



# ArcGIS Experience Builder: Customizing and Extending

Gavin Rehkemper

David Martinez

2021 ESRI  
DEVELOPER SUMMIT

# Agenda

- Developer tools
- Widgets
- Tips and tricks

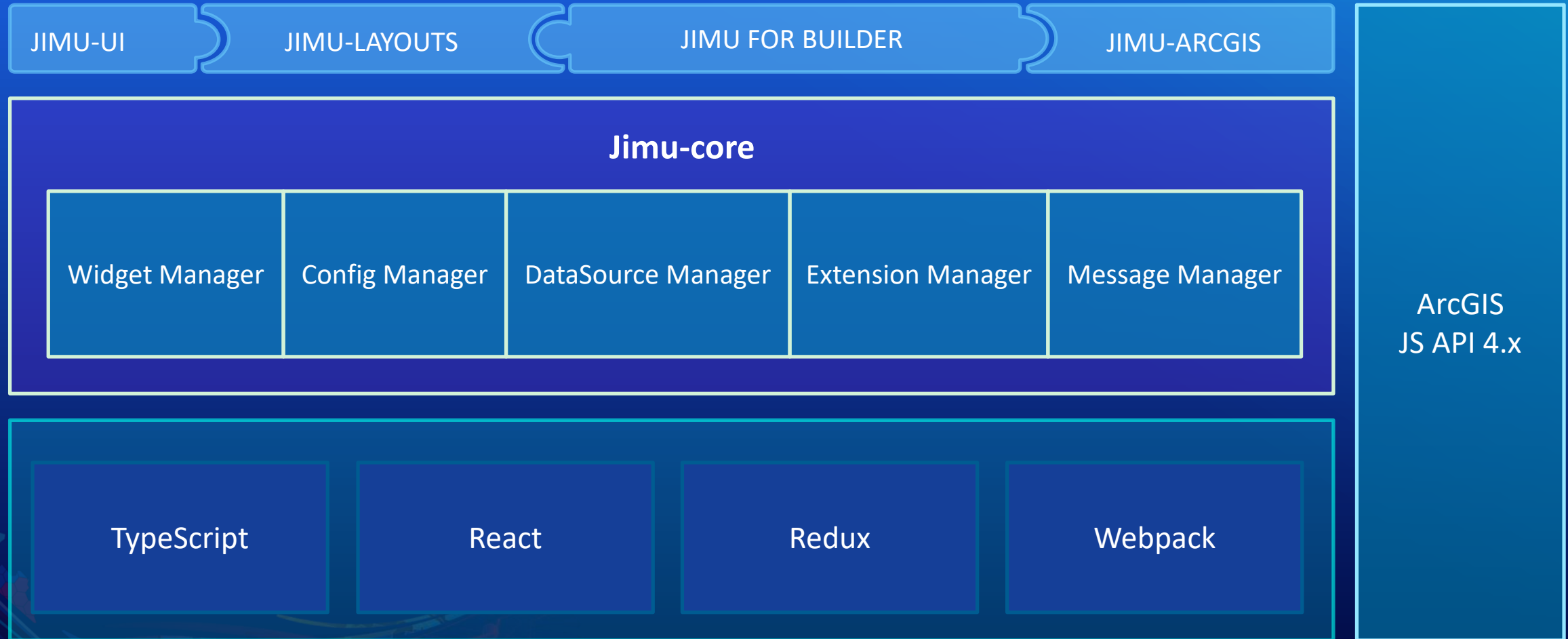


# Things you will need to know

- React
- TypeScript
- Jimu



# Technology stack

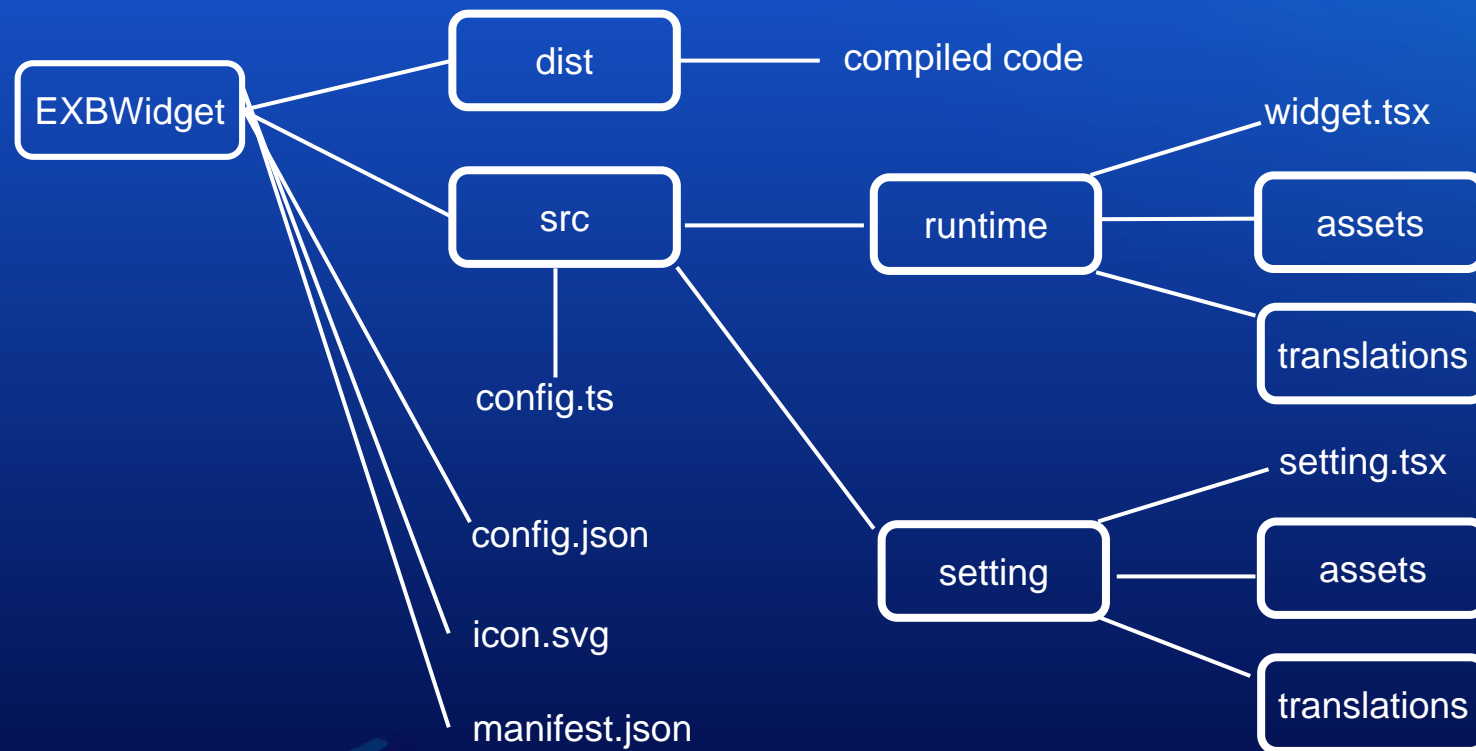




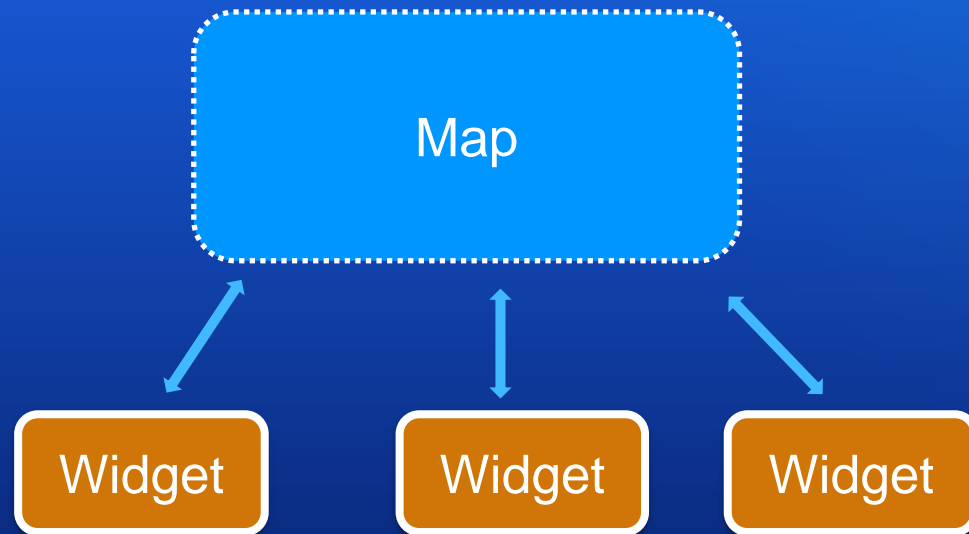
# Widgets

Building blocks to create an experience

# Widget Structure



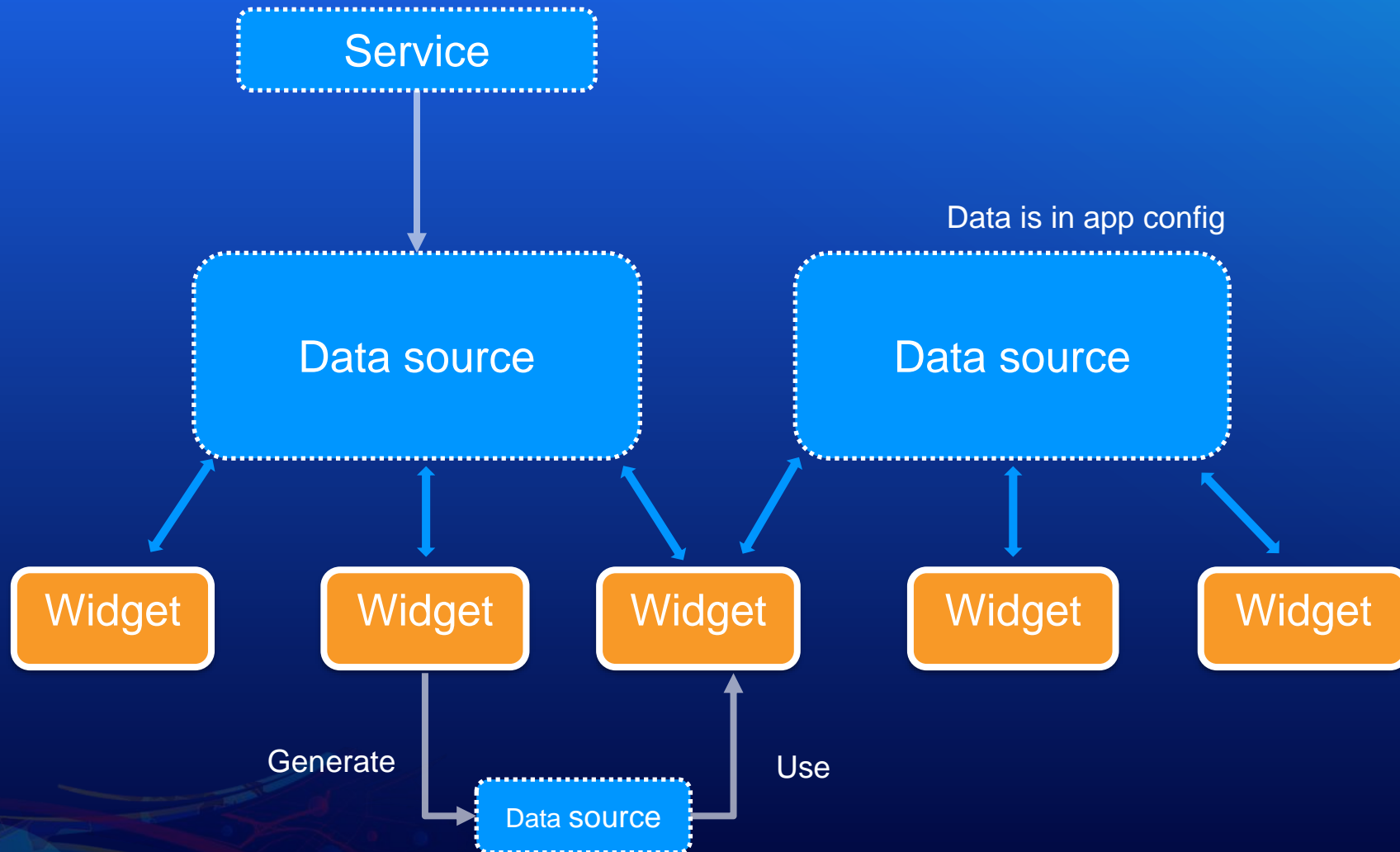
