INTRO TO UI-ROUTER + TYPESCRIPT

Jamal O'Garro
Software Engineer

TALK I: UI-ROUTER

WHAT IS UI-ROUTER?

- Routing framework for AngularJS
- Fully replaces the default Angular router
- Created by AngularUI team (UI-Bootstrap, NG-Grid, etc.)

WHY USE UI-ROUTER?

- Multiple Views
- State Machine
- Nested Views

MULTIPLE VIEWS

- (Old) Angular router only allowed single views
- Forced the use of partials to simulate multiple views
- UI-Router allows multiple views
- Each view can have it's own controller

STATE MACHINE

- Uses state machine design pattern

- Routes are states and URLs are state properties

NESTED VIEWS

- Provides more flexibility when building an app
- Allows for more modular code
- Helps you build apps quicker

HOW DO WE DEFINE STATE?

- Must have a unique name
- Must have a template
- Controllers are optional
- URLs are optional

UI-VIEW

- Directive that tells $state where templates go
- Renders the view associated with a given state
- Kind of like ng-view but not really . . .

UI-SREF

- binds anchor tag to a specific state

NESTED STATES

- Parent state must exist
- No two states can have the same name
- When a child state is active so is the parent
- Child states load their templates into the paren't ui-view directive
- Inherit resolved dependencies and data properties
  - Controllers, templates, URLs are not inherited

MULTIPLE NAMED VIEWS

- Views object - allows us to add several views, templates and controllers to a given state

LET'S LOOK AT SOME CODE!

TALK II: TYPESCRIPT

WHAT IS TYPESCRIPT?

- Superset of JavaScript
- JavaScript w/ Strong Typing + Other features
- Made by Microsoft
- Will be supported in Angular 2.0

HOW DOES IT LOOK?

```
class Car {
  engine: string;

  constructor(engine: string) {
    this.engine = engine;
  }
}

var m6 = new Car('V8');
console.log(m6);

// Adding function to prototype
class UsedCar {
  engine: string;

  constructor(engine: string) {
    this.engine = engine;
  }

  start() {
    return "Started " + this.engine;
  }
}
```

STRONG TYPING

```
var isSelected: boolean = true;
```

CLASSES

```
class Car {
  engine: string;

  constructor(engine: string) {
    this.engine = engine;
  }
}
```

INHERITANCE

```
class ManlyTruck extends Auto {
  bigTires: boolean;
  constructor(engine: string, bigTires: boolean) {
    super(engine);
    this.bigTires = bigTires;
  }
}
```

MODULES

```
module Shapes {
  class Rectangle {
    constructor(public height: number, public width: number) {
    }
  }

  var rect1 = new Rectangle(10, 4);
}
```

INTERFACES

```
interface ICar {
  engine: string;
  color:string;
}

class NewCar implements ICar {
  constructor(public engine:string, public color:string) {
  }
}
```

HOW TO GET STARTED

- typescriptlang.org
- NPM install -g typescript
- Visual Studio / Webstorm / Sublime

RESOURCES

- Gulp
- Official TypeScript Handbook
- Official TypeScript Tutorial
- TypeScript Playground

THANK YOU!

Jamal O'Garro
Software Engineer