

# **<angular/>**

## JavaScript Done Right

---

Elliott Sprehn (Google, Inc.)

[esprehn@gmail.com](mailto:esprehn@gmail.com)

<http://www.elliottsprehn.com/blog/>

<http://www.elliottsprehn.com/cfbugs/>

<http://www.angularjs.org/>

## ColdFusion rocks

---

- Easily create HTML templates
- Power for heavy lifting when you need it
- Solves the hard problems for me
- Dependency injection frameworks
  - ColdSpring, Lightwire, DI/1
- MVC frameworks
  - ColdBox, ModelGlue, Mach-ii, FW/1, ...

# Flex is sweet

---

- Flexible compiler
  - Custom metadata
- Two way data binding
- Reusable components
- Dependency injection frameworks
  - Swiz, SmartyPants, Spring Action Script
- MVC frameworks
  - PureMVC, Mate, Parsley, ...

# Development in JavaScript is getting better...

---

- Mature JS libraries
  - jQuery
  - Mootools
  - ...
- Powerful UI frameworks
  - jQuery UI
  - ExtJS
  - YUI
  - ...

... but, it could be awesome.

---

**<angular />**

# Declarative markup

---

- **ng:include**
  - Include HTML partials like cfinclude.
- **ng:repeat**
  - Loop over collections like cfloop.
- **ng:show / ng:hide**
  - Show or hide nodes based on a condition.
- **ng:class**
  - Add or remove classes from nodes.
- **ng:click / ng:change**
  - Handle user interactions with the page.

# Expression evaluation and filters

---

- **Expressions**

```
{{user.name}}
```

- **Filters**

```
{{user.dateRegistered | date: 'MM/yyyy'}}
```

- **Extensible (CFML on the client?)**

```
#user.name#
```

## Two way data binding

---

- **Form controls**

```
<input name="echo"> {{echo}}
```

- **Automatic evaluation**

```
<button ng:click="value = value + 1">
{{value}}
```



# Model-View-Controller

---

```
<form ng:controller="ProfileController" ng:submit="save()">
  <h2>Hello, {{user.name}}.</h2>
  <p>You can edit your profile details below.</p>
  <label>Email <input name="user.email"></label>
  <label>Address <textarea name="user.address"></textarea></label>
  <input type="submit" value="Save">
</form>
```

```
<script>
  function ProfileController() {
    this.user = {name: 'Elliott', email: 'esprehn@gmail.com'};
    this.save = function() {
      // Ajax request to save the user
    };
  };
</script>
```

# Dependency Injection

---

- Automatic injection

```
function ProfileController($resource, activityService_) { ... };
```

- Explicit injection

```
function ProfileController(resource, activityService) { ... };  
ProfileController.$inject = ['$resource', 'activityService'];
```

# RESTful resources

---

- Like ORM for REST
- Supports caching, bulking and JSONP
- Mocks provided for testing

```
<script>
  // $resource is automatically injected by name.
  function ProfileController($resource) {
    var Users = $resource('/users/me');
    this.user = Users.get();
    this.save = function() {
      Users.save(this.user);
    };
  };
</script>
```

# Service abstraction

---

- Define services for use in your controllers
- Easily swapped out for testing

```
angular.service('profileService', function($resource) {  
    return $resource('profiles/:id');  
});
```

```
function ProfileController(profileService_) {  
    // ...  
};
```

# Create your own ~~custom tags~~ widgets

---

- Define your own widgets

```
<app:ProfileEditor profile="user">  
</app:ProfileEditor>
```

- Reusable Components

```
<jqui:Button icon="ui-icon-gear">Click Me</jqui:Button>
```

- Attributes

```
<input name="tags" jqui:autocomplete="tagList">
```

# No more global state

---

- Controllers
- Services
- **Scopes**

```
<form ng:controller="ProfileController" ...>
  <h2>Hello, {{user.name}}.</h2>
  ...
  <div ng:repeat="comment in user.recentComments">
    <h4>
      {{comment.parentPost.title}}
      [<span ng:click="hideComment(comment)">Remove</span>]
    </h4>
    <p>{{comment.text}}</p>
  </div>
</form>
```

## Testability

---

- No DOM manipulation in controllers
- Mocks provided for XHR
- Easily mock out services
  - Angular services are not special
- JsTD and Jasmine integration

# End-to-End Runner

---

```
<script>
  describe('ProfileController', function() {
    it('should save profiles', function() {
      browser().navigateTo('/profiles/mine');
      expect(element('h2').text()).toEqual('Hello, Elliott. ');
      input('firstname').enter('Ethan');
      element('#save').click();
      browser().reload();
      expect(element('h2').text()).toEqual('Hello, Ethan. ');
    });
  });
</script>
```



# End-to-End DSL

---

```
<script>
  describe('ProfileController', function() {
    it('should save profiles', function() {
      browser().navigateTo('/profiles/mine');
      expect(profilePage().greeting()).
        toEqual('Hello, Elliott. ');
      profilePage().firstName().enter('Ethan');
      profilePage().save();
      browser().reload();
      expect(profilePage().greeting()).
        toEqual('Hello, Ethan. ');
    });
  });
</script>
```

# Questions

---

