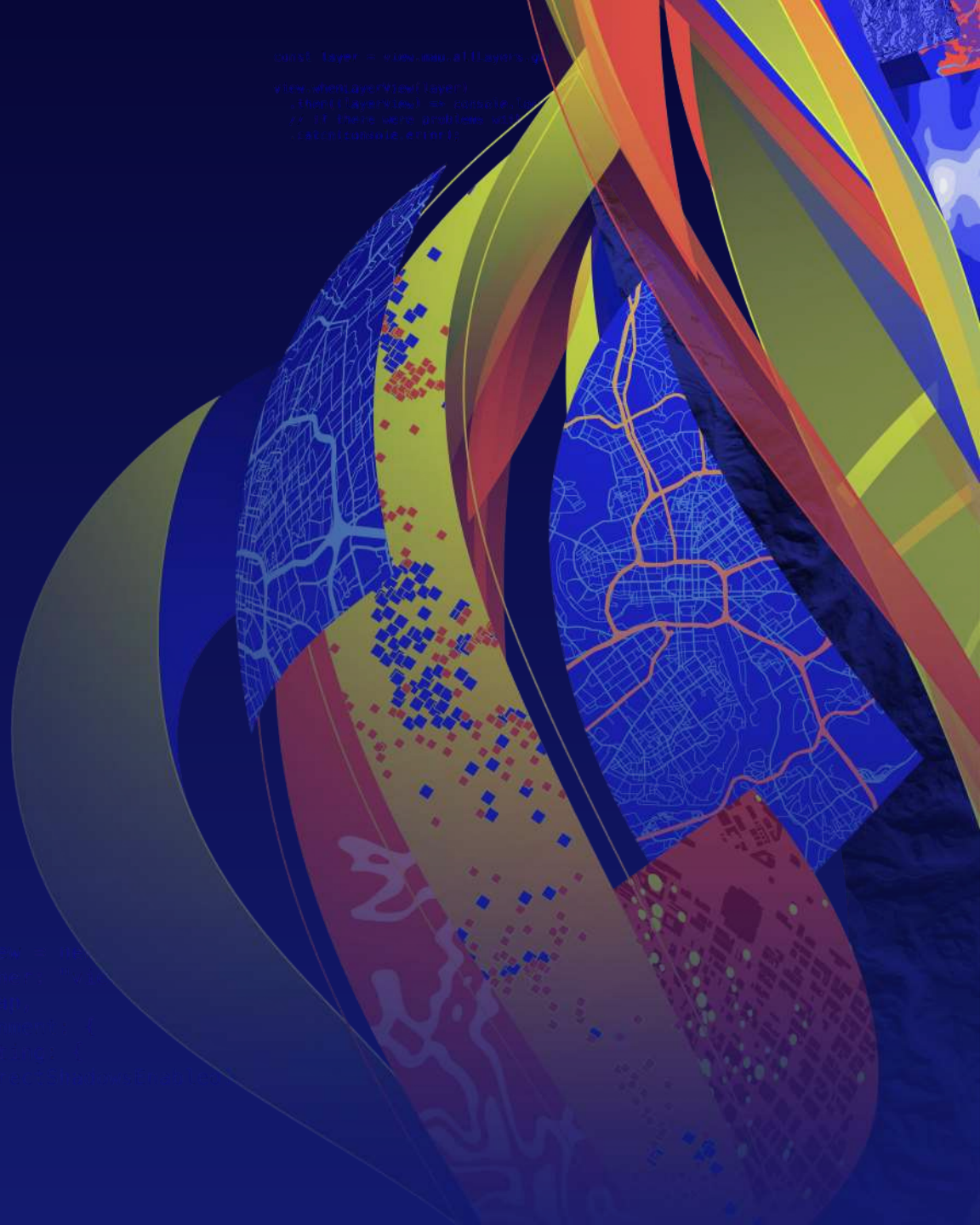
ESRI DEVELOPER SUMMIT 2023

Widget Styling and Custom Themes

Agenda

- Components and Widget Styling
- Themes
- Activity

