Building Your Own Widget with the ArcGIS API for JavaScript

Matt Driscoll – @driskull
JC Franco – @arfncode
Agenda

- Set up dev environment
- Create a...
  - Custom Class
  - Simple Widget
  - Custom Widget
- Enhance Custom Widget
Setting up the Dev Environment
Developer environment

JS API + TypeScript
TypeScript
interface Person {
    name: string;
    age: number;
}

var person: Person = { name: "Franco", age: 33 };

person.age = "24"; // TS2322: Type '"24"' is not assignable to type 'number'
person.height = 5.11; // TS2339: property 'height' does not exist on type 'Person'
// let and const
let canChange = 5;
const cannotChange = 5;

// fat arrow functions
const logName = (person) => console.log(person.name);

// template strings
const greeting = `Hello, my name is ${person.name} and I am ${person.age} years old.`;

// destructuring
const { name, age } = person;

// property shorthand
const shorthand = { person };
IDE Support

- Visual Studio
- WebStorm
- Sublime Text
- and more!
Demo: Dev Environment

- Install TypeScript + JS API
Demo Recap: Dev Environment

- Installed TypeScript + JS API typings
- Built mapping application
Creating a Class
esri/core/Accessor

- JavaScript API foundation
- Consistent developer experience

```javascript
// unified object constructor
const me = new Person({ name: "Franco", age: 33 });

// watch for changes to `age`
me.watch("age", singHappyBirthday);
```
// fetches webmaps from a portal and provides APIs to work with them
interface CustomClass {
  // used to fetch webmaps items
  portal: Portal;
  webMapGroupId: string;

  // active webmap and all fetched ones
  readonly active: PortalItem;
  readonly webMaps: PortalItem[];

  // will be updated with the active webmap
  view: MapView;

  // moves to the next webmap
  next(): void;
}
Demo Recap: Custom Class

- Implemented **CustomClass**
  - Extended **esri/core/Accessor**
  - Created properties with `@property`
  - Typed constructor arguments
  - Created public + private methods
Writing a Widget
About Widgets

• What?
  ▪ Encapsulated UI components
  ▪ Cohesive (integrated, unified)
  ▪ Single-purpose pieces of functionality

• Why?
  ▪ Reusable
  ▪ Interchangeable

• How?
  ▪ Extend esri/Widgets/Widget
esri/widgets/Widget

- Base widget class (View)
- Extends esri/core/Accessor
  - Properties
  - Watching properties
- Lifecycle
Lifecycle

- constructor
- postInitialize
- render
- destroy
**render**

- Defines UI
- Reacts to state changes
- Uses JSX (VDOM)
Demo: Simple View

Write simple widget
Demo Recap: Simple View

- Extended `esri/widgets/Widget`
- Implemented `render()`
- Added a `renderable()` property
- Added `onclick` event
- Added CSS Object + BEM Methodology
- Toggled property with event to re-render
Improving Our Widget
Architecture

- Separation of concerns
  - Views + ViewModels
  - UI replacement
  - Easier integration
Views

• Extend `esri/widgets/Widget`
• Rely on ViewModel
• Focus on UI
ViewModels

- Extend `esri/core/Accessor`
- Provide APIs to support View
- Focus on business logic
View + ViewModel in action

- View renders the state of the VM
  - Looks at properties on VM and renders accordingly
- User interacts with View (property/method)
  - Causes a change on VM or View
- View updates
  - Renders again due to changes on VM
Demo: Updated View

- Create `WebMapShowCase` to use `CustomClass` as VM
- Render details from the `active` portal item
Demo Recap: Update View

- Paired View and ViewModel
- Rendered property from ViewModel
- Wired up interactivity
- Learned to apply styles
- Dynamically rendered UI based on a property value
Going Further
Going Further

- Improve UX
- Internationalization (i18n)
Demo: Going Further (UX)

- Auto-cycle through webmaps
- Show timer in UI
- Support play/pause
Demo: Going Further (i18n)

- Localize widget
Demo Recap: Going Further

- Improved UX
- Added support for extra locale
Final Recap

- Setup dev environment
- Wrote a custom class
- Developed a custom Widget
- Enhanced a Widget
- Went further
Additional Resources

- Implementing Accessor
- Setting up TypeScript
- Widget Development
- JavaScript API SDK
- Styling
- Widget Patterns
Please Take Our Survey on the App

Download the Esri Events app and find your event

Select the session you attended

Scroll down to find the feedback section

Complete answers and select “Submit”
Where can I find the slides/source?

esriurl.com/developwidgetsuc2018