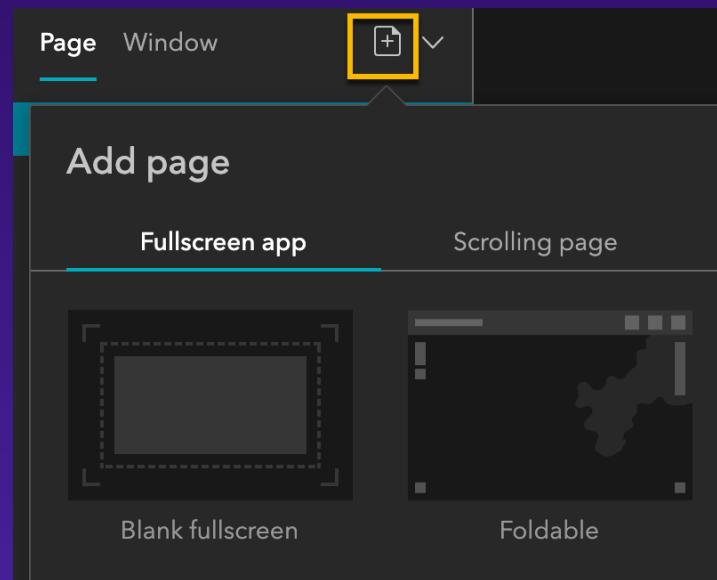


Layout Widgets



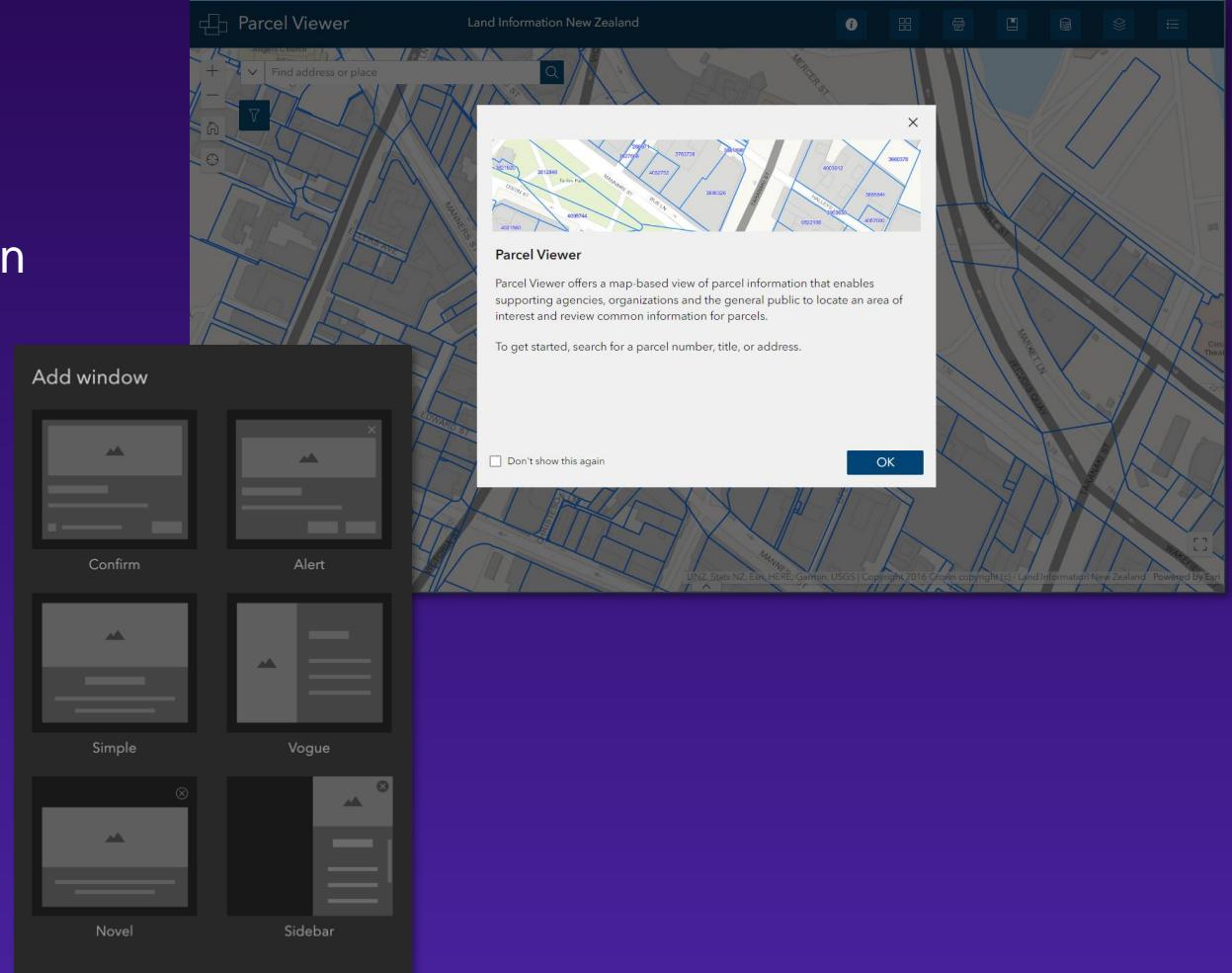
Page

- Header, Footer, Open with window
- Outline helps navigate widgets on the page
- Page templates



Window

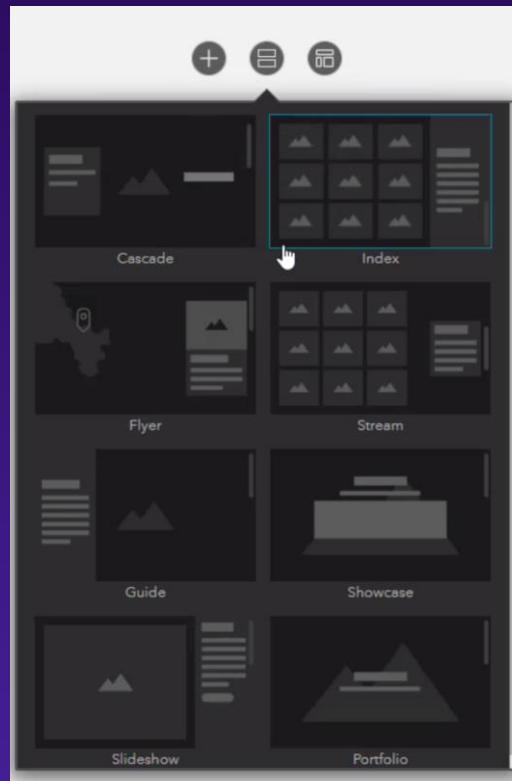
- Display Splash screens or information
- Fixed mode
 - shown in a fixed position like a splash screen
- Anchored mode
 - shown around the widget that triggers it
- Window templates



Screen Group

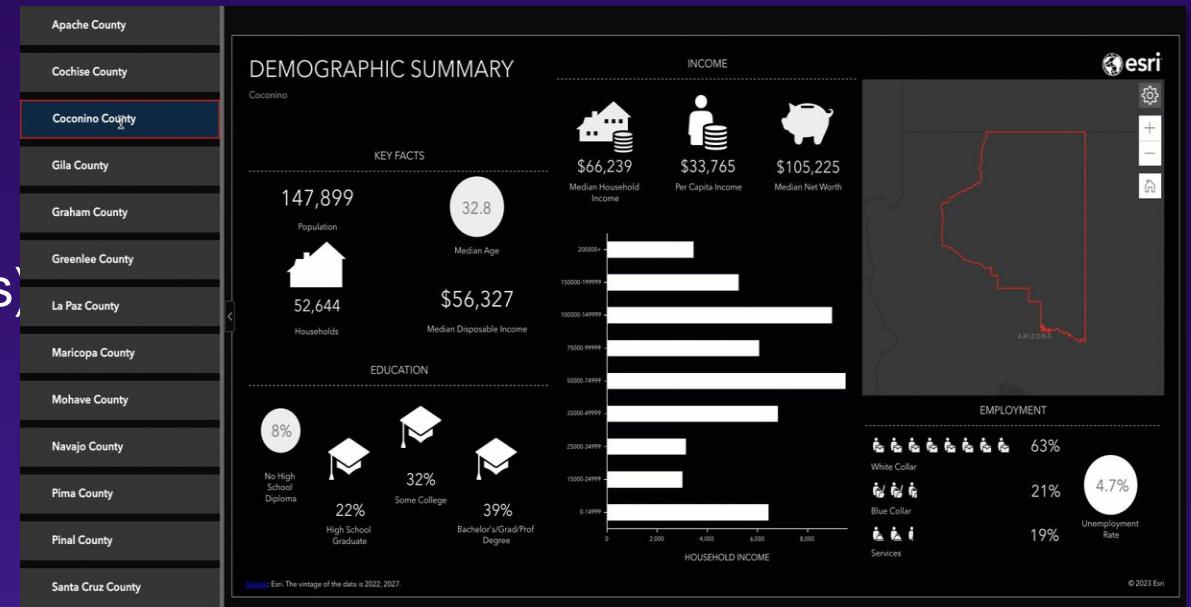
A special layout with multiple scrolling screens

Engaging
Experience

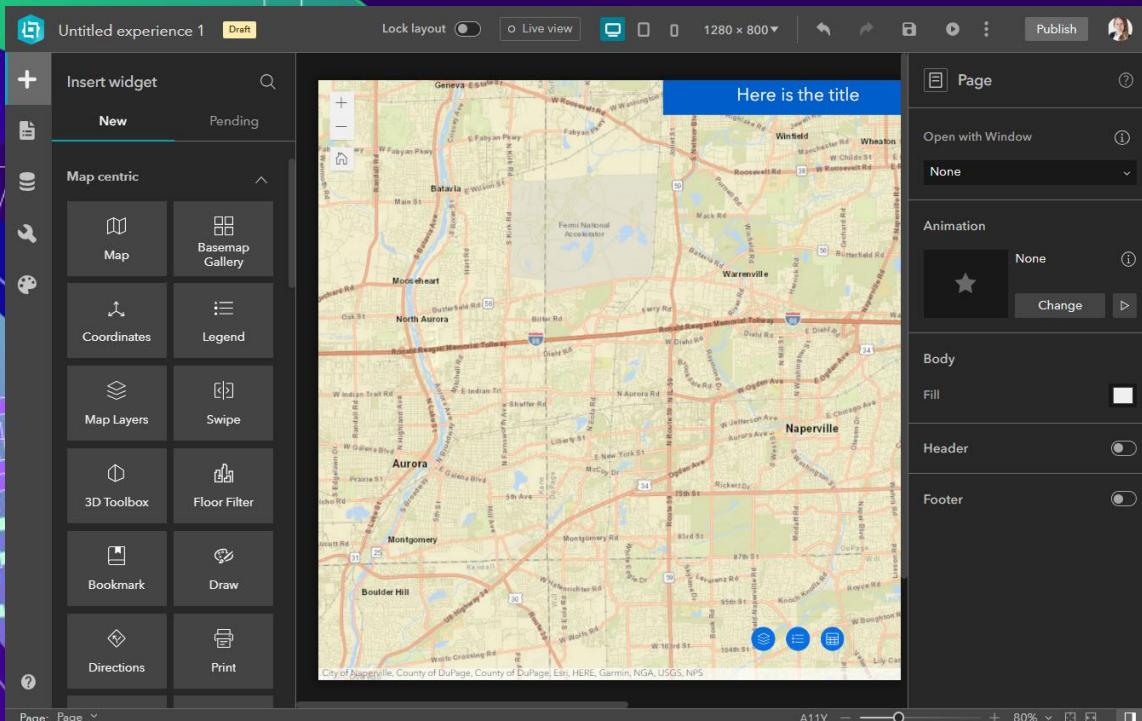


Integration with ArcGIS Apps

- Survey widget (ArcGIS Survey123)
- Business Analyst widget
- Floor Filter widget (ArcGIS Indoors)
- App URL parameter (ArcGIS Dashboards)



```
var mapView = new MapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.73]  
});
```



Quick Tour

Christine Wiltawsky

```
= new RouteParameters({  
  routeSet:{  
    features:  
  },  
  return  
})
```