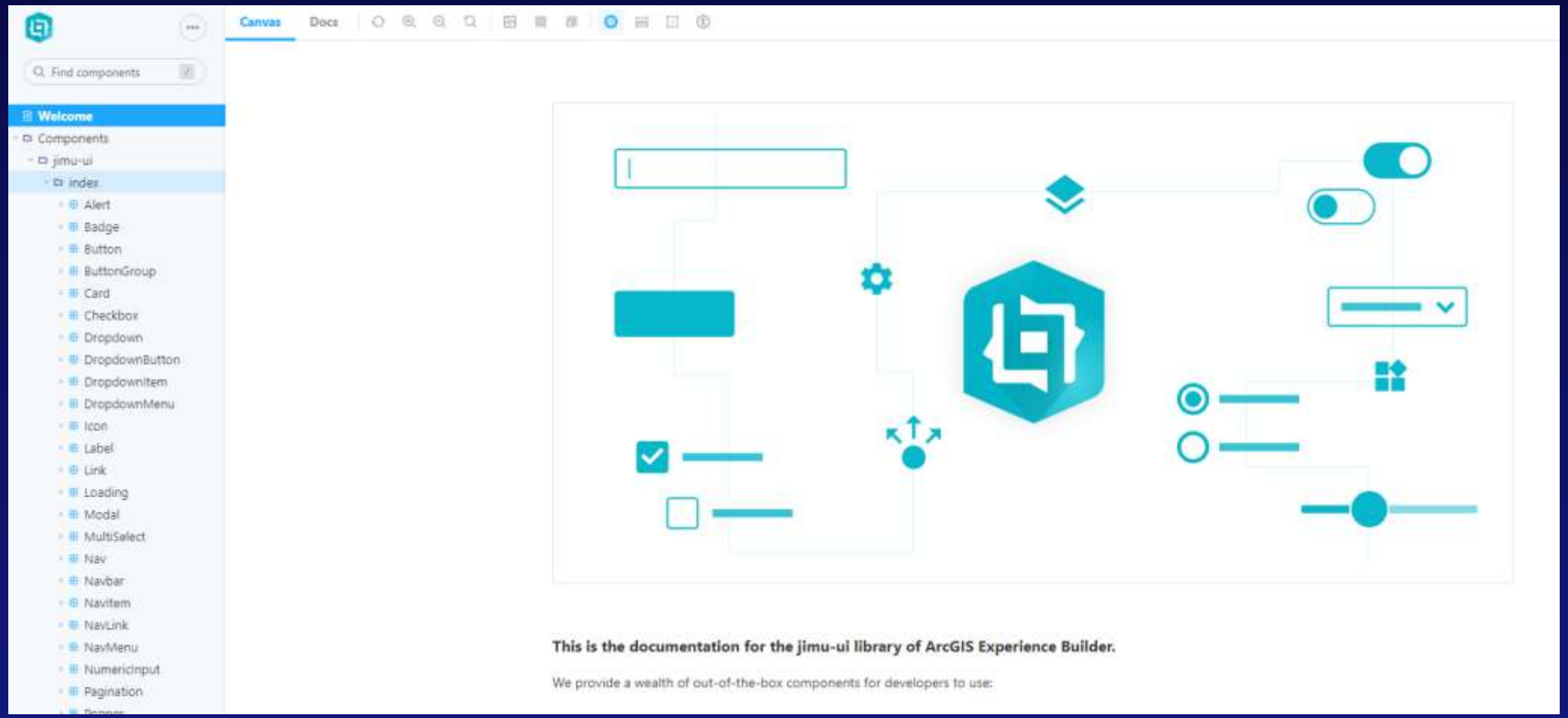
Components and Widget Styling



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

```
// 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>;
  }
}
```

