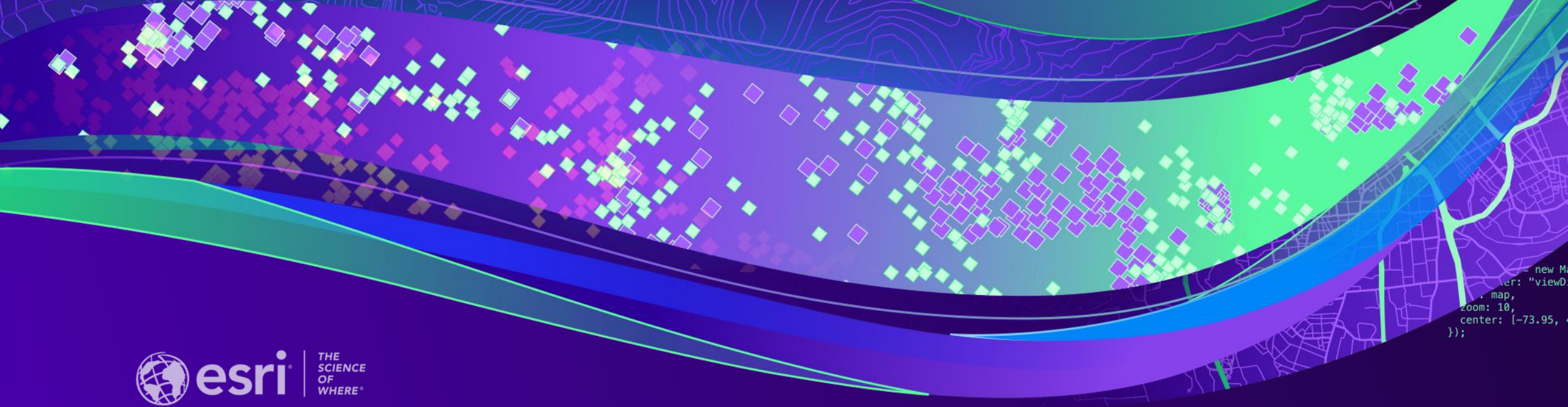


ESRI EUROPEAN DEVELOPER SUMMIT 2023

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

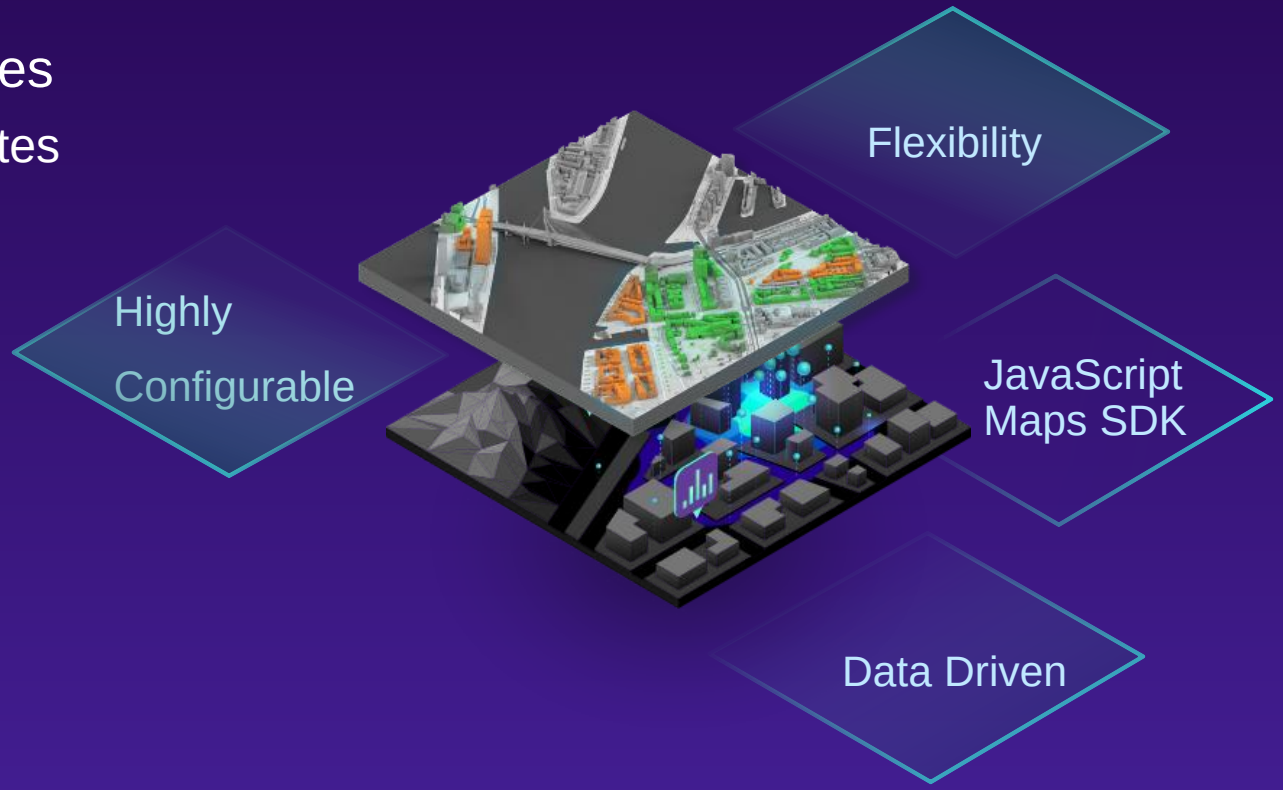
Introduction to ArcGIS Experience Builder

Christine Wiltawsky

What Is ArcGIS Experience Builder?

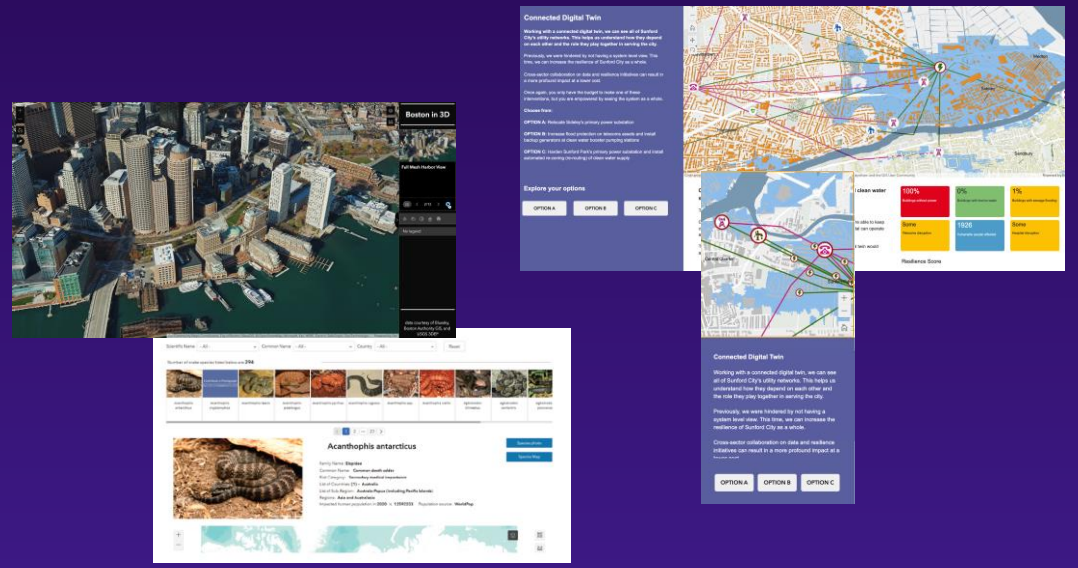
Create web apps you envision

- 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

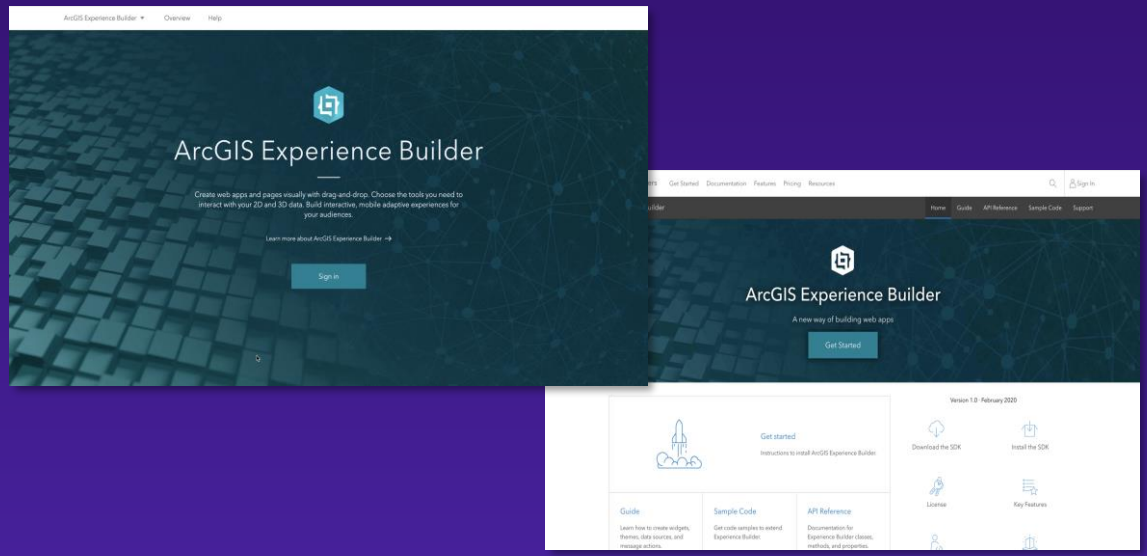


Available in ArcGIS Online, ArcGIS Enterprise, and developer edition

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

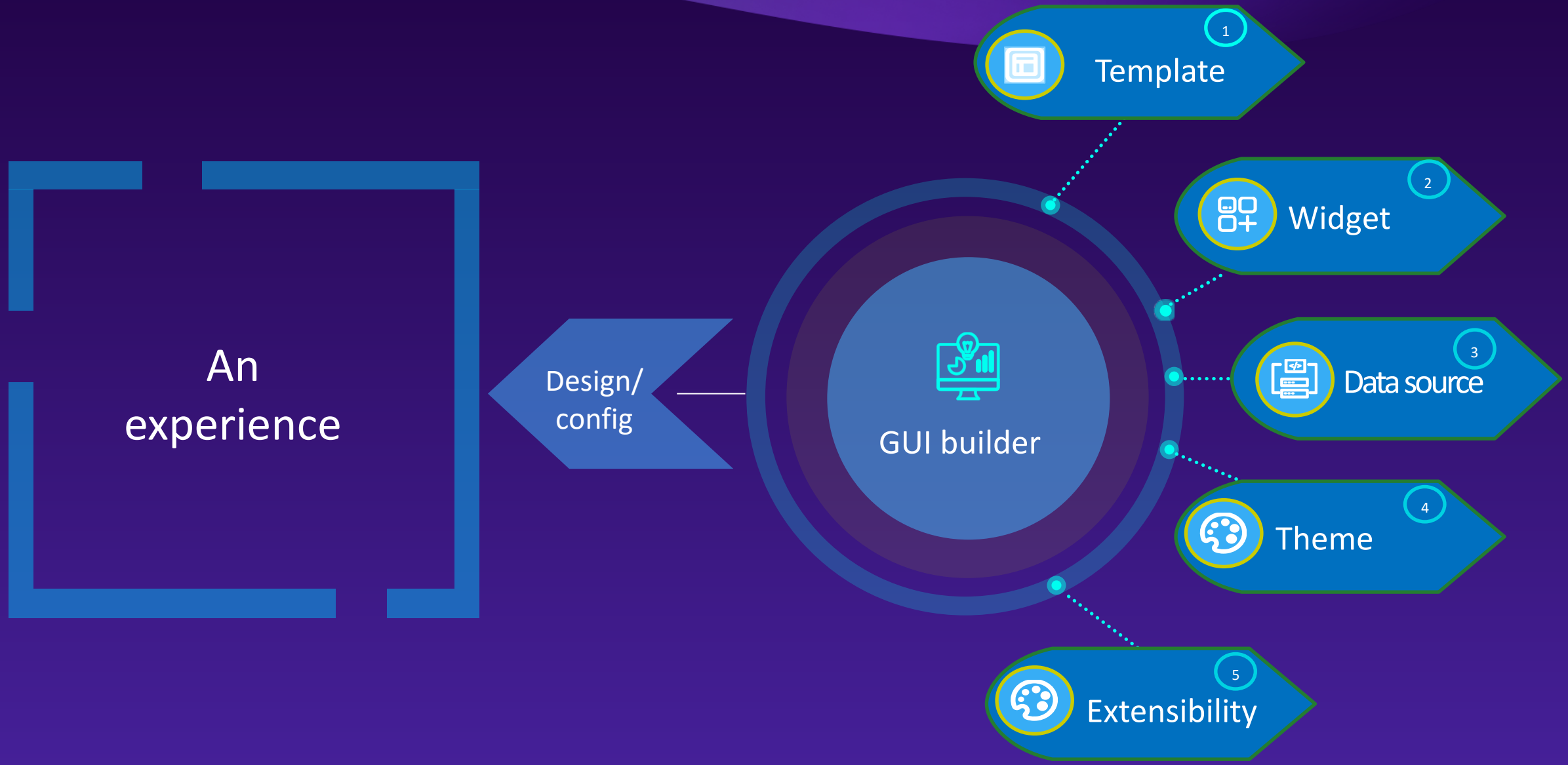
- Rolling release with ArcGIS Online updates
- Fixed release packaged with each version of ArcGIS Enterprise
- Fixed releases individually available with ArcGIS Experience Builder Developer Edition

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/

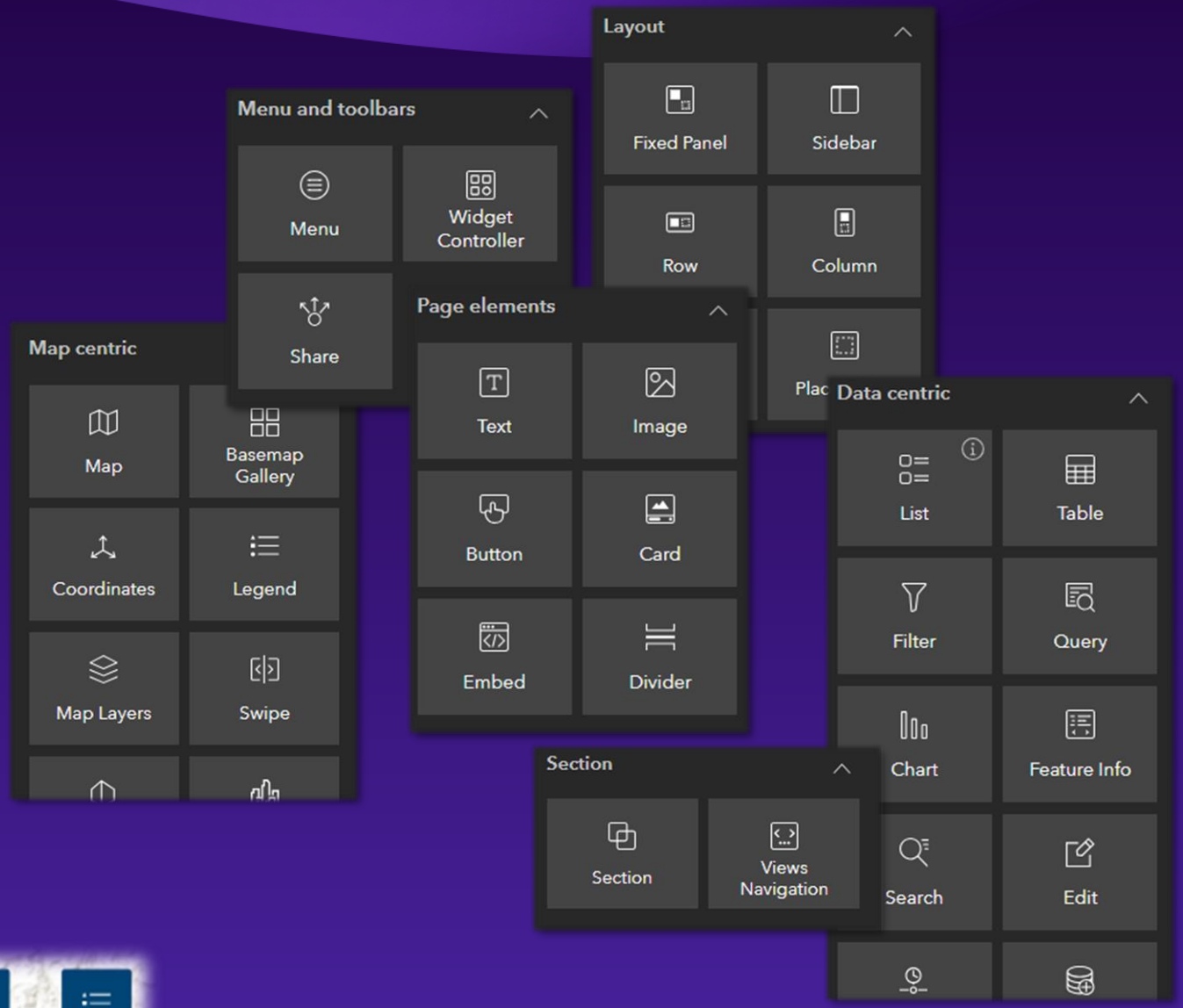
ArcGIS Experience Builder Licensing

- 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
1
Template
2
Widget
3
Data source
4
Theme
5
Extensibility

