Agenda

- What can be customized
- Customization approaches with demos
- Q & A
Customizing Widgets

- Theming
  - Changing styles: colors, sizing, font, etc.
- Implementing widget in a different framework
- Altering presentation of a widget
Customization Approaches

- Authoring a theme
- Recreating a view
- Extending a view
Ready?

CONTINUE?
Y/N
Level I

The 

Theming
Level I: Theming

Why Theme?

- Match branding.
- Match the map.
- Contrast with the map. Based on the environment.
- User-specific (e.g. bigger buttons)
A powerful scripting language for producing CSS.
Why Sass?

- It's modular.
- It's DRY.
- It makes theming easy.
Let's Create a Theme!

1. Create your theme directory.
   - esri/themes/[your-theme-name]/
2. Create a Sass file in your theme directory.
   - main.scss
   - @import "../base/core";
3. Compile.
4. Include the compiled CSS in your app.

<!-- in your app -->
<link rel="stylesheet" href="esri/themes/[your-theme-name]/main.css">
Theme Structure

Avoid writing a bunch of CSS selectors.
Include the theme "core" and override the default values.
Theme Structure

The theme core brings in three main variable files:

- `base/_colorVariables.scss`
- `base/_sizes.scss`
- `base/_type.scss`

These set the default values.
Theme Structure

Default

Any value assignment overrides the `!default` value.

```
// Inside base/_colorVariables.scss
$background_color : #fff !default ;
```

```
// Inside esri/themes/[your-theme-name]/main.scss
$background_color : #cc4b09;
```

But wait...there's more!
Theme Structure

Override the four main color variables...

```
$text_color : #fff; // white
$background_color : #cc4b09; // mario
$anchor_color : #ffbaaa; // luigi
$button_text_color : #ffbaaa; // luigi
```

...then magic!

Theming Guide
Demo: Theming
Level I: Theming Recap

- Use Sass for easy theming.
- Theme structure
  - Color
  - Size
  - Typography
- Use the core and override values.
LEVEL UP!
Level II: Widget Composition

Widgets are composed of Views & ViewModels

- Reusable
- UI replacement
- Framework integration
Level II: Views

- Presentation of the Widget
- Uses ViewModel APIs to render the UI
- View-specific logic resides here
Level II: Working with Views

API Exploration

- LayerList Doc
- LayerList.tsx (View)
- LayerList.scss (Theme)
- LayerList Sample
Views: Let's customize!

Let's create a custom widget view.

Using...

- jQuery
  - Plugin creation
- Bootstrap
Create a custom LayerList view for a Bootstrap app

- **Demo Steps**
- **Demo**
Level II: Views Recap

What have we learned about Widget Views?

- Views
  - Face of the widget
  - Renders the viewModel brains
  - View separation allows framework integration
  - Views can be downloaded on API docs
  - Can create views in other frameworks using ViewModels
LEVEL UP!
LEVEL III

Extending a View
Level III: Extending a View

- Why?
  - Reusable
  - Same ecosystem

- How?
  - JS API v4.4
  - esri/widgets/Widget
  - TypeScript
esri/widgets/Widget

• Provides lifecycle
• API consistency
Lifecycle

- constructor
- postInitialize
- render
- destroy
render

- Defines UI
- Reacts to state
- Uses JSX
TypeScript

- Typed JavaScript
- JS of the future, now
- IDE support
  - Visual Studio
  - WebStorm
  - Sublime
  - and more!
Demo: Extending a View

Demo Steps
Level III: Extending a View

Recap

- Reusable
  - View/ViewModel
- Same ecosystem  No
  - extra libraries
- Extended existing widget
  - Lifecycle
  - TypeScript
LEVEL UP!
LEVEL IV

Put it all together.
Conclusion

- Authored a theme
- Recreated a view
- Extended a view
Suggested Session

- Developing Your Own Widget with the ArcGIS API for JavaScript
Additional Resources

- Implementing Accessor
- Setting up TypeScript
- Widget Development JS
- API SDK
The source code!

esriurl.com/customwidgetsuc2017
Please Take Our Survey on the Esri Events App!

Download the Esri Events app and find your event

Select the session you attended

Scroll down to find the survey

Complete Answers and Select “Submit”
Questions?