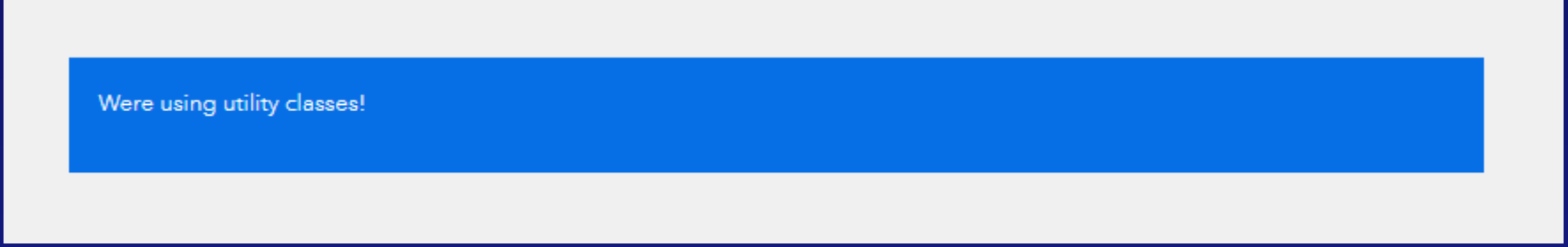
Styling your custom widget

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

Utility Classes

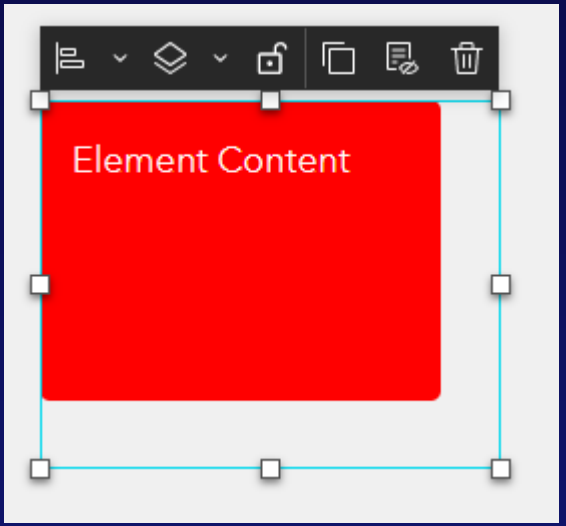
- Jimu UI provides the same CSS utility classes as Bootstrap to quickly apply styles to UI elements.
  - https://getbootstrap.com/docs/4.3/utilities

```
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  }
```



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.

style.css

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

```
.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

```
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.

```
const over = this.state.scale > 5000000;
const scaleStype = css`
  font-weight: ${over ? 'bold' : 'normal'};
  color: ${over ? 'red' : 'black'};
`;
```

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

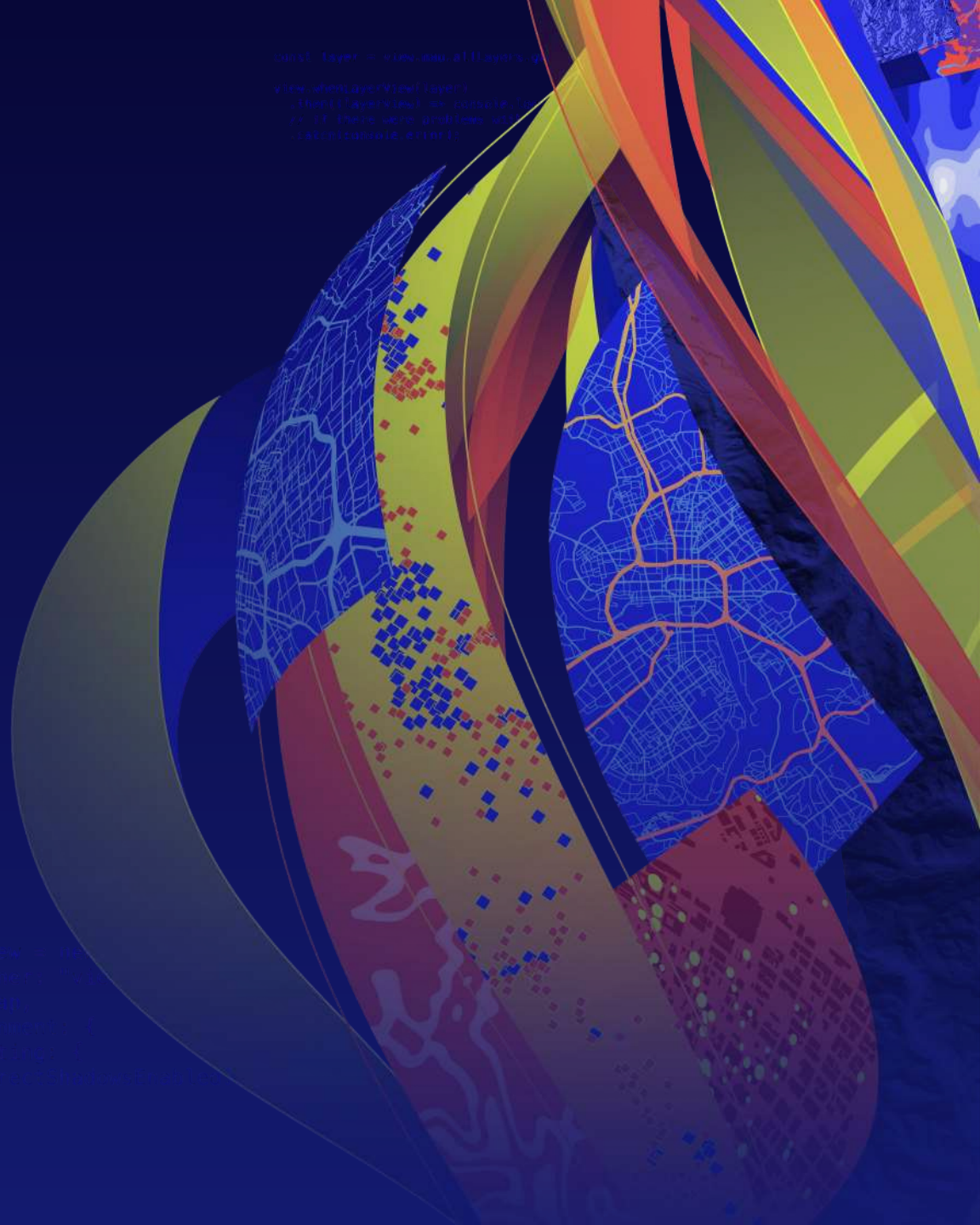
CSS in JS

- 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

```
// A styled button component:
const StyledButton = styled.button`
  color: white;
  border: none;
  border-radius: 5px;
  background-color: blue;
  transition: 0.15s ease-in all;
  &:hover {
    background-color: green;
  }
`;
```

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

Custom Themes



What is available

Basic:

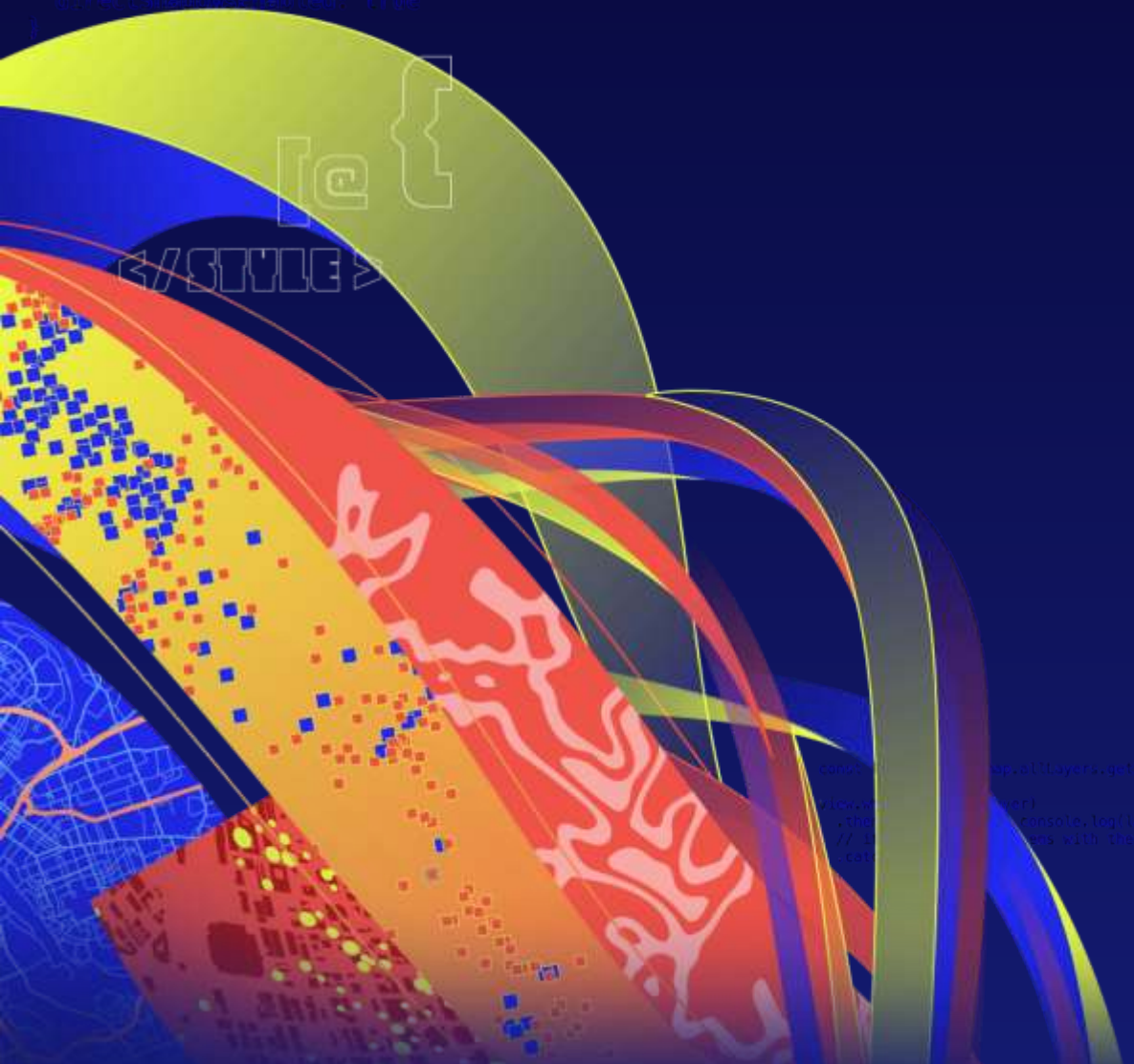
- Colors
- Typography
- Spacing

Advanced

- Style override via Sass CSS
- Assets: fonts, images, etc.