# Web AppBuilder – map centric app



All widgets hold a map object  
and can talk to map directly.



## ExB – data source centric app





# MapView/SceneView

- View concept same as ArcGIS JavaScript API
  - However, we wrap the view as JimuMapView
- Create a JimuMapView Object
  - MapViewManager.createJimuMapView()
- 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  
})
```

# Creating a widget



# Two ways to create a widget



- Class components
  - Make use of ES6 class and extend the component class in React.
  - Can maintain its own data with state.
  - Passes props (properties) down to class components and access them with `this.props`.
  - Uses the `render()` method.
- Function components
  - Basic JavaScript functions using arrows functions, but you can also use regular function keyword.
  - Can accept and use props.
  - Uses React Hooks to use state and other features.
  - There is no `render()` method.

# Create a widget using class component

```
//a custom pragma to transform your jsx into plain JavaScript
/** @jsx jsx */
//Import used to extend off BaseWidget class
import { AllWidgetProps, BaseWidget, jsx } from "jimu-core";
export default class Widget extends BaseWidget<AllWidgetProps, any> {

  render() {
    return (
      <div className="widget-starter jimu-widget" style={{ overflow: "auto" }}>
        <p>Hello world!</p>
        <p>Widget Name: {this.props.label}</p>
      </div>
    );
  }
}
```

# Create a widget using function component

```
/** @jsx jsx */
import { AllWidgetProps, jsx } from "jimu-core";

export default function Widget (props:AllWidgetProps) {
  return <div className="widget-starter jimu-widget" style={{ overflow: "auto" }}>
    <p>Hello world!</p>
    <p>Widget Name: {props.label}</p>
  </div>
}
```

# Creating a widget - Demo







# Tips & tricks

Discovering ways to get the most out of your development



# Top 3

- Change version of JS API
- Debugging
- Global state



# Change version of JS API

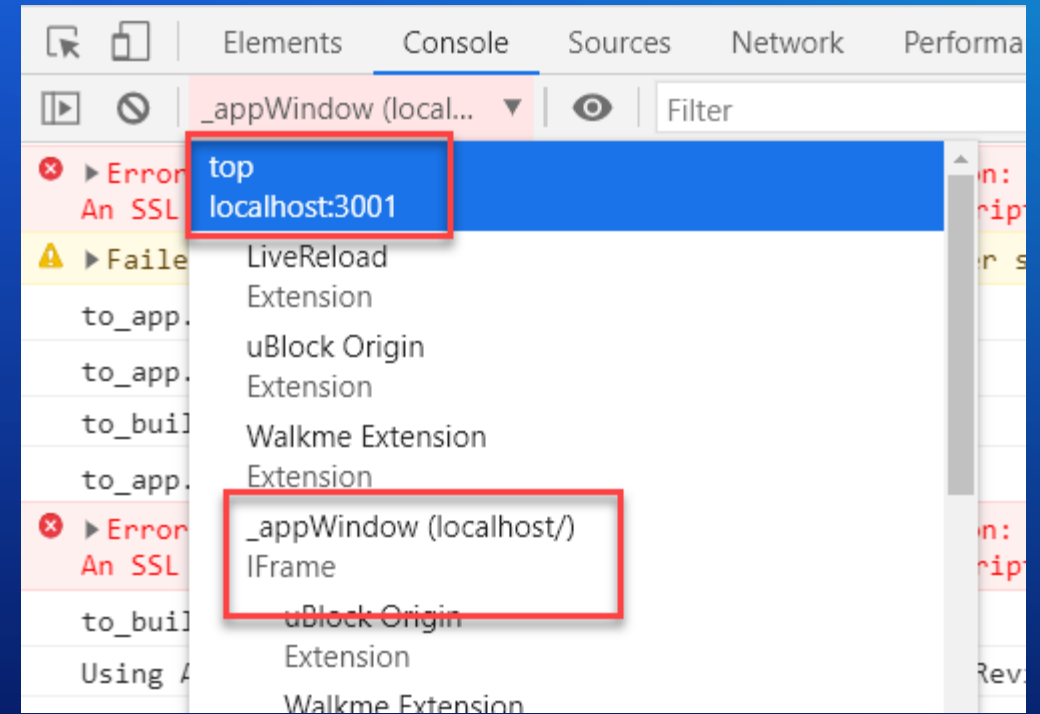
- Search for “arcgisJsApiUrl and replace the URL:
  - client/dist/experience/index.html
  - client/dist/template/index.html
  - client/dist/site/index.html
  - client/dist/builder/index.html
- Temporary change
  - <https://localhost?apiurl=//js.arcgis.com/4.18>

# Debugging

- VS Code Extensions
  - IntelliSense for CSS class names in HTML
  - vscode-styled-components
- Global Variables
  - `_appState`
  - `_dataSourceManager`
  - `_widgetManager`
  - `_sessionManager`

# Debugging

- In the Builder Page
  - “window.jimuConfig.isBuilder”



# Automatic Map Reference

- When you know the app your widget is in will only have one map
- Would like to reduce the configuration steps

