AngularJS Jump Start

SECTION 1:AngularJS - The Big Picture

08. The big picture

The Big Picture

Module
Config
Routes
View — $scope — Controller
Directives
*Factory

07. What are views

Controller
$scope
View
Directives
Filters

Views render the user interface:

- Contain HTML and CSS
- Bind to data provided by a controller via the $scope object
- Use directives to enhance HTML and render data

View — $scope — Controller

$scope is the "glue" (ViewModel) between a controller and a view

06. What is scope

05. What are controllers

View
$scope
Controller
Factory
Service

Controllers act as the "brain" for a view:

- Retrieve data from a factory/service and store it
- Handle events triggered by the view
- Know how to handle custom logic
- Rely on the $scope object to interact with the view

04. What are factories and services

Factory
Service

Factories and Services:

- Used to make RESTful calls
- Used to share data between controllers
- Used to handle custom logic
- Are Singletons

03. Components

Module

components you build for your applications

Modules are containers

Module

Modules are containers for:

- Controllers
- Routes
- Factories/Services
- Directives
- Filters

Module

Routes

UI		Logic/Data
View	$scope	Controller
Directives		Factory
Filters		Service

02.

- Use AngularJS directives to add data binding functionality into a page:

```
Name: <input type="text" ng-model="name" />{{ name }}
```

Data Binding Expression

AngularJS Directive

01. Steps to get started with AngularJS

- **Steps to get started with AngularJS:**
  1. Reference the AngularJS script in an HTML page
  2. Add an **ng-app** directive
  3. Bind data using directives

SECTION 2:Views, Directives and Filters

AngularJS Architecture Pattern

View — $scope — Controller

- **AngularJS relies on two key architecture patterns:**
  - Model-View-Controller (MVC)
  - Model-View-ViewModel (MVVM)

MV + MVVM = MV*

MVC + MVVM = MV*

Model
Controller
$scope
View

Request
Response

The role of $scope

- **$scope is "injected" into a controller**
- **Acts as the ViewModel**
- **Views bind to scope properties and functions**

AngularJS controllers

- **Controllers act as the "brain" for a view:**
  - Defines properties and methods
  - Handles showing/hiding controls and data in a view
  - Handles events triggered by a view
  - Knows how to retrieve data
  - Interacts with the view using the $scope object

Data Binding

Directives and Expressions

Filters and Pipes

- **AngularJS can "pipe" data through filtering, formatting, and sorting functions**
- **Uses the | character**

example: 9.9912 → Currency Filter → $9.99

key filters used in AngularJS

- currency
- date
- filter
- json
- limitTo
- lowercase
- number
- orderBy
- uppercase

Sorting and formatting data

```
<ul>
  <li data-ng-repeat="cust in customers | orderBy:'name'">
    {{ cust.orderTotal | currency }}
  </li>
</ul>
```

Format as currency

Order customers by name property

SECTION 3:Controllers, Scope and Modules

creating a controller

```
<div class="container" data-ng-controller="SimpleController as ctrl">
  <h1>Adding a Simple Controller</h1>
  <ul>
    <li data-ng-repeat="cust in ctrl.customers">
      {{ cust.name }} - {{ cust.city }}
    </li>
  </ul>
</div>

<script>
  function SimpleController() {
    this.customers = [
      { name: 'Dave Jones', city: 'Phoenix' },
      { name: 'Jamie Riley', city: 'Atlanta' },
      { name: 'Heedy Wahlin', city: 'Chandler' },
      { name: 'Thomas Winter', city: 'Seattle' }
    ];
  }
</script>
```

Server Header Response

```
<script>
  function SimpleController($scope) {
    $scope.customers = [
      { name: 'Dave Jones', city: 'Phoenix' },
      { name: 'Jamie Riley', city: 'Atlanta' },
      { name: 'Heedy Wahlin', city: 'Chandler' },
      { name: 'Jeff James', city: 'Seattle' }
    ];
  }
</script>
```

Using controller as

What is a module

Modules are containers for:

- Controllers
- Routes
- Factories/Services
- Directives
- Filters

Module

Modules are Containers

Modules are Containers

```
<html ng-app="moduleName">
```

Module
Config | Filter | Directive | Factory / Service | Controller
Routes

Defining a module

Defining a Module

- **Create a module using angular.module():**

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

Dependencies

Injecting dependencies into a Module

Injecting Dependencies into a Module

- **Modules may rely on functionality from other modules**
- **Helper modules can be "injected" into a module:**