Files

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

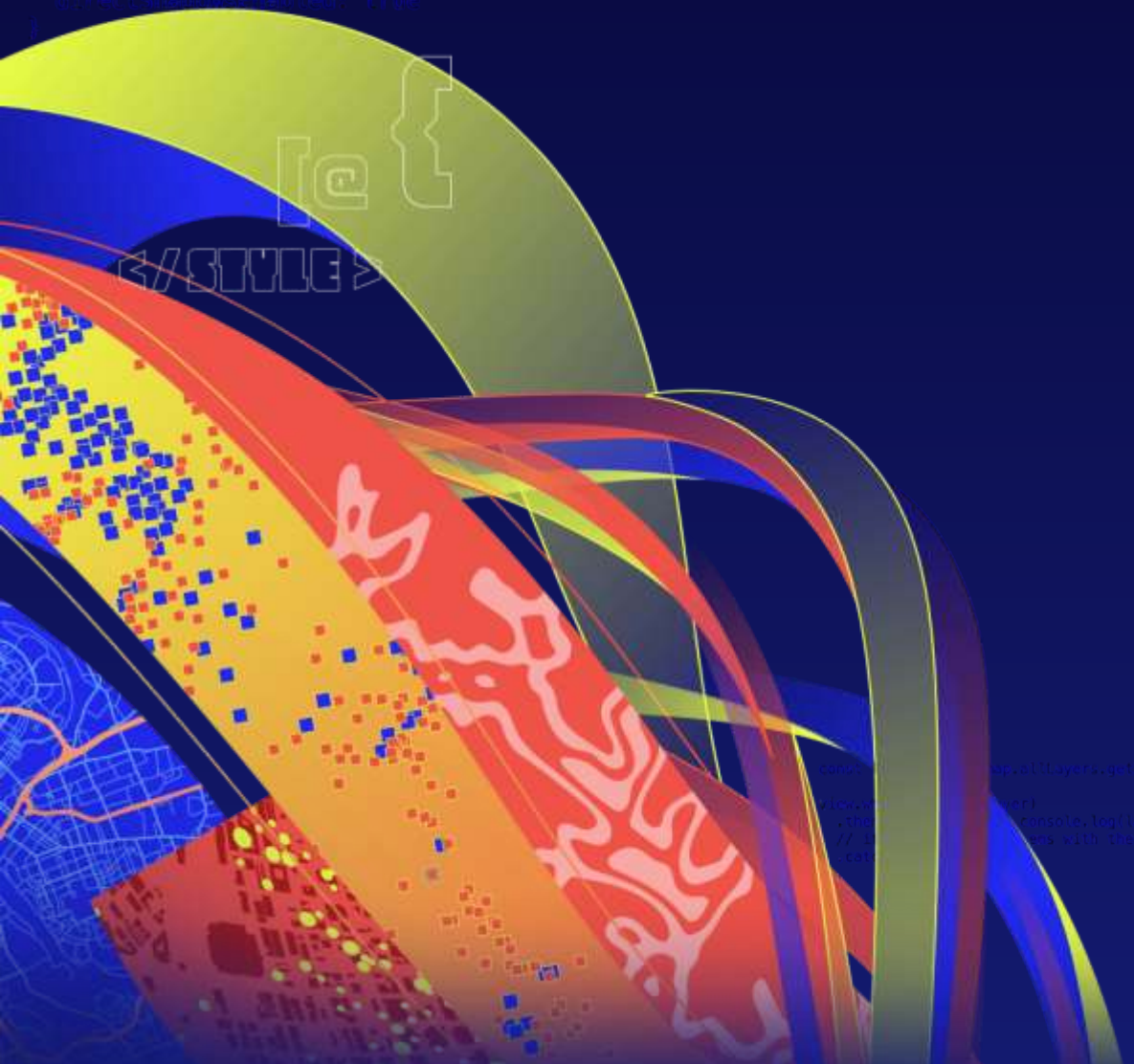


Basic Theme

Demo

Override CSS

- style.ts
  - Use the Emotion library for CSS-in-JS
- style.scss
  - More traditional way
  - Imported last