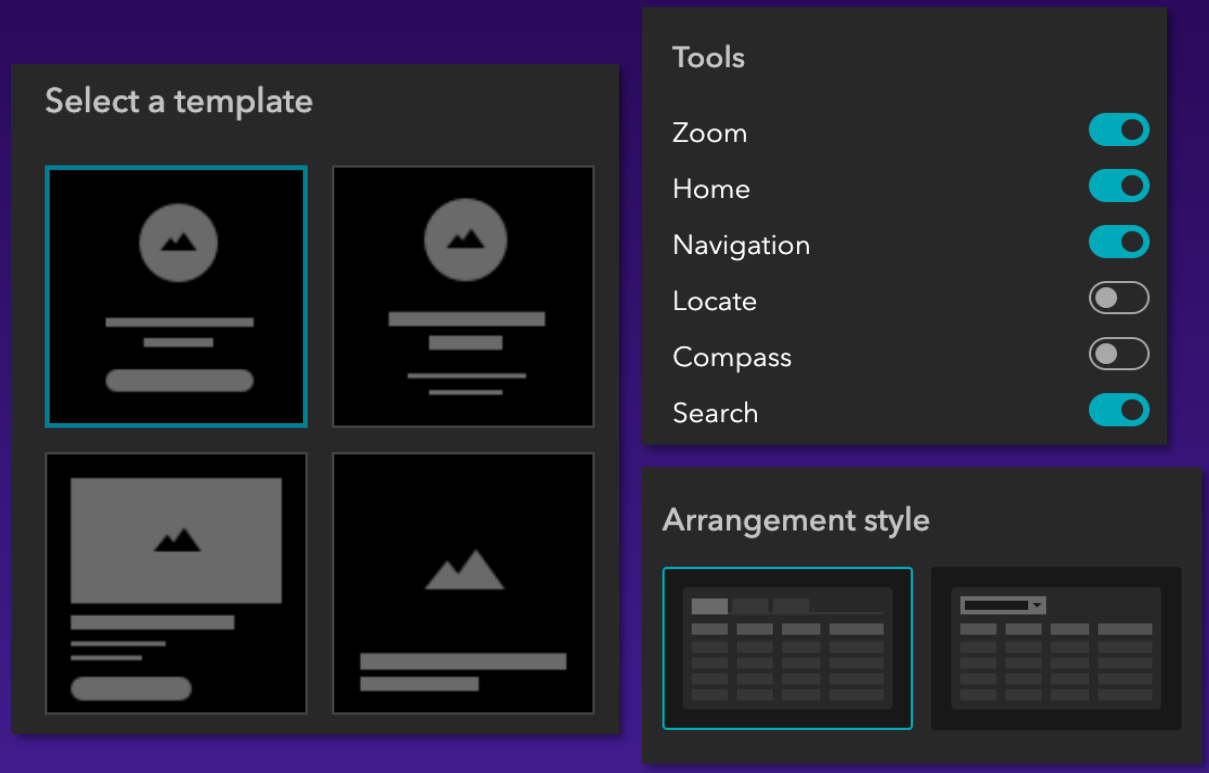
Types of Widgets

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



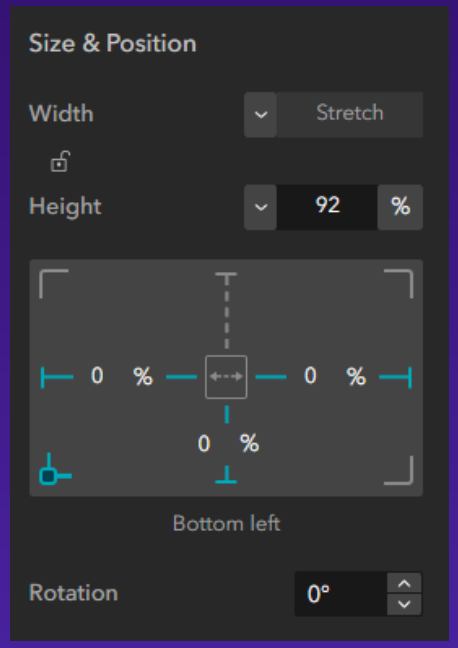
Widget Content

- Data sources
- Options
- Tools
- Modes
- Templates

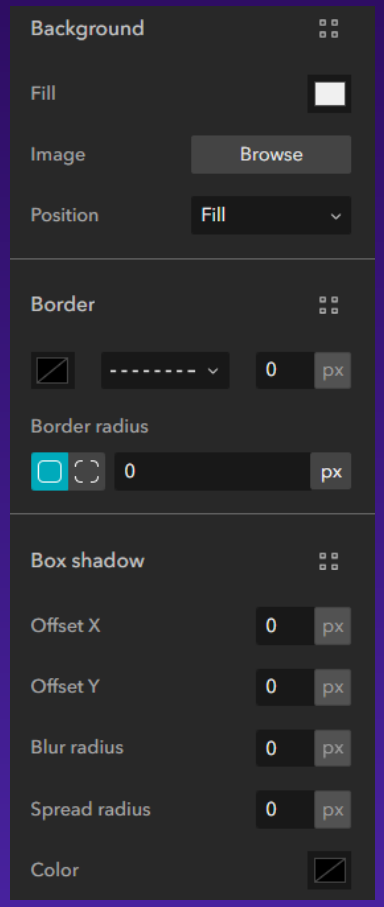


Widget Style

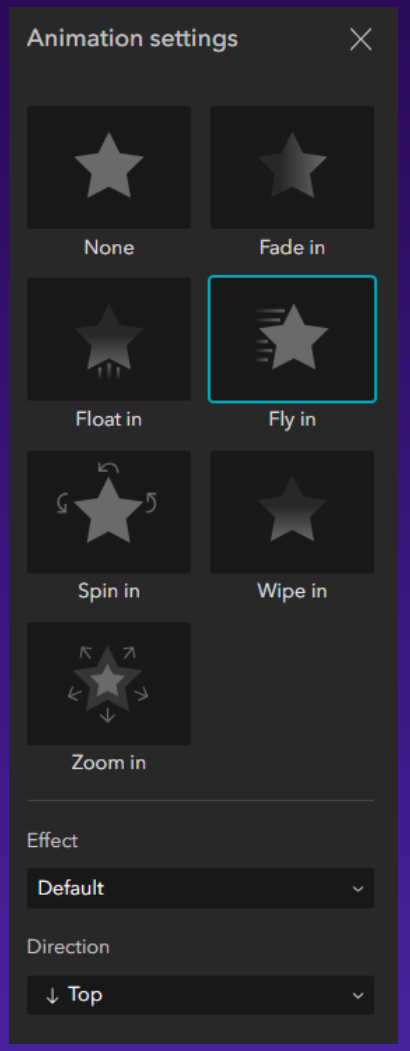
Size & Position

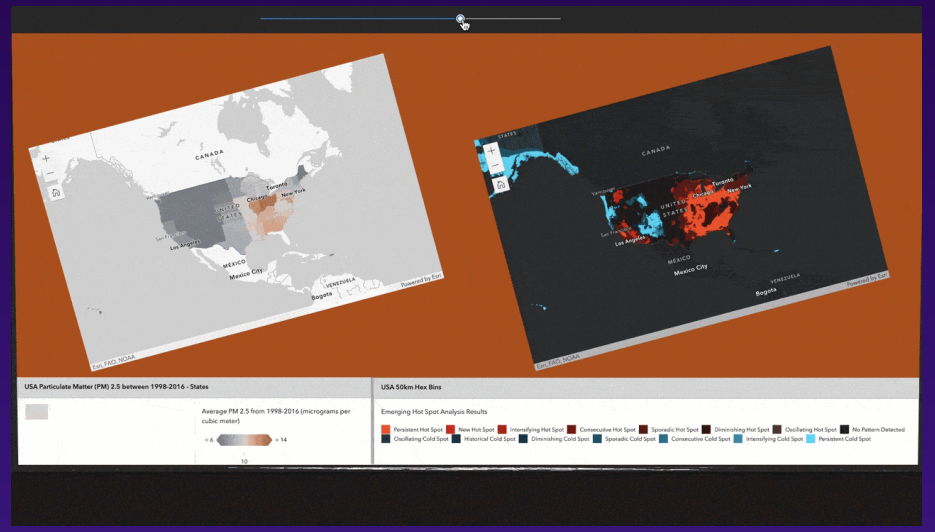


Background, Border & Box shadow



Animation



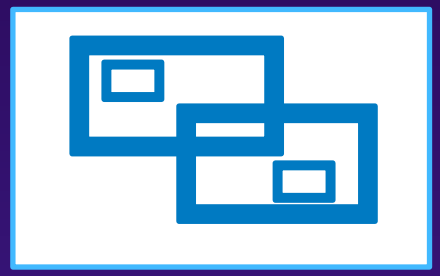


Layout Widgets

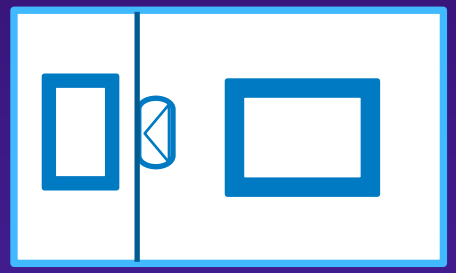
Row

Column

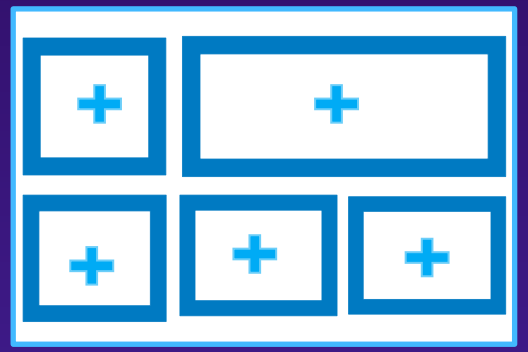
Fixed Panel



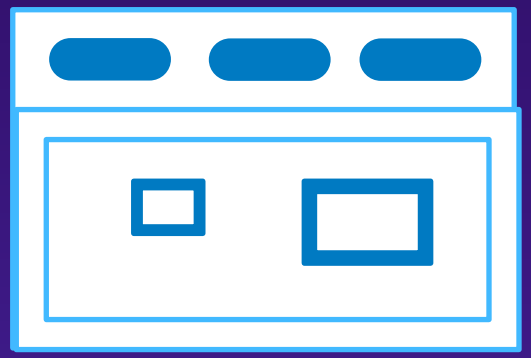
Sidebar



Grid

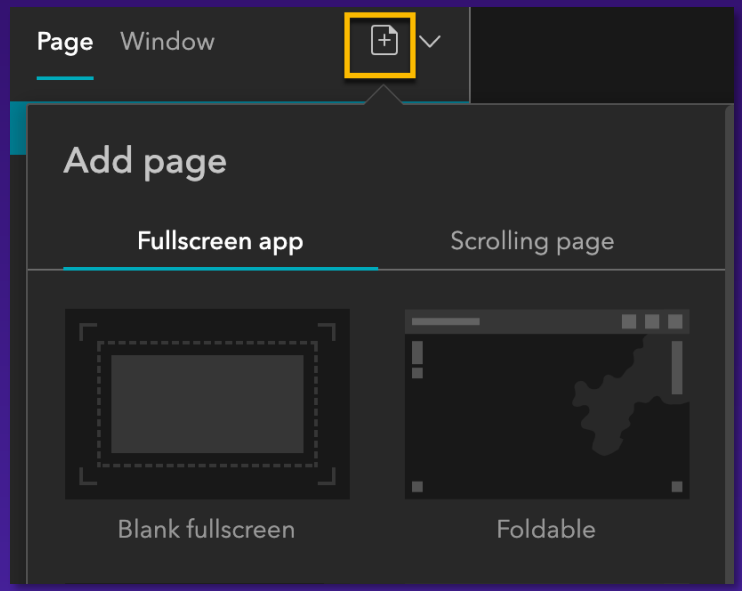


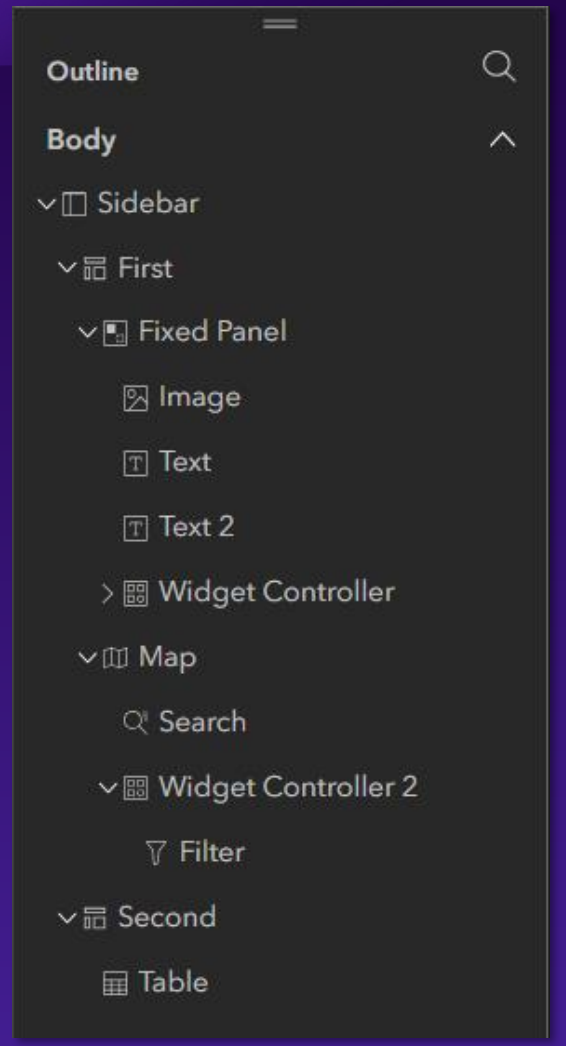
Section & View



Page

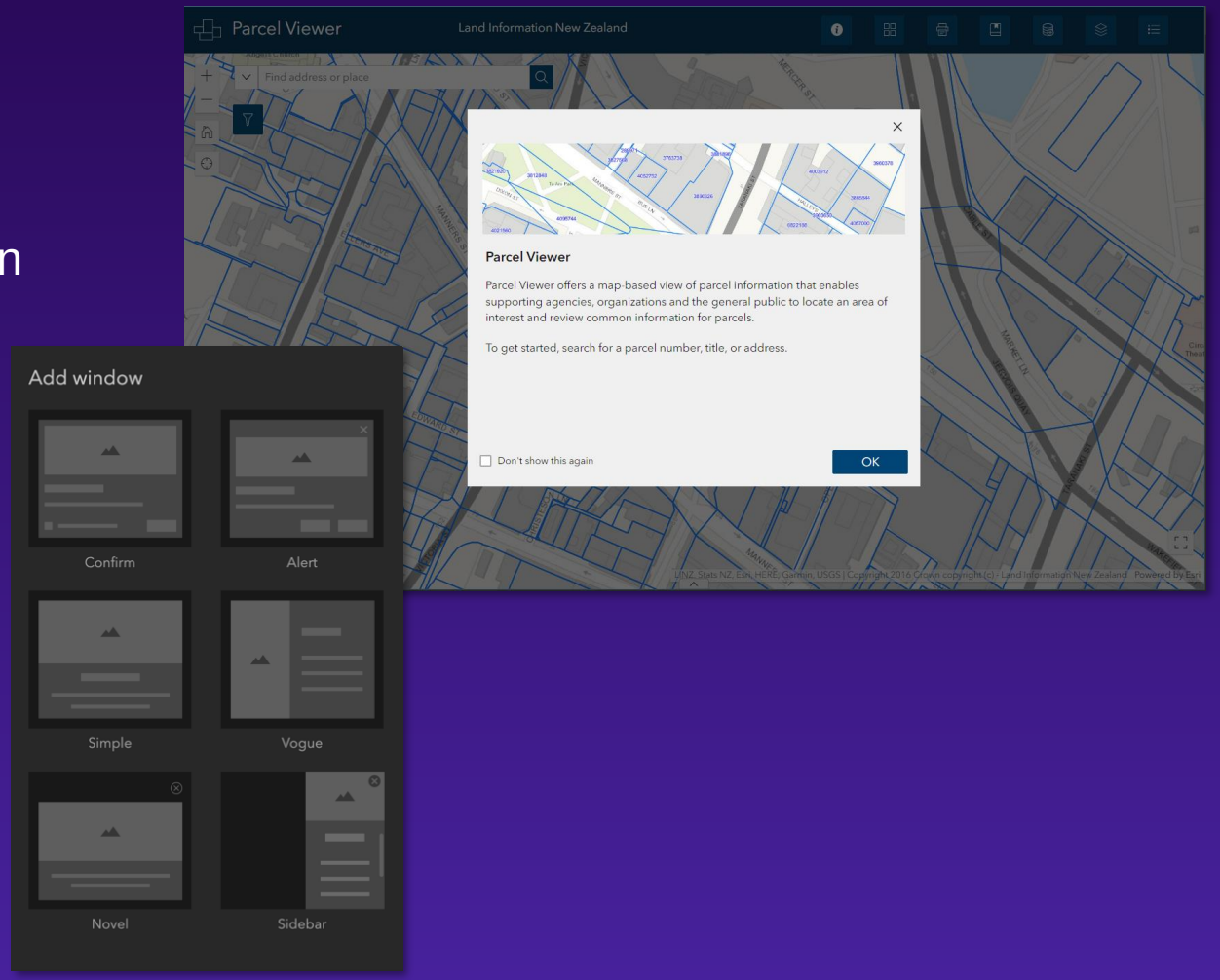
- Header, Footer, Open with window
- Outline helps navigate widgets on the page
- Page templates





Window

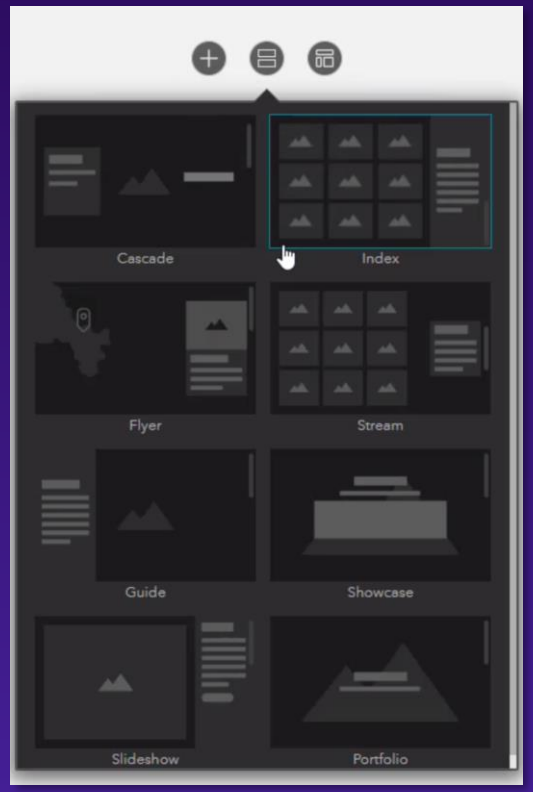
- Display Splash screens or information
- Fixed mode
  - shown in a fixed position like a splash screen
- Anchored mode
  - shown around the widget that triggers it
- Window templates