Introduction to Custom Widgets

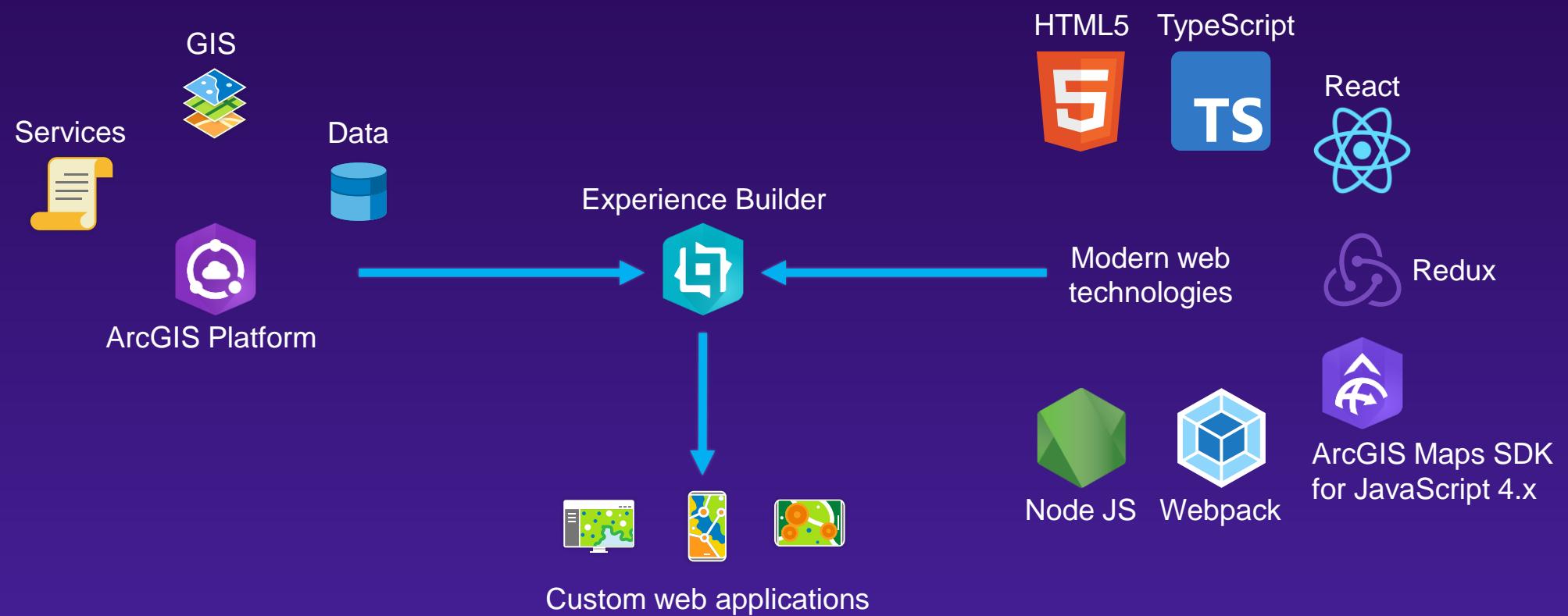
Christine Wiltawsky

```
const routeParams = new RouteParam({  
  stops: new FeatureSet({  
    features: view.graphics.toArray()  
  }),  
  returnDirections: true  
});
```

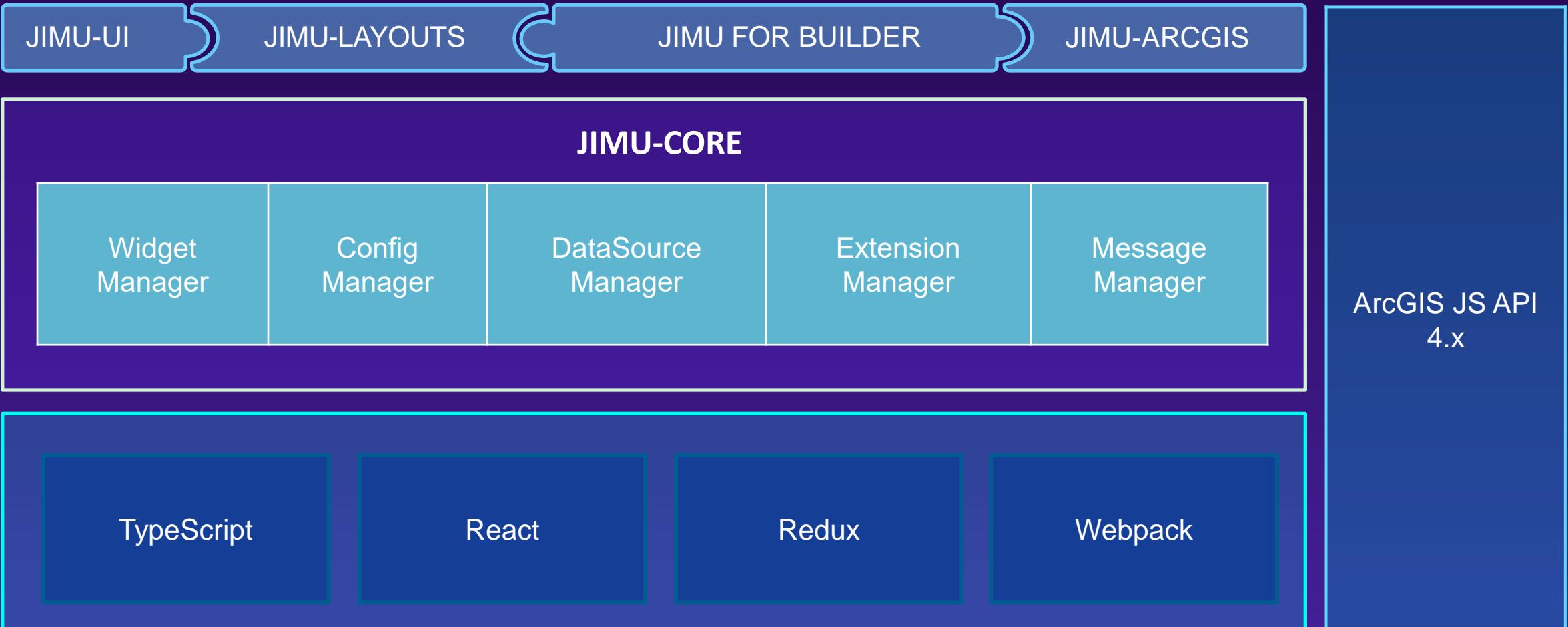
```
var mapView =  
  container: "view",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.7],  
  ...);
```

Technology Stack

Introduction to ArcGIS Experience Builder



Technology Stack



Developer Edition File Structure

Custom widgets

```
└─ ARCGISEXPERIENCEBUILDER
    ├─ client
    ├─ server
    └─ 3rd-party-license.txt
```

{ } package-lock.json

ⓘ readme.txt

{ } version.json

```
└─ client
    ├─ _mocks_
    ├─ dist
    ├─ dist-prod
    ├─ dist-report
    ├─ jimu-arcgis
    ├─ jimu-core
    ├─ jimu-for-builder
    ├─ jimu-for-test
    ├─ jimu-icons
    ├─ jimu-layouts
    ├─ jimu-theme
    ├─ jimu-ui
    ├─ node_modules
    ├─ scripts
    ├─ types
    ├─ webpack
    └─ your-extensions
        ├─ themes
        └─ widgets
    { } manifest.json
```

ⓘ .eslintignore

ⓘ .eslintrc.js

JS jest-systemjs-transformer.js

JS jest.config.js

{ } lerna.json

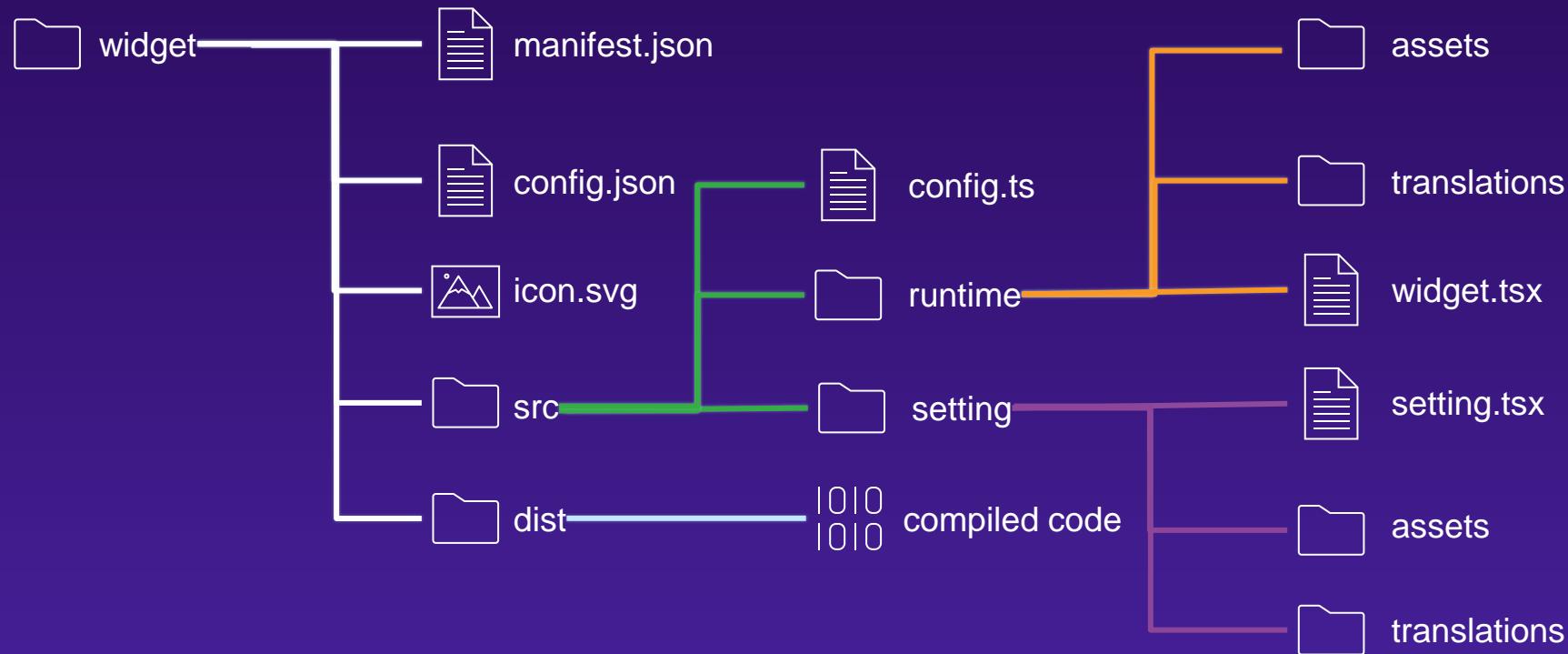
{ } package-lock.json

{ } package.json

TS tsconfig.json

⚡ webpack.config.js

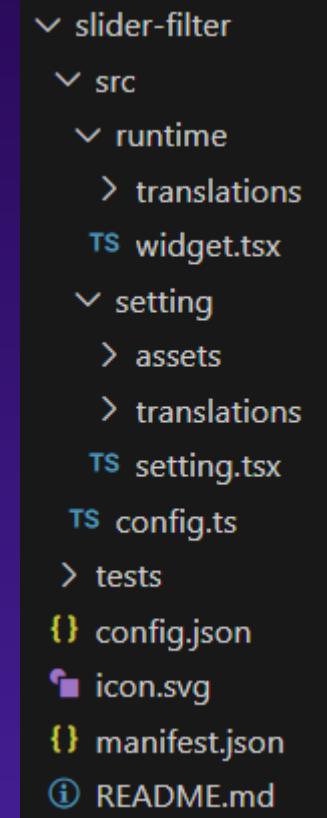
Widget Structure



Widget File Structure

Custom widgets

- **widget.tsx** controls widget behavior during runtime
- **setting.tsx** controls widget behavior during the app-building stage
- **config.ts** defines the structure of the widget's configuration
- **config.json** sets the widget's default configuration
- **manifest.json** allows the widget to interact with the rest of ArcGIS Experience Builder



Widget Development

Custom widgets

- TypeScript and React
- Jimu Framework for ArcGIS Experience Builder
 - jimu-core
 - jimu-arcgis
 - jimu-for-builder
 - jimu-ui
 - jimu-for-test
- ArcGIS Maps SDK for JavaScript
- Calcite Design system

积木 (Jīmù) is Mandarin for
“building block”

Build a basic Widget

Christine Wiltawsky

```
const routeParams = new RouteParam.  
stops: new FeatureSet({  
    features: view.graphics.toArray()  
}),  
returnDirections: true  
});
```

```
var mapView =  
    container: "v_  
    map: map,  
    zoom: 10,  
    center: [-73.95, 40..  
]);
```

Steps to set up your development environment

- Install Prerequisites (NodeJS)
- Download Experience Builder Developer Edition
- Unzip the files
- Command line in /server directory:
 - npm ci
 - npm start
- Command line in /client directory:
 - npm ci
 - npm start

Add widget to development application

- Open Experience Builder
- Add your widget
- Save and Publish

Activity

30 Minutes

- Review the “Getting Started” documentation:
<https://developers.arcgis.com/experience-builder/guide/getting-started-widget/>
- Set up your text editor (VS Code or similar)
- Duplicate the Simple widget and add it to an Experience
- Open the “client/” folder in your IDE, make a small code change to make sure your dev environment is working
- Bonus:
 - Update the “defaultSize” property of the manifest.json and see how that affects your widget
 - Update the icon of your widget.

REACT basics

- A Widget is a React component
 - <https://react.dev/learn>
- Hooks
 - <https://react.dev/reference/react/hooks>
 - useState
 - useEffect
- Functions instead of Class Components
- JSX
 - <https://developers.arcgis.com/experience-builder/guide/widget-ui/#writing-jsx>
 - <https://react.dev/learn/writing-markup-with-jsx>

TypeScript

Defining Types

- <https://typescriptlang.org/>

```
const user = {  
    name: "Hayes",  
    id: 0,  
};
```

```
interface User {  
    name: string;  
    id: number;  
}
```

```
const user: User = {  
    name: "Hayes",  
    id: 0,  
};
```

TypeScript

Types by Inference

- <https://typescriptlang.org/playground>

```
let helloWorld = "Hello World";
// ^ = let helloWorld: string
```

TypeScript

Unions

```
type MyBool = true | false;
```

```
type WindowStates = "open" | "closed" | "minimized";
```

```
type LockStates = "locked" | "unlocked";
```

```
type OddNumbersUnderTen = 1 | 3 | 5 | 7 | 9;
```

```
function getLength(obj: string | string[]) {  
    return obj.length;  
}
```

TypeScript

Generics

```
type StringArray = Array<string>;  
type NumberArray = Array<number>;  
type ObjectWithNameArray = Array<{ name: string }>;
```

Localization

Christine Wiltawsky

```
const routeParams = new RouteParam.  
  stops: new FeatureSet({  
    features: view.graphics.toArray()  
  }),  
  returnDirections: true  
});
```

```
var mapView =  
  container: "v_  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40..  
]);
```

What is it?