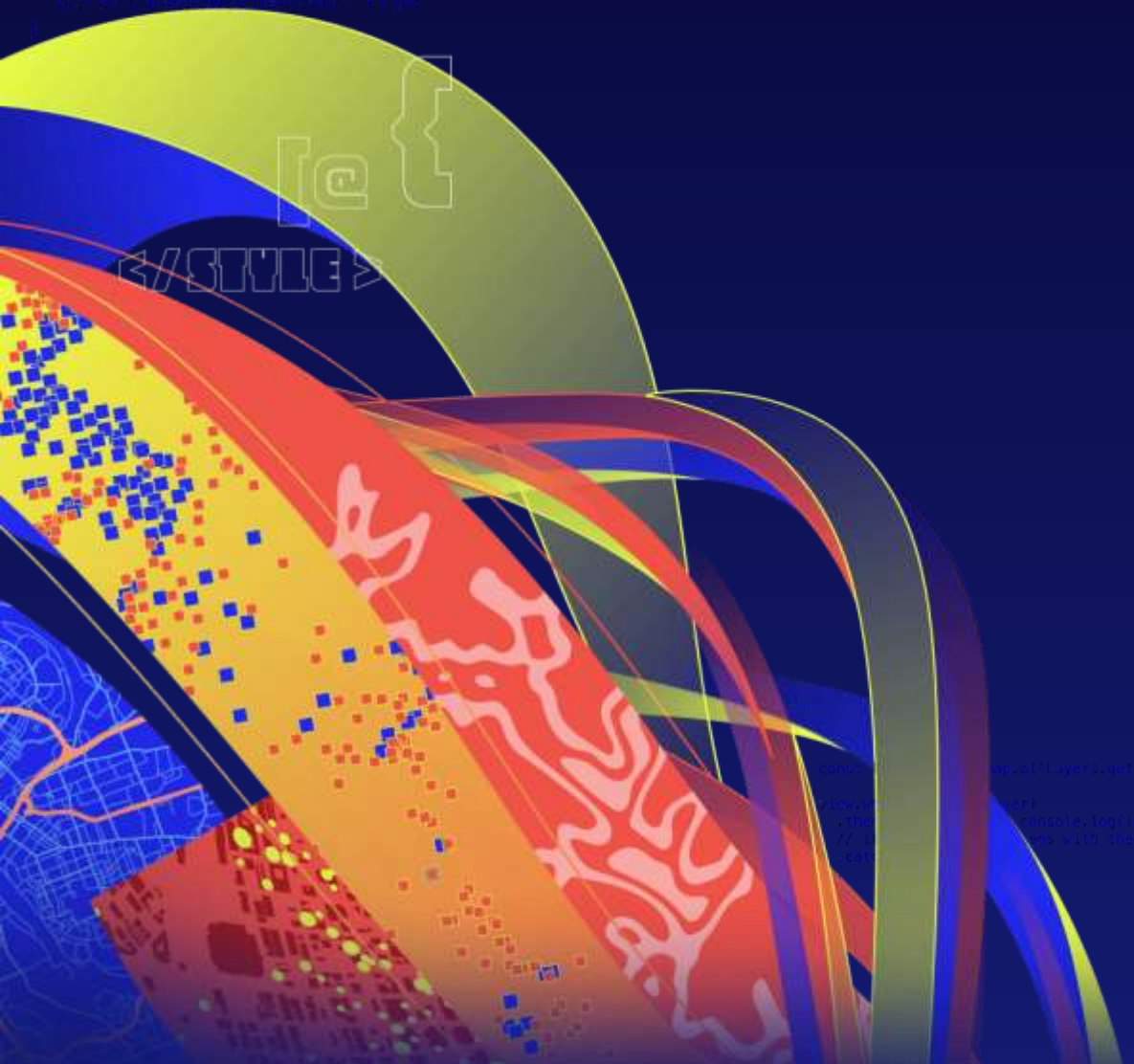
Override CSS

Demo

Custom Fonts

- Use the “Override CSS” method:

```
@import url('https://fonts.googleapis.com/css?family=Open+Sans');
```