Screen Group

A special layout with multiple scrolling screens

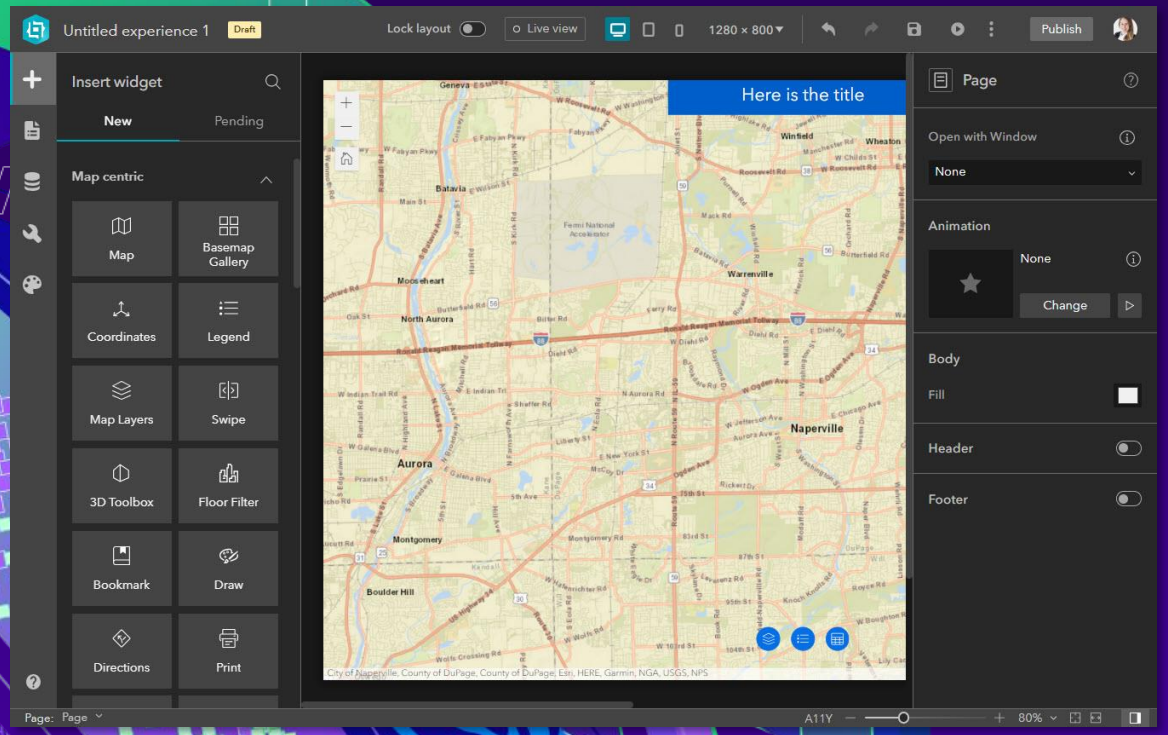
Engaging Experience



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

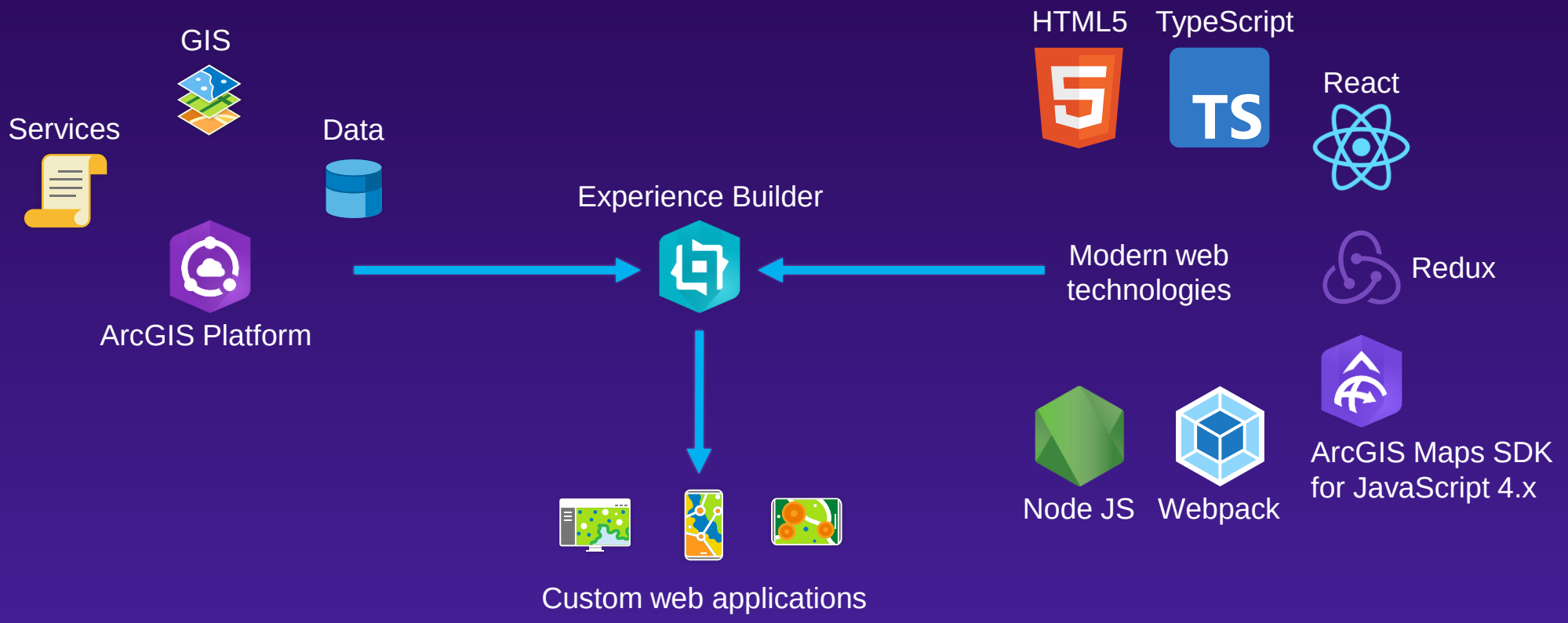
Christine Wiltawsky

Introduction to Custom Widgets

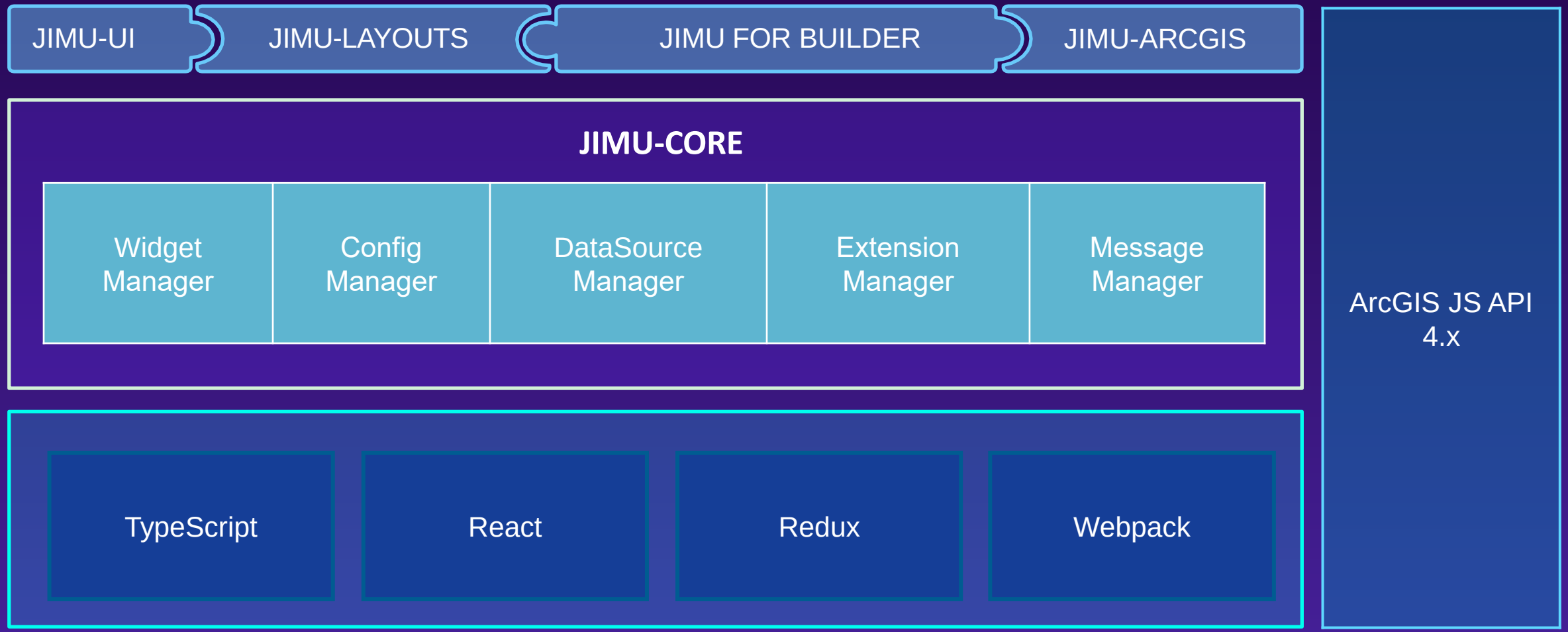
Christine Wiltawsky

Technology Stack

Introduction to ArcGIS Experience Builder

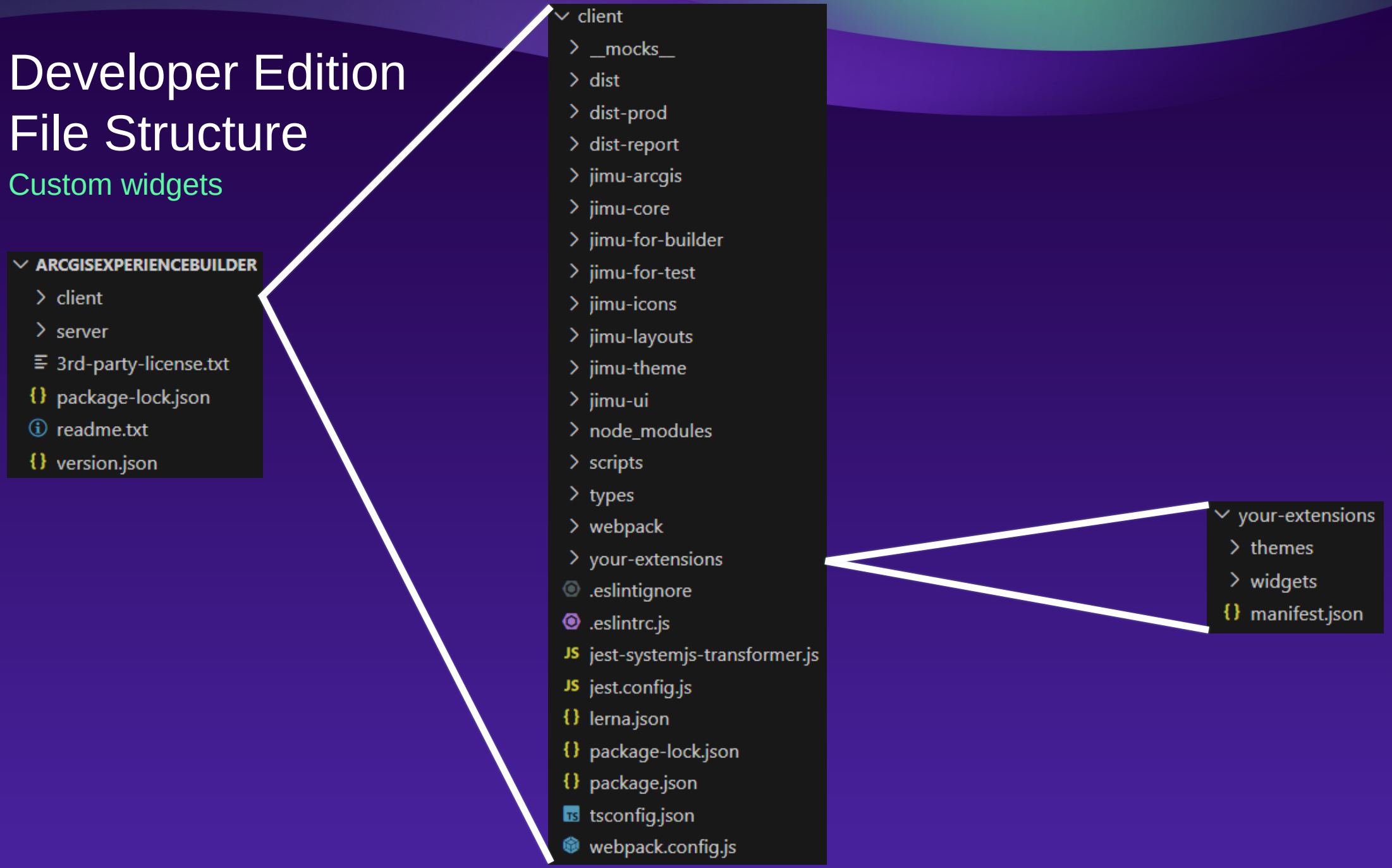


Technology Stack

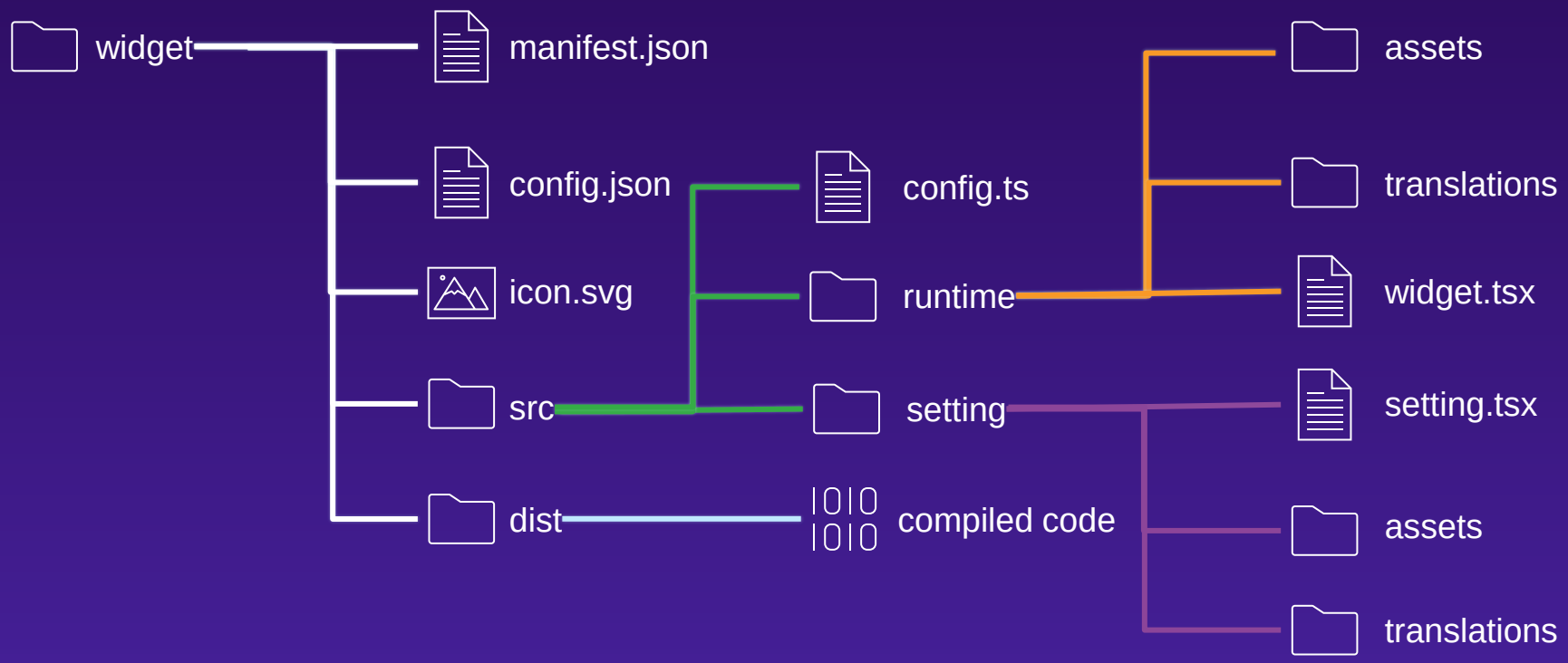


Developer Edition File Structure

Custom widgets



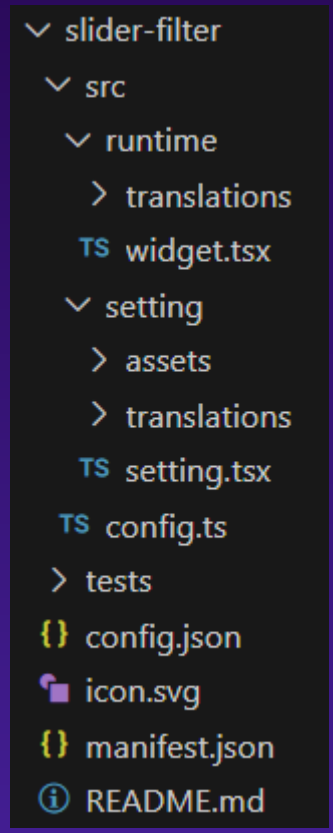
Widget Structure



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

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

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”
- …

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

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

```
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