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

External module

Adding controller to a module : Option 1

Adding a Controller to a Module: Option 1

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

demoApp.controller('SimpleController', function ($scope) {
  $scope.customers = [
    { name: 'Dave Jones', city: 'Phoenix' },
    { name: 'Jamie Riley', city: 'Atlanta' },
    { name: 'Heedy Wahlin', city: 'Chandler' },
    { name: 'Thomas Winter', city: 'Seattle' }
  ];
});
```

Define a Module

Define a Controller

Option 1

Adding a Controller to a Module: Option 2

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

angular.module('demoApp').controller('SimpleController',
function ($scope) {
  $scope.customers = [
    { name: 'Dave Jones', city: 'Phoenix' },
    { name: 'Jamie Riley', city: 'Atlanta' },
    { name: 'Heedy Wahlin', city: 'Chandler' },
    { name: 'Thomas Winter', city: 'Seattle' }
  ];
});
```

Reference module

Define a Module

Option 2

Adding a Controller to a Module: Option 3

```
var demoApp = angular.module('demoApp', []);
...
(function() {
  var SimpleController = function ($scope) {
    $scope.customers = [
      { name: 'Dave Jones', city: 'Phoenix' },
      { name: 'Jamie Riley', city: 'Atlanta' }
    ];
  };

  angular.module('demoApp')
    .controller('SimpleController',SimpleController);
})();
```

Option 3

dealing with minification of $scope

```
angular.module('demoApp')
  .controller('SimpleController', ['$scope', function ($scope) {
    $scope.customers = [...];
  }]);
```

Option 1

```
var SimpleController = function ($scope) {
  $scope.customers = [...];
};

SimpleController.$inject = ['$scope'];

angular.module('demoApp')
  .controller('SimpleController', SimpleController);
```

Option 2

SECTION 4:Routing

The ngRoute module

- **AngularJS routing functionality is located in the ngRoute module**
- **Add a <script> that loads angular-route.js**
- **Reference ngRoute in your module:**

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

Routing and modules

- **Routes configured by calling angular.module.config()**
- **$routeProvider injected dynamically**

$routeProvider
when '/customers' → customers.html / customersController
when '/orders' → orders.html / ordersController

Routing overview

- **AngularJS routes associate a view with a controller**

/customers → customers.html / customersController

- **Relies on ngRoute module (separate script)**
- **Routes configured using $routeProvider**

Using $routeProvider to configure routes

Using $routeProvider to Configure Routes

```
var app = angular.module('customersApp', ['ngRoute']);

app.config(function ($routeProvider) {
  //Routes go here
});
```

$routeProvider injected dynamically

```
var app = angular.module('customersApp', ['ngRoute']);

app.config(function ($routeProvider) {
  $routeProvider
    .when('/',
      {
        controller: 'CustomersController',
        templateUrl: '/app/views/customers.html'
      })
    .when('/orders',
      {
        controller: 'OrdersController',
        templateUrl: '/app/views/orders.html'
      })
    .otherwise({ redirectTo: '/' });
});
```

Otherwise

3 things you have to specify

1. Path
2. Controller
3. View

Route parameters

- **Route parameters can be defined using the : character**

```
app.config(function ($routeProvider) {
  $routeProvider
    .when('/editCustomer/:customerId',
      {
        controller: 'CustomerEditController',
        templateUrl: '/app/views/customerEdit.html'
      });
});
```

Route parameter

Example Route: /editCustomer/254

Accessing route parameters in a Controller

- **Route parameters can be accessed through the $routeParams provider:**

```
var app = angular.module('customersApp');

app.controller('CustomerEditController',
  function($scope, $routeParams) {

  $scope.customerId = $routeParams.customerId;

});
```

Route Parameter operator

Using the ng-view directive

Dynamically loaded views are injected into the shell page as a module loads:

```
<div ng-view></div>
OR
<ng-view></ng-view>
```

View1

Summary

- **AngularJS provides built-in support for routing and history management**
- **Routes associate a view with a controller**
- **Reference the ngRoute module**
- **$routeProvider injected into angular.module.config()**

SECTION 5:Factories and Services

Factories and services

- **AngularJS supports the concept of factories and services**
- **Includes several built-in factories/services**
- **Singletons that perform re-useable tasks:**
  - Ajax calls
  - Business rules
  - Calculations
  - Share data between controllers