```
render() {  
  return (  
    <div className="widget-demo jimu-widget m-2">  
      <JimuMapViewComponent  
        useMapWidgetIds={this.getArbitraryFirstMapWidgetId()}  
        onActiveViewChange={this.activeViewChangeHandler}  
      />  
    </div>  
  );  
}
```

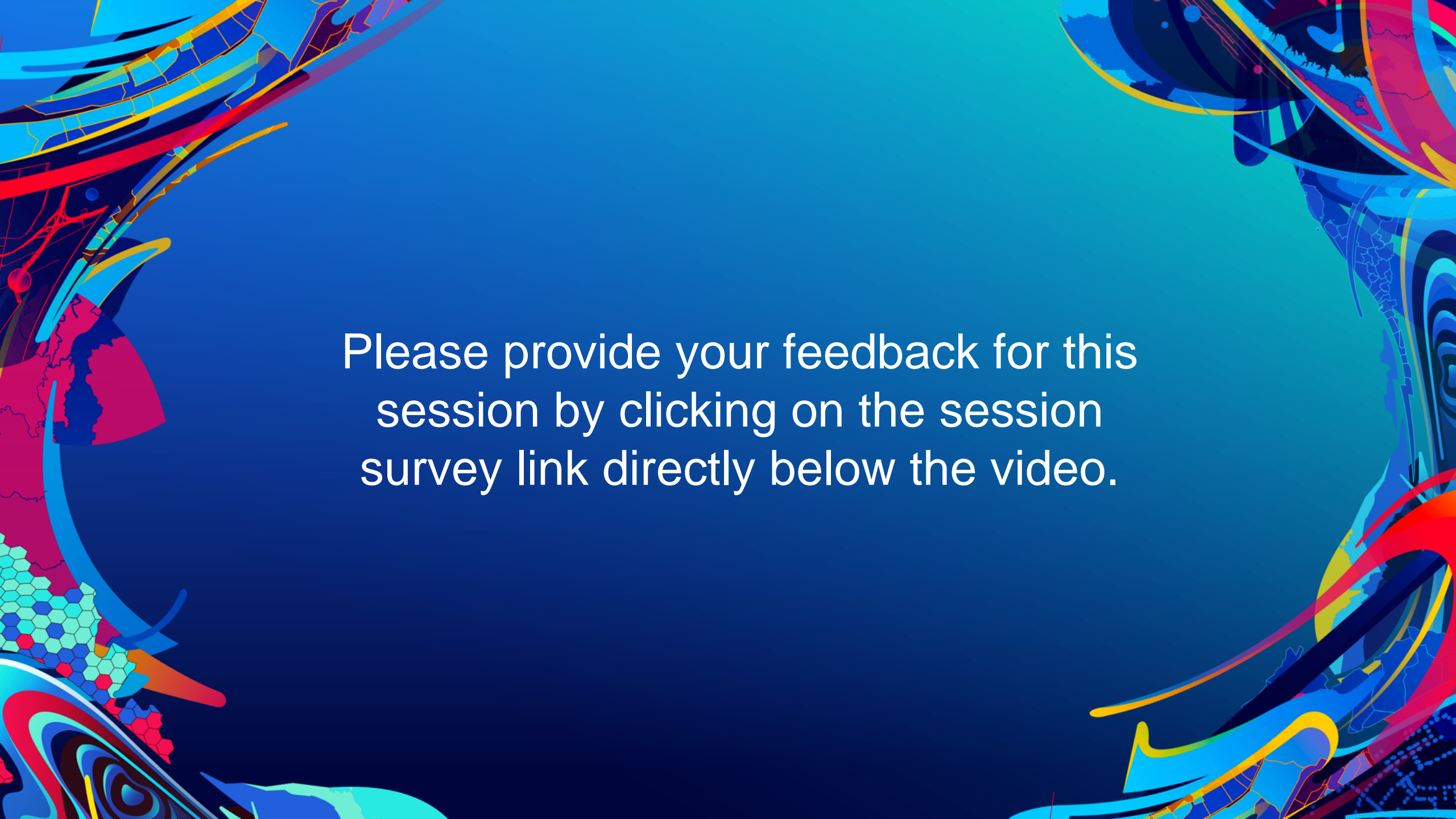
# Automatic Map Reference

```
getArbitraryFirstMapWidgetId = (): ImmutableArray<string> => {  
  const appState: any = window._appState;  
  // Loop through all the widgets in the config and find the "first"  
  // that has the type (uri) of "arcgis-map"  
  const arbitraryFirstMapWidgetInfo: { [key: string]: any } = Object.values(appState.appConfig.widgets).find((widgetInfo: any) => {  
    return widgetInfo.uri === 'widgets/arcgis/arcgis-map/'  
  });  
  
  let useMapWidgetIds = Immutable([]); // empty by default  
  if (arbitraryFirstMapWidgetInfo) {  
    useMapWidgetIds = Immutable([arbitraryFirstMapWidgetInfo.id]);  
  }  
  return useMapWidgetIds;  
}
```

# Resources

- Code for this presentation : <https://esriurl.com/ds21exb>
- Samples repo: <https://github.com/Esri/arcgis-experience-builder-sdk-resources>
- Doc site: <https://developers.arcgis.com/experience-builder/>
  - Guide
  - Sample code
  - API reference
- Post questions and feedback: <https://community.esri.com/t5/arcgis-experience-builder/ct-p/arcgis-experience-builder>





Please provide your feedback for this session by clicking on the session survey link directly below the video.



esri®

THE  
SCIENCE  
OF  
WHERE®