- “Localization” / “Translation” / “i18n”
- No “hardcoded strings” within your code

Why?

- Easier at the beginning!
- Not multi-language? May be requirement later
- Cleaner code - no “hardcoded strings” within your code

Localization in Experience Builder

- OOTB widgets are already localized/translated
- Tools are built-in!

manifest.json: translatedLocales

- Within manifest.json
- What languages are supported

```
"translatedLocales": [  
    "en",  
    "es",  
    "zh-cn"  
]
```

Translation Files

- “runtime/translations/default.ts”
- “runtime/translations/es.js”
- “runtime/translations/zn-ch.js”
- ...

```
export default {  
    _widgetLabel: 'My Widget',  
    str1: 'String 1',  
}
```

- “setting/translations/default.ts”
- “setting/translations/es.js”

```
System.register([], function (_export) {return {execute: function () {_export({  
    // the strings  
    _widgetLabel: 'Translated Widget Name',  
    str1: 'Translated String 1',  
})}}});
```

Using the strings

- First:

```
import defaultMessages from "./translations/default";
```

- Usually:

```
<FormattedMessage id="str1" defaultMessage={defaultMessages.str1}/>
```

- Direct:

```
props.intl.formatMessage({id: 'str1', defaultMessage: defaultMessage.str1})
```

Debugging

- URL param Format:

&locale=es

Activity

10 Minutes

- Add translated strings to the widget you created in the previous lesson.
- Bonus:
 - Add multiple language support
- Documentation:
<https://developers.arcgis.com/experience-builder/guide/extend-base-widget/#i18n-support>

Connect Widget with a map

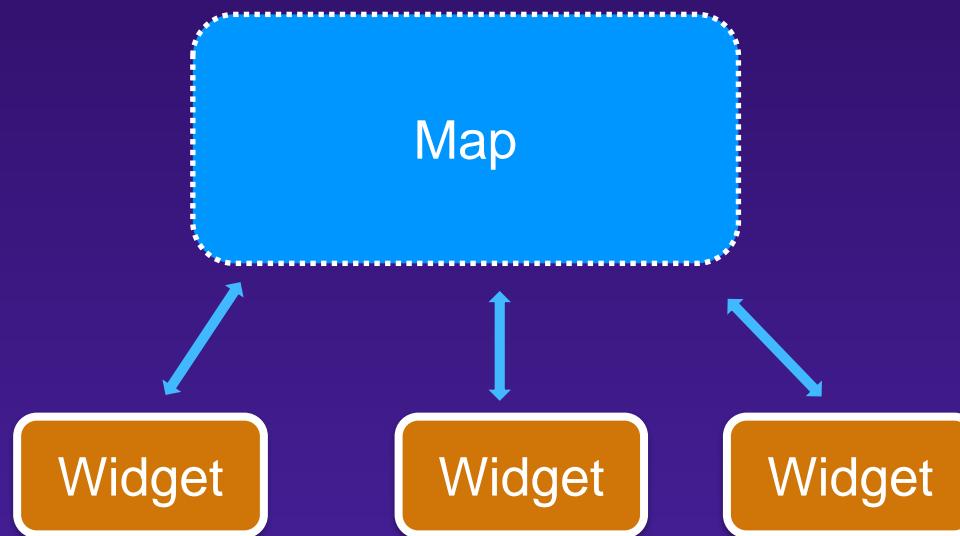
Christine Wiltawsky

```
const routeParams = new RouteParam({  
  stops: new FeatureSet({  
    features: view.graphics.toArray()  
  }),  
  returnDirections: true  
});
```

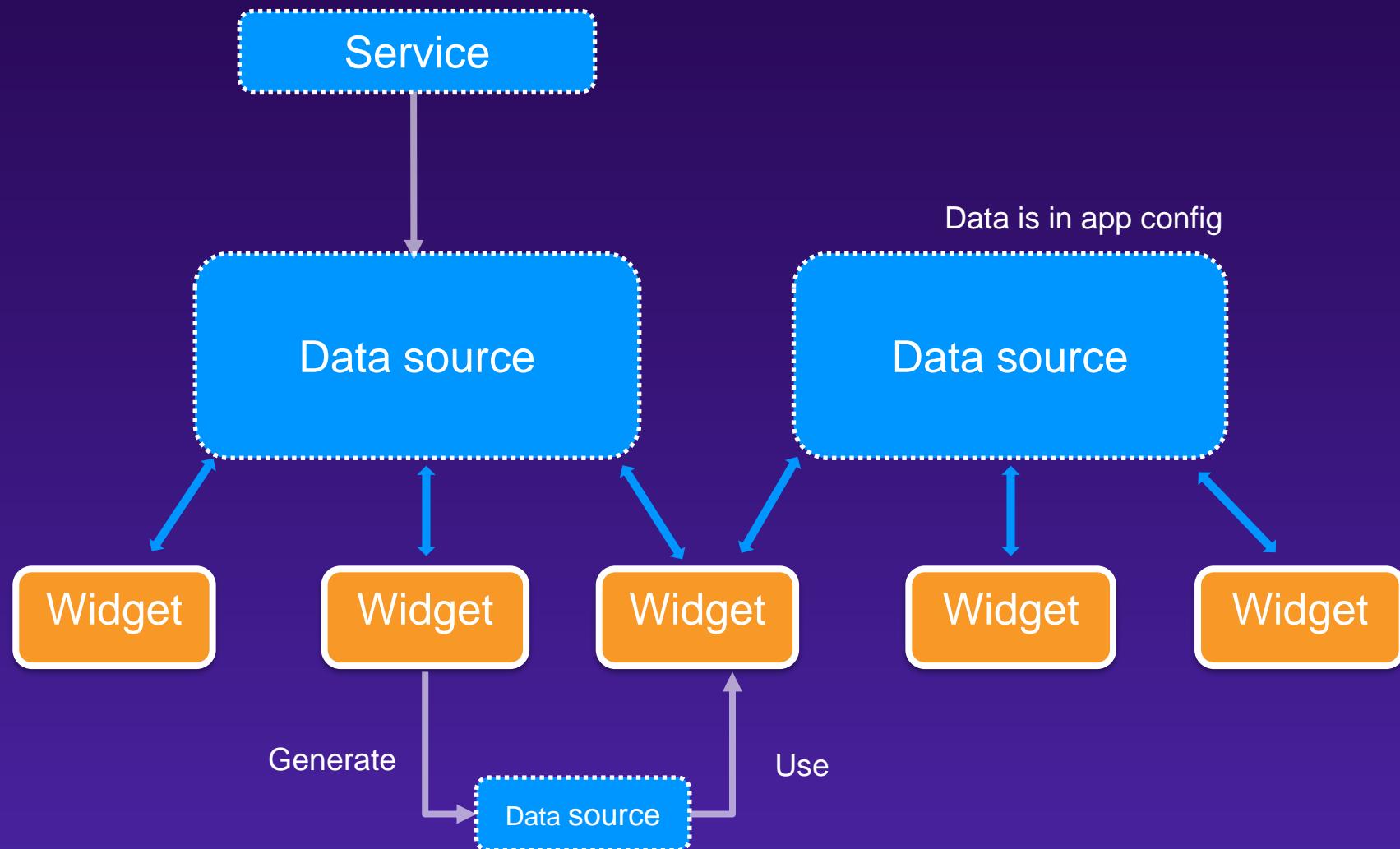
```
var mapView =  
  container: "view",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.7],  
  style: "mapbox://style/...");
```

Web AppBuilder

- Map-centric app
- All widgets holds a map object and can communicate to the map directly



ArcGIS Experience Builder



MapView/SceneView

- View concept same as ArcGIS JavaScript API
 - ExB wrapper for the "view" as JimuMapView
- JimuMapView objects has these main properties:
 - view: map/scene view object
 - dataSourceId: datasource to create the view
 - mapWidgetId: the widget that creates the objects
 - jimuLayerViews: the layer view object wrapper

```
MapViewManager.getInstance().createJimuMapView({
  mapWidgetId: this.props.id,
  view: new MapView(options),
  datasourceId: webmapDs.id,
  isActive: true
})
```

Activity

Part 1- 15 Minutes

- Create “src/settings/settings.tsx”
 - Add the MapWidgetSelector to allow the widget author to choose their map
 - Add a text input for the exampleConfigProperty
 - Update widget.tsx to use JimuMapViewComponent to get a reference to the map
 - Bonus: use console.log() to list the layer names from your map
-
- <https://github.com/Esri/arcgis-experience-builder-sdk-resources/tree/master/widgets/get-map-coordinates-function>
 - <https://developers.arcgis.com/experience-builder/guide/get-map-coordinates/>

Activity

Part 2 - 15 Minutes

- Set up the map listener to show the lat/lion and scale (like demo)
- Add “zoom level” to display
- Add an additional event listener so the lat/lion updates based on the mouse clicked
- Bonus: Make “zoom level” visibility configurable through the settings panel

Actions

Christine Wiltawsky

```
const routeParams = new RouteParam.  
stops: new FeatureSet({  
    features: view.graphics.toArray()  
}),  
returnDirections: true  
});
```

```
var mapView =  
    container: "v_  
    map: map,  
    zoom: 10,  
    center: [-73.95, 40..  
});
```

Widget Action

- Widgets communicate with each other
- Data actions are opt-in
- Message actions are automatic

A screenshot of a Table widget titled "Interior Landmarks". The table has columns for BoroughID and Name. A context menu is open over the second row, with the "Data action" option highlighted by a yellow arrow. The menu items include "Export to JSON", "Export to CSV", "Export to GeoJSON", "Export all", "Export selected", "Zoom to", "Pan to", and "Show on map".

Data action

The "Data action" configuration interface for "Table 1". It shows a toggle switch labeled "Enable data action" which is turned on. Below it, a list of data actions is shown with checkboxes:

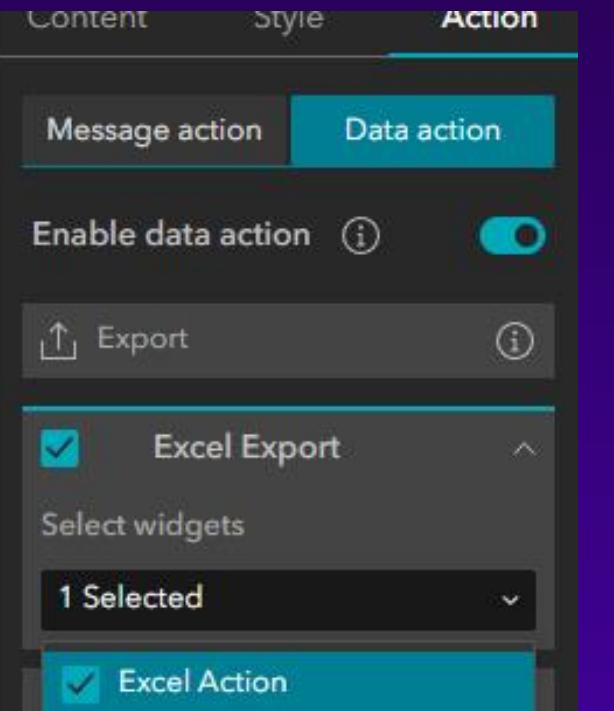
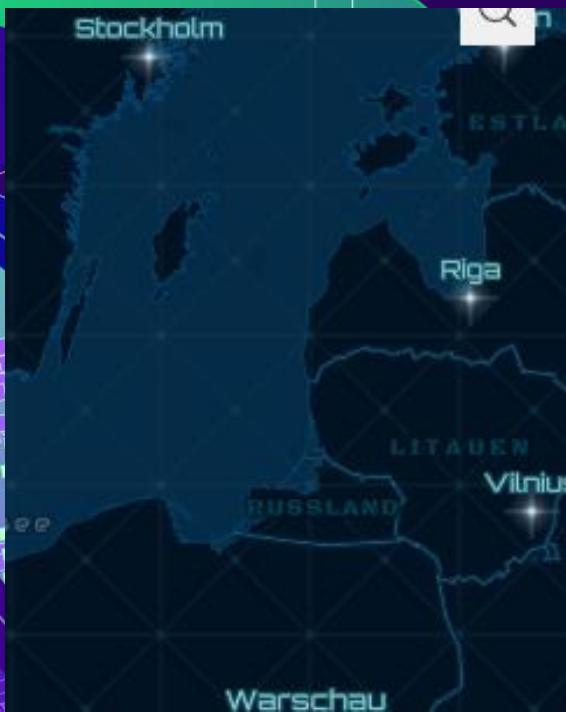
- ✓ Pan to
- ✓ Show on map
- ✓ Zoom to

A "Data action" button is also present.

