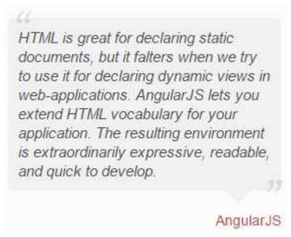
AngularJS by Google

Greenhorn

Darryl Ng
issnes@nus.edu.sg
Institute of System Science
National University of Singapore

What is AngularJS?

- Frontend JavaScript MVVM based framework
  - Developed by Google
  - Open source, available at http://angularjs.org/
- Encourages good 'software engineering' development style in SPA
  - Goal is to develop business applications
  - Lots of modules/plugins



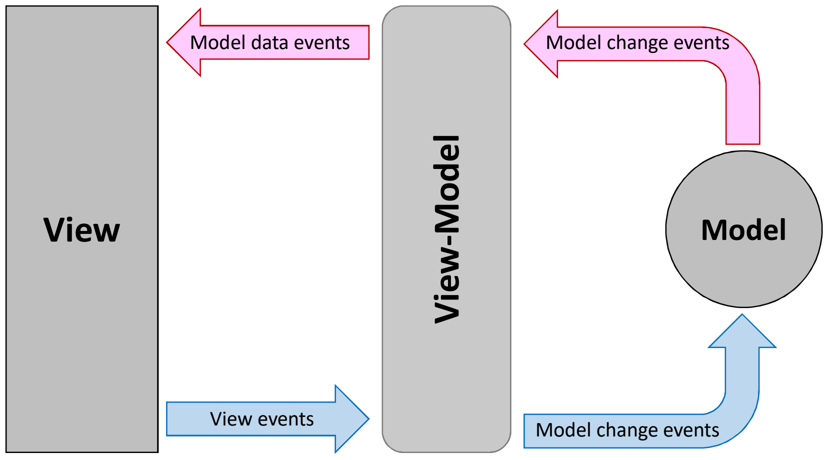
"HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop."

AngularJS

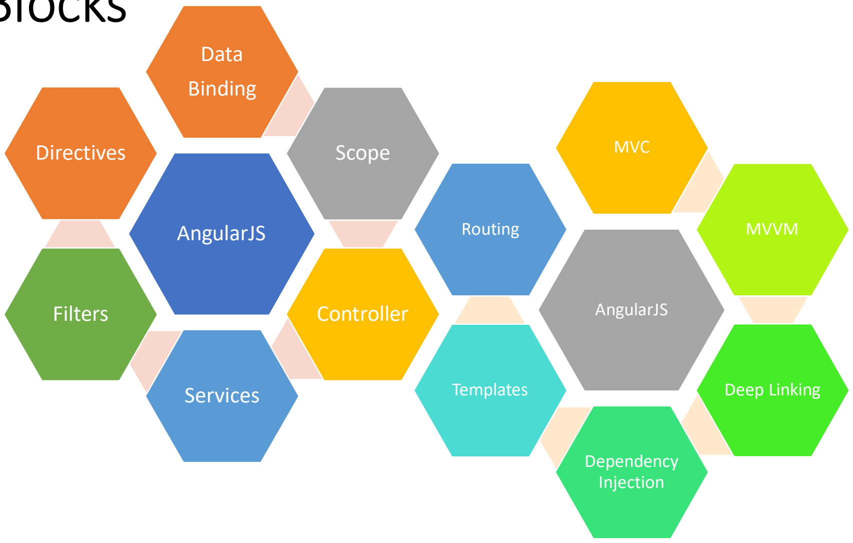
Features Overview

- Based on MVVM
  - Clear separation of model, view and view-models
- Bi directional binding
  - Models can be bind to views
  - Changes in the views are propagated to the model and vice versa
- Dependency injection
  - Many Angular and user objects are available through injection
- Service and factories abstraction
  - Calls to RESTful web services and other operations can be represented as services and factories
- Developed with testing

Model View View-Model



Key Blocks

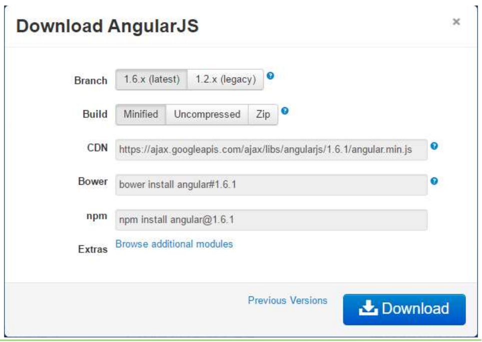


Pros and Cons

✓SPA, clean and maintainability
✓Rich and responsive data binding capability to HTML
✓Unit testable
✓Dependency injection and separation of concerns
✓Reusable components
✓Achieve more with less coding
×Security
×Degradable

Dev Environment

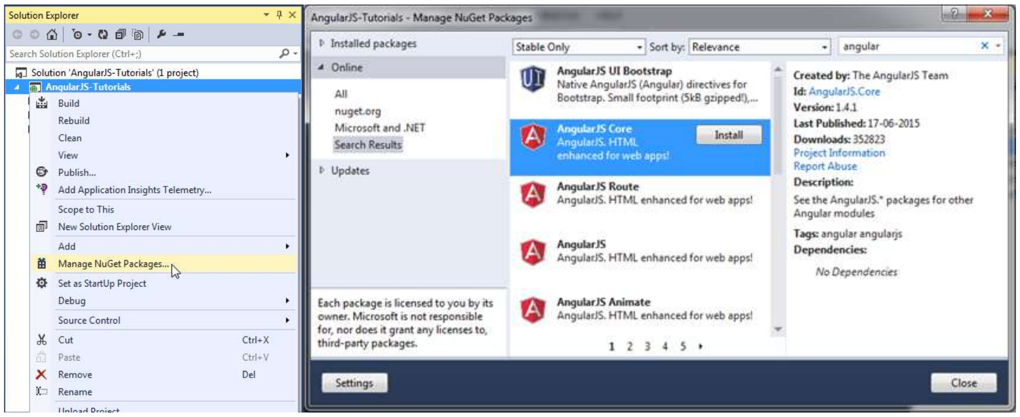
- AngularJS Library
  - https://angularjs.org/
  - https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js
- Editor/IDE
  - Sublime Text, Notepad++, Eclipse, Visual Studio
- Browser
  - Chrome, Firefox
- Web server
  - IIS, Apache


Download AngularJS

Branch	1.6.x (latest) 1.2.x (legacy)
Build	Minified Uncompressed Zip
CDN	https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.min.js
Bower	bower install angular#1.6.1
npm	npm install angular@1.6.1
Extras	Browse additional modules

Visual Studio AngularJS project

- AngularJS library from NuGet package manager



AngularJS UI Bootstrap Native AngularJS (Angular) directives for Bootstrap. Small footprint (5kB gzipped!),...

AngularJS Core AngularJS. HTML enhanced for web apps!

AngularJS Route AngularJS. HTML enhanced for web apps!

AngularJS AngularJS. HTML enhanced for web apps!

AngularJS Animate AngularJS. HTML enhanced for web apps!

Created by: The AngularJS Team
Id: AngularJS.Core
Version: 1.4.1
Last Published: 17-06-2015
Downloads: 352823
Project Information
Report Abuse
Description:
See the AngularJS.* packages for other Angular modules
Tags: angular angularjs
Dependencies:
No Dependencies

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

HTML5 Support

- AngularJS application is denoted by **`the data-ng-app`** attribute
  - All elements under the `data-ng-app` element under the control of AngularJS
    - All `data-ng-*` attribute are said to be AngularJS directive
  - The boundary of an AngularJS application can be page or a form within that page
  - use `data-ng-app="..."` instead of ng-app because `data-*` is a valid HTML5 syntax

```
<form method="POSTS" action="process" data-ng-app="">

<form method="POSTS" action="process" ng-app>
```

AngularJS Application Overview

- Applications are defined as a module
  - A module have a name – the application's name
  - Module can have dependencies
- Modules can include the following
  - One or more controllers
  - Configuration typically routes
  - Services
  - Factories
- Specify what dependencies to inject

Directives

- Compile and link functions
- Template
- Replace
- Scoe
- Controller
- Priority
- Restrict

Directives - 1

- **`ng-app`** – starts an AngularJS Application to HTML
- **`ng-init`** – initializes application data
- **`ng-model`** – defines the model that is variable to be used in AngularJS Application to HTML input controls
- **`ng-repeat`** – repeats html elements for each item in a collection
- **`ng-bind`** – binds the AngularJS Application data to HTML tags
- **`ng-controller`** – attaches the controller of MVC to the view

Directives - 2

- **ng-show** – display HTML element based on the value of the specified expression
- **ng-readonly** – makes HTML element read-only based on the value of the specified expression
- **ng-disabled** – sets the disable attribute on the HTML element if specified expression evaluates to true
- **ng-if** – removes/recreates HTML element based on an expression
- **ng-click** – specifies custom behavior when an element is clicked

Directive Syntax

- applied to DOM element
- non-mandatory to use ng- syntax only
- start with **x-** or **data-**
- **-** can be replaced with **:** or **_** or both
- mix with **data-** or **x-**

```
<!DOCTYPE html>
<html>
  <head>
    <script src="~/Scripts/angular.js"></script>
  </head>
  <body ng-app>
    Enter Name: <input type="text" ng-model="name" /> <br />
    data-ng-bind: <span data-ng-bind="name"></span><br />
    data-ng:bind: <span data-ng:bind="name"></span><br />
    data:ng:bind: <span data:ng:bind="name"></span><br />
    x:ng:bind: <span x:ng:bind="name"></span><br />
    ng:bind: <span ng:bind="name"></span><br />
    x-ng-bind: <span x-ng-bind="name"></span><br />
    x_ng_bind: <span x_ng_bind="name"></span><br />
    ng_bind: <span ng_bind="name"></span>
  </body>
</html>
```

Overview of an AngularJS Application

```
<!DOCTYPE html>
<html>
<head>
  <script src="~/Scripts/angular.js">
  </script>
</head>
<body ng-app="MyApp">
  Enter Your Name:
  <input type="text" ng-model="name" />
  <br />
  Hello <label ng-bind="name"></label>
</body>
</html>
```

Include AngularJS library

Specify that everything under this node is an AngularJS application by the name of `MyApp`

Bind the input to the expression **{{ }}**

jQuery Equivalent

```
<!DOCTYPE html>
<html>
<head>
  <script src="~/Scripts/jquery-1.10.2.min.js"></script>
</head>
<body>
  Enter Your Name: <input type="text" id="txtName" />
  <br />
  Hello <label id="lblName"></label>

  <script>
    $(document).ready( function< () {
      $('#txtName').keyup(function () {
        $('#lblName').text($('#txtName').val());
      });
    });
  </script>
</body>
</html>
```

AngularJS Building Blocks

```
<!DOCTYPE html>
<html>
<head>
  <title>First AngularJS Application</title>
  <script src= "~/Scripts/angular.js"></script>
</head>
<body ng-app="MyFirstApp" >
  <h1>First AngularJS Application</h1>
  Enter Numbers to Multiply:
  <input type="text" ng-model="Num1" /> x <input type="text" ng-model="Num2" /> =
  <span>{{Num1 * Num2}}</span>
</body>
</html>
```

Template

Directives

Expression

JavaScript code executed inside brackets {{ }}.

ng-app Directive

- starting point of AngularJS Application
- automatically initializes the AngularJS framework
- placed at the root of an HTML document
  - e.g. **`<html>`** or **`<body>`** tag

```
<!DOCTYPE html>
<html>
  <head>
    <title>ng-app Directive</title>
    <script src="../Scripts/angular.min.js"></script>
  </head>
  <body>
    <div> {{2/2}} </div>
    <div id="myDiv" ng-app> {{5/2}}
      <div> {{10/2}} </div>
    </div>
    <div> {{2/2}} </div>
  </body>
</html>
```

Angular features not supported outside of ng-app

Angular features supported inside ng-app

Expression - 1

- JavaScript expression
- braces - {{ expression }}

✗cannot contain conditions, loops, exceptions or regular expressions
  - e.g. if-else, ternary, for loop, while loop etc

✗cannot declare functions

✗cannot contain comma or void

✗cannot contain return keyword

Expression - 2

```
<html ng-app>
  <head>
    <script src="~/Scripts/angular.js"></script>
  </head>
  <body>
    <h1>AngularJS Expression Demo:</h1>
    <div> {{"Hello World"}} <br />
          {{100}} <br />
          {{true}} <br />
          {{10.2}} <br />
          2 + 2 = {{2 + 2}} <br />
          2 - 2 = {{2 - 2}} <br />
          2 * 2 = {{2 * 2}} <br />
          2 / 2 = {{2 / 2}} <br />
          {{"Hello" + " World"}} <br />
          {{100 + 100 }} <br />
          {{true + false}} <br />
          {{10.2 + 10.2}} <br />
    </div>
  </body>
</html>
```

Expression - 3

```
<!DOCTYPE html>
<html>
  <head>
    <script src="~/Scripts/angular.js"></script>
  </head>
  <body>
    <div ng-app ng-init="greet='Hello World!'; amount= 10000;rateOfInterest = 10.5;
      duration=10; myArr = [100, 200]; person = { firstName:'Steve', lastName :'Jobs'}">
      {{ (amount * rateOfInterest * duration)/100 }}<br />
      {{myArr[1]}} <br />
      {{person.firstName + " " + person.lastName}}
    </div>
  </body>
</html>
```

Conditional Display - 1

- Show and Hide
- Switch
- If
- Include

Conditional Display - 2
Show and Hide

```
<!DOCTYPE html>
<html>
<head>
  <script src="angular.js"></script>
</head>

<body ng-app>
  <h3>1. Show</h3>
  <label>Show the square: <input type="checkbox" ng-model="mustShow" /></label><br />
  <div ng-show="mustShow" style="width: 50px; height: 50px; background-color: red;">
  </div> <br /> <br />
  <h3>2. Hide</h3>
  <label>Hide the square: <input type="checkbox" ng-model="mustHide" /></label><br />
  <div ng-hide="mustHide" style="width: 50px; height: 50px; background-color: green;">
  </div>
</body>
</html>
```

Conditional Display - 3
Switch

```
<!DOCTYPE html>
<html>
<head>
  <script src="angular.js"></script>
</head>

<body ng-app>
  <label>Type the number you want to show (1 to 5): <input type="text" ng-model="showNumber" />
  </label><br />
  <div ng-switch="showNumber">
    <div ng-switch-when="1" style="width: 50px; background-color: red; text-align: center;">1</div>
    <div ng-switch-when="2" style="width: 50px; background-color: green; text-align: center;">2</div>
    <div ng-switch-when="3" style="width: 50px; background-color: yellow; text-align: center;">3</div>
    <div ng-switch-when="4" style="width: 50px; background-color: fuchsia; text-align: center;">4</div>
    <div ng-switch-when="5" style="width: 50px; background-color: orange; text-align: center;">5</div>
    <div ng-switch-default style="width: 50px; background-color: lightgray; text-align: center;">None</div>
  </div>
</body>
</html>
```

ng-if, ng-readonly, ng-disabled

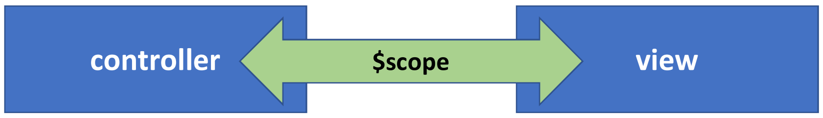
```
<!DOCTYPE html>
<html>
<head>
  <script src="~/Scripts/angular.js"></script>
  <style> div { width: 100%; height: 50px; display: block; margin: 15px 0 0 10px; } </style>
</head>
<body ng-app ng-init="checked=true" >
  Click Me: <input type="checkbox" ng-model="checked" /> <br />
  <div> New: <input ng-if="checked" type="text" /> </div>
  <div> Read-only: <input ng-readonly="checked" type="text" value="This is read-only." /> </div>
  <div> Disabled: <input ng-disabled="checked" type="text" value="This is disabled." /> </div>
</body>
</html>
```

Controller

- **`$scope`** object
- maintains application data and behavior
- attach properties and methods to the $scope object inside a controller function
- add/update the data and attach behaviors to HTML elements

$rootScope & $scope

- built-in object
- contains application data and methods
  - create properties to a $scope object inside a controller function
  - assign a value or function to it
- middleware between a controller and HTML
- Bi-directional data transfer between controller and view



$scope methods & events

`$new()`	Creates new child scope
`$watch()`	Register a callback to be executed whenever model property changes
`$watchGroup()`	Register a callback to be executed whenever model properties changes Specify an array of properties to be tracked
`$watchCollection()`	Register a callback to be executed whenever model object or array property changes
`$digest()`	Processes all of the watchers of the current scope and its children
`$destroy()`	Removes the current scope (and all of its children) from the parent scope
`$eval()`	Executes the expression on the current scope and returns the result
`$apply()`	Executes an expression in angular outside the angular framework
`$on()`	Register a callback for an event
`$emit()`	Dispatches the specified event upwards till $rootScope
`$broadcast()`	Dispatches the specified event downwards to all child scopes

Module

- AngularJS applications are compose of one or more modules
  - Module is an abstraction
  - Allows us to group related concerns/functions into modules
  - Follows naturally from the concept of an AngularJS application; natural for applications to be broken into modules
- Why use modules?
  - Code can be packaged for reuse
  - Modules can express dependency on other modules so they can be loaded correctly even if their files are not in the correct order
- You can define the following in a module
  - Controllers, services, factories, routes, etc.

Defining a Module

- A module is created using **`angular.module()`**
  - Takes 2 parameters
    - A module name
    - An array name of module in which the module is dependent on
- Use the following methods to register controllers, factories, services with the module
  - `controllers(),service(),factory()`
- Specify the name of the module in **`data-ng-app`** for the view to use that module

Defining a Module Example

Module name

No dependency

```
var MyApp = angular.module("MyApp", []);
```

```
<html data-ng-app="MyApp">
...
```

Module Example with Dependency

ctrls.js

```
var ctrls = angular.module("MyApp.Ctrls", []);
...
```

app.js

```
var myApp = angular.module("MyApp"
     , ["MyApp.Ctrls", "ngResource"]);
...
```

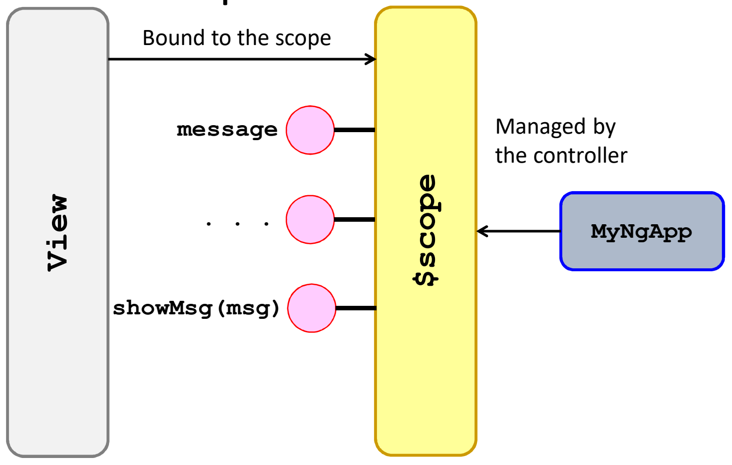
Dependency

```
<html data-ng-app="MyApp">

  <head>
    <script src="js/app.js"><script>
    <script src="js/ctrls.js"><script>
    ...
```

Loading order is unimportant

Controller and Scopes



Controller – 1 Example

1. Specify controller using ng-controller

2. Create an Angular App module

3. Create a controller

```
<!DOCTYPE html>
<html>
  <head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
  </head>
  <body ng-app="myNgApp">
    <div ng-controller="myController"> {{message}} </div>
    <script>
      var ngApp = angular.module('myNgApp', []);
      ngApp.controller('myController', function ($scope) {
        $scope.message = "Hello World!";
      });
    </script>
  </body>
</html>
```

5. Use property created inside controller

4. Attach a property to $scope object

Controller - 2
Attaching behaviours

```
<!DOCTYPE html>
<html>
  <head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
  </head>
  <body ng-app="myNgApp">
    <div ng-controller="myController">
      Enter Message: <input type="text" ng-model="message" /> <br />
      <button ng-click="showMsg(message)">Show Message</button>
    </div>
    <script>
      var ngApp = angular.module('myNgApp', []);
      ngApp.controller('myController', function ($scope) {
        $scope.message = "Hello World!";
        $scope.showMsg = function (msg) {
          alert(msg);
        };
      });
    </script>
  </body>
</html>
```

Controller - 3
Attaching object

```
<!DOCTYPE html>
<html>
  <head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
  </head>
  <body ng-app="myNgApp">
    <h2>Student Information:</h2>
    <div ng-controller="myController">
      First Name: {{student.firstName}} <br />
      Last Name: {{student.lastName}}
    </div>
    <script>
      var ngApp = angular.module('myNgApp', []);
      ngApp.controller('myController', function ($scope) {
        $scope.student = { firstName: 'James', lastName: 'Bond' };
      });
    </script>
  </body>
</html>
```

Modularize AngularJS Application

- Separation of JavaScript files for each module

myNgApp.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>AngualrJS Controller</title>
    <script src="~/Scripts/angular.js"></script>
  </head>
  <body ng-app="myNgApp">
    <h2>Student Information:</h2>
    <div ng-controller="myController">
      First Name: {{student.firstName}} <br />
      Last Name: {{student.lastName}}
    </div>
    <script src="app.js"></script>
    <script src="myController.js"></script>
  </body>
</html>
```

app.js

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

myController.js

```
ngApp.controller('myController', function ($scope) {
  $scope.student = {
       firstName: 'James',
       lastName: 'Bond'
  };
});
```

Events - 1

- **ng-blur**
- **ng-change**
- **ng-click**
- **ng-dblclick**
- **ng-focus**
- **ng-keydown**
- **ng-keyup**
- **ng-keypress**
- **ng-mousedown**
- **ng-mouseenter**
- **ng-mouseleave**
- **ng-mousemove**
- **ng-mouseover**
- **ng-mouseup**

```
<!DOCTYPE html>
<html>
  <head>
    <script src="~/Scripts/angular.js"></script>
  </head>
  <body ng-app="myApp">
    <div ng-controller="myController">
      Enter Password: <input type="password" ng-model="password" /> <br />
      <button ng-click="DisplayMessage(password)">Show Password</button>
    </div>
    <script>
      var myApp = angular.module('myApp', []);
      myApp.controller("myController", function ($scope, $window) {
        $scope.DisplayMessage = function (value) {
          $window.alert(value)
        }
      });
    </script>
  </body>
</html>
```

Events - 2

```
<!DOCTYPE html>
<html>
  <head>
    <script src="~/Scripts/angular.js"></script>
    <style> .redDiv { width: 100px; height: 100px;
background-color: red; padding:2px 2px 2px 2px; }
            .yellowDiv { width: 100px; height: 100px;
background-color: yellow; padding:2px 2px 2px 2px; }
    </style>
  </head>
  <body ng-app>
    <div ng-class="{redDiv: enter, yellowDiv: leave}"
         ng-mouseenter="enter=true;leave=false;"
         ng-mouseleave="leave=true;enter=false">
       Mouse <span ng-show="enter">Enter</span>
             <span ng-show="leave">Leave</span>
    </div>
  </body>
</html>
```

Forms

- Text Input
- CheckBox Input
- Radiobutton Input
- Select
- Track Changes
- Validation
- Nested Forms
- Form submission
- Form Reset

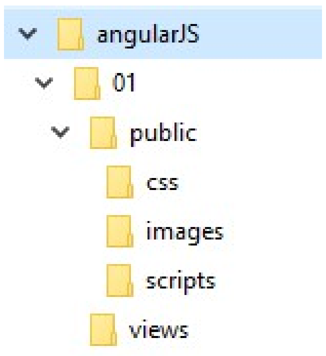
Forms – 1
Hosting using NodeJS (memberApp.js)

```
var express = require('express');
var app = express();
var path = require('path');
app.use(express.static(path.join(__dirname, 'public')));


// viewed at http://localhost:1337
app.get('/', function(req, res) {
    res.sendFile(path.join(__dirname + '/views/memberApp.html'));
});

app.listen(1337);

console.log("Running at Port 1337");
```



Forms – 2
HTML page (memberApp.html)

```
<!DOCTYPE html>
<html ng-app="memberApp">
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.js"></script>
</head>
<body ng-controller="memberController">
    <h1>Member Information:</h1>
    <form ng-submit="submitMemberForm()" >
        <label for="firstName" >First Name: </label><br />
        <input type="text" id="firstName" ng-model="member.firstName" /> <br />
        <label for="surName">Surname: </label><br />
        <input type="text" id="surName" ng-model="member.surName" /> <br />
        <label for="secondname" >Second Name:</label><br />
        <input type="text" id="secondName" ng-model="member.secondName" /> <br /><br />
        <input type="submit" value="Submit" />
        <input type="reset" ng-click="resetForm()" value="Reset" />
    </form>
    <script src="/../scripts/app.js"></script>
    <script src="/../scripts/memberController.js"></script>
</body>
</html>
```

Forms – 3 Declare Angular Module

app.js

```
//1. create app module
       var memberApp = angular.module('memberApp', []);
```

Forms – 4 - controller

memberController.js

```
//2. create controller
memberApp.controller("memberController", function ($scope, $http) {

    //3. attach originalMember model object
    $scope.originalMember = {
        firstName: 'