Pittsburgh TechFest

Intro to Angular

Richard Ashkettle

Technical Architect - TeleTracking

Email: richard.ashkettle@gmail.com
twitter: @rick_ashkettle
web: ashkettle.com

What is Angular?

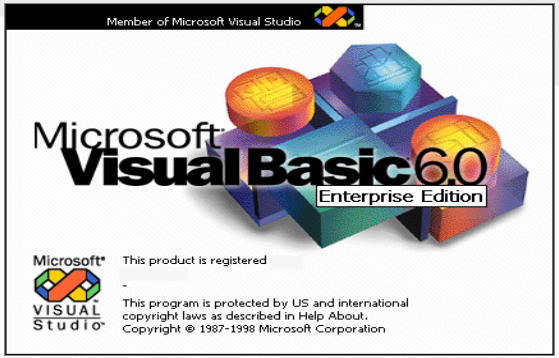
MV* Framework

Not particularly opinionated

Data Binding

Allows you to do more with less code

Easy for new team members to pick up the basics



Philosophy

AngularJS is built around the belief that declarative programming should be used for building user interfaces and connecting software components, while imperative programming is better suited to defining an application's business logic.

Bootstrapping

Most commonly as ng-app directive (declaring root element)

Spins up new Injector

Builds and Links Directives

Module

- Simply a Container
  - Allows for organization of code and injector configuration

```
var myModule = angular.module('myModule', []);

// add some directives and services
myModule.service('myService', ...);
myModule.directive('myDirective', ...);
```

$Scope

Very confusing term

Scope is really the data.

$Scope has scope, thus $Scope can be out of scope.

Use Controller As pattern to avoid this abomination.

Controllers

A constructor function used to store $Scope

```
myApp.controller('GreetingController', ['$scope', function($scope) {
  $scope.greeting = 'Hola!';
}]);
```

Logic should be in Services, not in the controller.

Directives

DOM element markers that allow scripts to be run in the template.

These appear as declarative elements, attributes or classes.

ng-bind, ng-repeat are some typical examples..

This was the beginning of the modern web

Templates

Pretty much what you expected

```
<html ng-app>
 <!-- Body tag augmented with ngController directive  -->
 <body ng-controller="MyController">
  <input ng-model="foo" value="bar">
  <!-- Button tag with ng-click directive, and
       string expression 'buttonText'
       wrapped in "{{ }}" markup -->
  <button ng-click="changeFoo()">{{buttonText}}</button>
  <script src="angular.js">
 </body>
</html>
```

Services

This is where your true logic should be.

Many provided by Angular ($http for instance)

Lazy instantiation

Singletons - yes, these are all singletons

Seriously easy to mock and test

Routers?

Angular has the worst Router known to modern man

However the Angular-UI router can be used and is a quite viable routing solution.

Pros

Easy to get started on initially

Directives

2 way binding

Easily Mockable Services

Filters

Built-in templating engine that really works

Cons

Digest Cycle inefficiencies (slow)

Overuse of Singletons causes bad architecture

Angular projects tend to have their own soup

Hard to really know well

$Scope is a mess (controller as fixes this somewhat)

2 way binding

Q&A

Please feel free to contact me if you have any additional questions.