Message action

The "Message action" configuration interface for "Map 2". It shows a "Trigger" section with "Extent Changes" selected. Below it, a "Map 1" section with "Zoom to" and a "+ Add action" button.

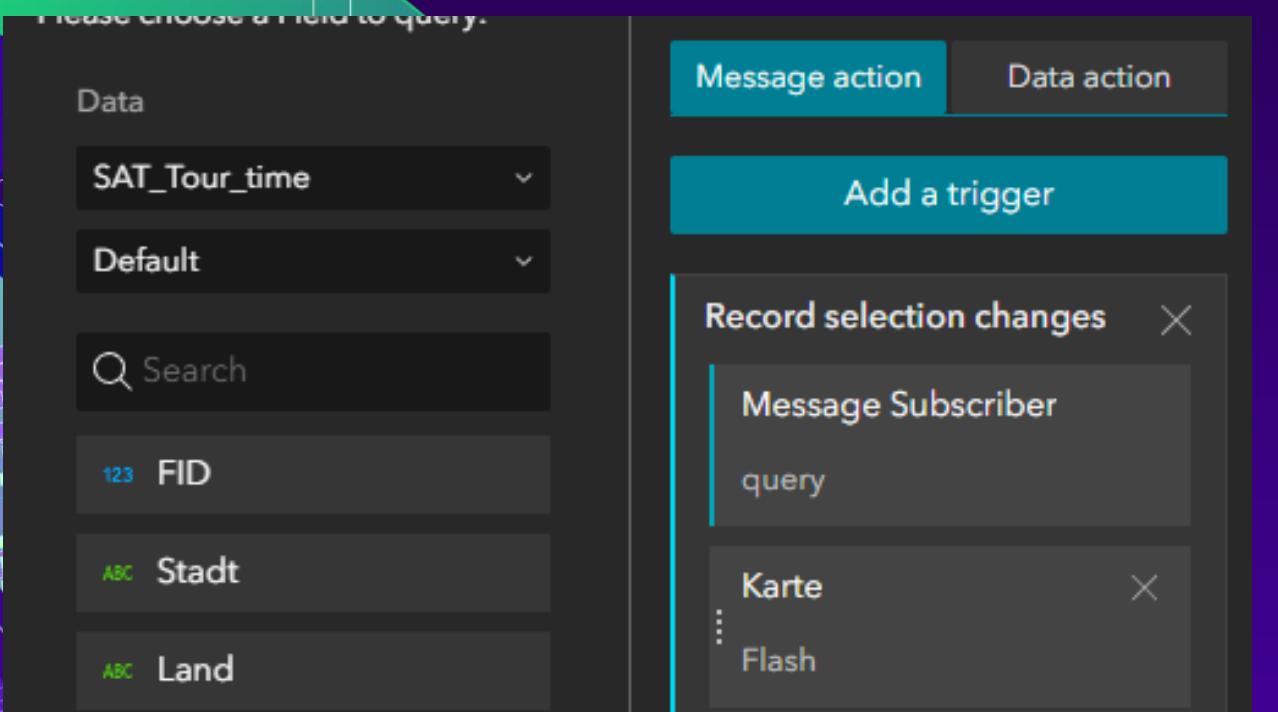
```
var mapView = new MapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.73]  
});
```



Data Action

Receive Data

```
var mapView = newMapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.73]  
});
```



Message Action

Perform process on trigger

Message Action

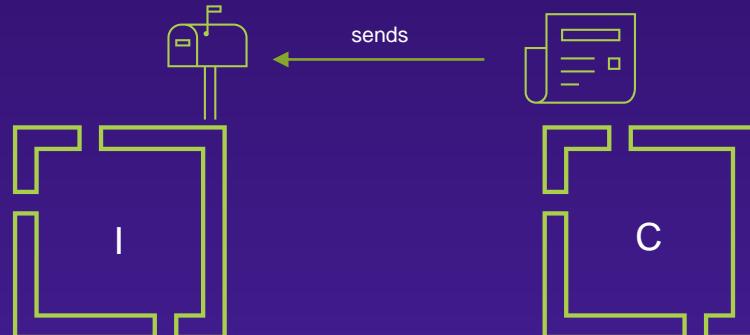
- A trigger broadcasts a message
- Targets listen to the message
- Actions provide Target behavior

Trigger	Target	Action
Extent Changes from Map 1	Map 2	Zoom to the same extent as Map 1
	Framework (data)	Filter Data Records For example, List 1 displays only features within the current map extent

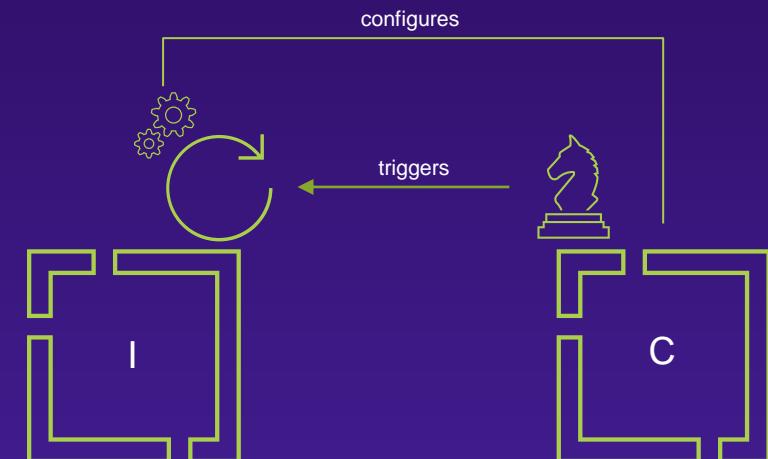
Actions

Send data and messages to other widgets

Data action



Message action



Samples Actions

- Message Action Subscribe
 - <https://github.com/Esri/arcgis-experience-builder-sdk-resources/tree/master/widgets/message-subscriber>
- Message Action Publisher
 - Sample: Table Widget
 - <https://github.com/esride-nik/ExB-workshop/tree/master/widgets/messagePublisher>
- Data Action
 - Sample Table Widget
 - <https://github.com/esride-nik/ExB-workshop/blob/master/widgets/excelExport/src/data-actions/excel-export.ts>
- <https://developers.arcgis.com/experience-builder/guide/core-concepts/data-action/>
- <https://developers.arcgis.com/experience-builder/guide/core-concepts/message-action/>

DataSources

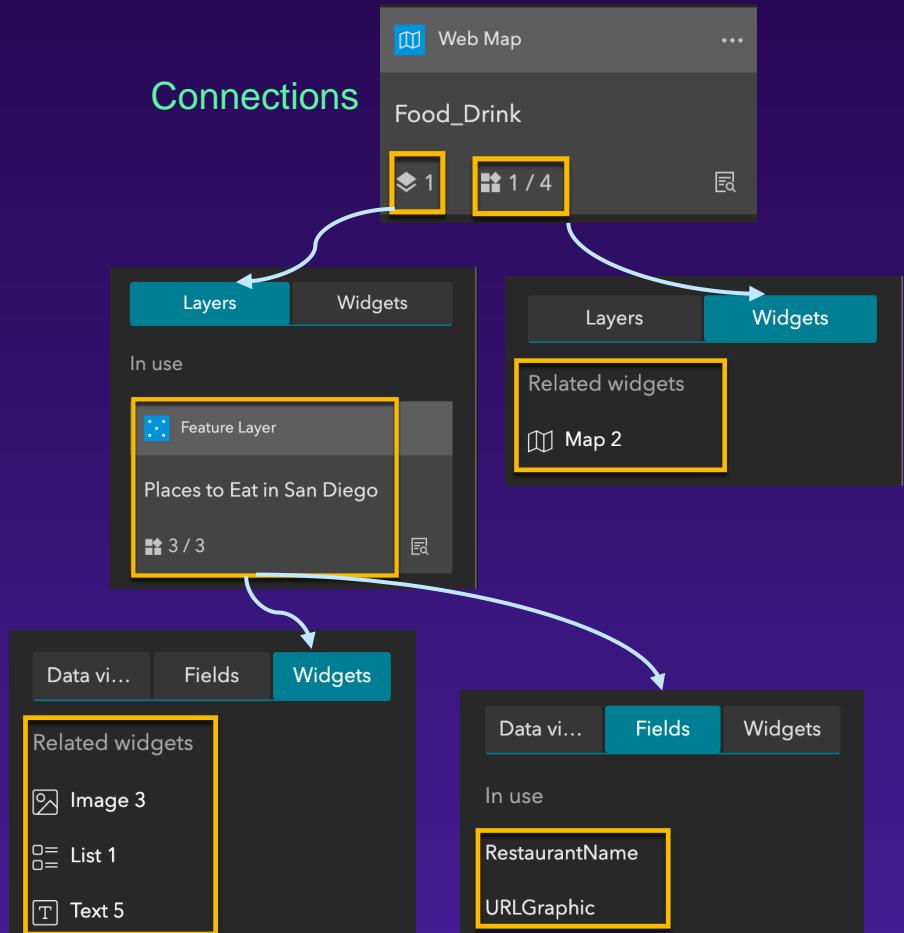
Christine Wiltawsky

```
var mapView =  
  container: "v_..  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40..  
});
```

```
const routeParams = new RouteParam.  
  stops: new FeatureSet({  
    features: view.graphics.toArray()  
  }),  
  returnDirections: true  
});
```

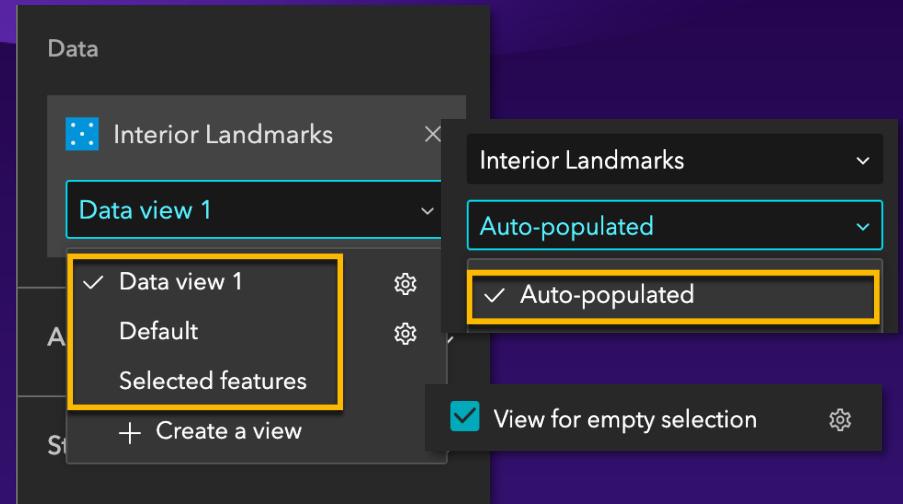
Data Source

- Manage and share data at the app (framework) level
 - Hosted feature layers
 - Feature & map services
 - Feature collections
 - Scene layers
- Show connections between Map, Layer, Field, Widget



Data Source Views

- Default view
- Selected features' view
 - Use **View for empty selection** to choose the default feature
- Auto-populated view
- Data view

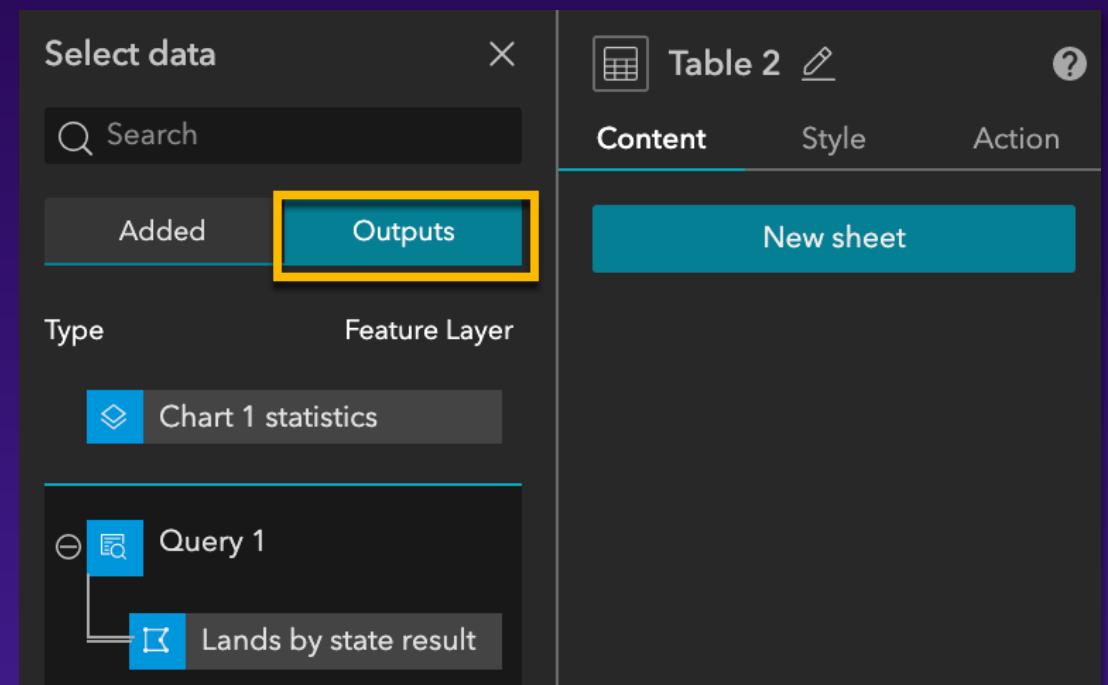


A screenshot of the "Data view 1" configuration pane. It displays a table of records with the following columns: Distribute..., GlobalID, website, In Places, Notes, OBJECTID, and photo. The "Sort" tab is currently selected, showing a sorting interface with up and down arrows and a "Add a sort field" button. Below the table, there are tabs for "Data vi...", "Fields", and "Widgets", along with a "Create a view" button. The table itself contains 14 rows of data, each with a "Public" status and a unique GlobalID.

Distribute...	GlobalID	website	In Places	Notes	OBJECTID	photo
Public	5b146479...		Yes	Park provi...	1	
Public	7177d821...		Yes	Park provi...	2	
Public	769c412b...		Yes	Park provi...	3	
Public	66ccd7cc...		Yes	Park provi...	4	
Public	660e80a2...		Yes	Park provi...	5	
Public	50ff3544...		Yes	Park provi...	6	
Public	41fe19ea...		Yes	Park provi...	7	
			Yes	Park provi...	8	
			Yes	Park provi...	9	
			Yes	Boat Ramp	10	
			Yes	Boat Ramp	11	
			Yes	Restroom	12	
			Yes	Restroom	13	
			Yes	Dinner Area	14	

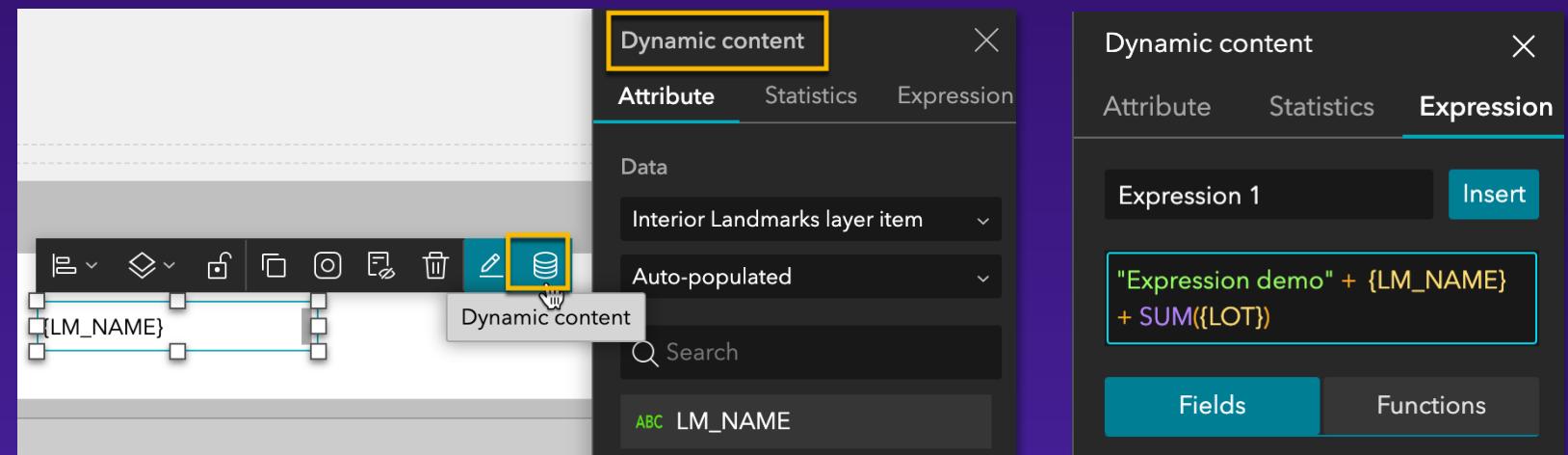
Output Data Source

- Generated by widgets at runtime
 - Chart, Query, Directions, Elevation Profile
- Other widget's data source
 - such as Table and Text



Dynamic Content

- Dynamically display attribute values, statistics or expressions
 - In Image, Text, Button, List, Card, and Embed
 - Must connect to data



Key points

- Data drives the experience
- Share data between widgets
- Pipe data from one widget to another
- Abstraction - changing a data source does not break the app
- <https://developers.arcgis.com/experience-builder/guide/widget-communication/>

Data source

- Defines how widgets access data
- jimu-core defines the DataSource interface with common methods and properties:
 - <https://developers.arcgis.com/experience-builder/api-reference/jimu-core/DataSource>
 - Custom data sources can be created by using the DataSource interface
 - Manage with DataSourceManager
 - <https://developers.arcgis.com/experience-builder/api-reference/jimu-core/DataSourceManager/>

Common classes and interfaces

<https://developers.arcgis.com/experience-builder/api-reference/jimu-core/>

- **DataSourceManager**
 - Provides access to all the datasources in the app
- **DataSourceComponent**
- **DataSource**
 - `getSelectedRecords()`
 - `nextRecord()`
 - `getStatus()`
 - `getData()`
 - `getId()`
- **QuerableDataSource**
 - `updateQueryParams`
 - `load`
 - `query()`
 - `getRealQueryParams()`
- **FeatureLayerDataSource**
- **WebMapDataSource/WebSceneData source**

How are data sources persisted?

- Saved in the dataSources property of the app config
 - Use of a datasource is declared with useDataSources Property
- config location:
 - server\public\apps\<app_id>\resources\config\config.json

```
1427  "dataSources": {  
1428    "dataSource_1": {  
1429      "id": "dataSource_1",  
1430      "type": "WEB_MAP",  
1431      "sourceLabel": "DS2023 - Regional Offices",  
1432      "itemId": "fecde6493224443ba8b2c77b633dd482",  
1433      "portalUrl": "https://arcgis-devlabs.maps.arcgis.com"  
1434    },  
1435    "dataSource_2": {  
1436      "id": "dataSource_2",  
1437      "type": "WEB_MAP",  
1438      "sourceLabel": "Regional Office Boundaries",  
1439      "itemId": "444d345cd3084d8c8a8e1e3c2eaca25f",  
1440      "portalUrl": "https://arcgis-devlabs.maps.arcgis.com"  
1441    },  
1442    "dataSource_3": {  
1443      "id": "dataSource_3",  
1444      "type": "WEB_MAP",  
1445      "sourceLabel": "Regional Office Boundaries",  
1446      "itemId": "444d345cd3084d8c8a8e1e3c2eaca25f",  
1447      "portalUrl": "https://arcgis-devlabs.maps.arcgis.com"  
1448    },  
1449  },  
1450  "messageConfigs": {  
1451    "messageConfig_1": {  
1452      "id": "messageConfig_1",  
1453      "widgetId": "widget_1"
```

Accessing data sources in your widget

- Widget dependent on map (e.g., bookmark, legend, survey)
 - `widget.tsx`
 - `JimuMapView`
 - `setting.tsx`
 - `JimuMapViewSelector`
- Widget is not dependent on map
 - `widget.tsx`
 - `DataSourceComponent`
 - `setting.tsx`
 - `DataSourceSelector`
 - `FieldSelector`

Activity

30 Minutes

- Download and install the ‘get-map-coordinates-function’ widget from github
 - <https://github.com/Esri/arcgis-experience-builder-sdk-resources/tree/master/widgets/get-map-coordinates-function>
 - Explore the code and add to an application
 - How is a reference to the map datasource made?
- Download and install the ‘listen-selection-change’ widget from github
 - <https://github.com/Esri/arcgis-experience-builder-sdk-resources/tree/master/widgets/listen-selection-change>
 - Explore the code in this widget and add to an application
 - How is the reference to a datasource accomplished in this code?
- ** Challenge: In the ‘listen-selection-change’ widget add a field selector to the settings to allow end users to modify which field is used to display on the widget.

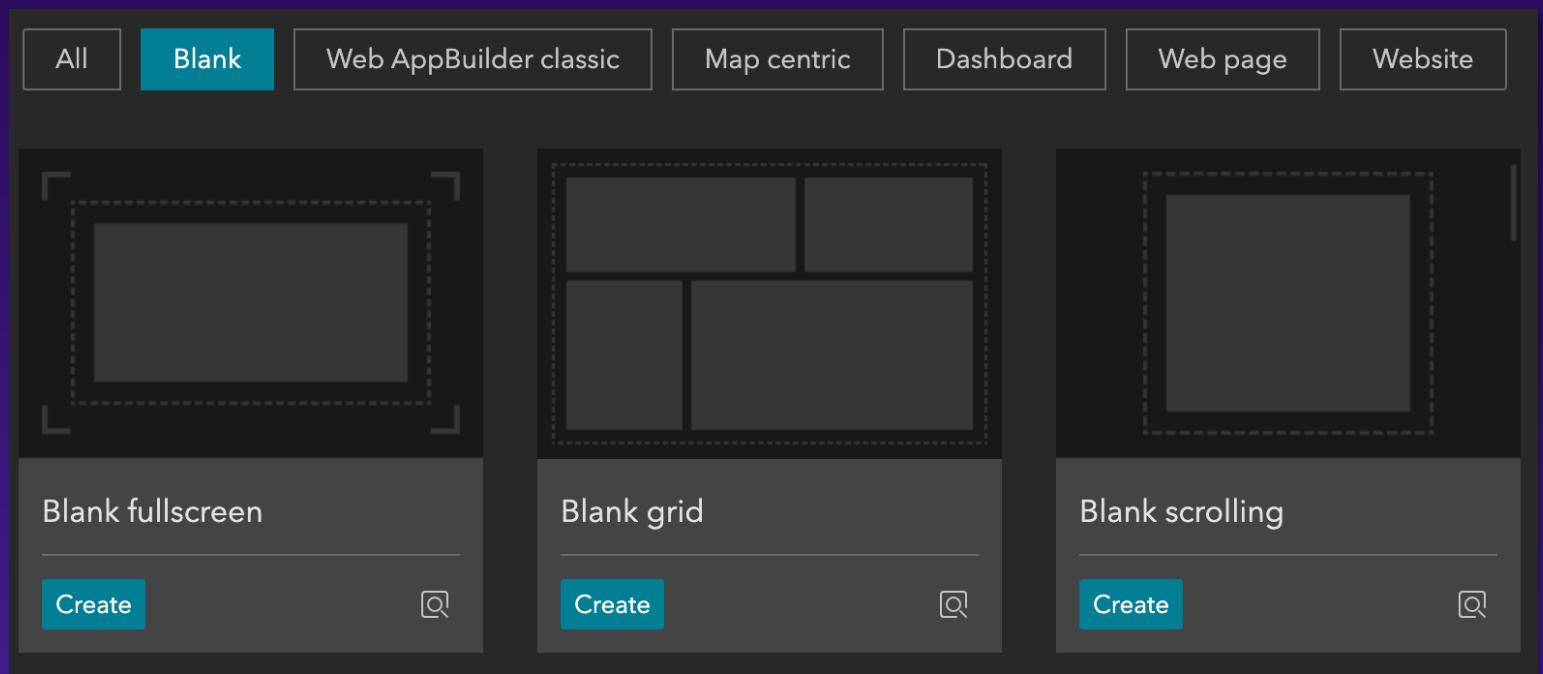
Templates

```
var mapView =  
  container: "v_..  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40..  
});
```

```
const routeParams = new RouteParam.  
  stops: new FeatureSet({  
    features: view.graphics.toArray()  
  }),  
  returnDirections: true  
});
```

Default Templates

- Fullscreen
- Scrolling
- Grid



Public Templates

Available in ArcGIS Online

The screenshot shows the ArcGIS Online interface with a dark theme. At the top, there are navigation tabs: Default, My templates, My favorites, My groups, My organization, **ArcGIS Online**, and Living Atlas. Below these tabs, there are four rows of template cards. Each card includes a preview image, the template name, and two buttons: 'Create' and 'View'.

- Row 1:**
 - Industrial Annual Report
 - Yellowstone Tour Directory
 - Grand Lyon 3D
 - Coastal tourism sites
- Row 2:**
 - Space Exploration
 - County Schools Editor
 - Hotel Booking
 - City News

The screenshot shows the ArcGIS Online interface with a dark theme, similar to the first one. At the top, there are navigation tabs: Default, My templates, My favorites, My groups, My organization, **ArcGIS Online**, and Living Atlas. Below these tabs, there are four rows of template cards. Each card includes a preview image, the template name, and two buttons: 'Create' and 'View'.