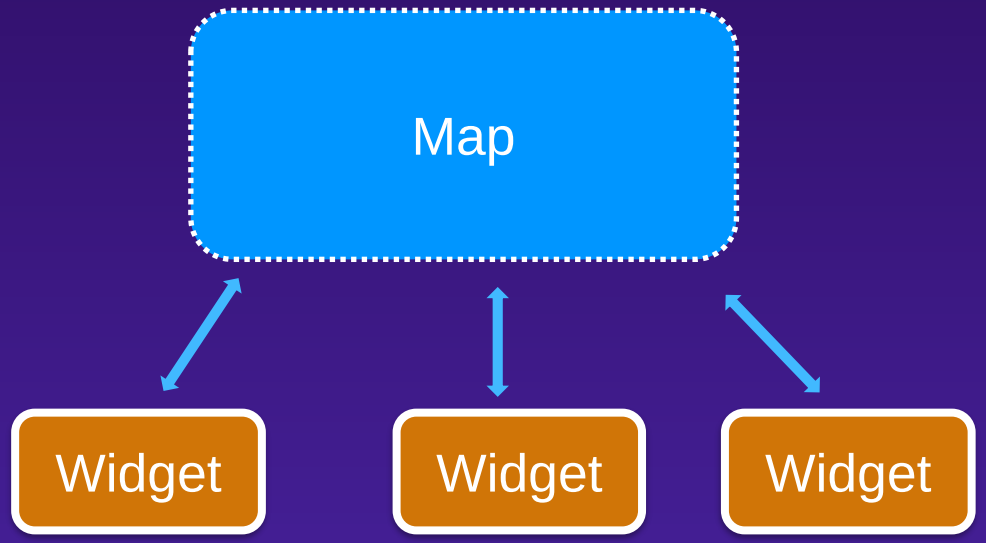
- Documentation: https://developers.arcgis.com/experience-builder/guide/extend-base-widget/#i18n-support

Connect Widget with a map

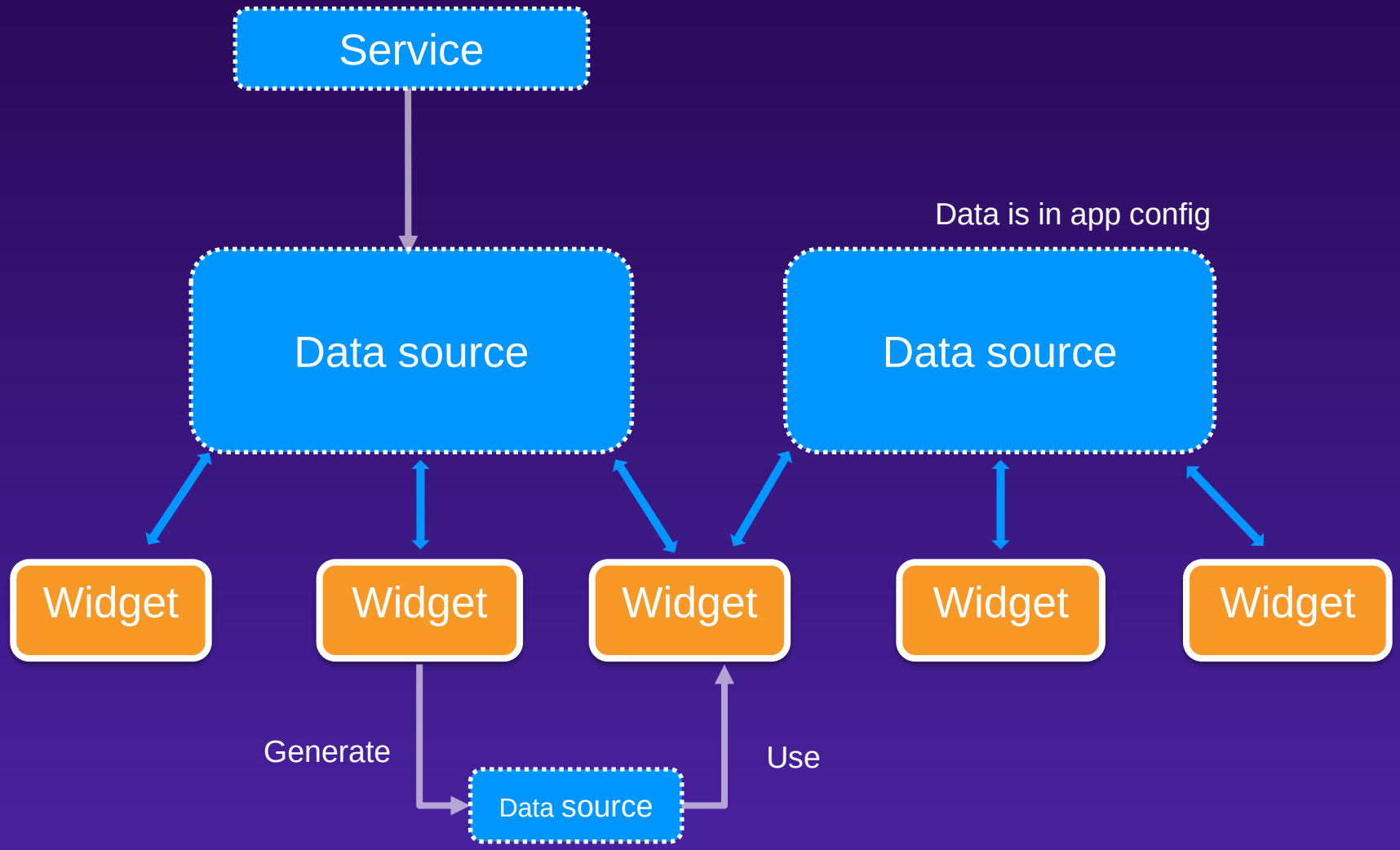
Christine Wiltawsky

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 as 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/lon and scale (like demo)
- Add “zoom level” to display
- Add an additional event listener so the lat/lon updates based on the mouse clicked

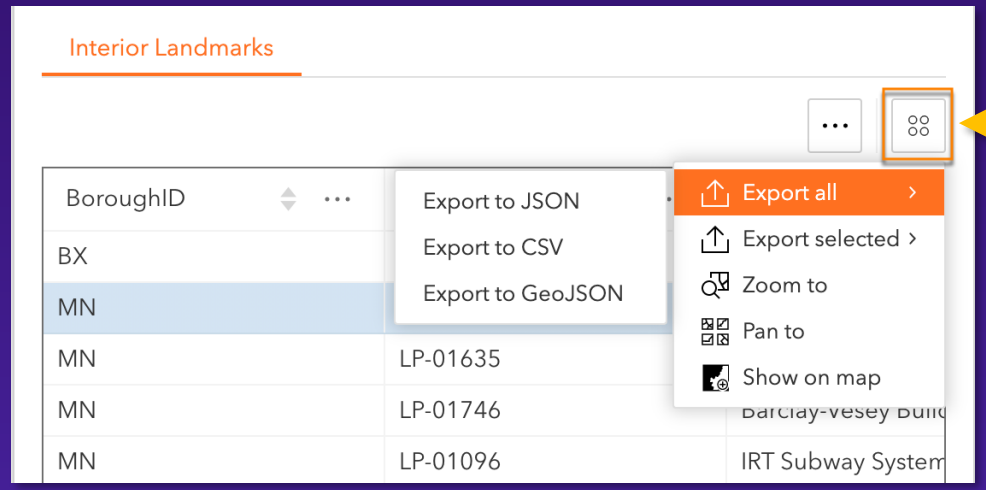
- Bonus: Make “zoom level” visibility configurable through the settings panel

Actions

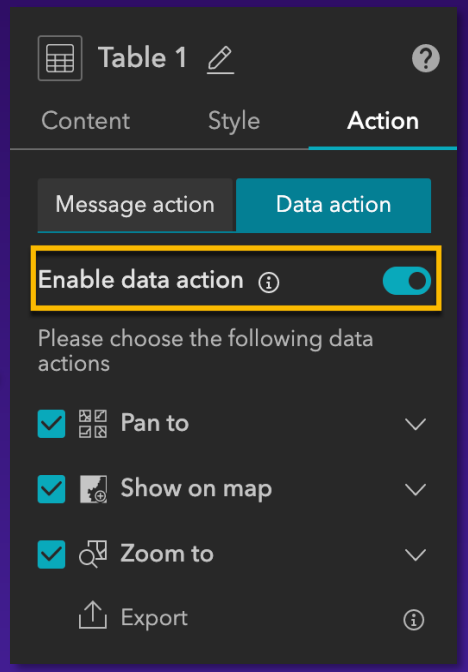
Christine Wiltawsky

Widget Action

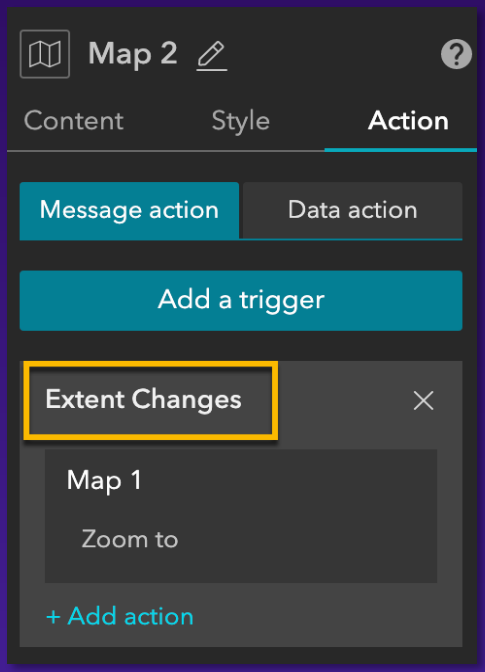
- Widgets communicate with each other
- Data actions are opt-in
- Message actions are automatic



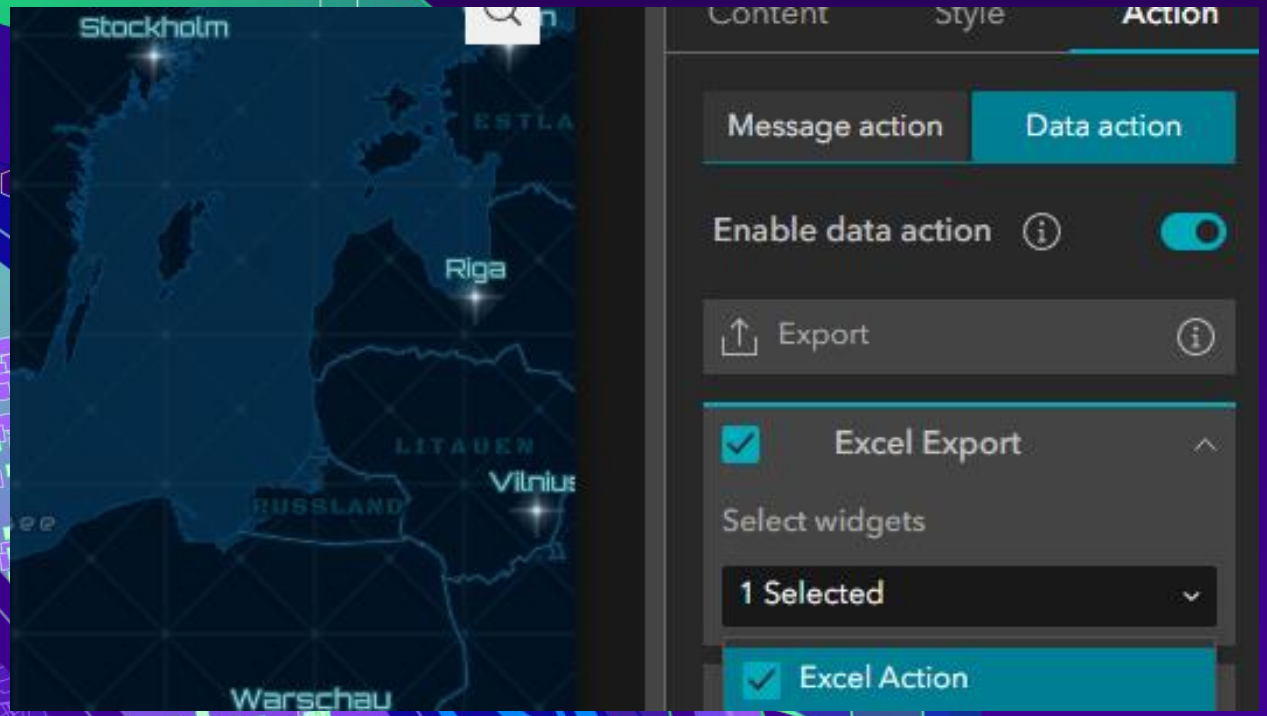
Data action



Message action

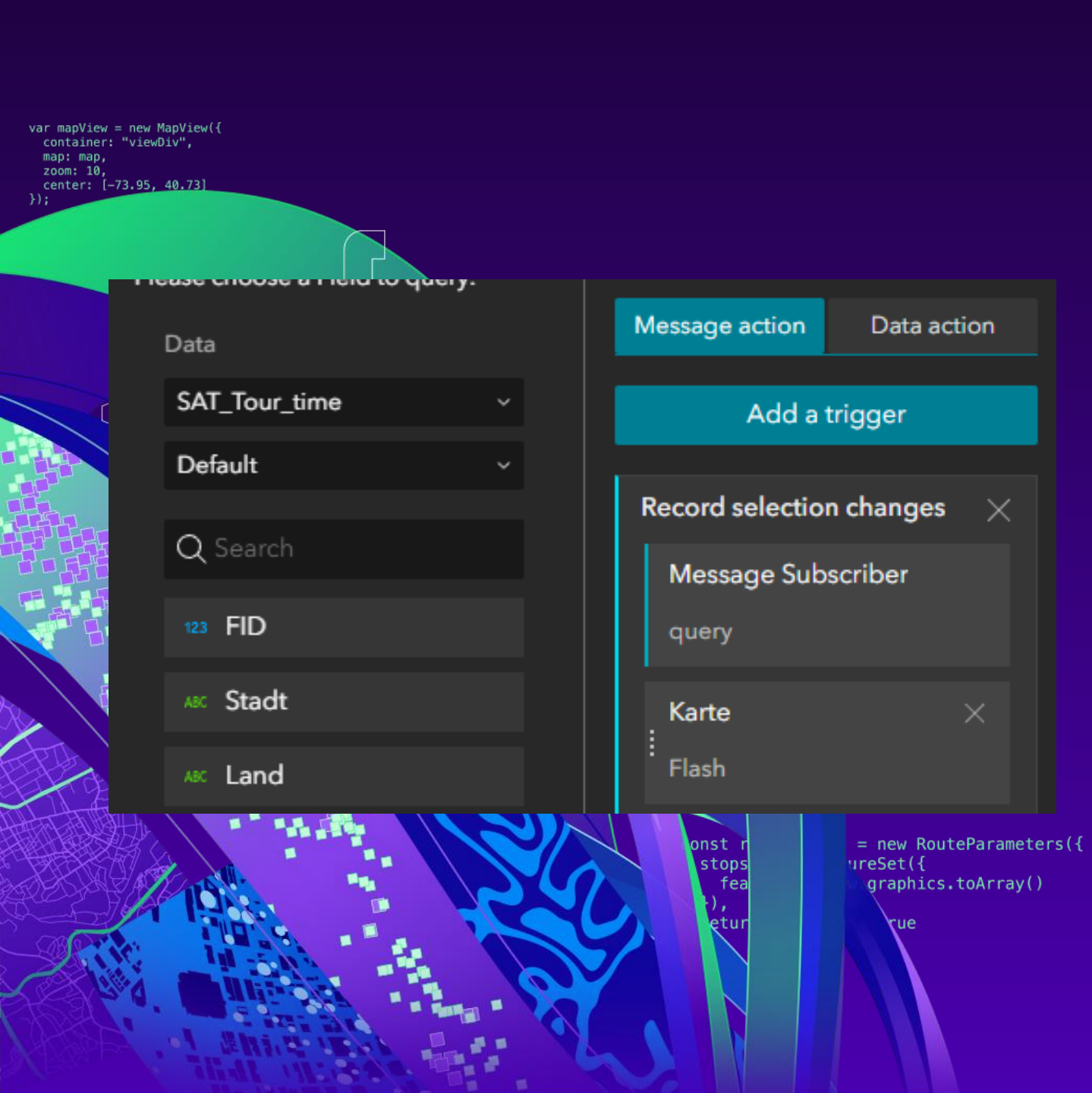