Creating factories, services and more

Module
Config | Filter | Directive | Factory / Service / Constant / Value | Controller
Routes

Build-in AngularJS services (most commonly used)

- **AngularJS provides many built-in services:**

$http	$timeout	$window
$location	$q	$rootScope
$interval	$filter	$log

- $http — ajax call
- $timeout — timeout (same as window.setTimeout)
- $window — window object
- $location — URL
- $q — asynchronous call and ajax
- $rootScope — create new scope
- $interval — set interval (repeating call after some time)
- $filter — programmatically to get custom filters
- $log — for logging

Creating a factory

- **Create a custom factory when you need to:**
  - Define re-useable tasks
  - Share code or state between controllers
- **Factories:**
  - Create and return a custom object
  - Created using the module.factory() function
  - Can be injected into other components
  - Can have dependencies

```
(function () {
  var customersFactory = function () {
    var customers = [...];

    return {
      getCustomers: function () {
        return customers;
      }
    };
  };

  angular.module('customersApp')
    .factory('customersFactory', customersFactory);
})();
```

alternate syntax

```
(function () {

  var customersFactory = function () {
    var customers = [...];
    var factory = {};
    factory.getCustomers = function () {
      return customers;
    };
    return factory;
  };

  angular.module('customersApp')
    .factory('customersFactory', customersFactory);

})();
```

Return code^{ed} object

What is a factory

What is a service

- **Similar to a factory as far as functionality**
- **Services:**
  - Service function represents the returned object as opposed to a custom object like in a factory
  - Created using the module.service() function
  - Can be injected into other components
  - Can have dependencies

Creating a Service

```
(function () {

  var customerService = function () {
    var customers = [ ... ];

    this.getCustomers = function () {
      return customers;
    };
  };

  angular.module('customersApp')
    .service('customerService', customerService)
});
```

Function represents the singleton object

Refers to this

Add service to module

Creating a service

Defining application values

- **An value can be created using module.value(key,value):**

```
angular.module('customersApp').value('version', '1.0');
```

Value operator

- **A constant can be created using module.constant(key,value):**

```
angular.module('customersApp').constant('version', '1.0');
```

Constant operator

Value and Constant

Services used to make ajax call

- **AngularJS services can be used to make Ajax calls**

$http | $resource

Making Ajax call from Factory / Service

- **The $http service provides Ajax functionality**
- **Uses the browser's XmlHttpRequest object**
- **Makes asynchronous requests**
- **Supports multiple HTTP verbs:**
  - $http.head
  - $http.get
  - $http.post
  - $http.put
  - $http.delete
  - $http.jsonp

AJAX

The $http service

```
(function () {

  var customersFactory = function ($http) {
    var factory = {};
    factory.getCustomers = function () {
      return $http.get('api/customers');
    };
    return factory;
  };

  angular.module('customersApp')
    .factory('customersFactory', customersFactory);

})();
```

Service injected

Make async call

Using the $http service

- **$http makes asynchronous calls:**
  - Relies on $q service's (deferred/promise APIs
  - Access data by calling .then() or .success()/.error()

```
customersFactory.getCustomers()
  .success(function (customers) {
    $scope.customers = customers;
  })
  .error(function (data, status, headers, config) {
    $log.warn('Error');
  });
```

Callback operator

Accessing $http response data

SECTION 6:UI and Animation

Directives and animation

Common Directives	Custom Directives
• ngClass • ngHide/ngShow • ngInclude • ngRepeat • ngSwitch • ngView	• Sanimate service

Directives and Supported Animations

Directive	Supported Animations
ngClass	Add and remove
ngHide/ngShow	Add and remove
ngInclude	Enter and leave
ngRepeat	Enter, leave, move
ngSwitch	Enter and leave
ngView	Enter and leave

The ngAnimate module

- **AngularJS animation functionality is located in the ngAnimate module**
- **Add a <script> that loads angular-animate.js**
- **Reference ngAnimate in your module:**

```
var demoApp = angular.module('customersApp', ['ngAnimate']);
```

Animation Techniques

- **Animations can be defined multiple ways using CSS:**
  - CSS3 Transitions

```
transition: 0.5s linear all;
```

  - CSS3 Animations

```
animation: 0.5s name-of-keyframe-animation;
```

Defining Animation with CSS

Animation Techniques

Animation Techniques
- Using CSS3 Transitions
- Using CSS3 Animations
- Using JavaScript Animations