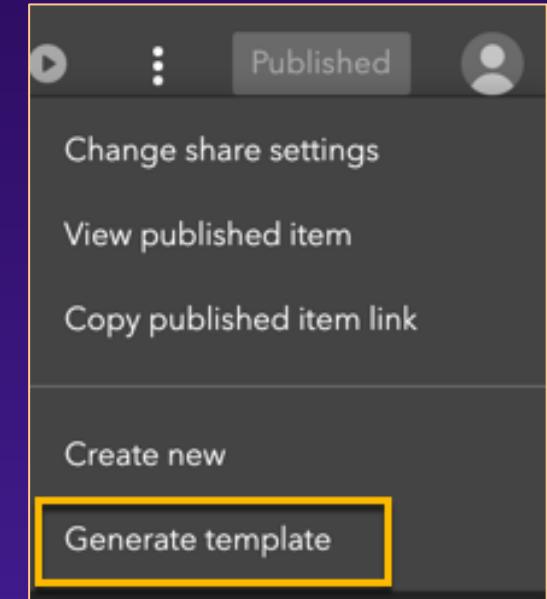
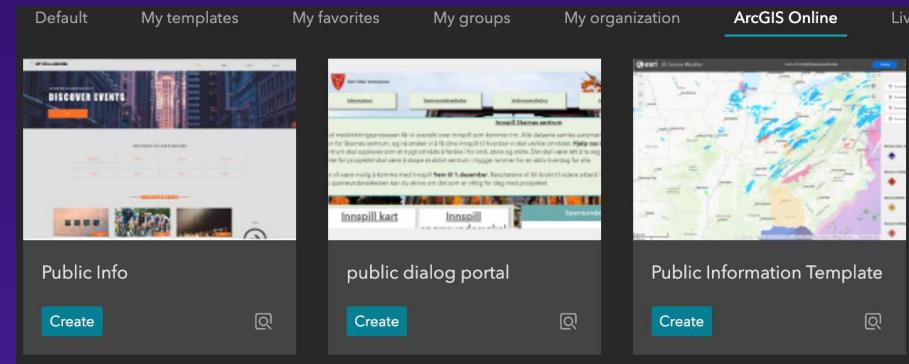
- Row 1:**
 - Demographic Summary of ...
 - City wide demographic an...
 - Analyze LA metropolitan us...
 - Key Demographic Facts wit...
- Row 2:**
 - Demographic and Spendin...
 - Analyze existing businesse...
 - Health Care and Insurance ...
 - Community Profile around ...

**Business
Analyst
Templates**

Showcase
New
Features

Generate templates

- Create your own starters for reuse and branding
- Shared with others



Themes

Theme

- Defines color and font
- Customized options
 - Color
 - Font style
 - Font size

Default

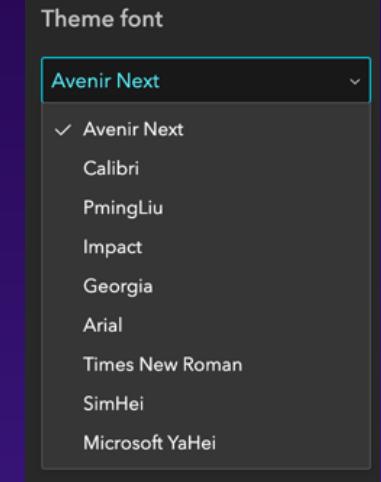
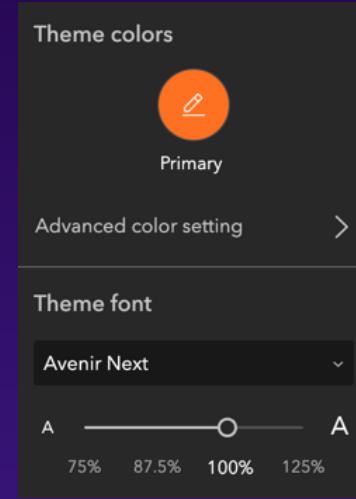
Dark

Vivid

Ink

Mysterious

Shared



Do I need a custom theme?

Template vs. Theme

- Template
 - Colors
 - Predefined Fonts
 - Logos
- Theme
 - Colors
 - Custom Fonts
 - Logos
 - Spacing
 - Style override via Sass CSS
 - Images
 - Borders

Basic Theme

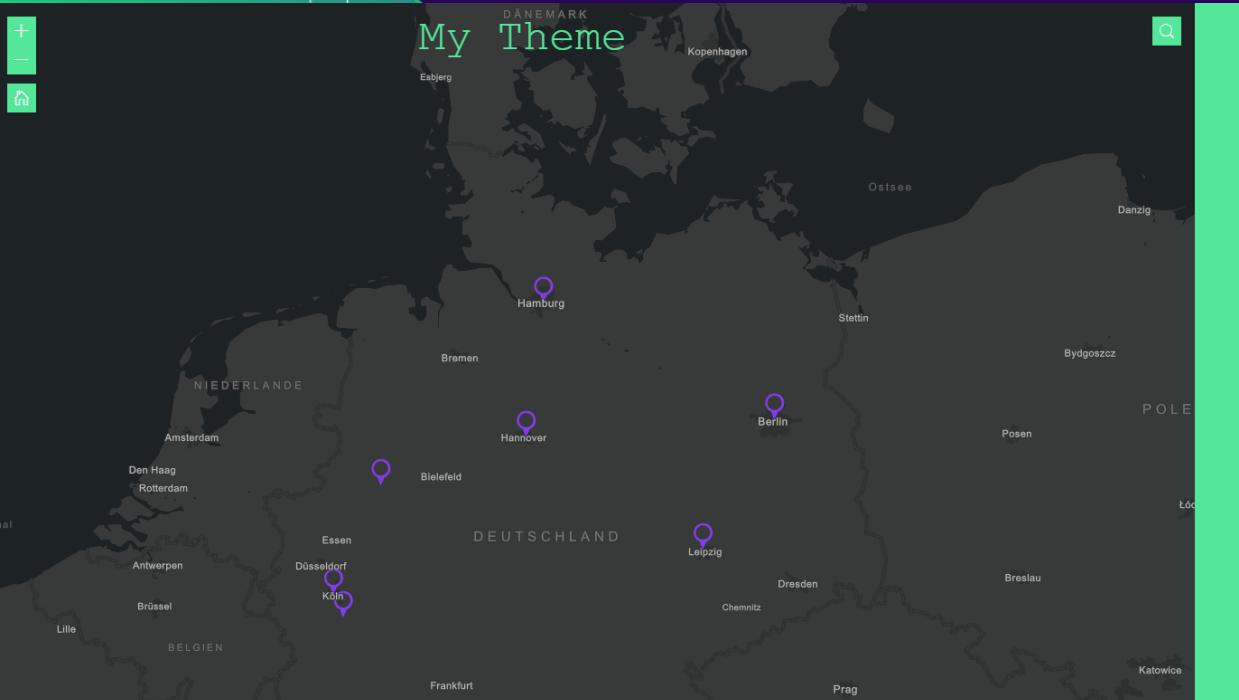
- Override theme variables in json

- Files

- variables.json
 - thumbnail.png
 - manifest.json

- Full theme variable json

```
var mapView = new MapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.73]  
});
```



Basic Theme

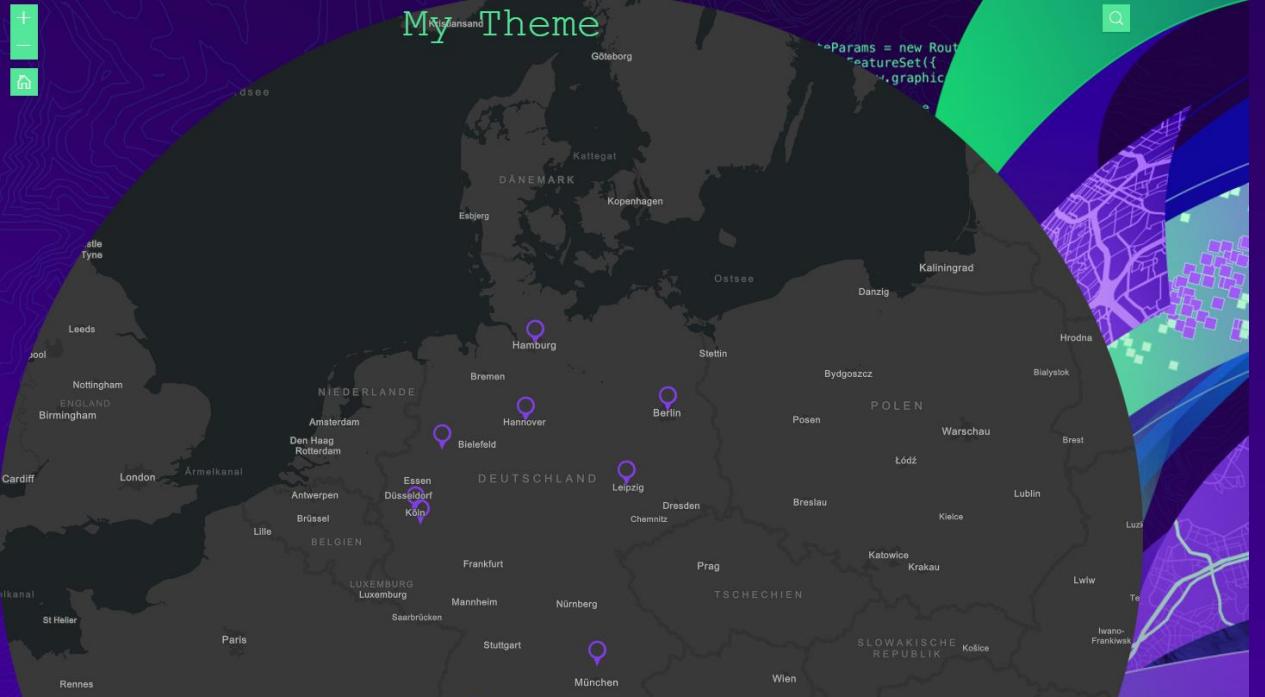
Christine Wiltawsky

Override CSS styles

Two different ways of writing styles

- style.ts
 - Use the Emotion library for CSS-in-JS
 - Recommended
- style.scss
 - More traditional way
 - Imported last
- Only one style file can be added to a theme folder

```
var mapView = new MapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.73]  
});
```



Override CSS

Christine Wiltawsky

Components and Widget Styling

Christine Wiltawsky

```
const routeParams = new RouteParam({  
  stops: new FeatureSet({  
    features: view.graphics.toArray()  
  }),  
  returnDirections: true  
});
```

```
var mapView =  
  container: "view",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.7],  
  style: "light"  
);
```

Jimu UI Library

- UI library of components for developers to use:
 - client\jimu-ui\index.d.ts
- Basic UI components:
 - button, dropdown, form controls,
 - icon, navigation, modal,
 - grid layout container, etc.
- Advanced UI components:
 - date picker,
 - resource selector,
 - expression builder, etc.

Storybook Documentation

The screenshot shows the Storybook interface for the **jimu-ui** library. The left sidebar lists various components under the **Welcome** category, including Alert, Badge, Button, ButtonGroup, Card, Checkbox, Dropdown, DropdownButton, DropdownItem, DropdownMenu, Icon, Label, Link, Loading, Modal, MultiSelect, Nav, Navbar, NavItem, NavLink, NavMenu, NumericInput, Pagination, and Popover. The **index** component is currently selected. The main canvas displays a collection of UI components arranged around a central hexagonal logo featuring a stylized 'J' and 'M'. These components include a text input field, a dropdown menu, a checkbox, a switch, a range slider, a multi-select dropdown, and a circular slider.

This is the documentation for the **jimu-ui** library of ArcGIS Experience Builder.

We provide a wealth of out-of-the-box components for developers to use:

developers.arcgis.com/experience-builder/api-reference/jimu-ui/

Using UI Components

```
3 // in widget.tsx:
4 import { React, AllWidgetProps } from 'jimu-core';
5 import { Button, Icon } from 'jimu-ui'; // import components
6
7 // Create an svg icon using Icon component:
8 const iconNode = <Icon icon={require('jimu-ui/lib/icons/star.svg')} />;
9
10 export default class Widget extends React.PureComponent<AllWidgetProps<IMConfig>, any> {
11   render() {
12     // Add Button component containing an icon to the widget:
13     return <Button type='primary'>{iconNode} primary button</Button>;
14   }
15 }
16
```

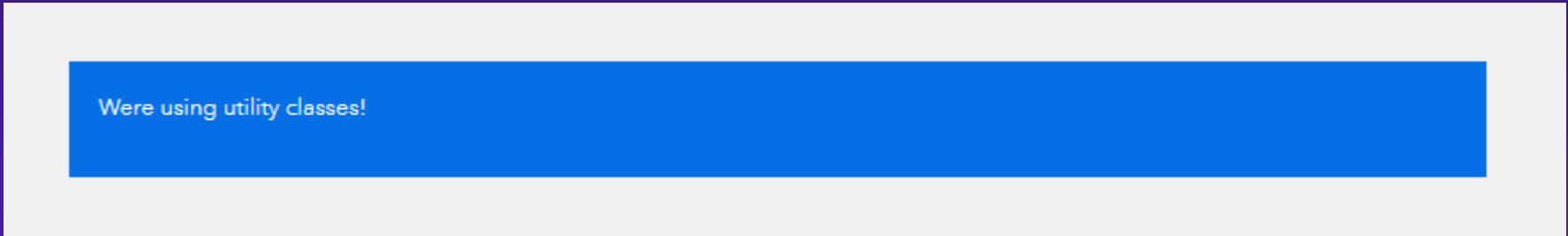
Styling your custom widget

- Utility Classes
- Inline CSS
- CSS/Sass style files
- CSS-in-JS
- Styled Components

Utility Classes

- Jimu UI provides the same CSS utility classes as [Bootstrap](#) to quickly apply styles to UI elements.
 - <https://getbootstrap.com/docs/4.3/utilities>

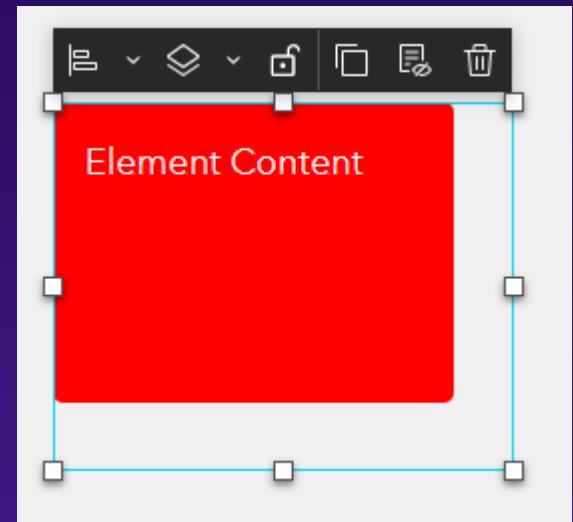
```
9  render() {  
10    return (  
11      <div className='w-100 p-3 bg-primary text-white'>  
12        <p>Were using utility classes!</p>  
13      </div>  
14    );  
15  }
```



Were using utility classes!

Inline CSS

```
12  render() {
13    const containerStyle = {
14      background: 'red',
15      color: 'white',
16      width: 200,
17      height: 150,
18      padding: '1rem',
19      borderRadius: 5,
20      fontSize: 'large',
21      fontWeight: 'bold',
22    };
23
24    return (
25      <div
26        style={containerStyle} // CSS styles applied
27      >
28        Element Content
29      </div>
30    );
31  }
```



Using CSS Style Sheets

- Define CSS styles in external stylesheet files and import them separately in the widget.
- Accepted stylesheet file extensions are: .css, .sass and .scss.

The image shows a code editor interface with two tabs: '# style.css' and 'TS widget.tsx'. The '# style.css' tab contains a CSS style sheet with a single class definition named 'containerStyle'. The 'widget.tsx' tab contains a TypeScript component named 'Widget' that imports the 'style.css' file and uses its styles in a render function.

```
# style.css
client > your-extensions > widgets > simple > src > # style.css > .containerStyle
1  .containerStyle {
2    background-color: 'red';
3    color: 'white';
4    width: 200;
5    height: 150;
6    padding: '1rem';
7    border-radius: 5;
8    font-size: 'large';
9    font-weight: 'bold';
10   }
11

TS widget.tsx
client > your-extensions > widgets > simple > src > runtime > TS widget.tsx > Widget > render
8   import './style.css';
9
10  export default class Widget extends React.PureComponent<
11    AllWidgetProps<IMConfig>,
12    any
13  > {
14    render() {
15      return (
16        <div
17          className='containerStyle' // CSS styles applied
18        >
19          Element Content
20        </div>
21      );
22    }
23  }
```

CSS in JS

- The “CSS prop”
 - CSS styles can be written in template literals, which allows you to write JS logics inside of CSS.

```
100
101  const over = this.state.scale > 5000000;
102  const scaleStyle = css`  

103    font-weight: ${over ? 'bold' : 'normal'};  

104    color: ${over ? 'red' : 'black'};  

105  `;  

106
```

```
<span css={scaleStyle}>Scale 1:{this.state.scale}</span>
```

CSS in JS

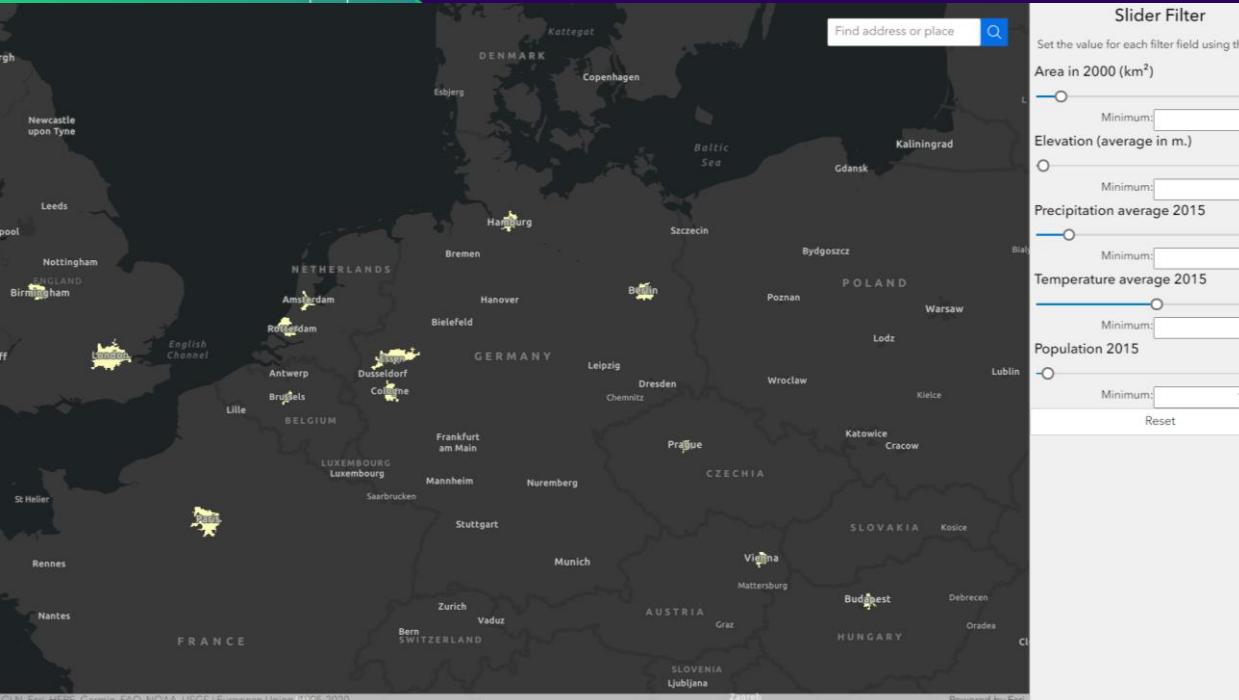
- Styled components

- Like css prop, but for components
- Style components using Template Literals
- The "styled" approach is perfect for creating reusable components within your widget

```
39 // A styled button component:  
40 const StyledButton = styled.button`  
41   color: white;  
42   border: none;  
43   border-radius: 5px;  
44   background-color: blue;  
45   transition: 0.15s ease-in all;  
46   &:hover {  
47     background-color: green;  
48   }  
49 `;
```

```
... //<Button type='primary' onClick={this.resetView}>  
... // Reset  
... //</Button>  
... // A styled button component:  
... <StyledButton onClick={this.resetView}>Reset</StyledButton>
```

```
var mapView = new MapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.73]  
});
```



Calcite Components

```
= new RouteParameters({  
  routeSet:{  
    featureSet:  
      stops:  
        featureSet:  
          graphics.toArray()  
    },  
    returnRoute:  
      true  
  })
```

Mobile Layouts

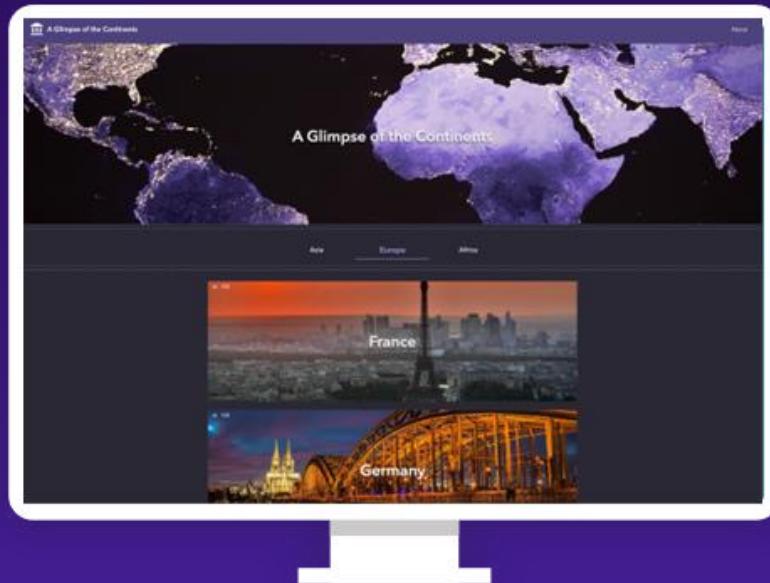
Christine Wiltawsky

```
const routeParams = new RouteParam.  
stops: new FeatureSet({  
    features: view.graphics.toArray()  
}),  
returnDirections: true  
});
```

```
var mapView =  
    container: "v_...  
    map: map,  
    zoom: 10,  
    center: [-73.95, 40..  
]);
```

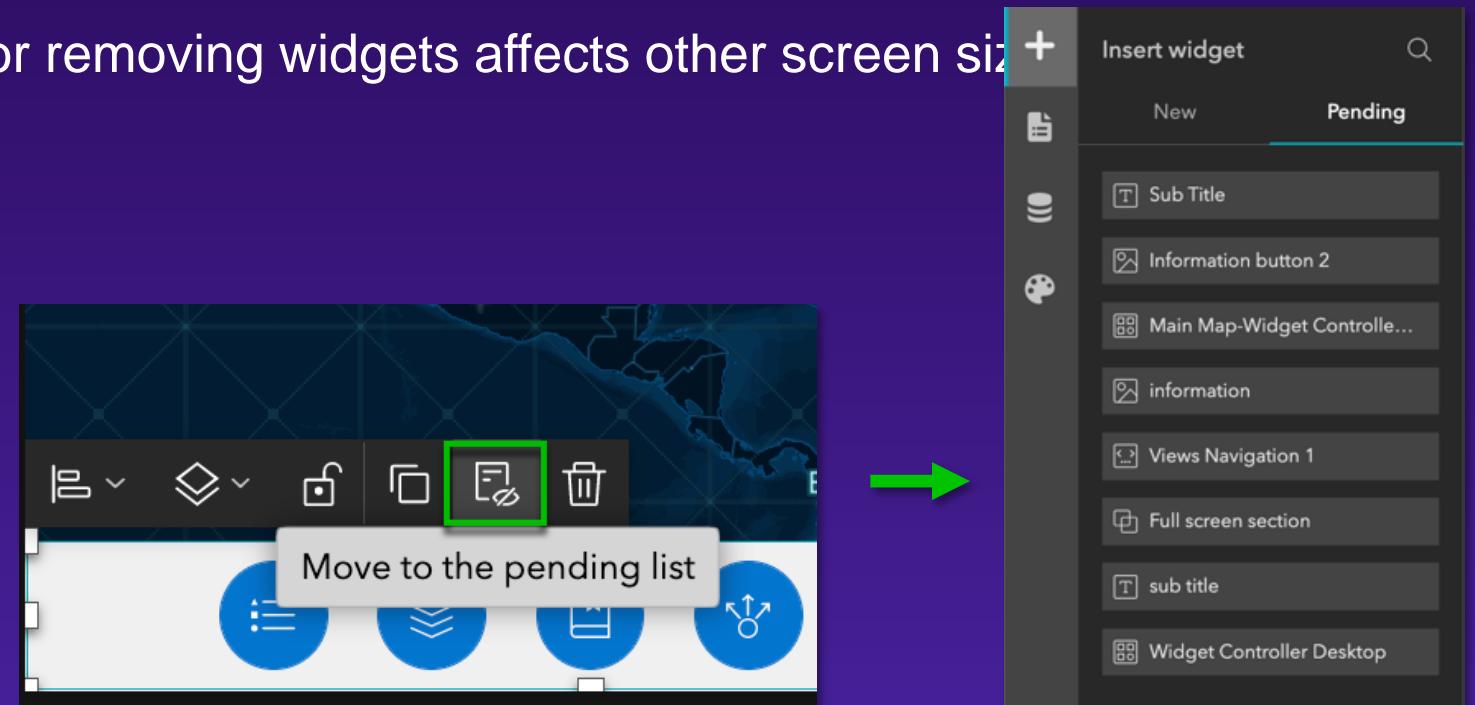
Mobile Optimization

- Configure apps on Mobile, Tablet, and Desktop differently with one URL



Keys for Mobile Optimization

- Auto vs Custom mode
- Changing position, size, and style of widgets won't affect other screen sizes
- Changing content of widgets or removing widgets affects other screen sizes
 - Move to the pending list



```
var mapView = new MapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.73]  
});
```



Mobile Optimization

Christine

```
= new RouteParameters({  
  featureSet:{  
    graphics.toArray()  
  },  
  true  
});
```

Deployment

Christine Wiltawsky

```
const routeParams = new RouteParam.  
  stops: new FeatureSet({  
    features: view.graphics.toArray()  
  }),  
  returnDirections: true  
});
```

```
var mapView =  
  container: "v_  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40..  
]);
```

Two Patterns

- ArcGIS Enterprise
 - Deploy custom widgets
- Developer edition
 - Host app on a web server

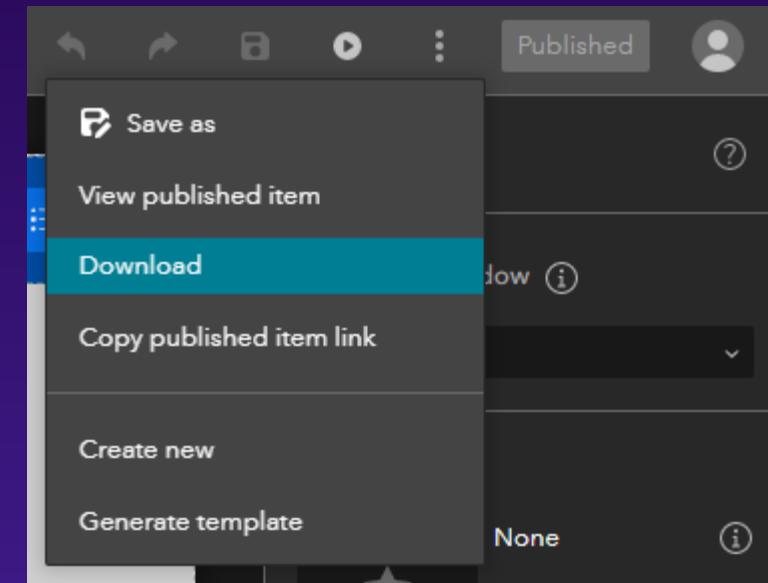
Deployment: ArcGIS Enterprise

- Deploy custom widgets to Experience Builder within ArcGIS Enterprise
- Users within ArcGIS Enterprise can build Experiences with those custom widgets
- ArcGIS Enterprise 11.0 or higher

Detailed instructions: esriurl.com/exb-widget-enterprise

Deployment: Developer Edition

- Build the Experience with your custom widgets
- Download as ZIP
- Unzip files
- Add clientId to cdn/1/config.json
- Host the files on a web server
- Command
 - node -e "require('./server/src/middlewares/dev/apps/app-download.js').zipApp('0', 'app.zip', 'clientID');"



Automated Deployment

Christine Wiltawsky

```
const routeParams = new RouteParam.  
stops: new FeatureSet({  
    features: view.graphics.toArray()  
}),  
returnDirections: true  
});
```

```
var mapView =  
    container: "v.  
    map: map,  
    zoom: 10,  
    center: [-73.95, 40..  
});
```

Why automation?

- Faster builds
- Consistent builds
- Easier for testers
- The "reference" application

How to automate

- Store the reference version of each app
- Auto-build using your CI/CD env of choice:
 - Azure DevOps
 - GitHub Actions
 - GitLab Actions
 - Jenkins
 - Etc.

Folder structure

- manifest.json
- apps/
 - 0/
 - 1/
 - 2/
 - ...
- widgets/
 - custom-widget-1/
 - custom-widget-2/
 - ...

Automatic build

- Create file:
 - `github/workflows/build-app.yml`
- Demo repo:
 - <https://github.com/EsriDevEvents/arcgis-experience-builder-customizing-and-extending-ds-europe-2023>

What's New / Road Ahead

Christine Wiltawsky

```
const routeParams = new RouteParam.  
stops: new FeatureSet({  
    features: view.graphics.toArray()  
}),  
returnDirections: true  
});
```

```
var mapView =  
    container: "v_...  
    map: map,  
    zoom: 10,  
    center: [-73.95, 40..  
]);
```

New in 2023

Enhancements

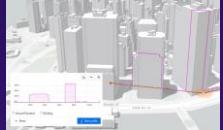
Slice



Editing Related Records



Elevation Profile



Web Scenes sync up



Mobile Optimization

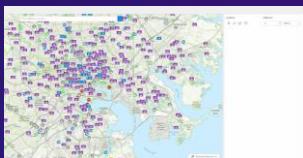


New widgets

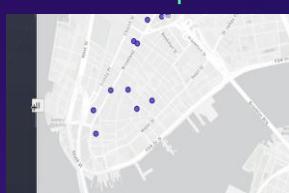
Add Data



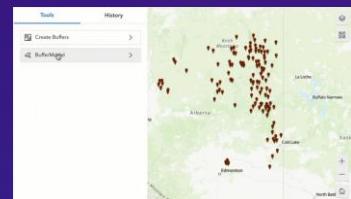
Near Me



Swipe



Analysis

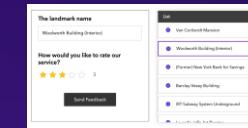


Basemap Gallery

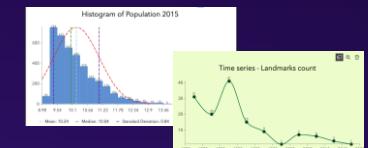


Enhancements

Business Analyst



Charts



Elevation Profile



Survey



Widget Controller



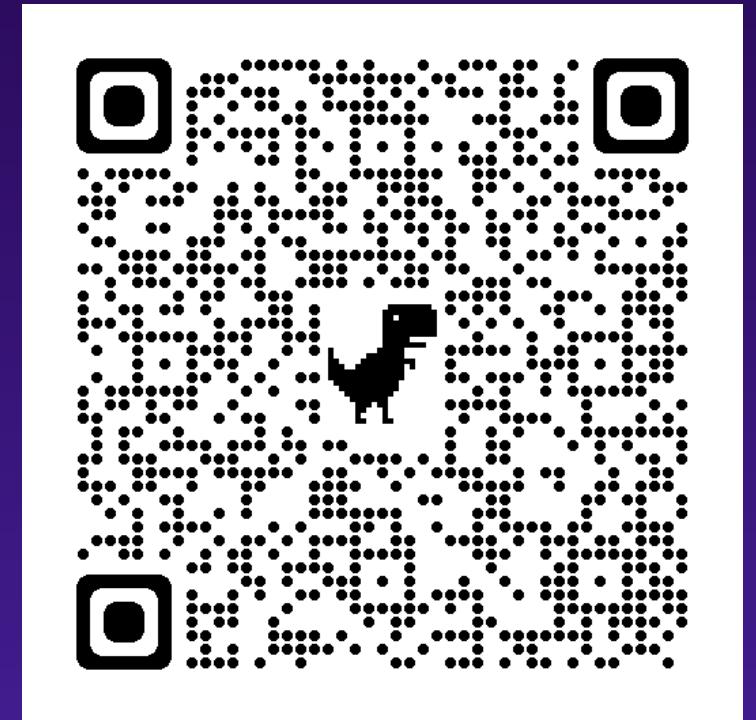
Charts

Roadmap

- Q1 2024
 - Select Widget
 - Group Filter
 - App URL
- Q2 2024
 - Image Measurement Widget
 - Reporting
 - Batch Attribute Editing
 - Parameter

Helpful Resources

The screenshot shows a web page with a blue header bar. The main content area has a white background. At the top left, there's a breadcrumb navigation: Home > All Communities > Products > ArcGIS Experience Builder > ArcGIS Experience Builder Documents. Below this, a section titled "Helpful resources to get you started" is displayed. This section includes a summary, a "Help Documentation Site" section with a list of links, and a "Blog Articles" section at the bottom. On the right side of the page, there are sections for "Version history" and "Contributors". A "Subscribe" button is located in the top right corner of the main content area.

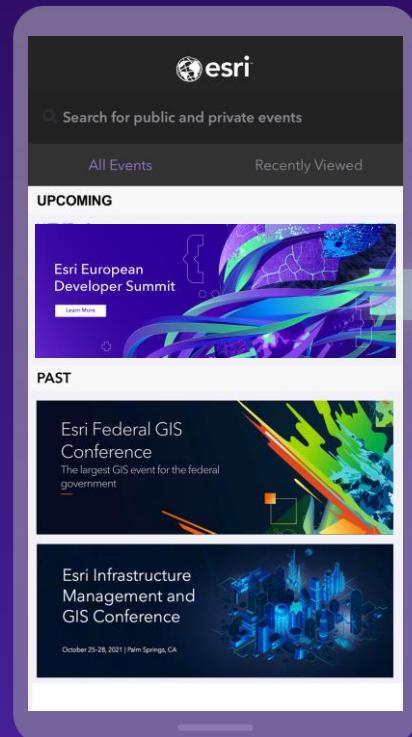


Helpful Resources

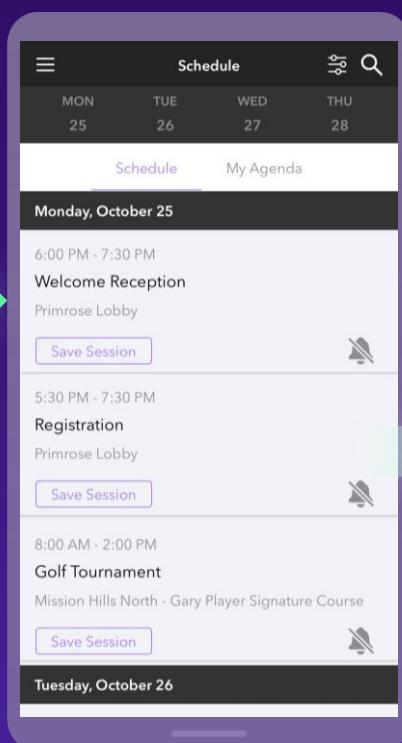
- [Entwicklerdokumentation inkl. API-Referenz und Beispielen](#)
- [Esri Community](#)
- [Samples on GitHub](#)
- [Storybook / jimu-ui library](#)
- [Gallery](#)

Please Share Your Feedback in the App

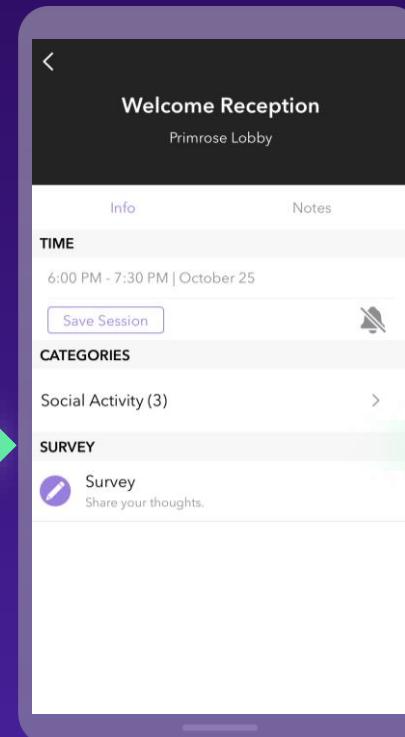
Download the Esri Events app and find your event



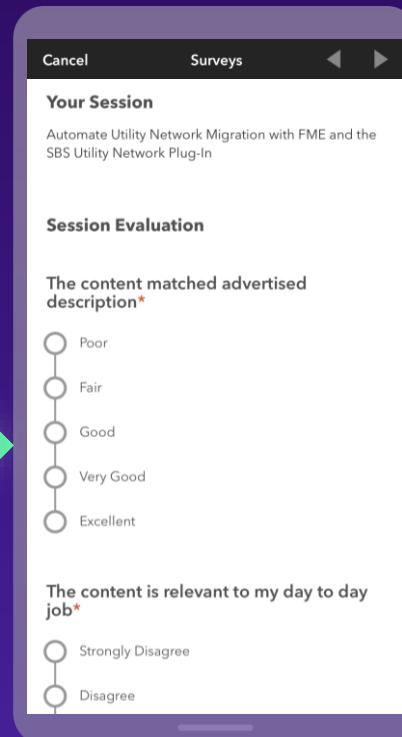
Select the session you attended



Scroll down to "Survey"



Log in to access the survey





```
const routeParams = new RouteParam.  
stops: new FeatureSet({  
    features: view.graphics.toArray()  
}),  
returnDirections: true  
});
```

Connect With Us On Social

And Join the Conversation Using #EsriDevSummit



twitter.com/EsriDevs



twitter.com/EsriDevEvents



youtube.com/@EsriDevs



links.esri.com/DevVideos



github.com/Esri



github.com/EsriDevEvents



links.esri.com/EsriDevCommunity

```
var mapView =  
    container: "v..  
    map: map,  
    zoom: 10,  
    center: [-73.95, 40..  
});
```



THE
SCIENCE
OF
WHERE®

Copyright © 2023 Esri. All rights reserved.

```
const routeParams = new RouteParams({  
    stops: new FeatureSet({  
        features: view.graphics  
    }),  
    returnDirections: true  
});
```

```
var mapView = newMapView({  
    container: "view",  
    map: map,  
    zoom: 10,  
    center: [
```

LIVE
BY
THE
CODE

Test



```
= new Map({  
    viewer: "viewD",  
    map,  
    zoom: 10,  
    center: [-73.95, 40.75]  
});
```

ESRI EUROPEAN DEVELOPER SUMMIT 2023

```
var mapView = new MapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40.73]  
});
```



```
const routeParameters = new RouteParameters({  
  stops: featureSet({  
    type: "Feature",  
    geometry: "Point",  
    properties: {  
      address: "123 Main St, New York, NY 10001",  
      lat: 40.7128,  
      lon: -74.0139  
    }  
  }),  
  returnStops: true  
});
```



```
const routeParams = new RouteParam.  
  stops: new FeatureSet({  
    features: view.graphics.toArray()  
  }),  
  returnDirections: true  
});
```

```
var mapView =  
  container: "v_  
  map: map,  
  zoom: 10,  
  center: [-73.95, 40..  
]);
```



THE
SCIENCE
OF
WHERE®

Copyright © 2023 Esri. All rights reserved.

```
var mapView = new MapView({  
  container: "viewContainer",  
  map: map,  
  zoom: 10,  
  center: [
```

```
const routeParams = new Route.  
stops: new FeatureSet({  
  features: view.graphics  
}),  
returnDirections: true  
});
```