```
var mapView = new MapView({
  container: "viewDiv",
  map: map,
  zoom: 10,
  center: [-73.95, 40.73]
});
```



Data Action

Receive Data



Message Action

Perform process on trigger

Message Action

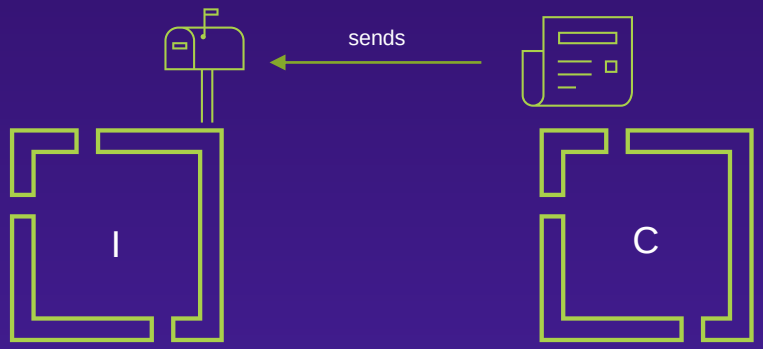
- A trigger broadcasts a message
- Targets listen to the message
- Actions provide Target behavior

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

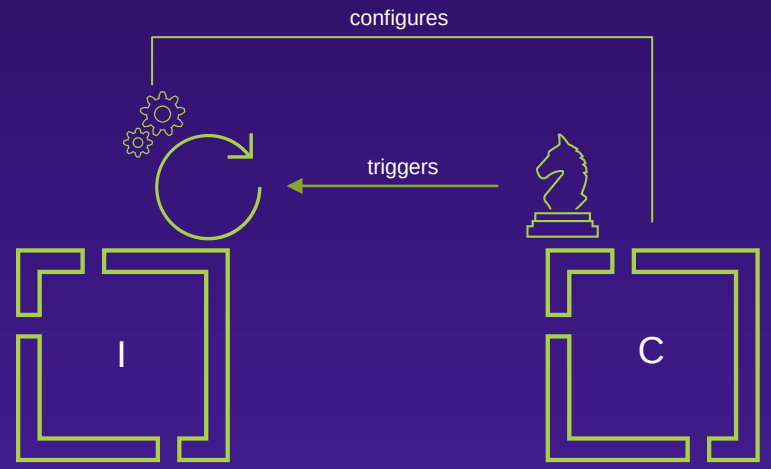
Actions

Send data and messages to other widgets

Data action



Message action



Samples Actions

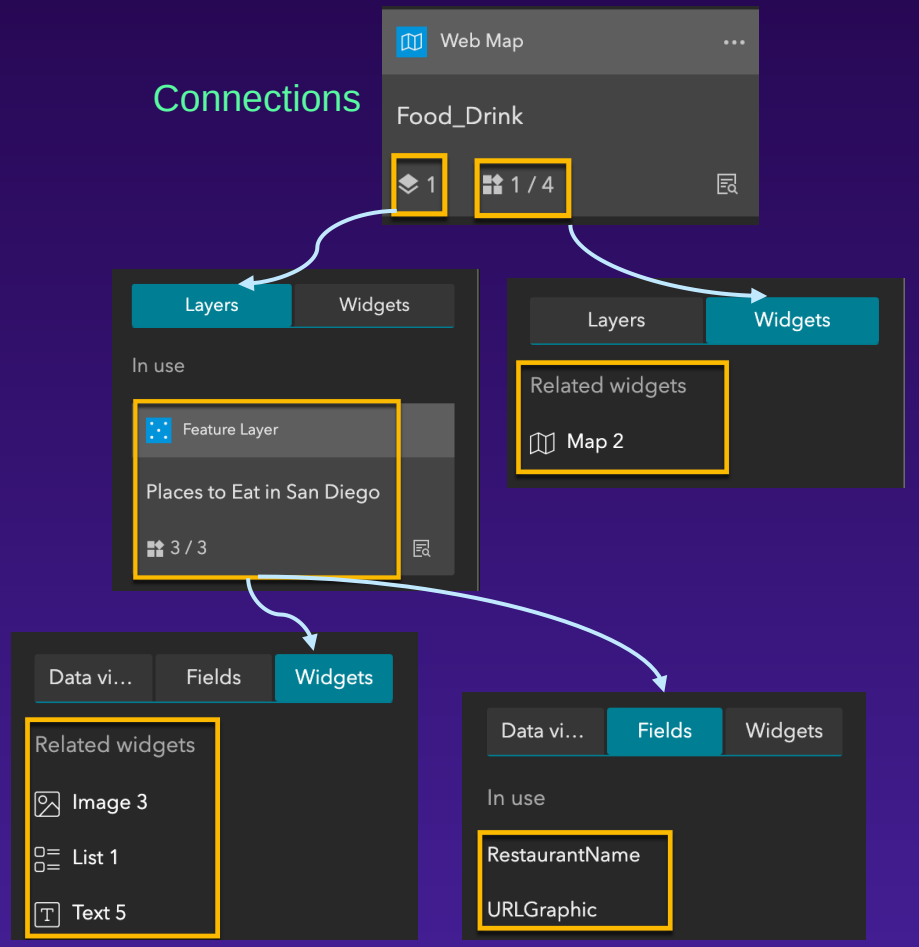
- Message Action Subscribe
  - https://github.com/Esri/arcgis-experience-builder-sdk-resources/tree/master/widgets/message-subscriber
- Message Action Publisher
  - Sample: Table Widget
  - https://github.com/esride-nik/ExB-workshop/tree/master/widgets/messagePublisher
- Data Action
  - Sample Table 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

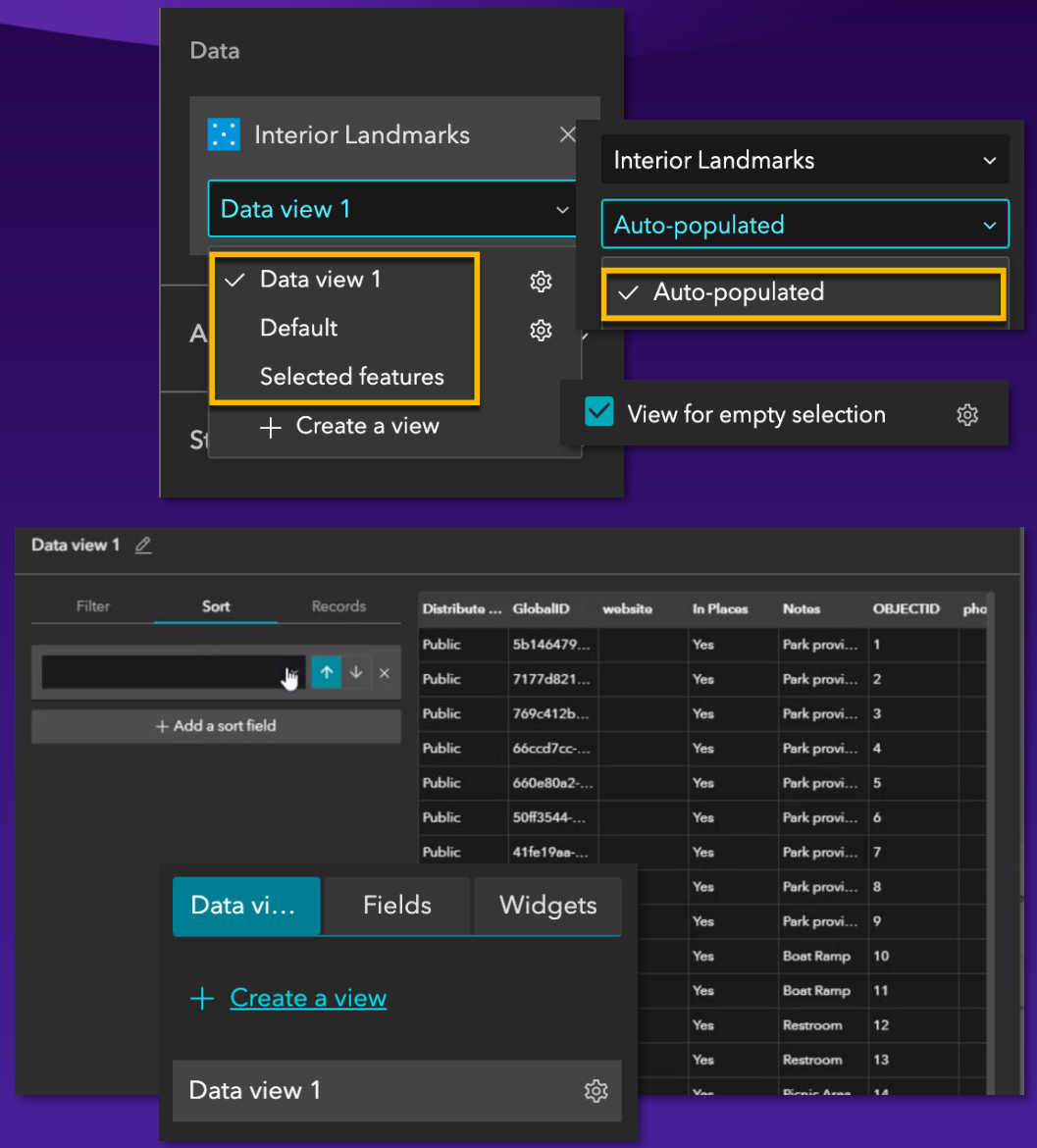
Data Source

- 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



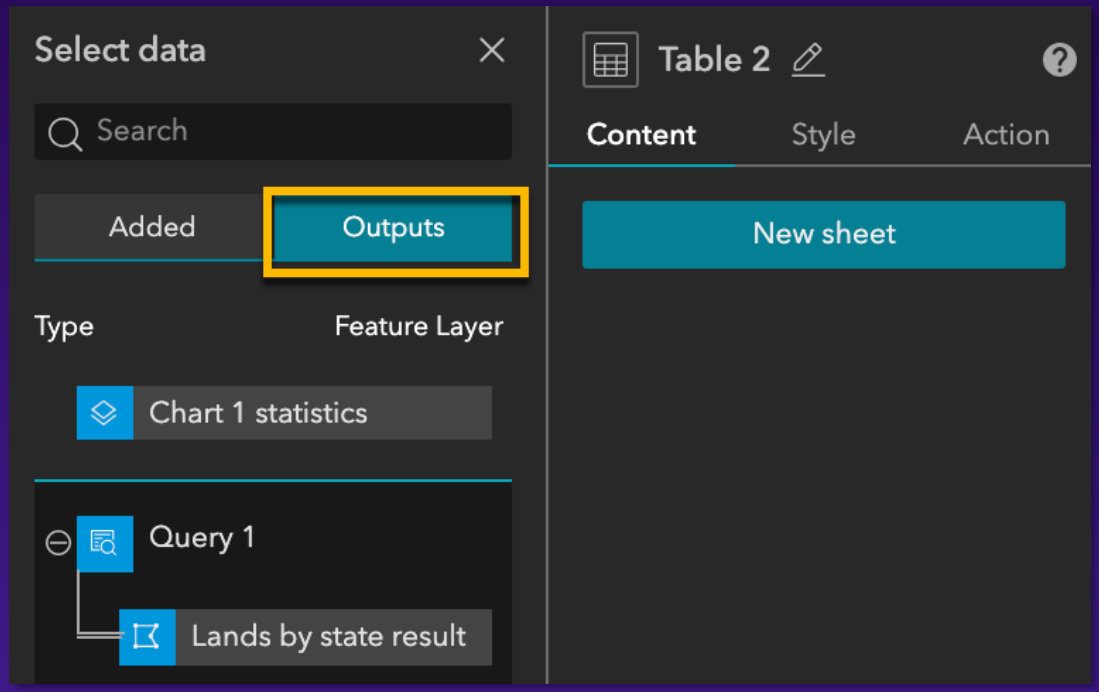
Data Source Views

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



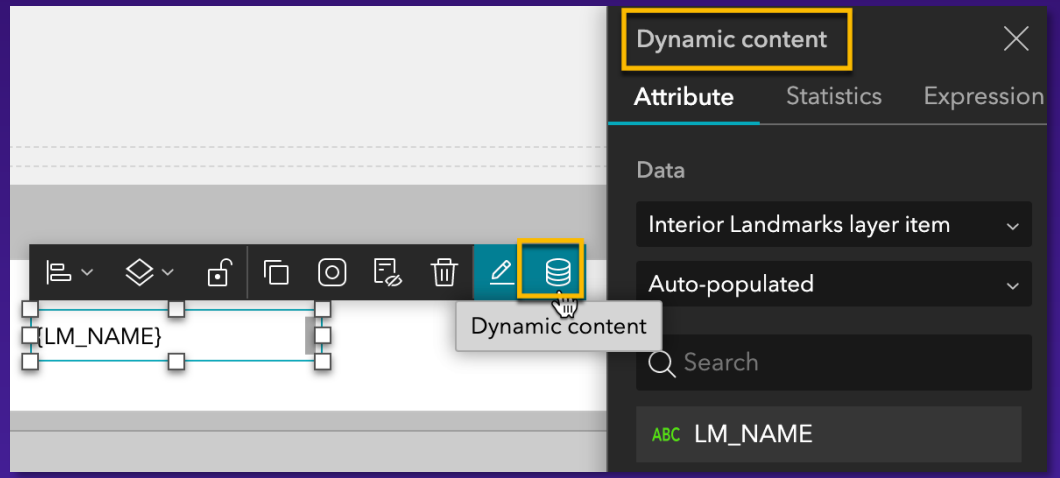
Output Data Source

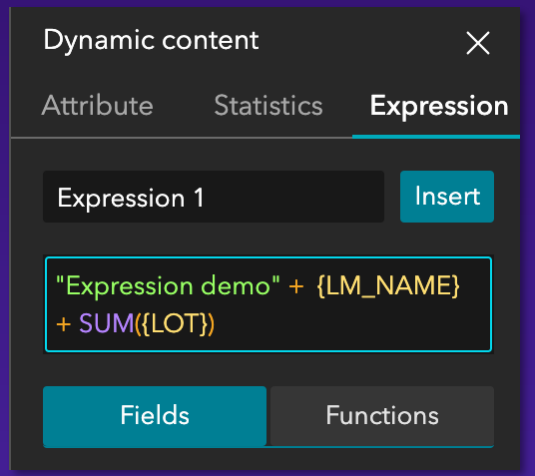
- Generated by widgets at runtime
  - Chart, Query, Directions, Elevation Profile
- Other widget's data source
  - such as Table and Text



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/

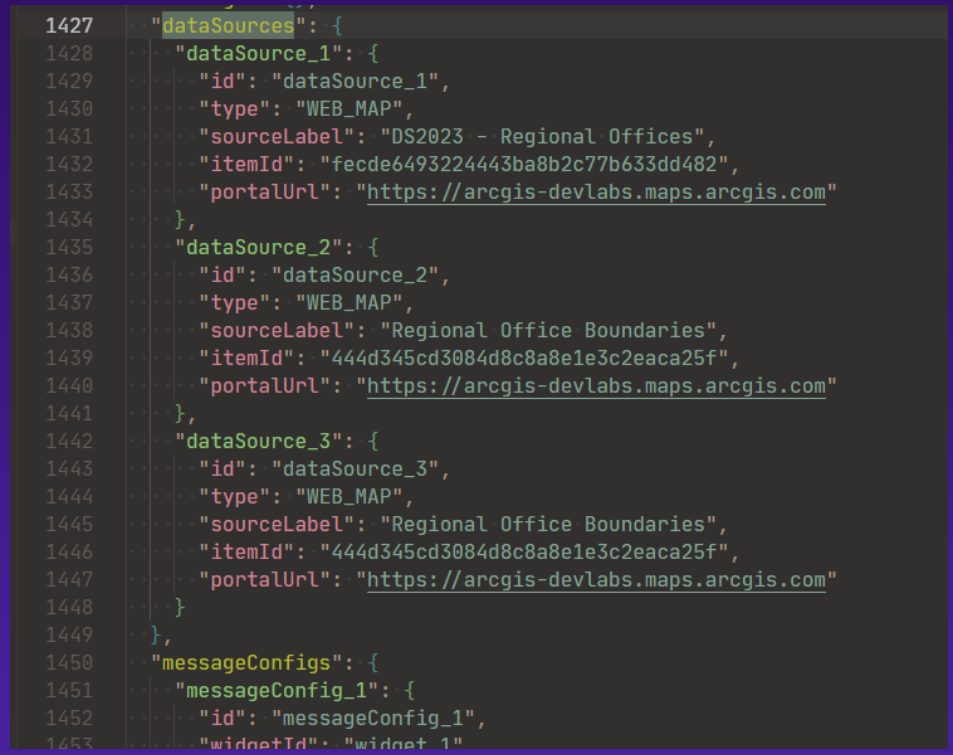
Common classes and interfaces

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()
- QueriableDataSource
  - updateQueryParams
  - load
  - query()
  - getRealQueryParams()
- FeatureLayerDataSource
- WebMapDataSource/WebSceneDataSource

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



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?

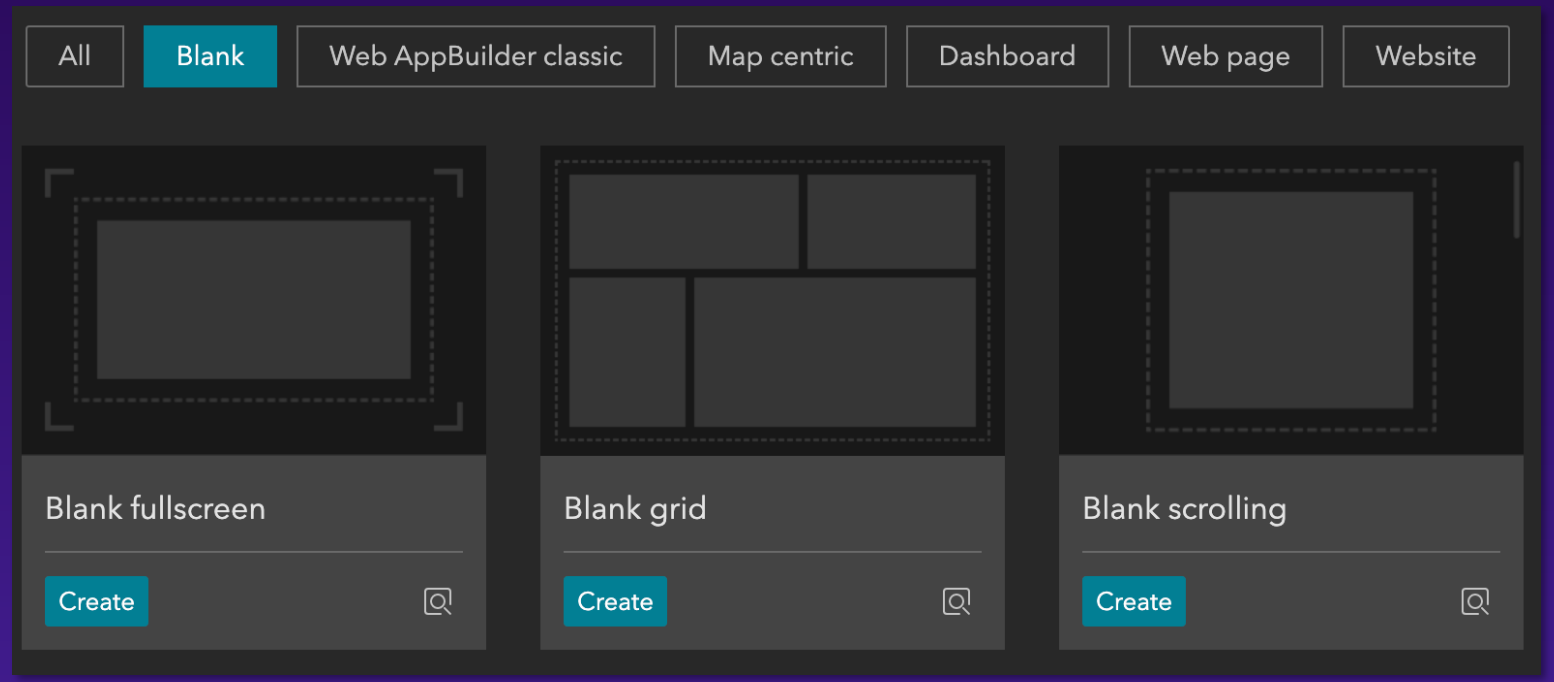
- ** Challenge: In the ‘listen-selection-change’ widget add a field selector to the settings to allow end users to modify which field is used to display on the widget.

Templates


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..
});

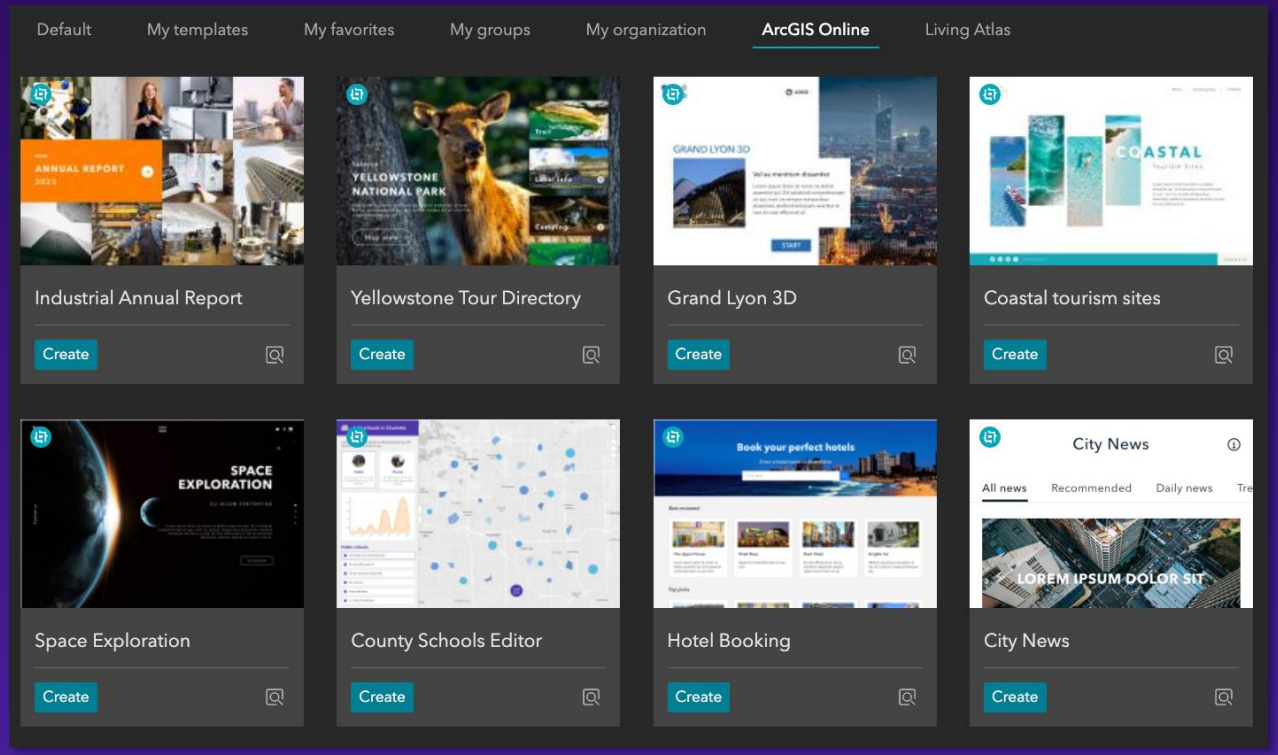
Default Templates

- Fullscreen
- Scrolling
- Grid

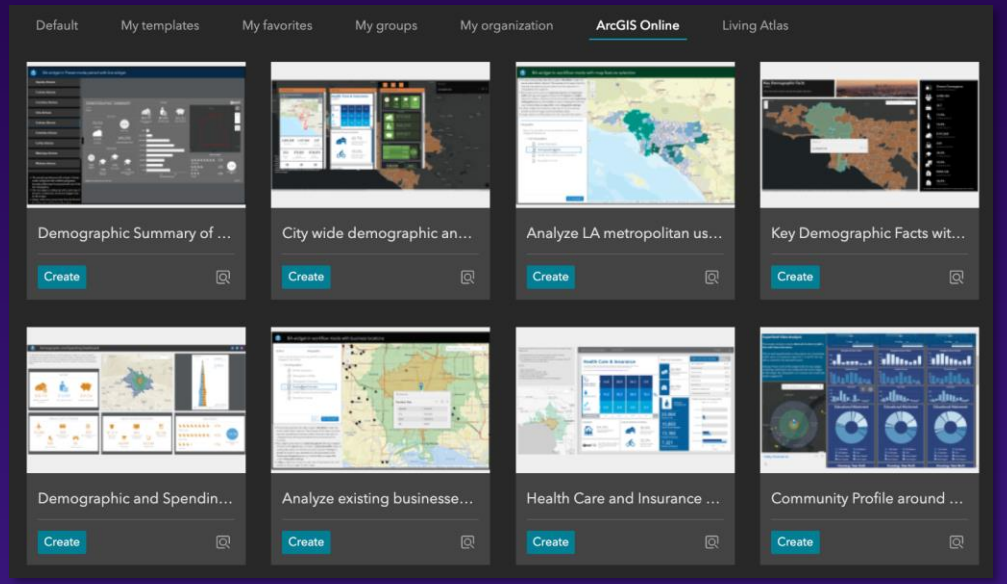


Public Templates

Available in ArcGIS Online



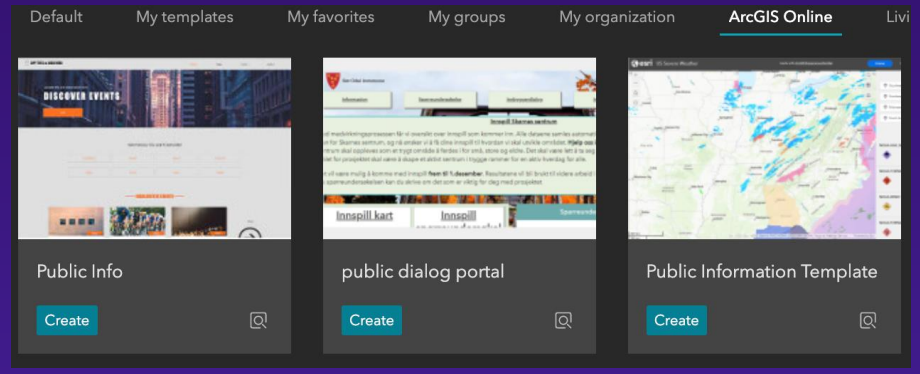
Showcase New Features

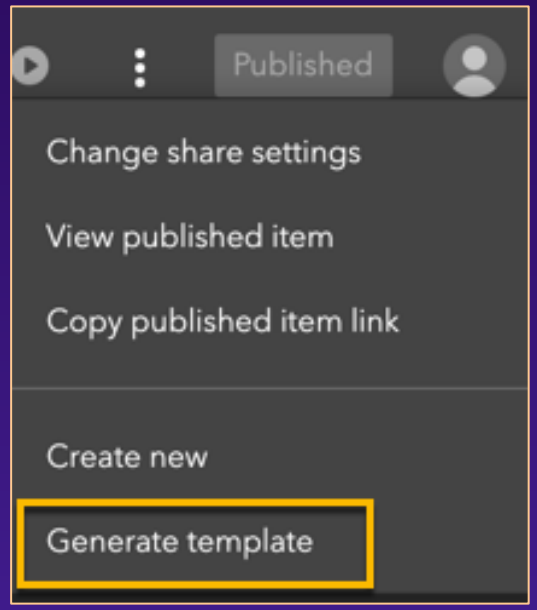


Business Analyst Templates

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

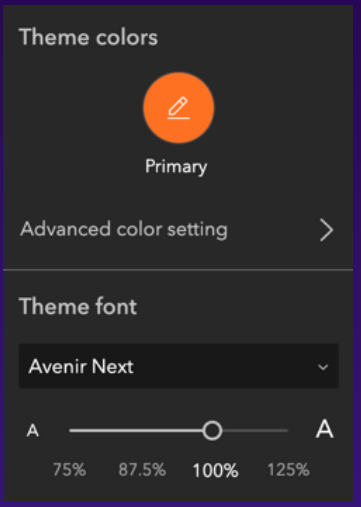
Dark

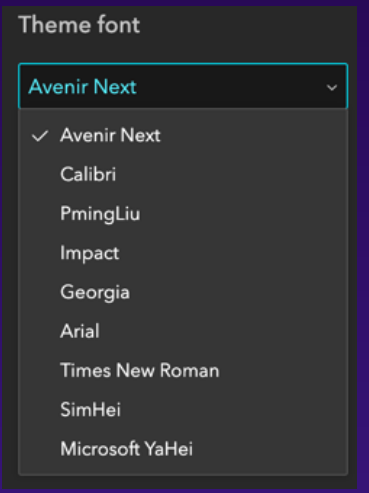
Vivid

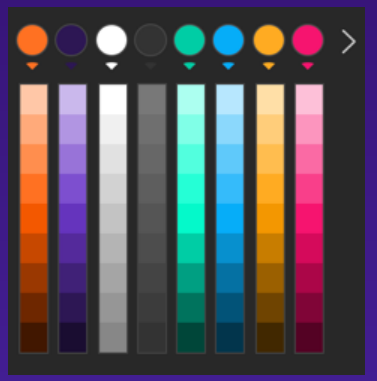
Ink

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

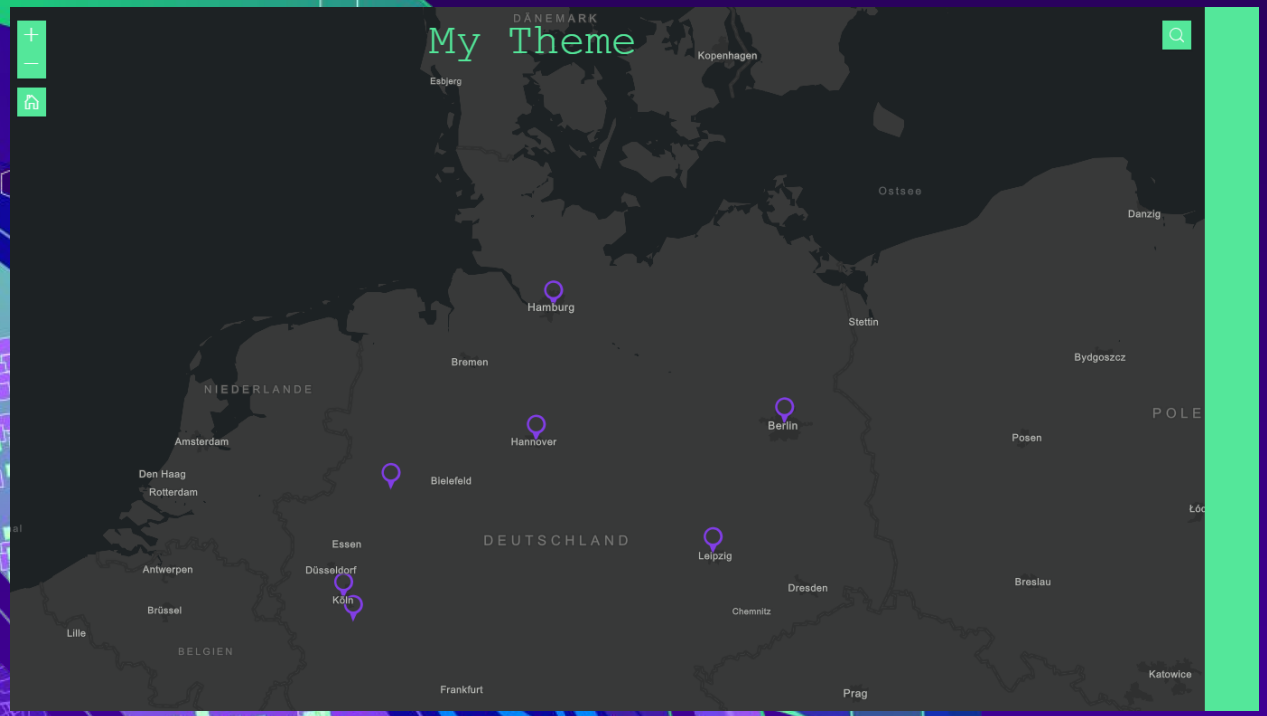
Basic Theme

- 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]
});
```




```
= new RouteParameters({
ureSet({
graphics.toArray()
rue
```