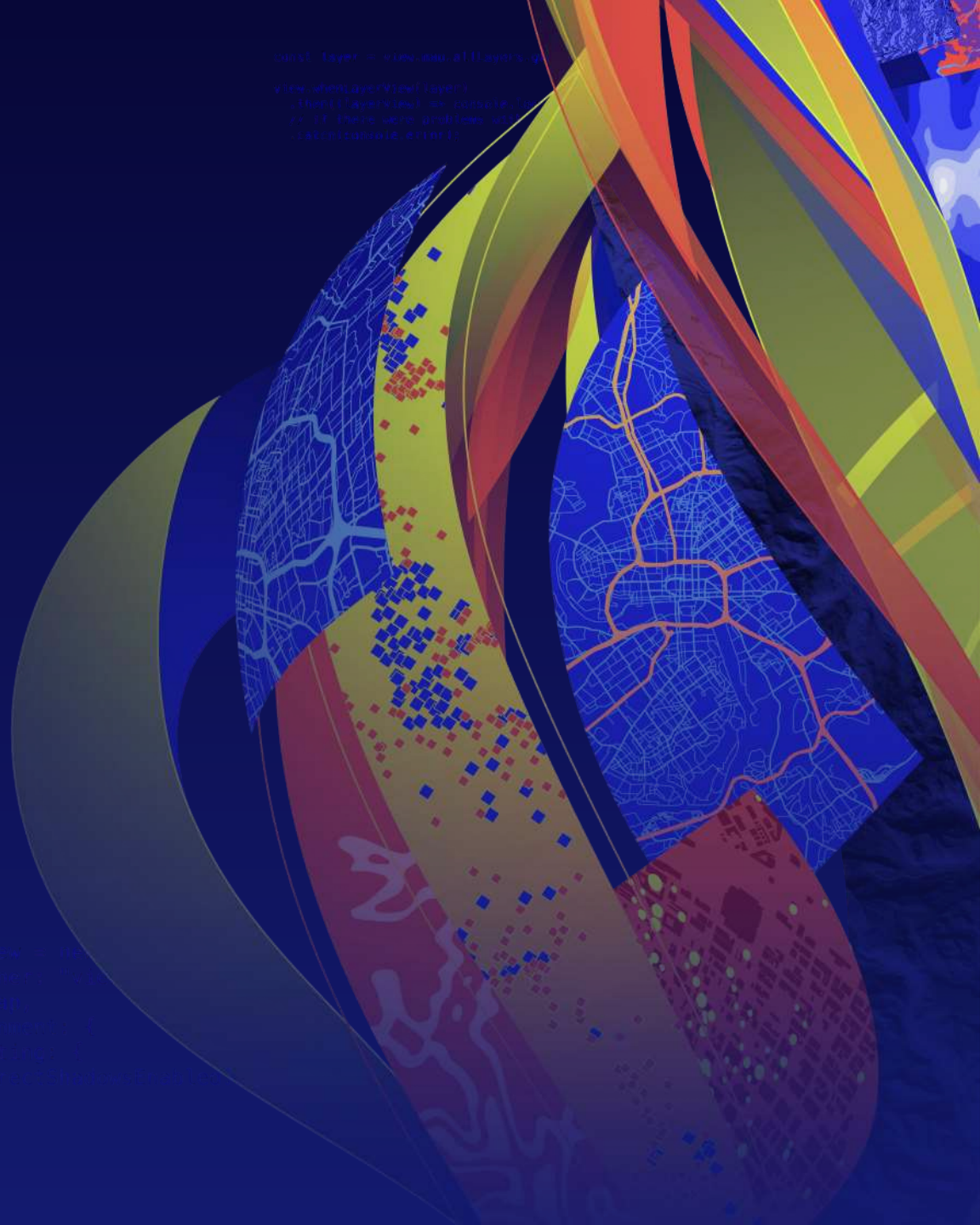


Custom Fonts

Demo

Walk through

Custom Theme



Activity

30 Minutes

- Add some custom styles to your widget from earlier today.
- Create a custom theme, using your organization's brand and colors.
  - If you don't have one, use the Olympics 2024 colors/theme: https://www.paris2024.org/en/design
- Bonus:
  - Also include the Paris2024 font (*is this possible?*)
- Documentation: https://developers.arcgis.com/experience-builder/guide/theme-development/

esri® | THE SCIENCE OF WHERE®