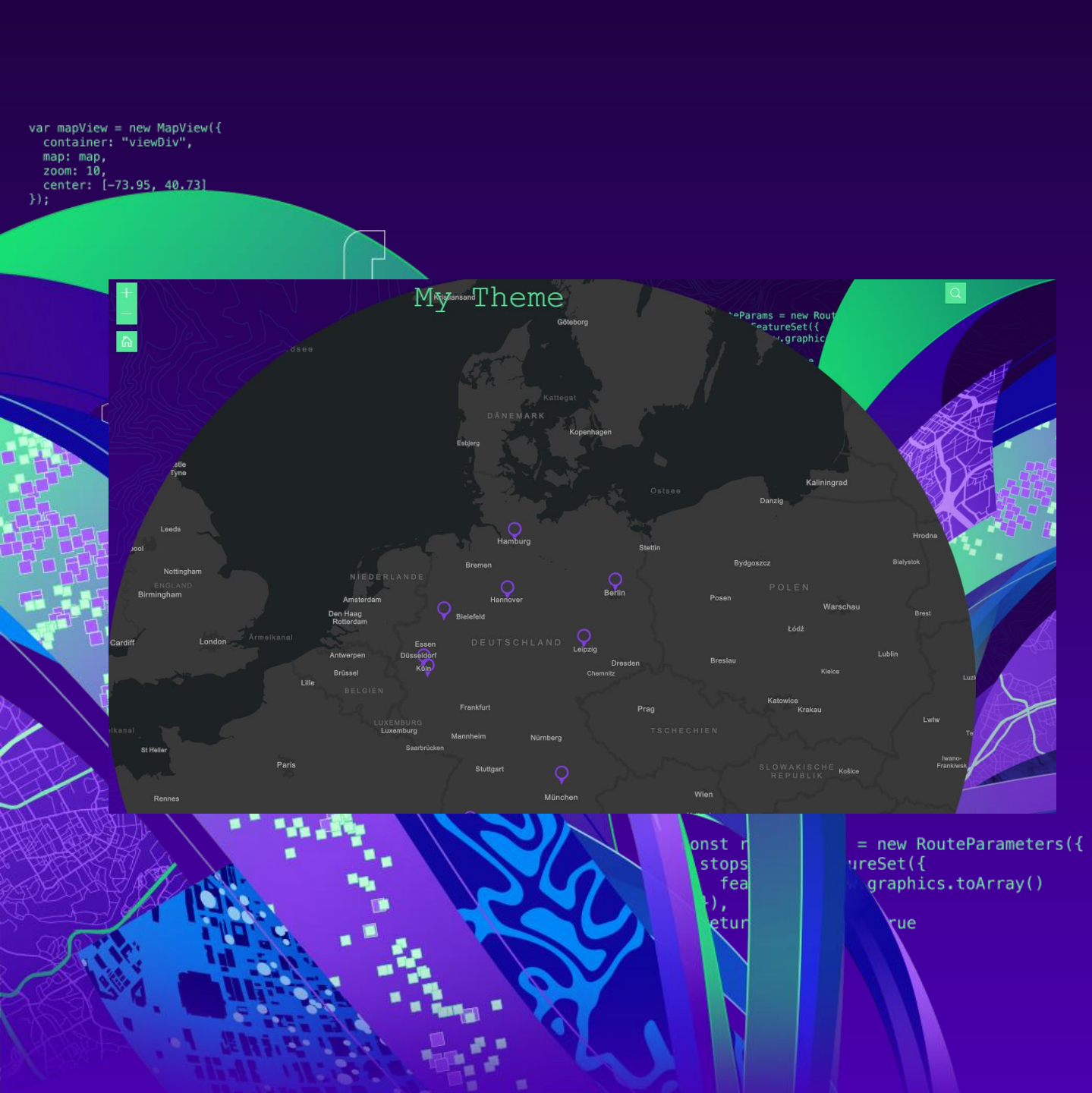
Basic Theme

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



Override CSS

Christine Wiltawsky

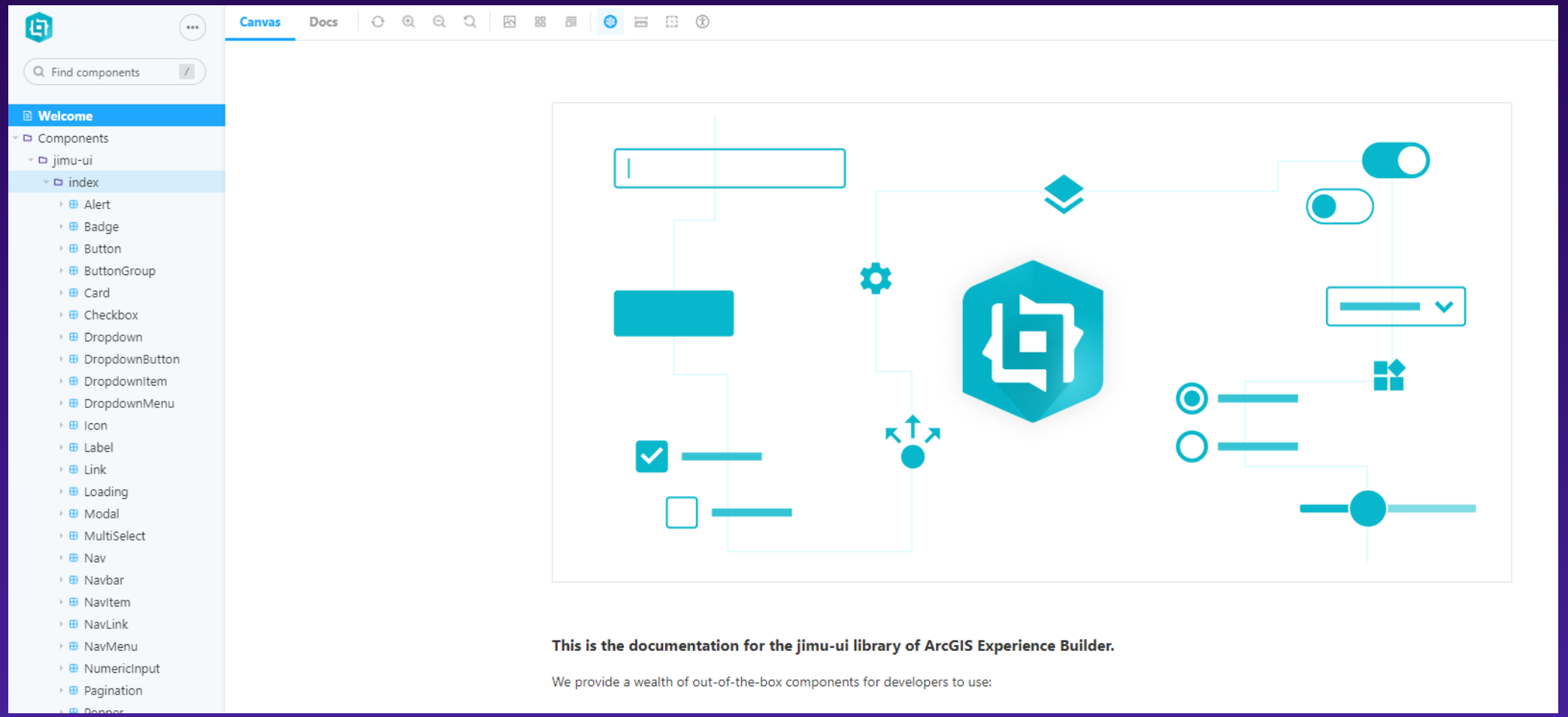
Components and Widget Styling

Christine Wiltawsky

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



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

Using UI Components

```tsx
// in widget.tsx:
import { React, AllWidgetProps } from 'jimu-core';
import { Button, Icon } from 'jimu-ui'; // import components

// Create an svg icon using Icon component:
const iconNode = <Icon icon={require('jimu-ui/lib/icons/star.svg')} />;

export default class Widget extends React.PureComponent<AllWidgetProps<IMConfig>, any> {
  render() {
    // Add Button component containing an icon to the widget:
    return <Button type='primary'>{iconNode} primary button</Button>;
  }
}

```

Styling your custom widget

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

Utility Classes

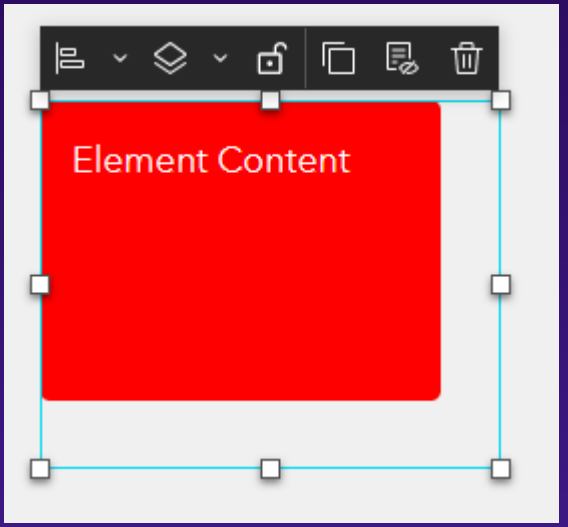
- Jimu UI provides the same CSS utility classes as Bootstrap to quickly apply styles to UI elements.
  - https://getbootstrap.com/docs/4.3/utilities

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

Inline CSS

```
render() {
  const containerStyle = {
    background: 'red',
    color: 'white',
    width: 200,
    height: 150,
    padding: '1rem',
    borderRadius: 5,
    fontSize: 'large',
    FontWeight: 'bold',
  };

  return (
    <div
      style={containerStyle} // CSS styles applied
    >
      Element Content
    </div>
  );
```



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.

style.css

client > your-extensions > widgets > simple > src > # style.css > .containerStyle

```css
.containerStyle {
  background-color: 'red';
  color: 'white';
  width: 200;
  height: 150;
  padding: '1rem';
  border-radius: 5;
  font-size: 'large';
  font-weight: 'bold';
}

```

widget.tsx

client > your-extensions > widgets > simple > src > runtime > widget.tsx > Widget > render

```tsx
import '../style.css';

export default class Widget extends React.PureComponent<
  AllWidgetProps<IMConfig>,
  any
> {
  render() {
    return (
      <div
        className='containerStyle' // CSS styles applied
      >
        Element Content
      </div>
    );
```

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 scaleStype = css`
103       font-weight: ${over ? 'bold' : 'normal'};
104       color: ${over ? 'red' : 'black'};
105     `;
106
```

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

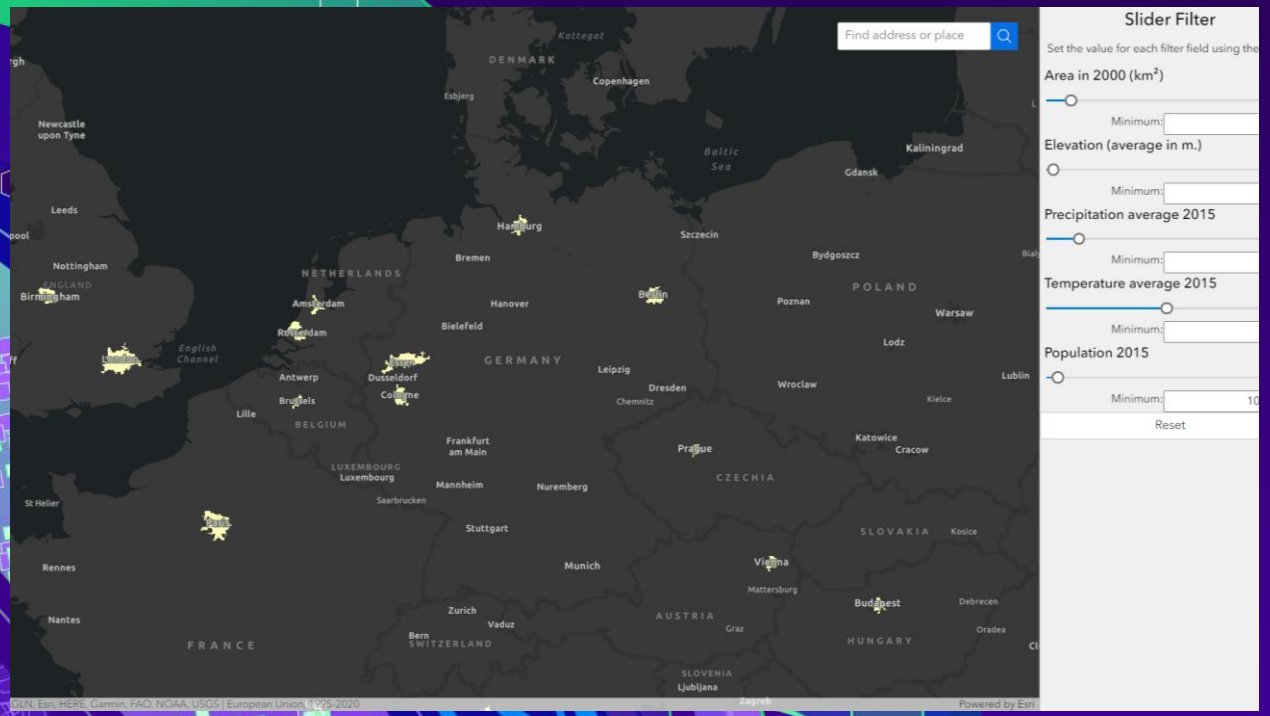
CSS in JS

- Styled components
  - Like css prop, but for components
  - Style components using Template Literals
  - The "styled" approach is perfect for creating re-usable 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

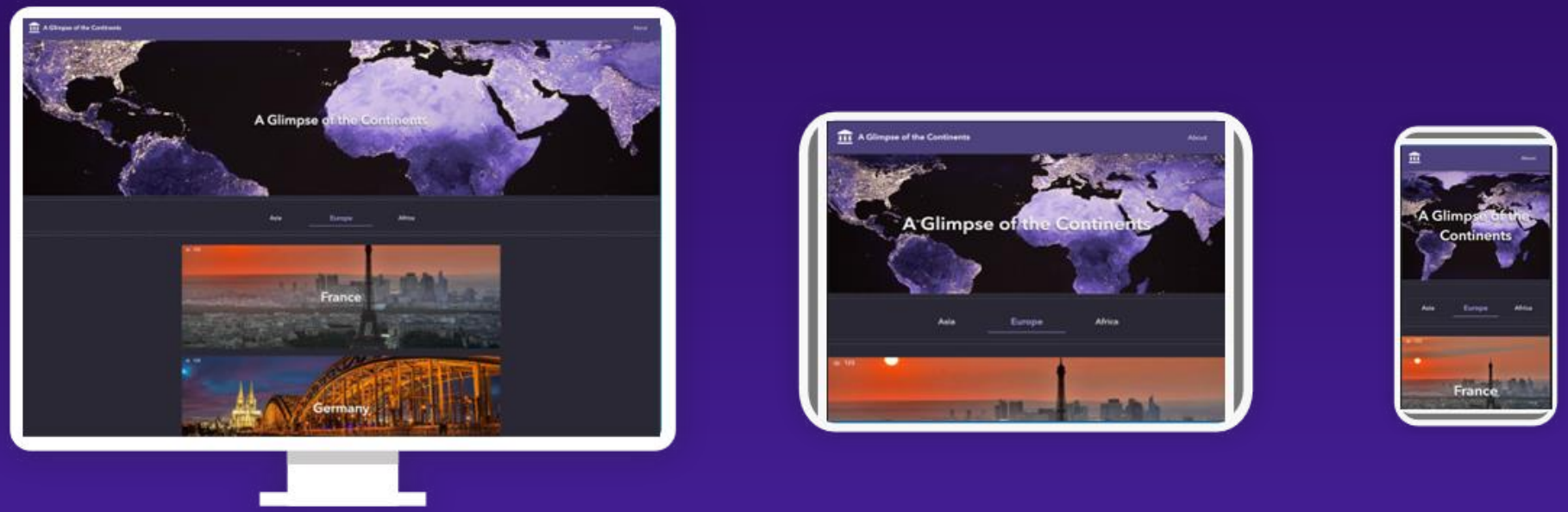
```
= new RouteParameters({
ureSet({
graphics.toArray()
rue
```

Mobile Layouts

Christine Wiltawsky

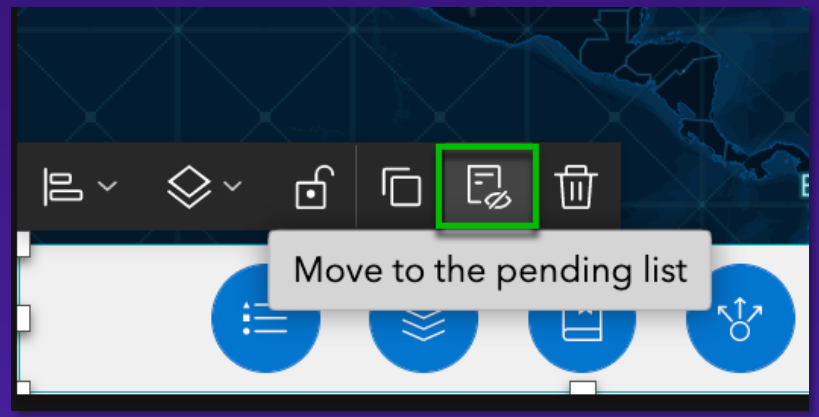
Mobile Optimization

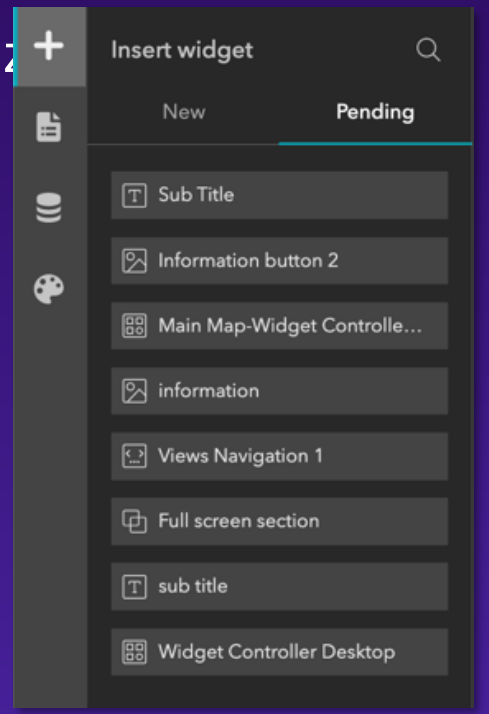
- Configure apps on Mobile, Tablet, and Desktop differently with one URL



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 siz
  - Move to the pending list





Mobile Optimization

Christine

Deployment

Christine Wiltawsky

Two Patterns

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

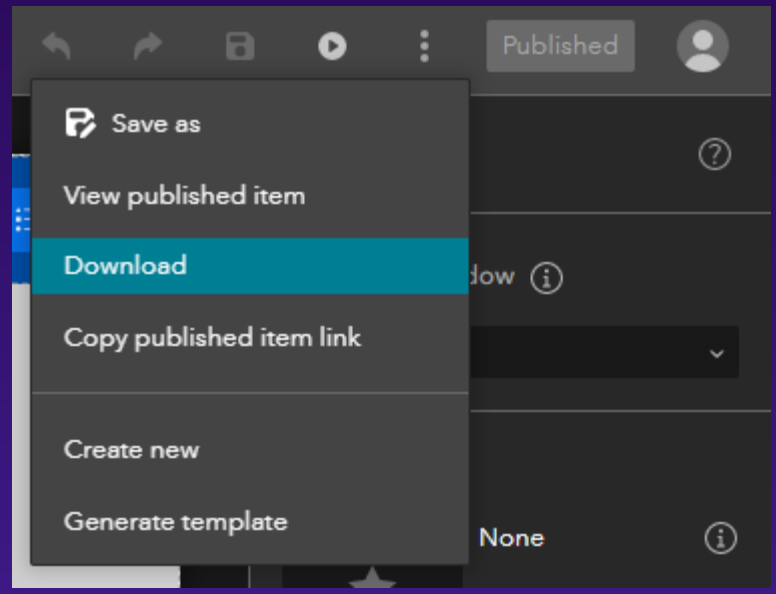
Deployment: ArcGIS Enterprise

- 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

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

New in 2023

Enhancements

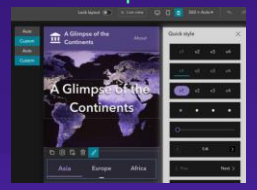
Slice

Elevation Profile

Editing Related Records

Web Scenes sync up

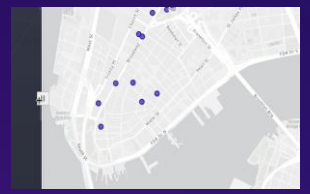
Mobile Optimization



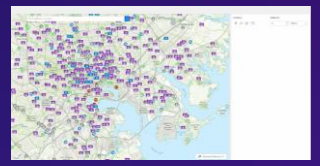
New widgets

Add Data

Swipe

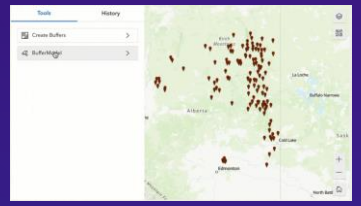


Near Me



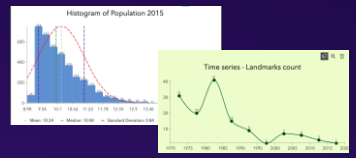
Basemap Gallery

Analysis

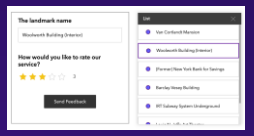


Enhancements

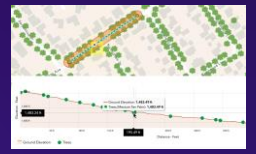
Business Analyst



Charts



Survey



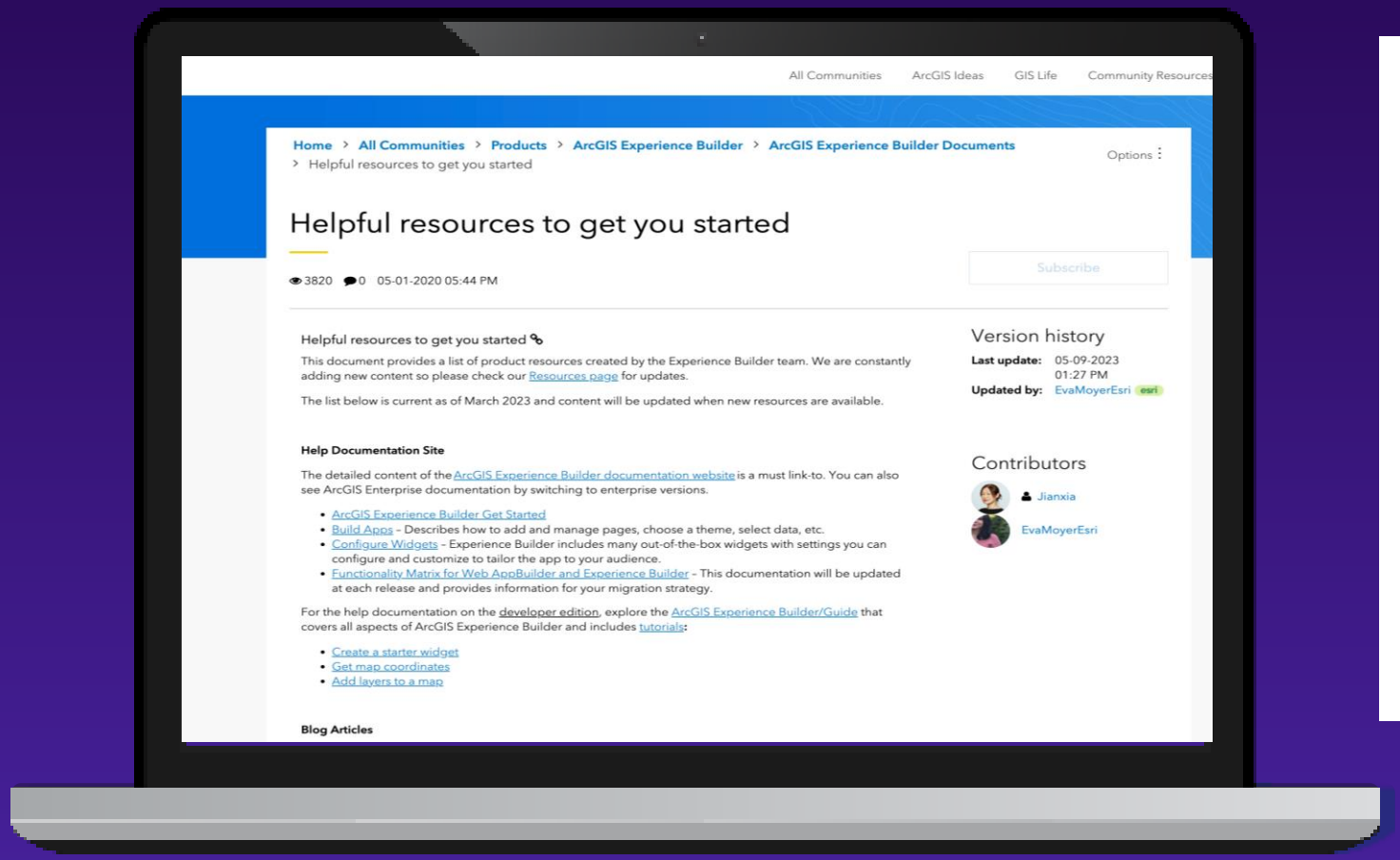
Elevation Profile

Widget Controller

Roadmap

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

Helpful Resources

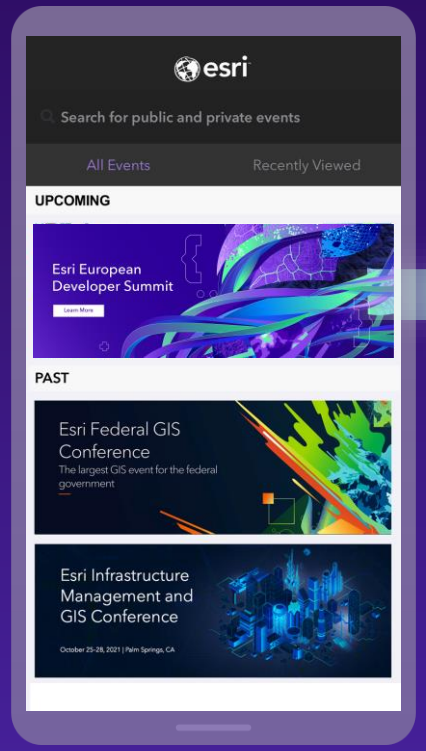


Helpful Resources

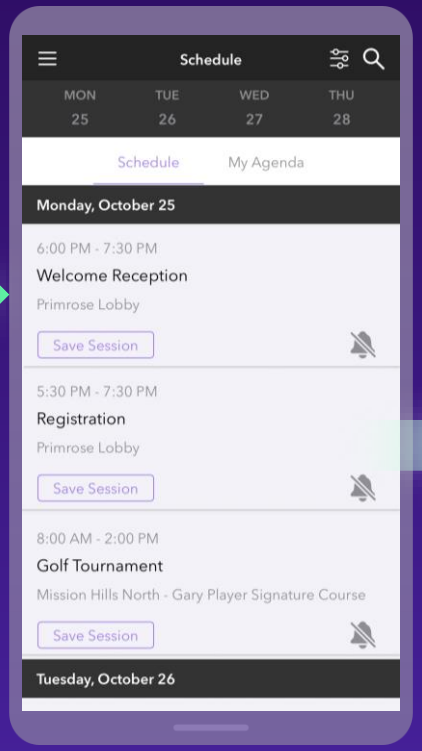
- Entwicklerdokumentation inkl. API-Referenz und Beispielen
- Esri Community
- Samples on GitHub
- Storybook / jimu-ui library
- Gallery

Please Share Your Feedback in the App

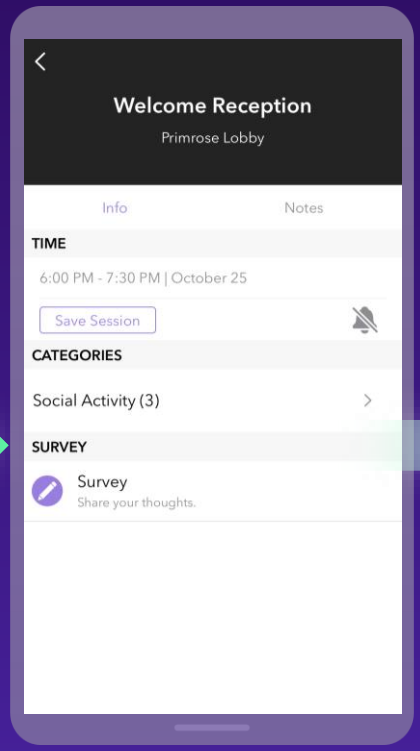
Download the Esri Events app and find your event



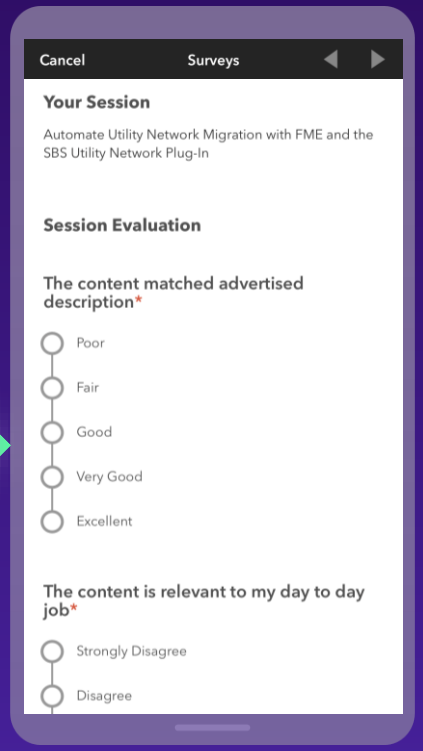
Select the session you attended



Scroll down to “Survey”



Log in to access the survey

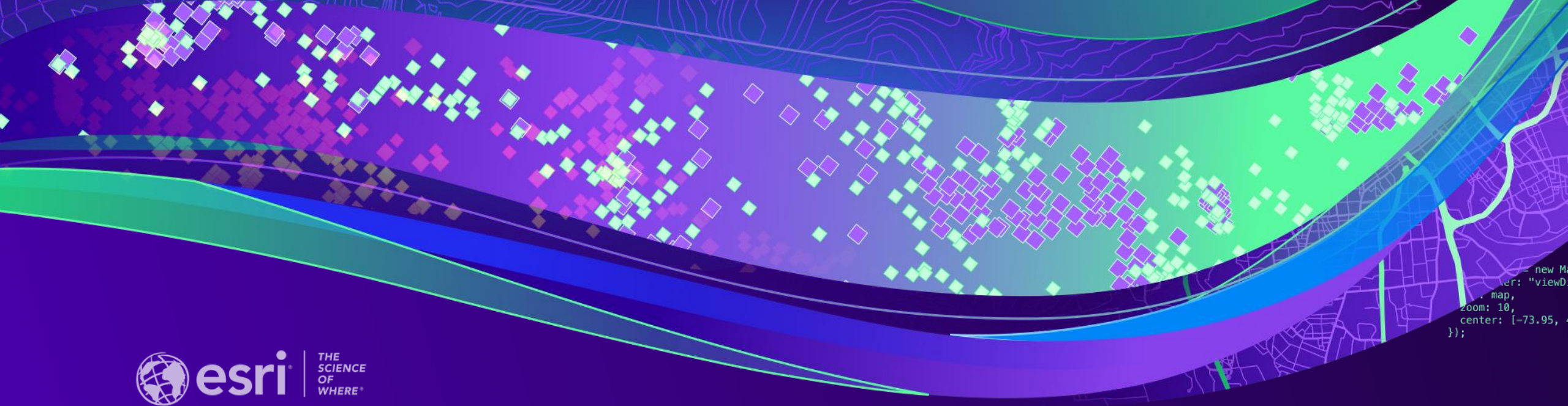


Connect With Us On Social

And Join the Conversation Using #EsriDevSummit

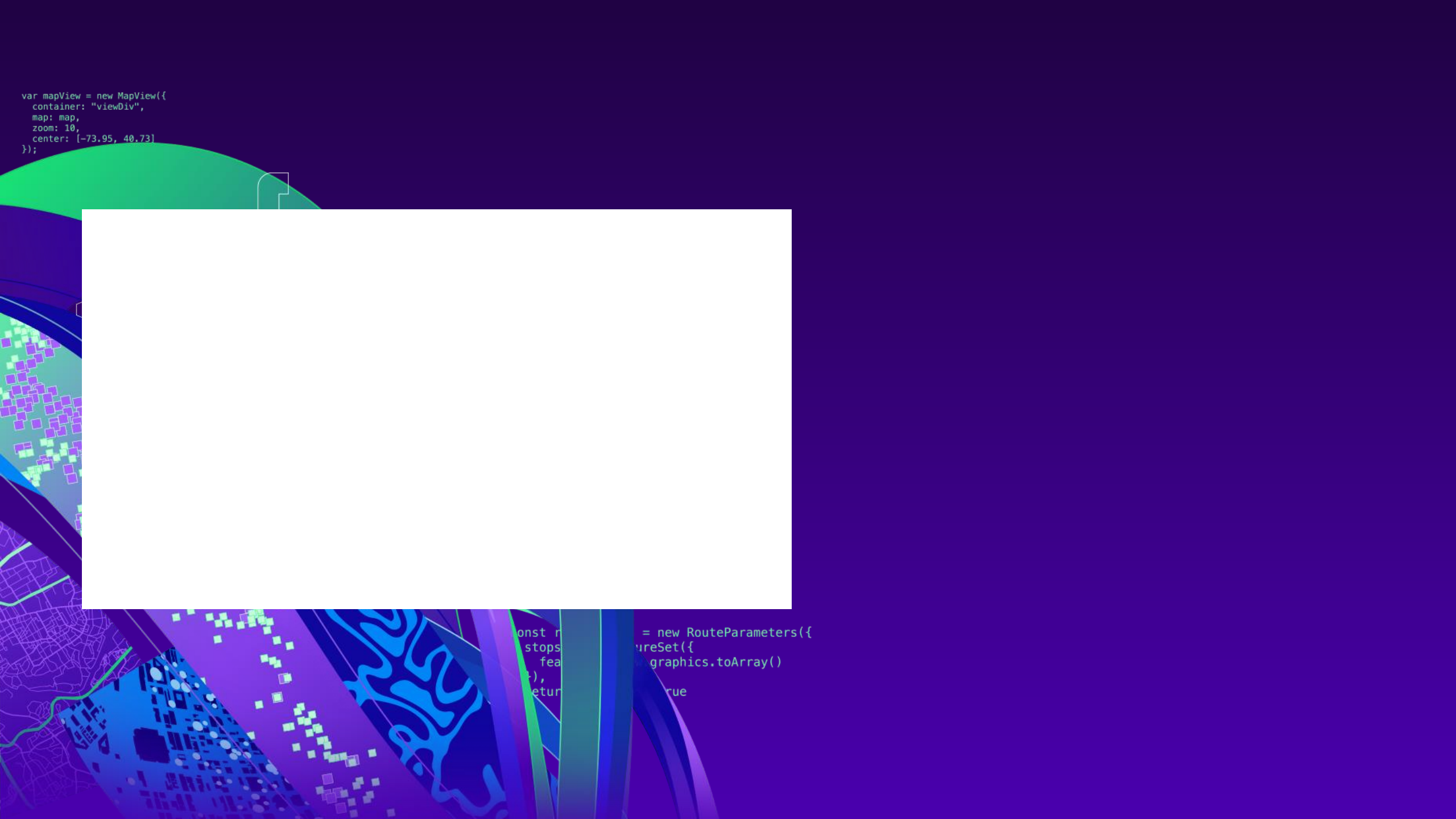
- twitter.com/EsriDevs
- twitter.com/EsriDevEvents
- youtube.com/@EsriDevs
- links.esri.com/DevVideos
- github.com/Esri
- github.com/EsriDevEvents
- links.esri.com/EsriDevCommunity

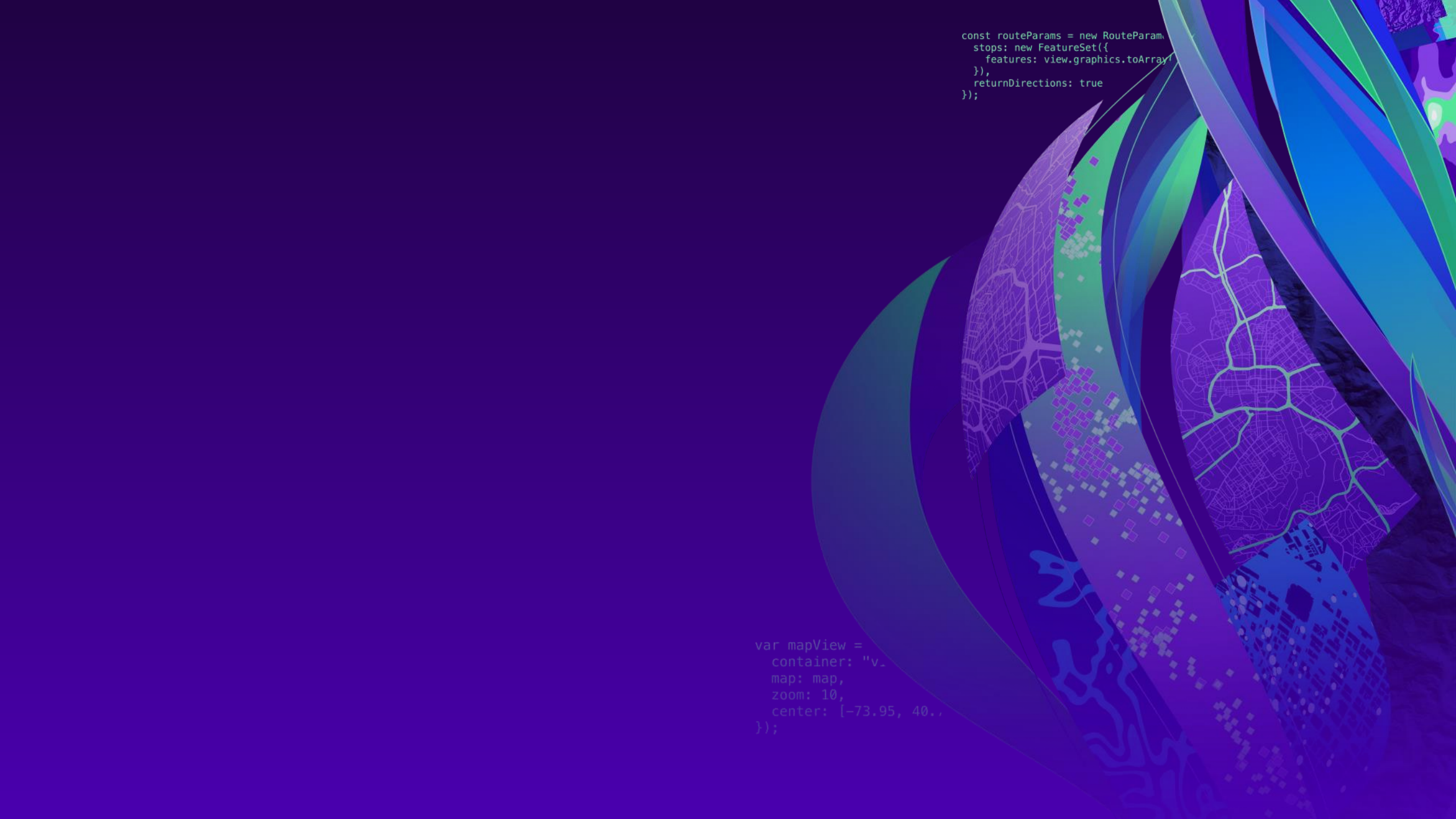
Test



esri | THE SCIENCE OF WHERE®

ESRI EUROPEAN DEVELOPER SUMMIT 2023


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


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.
});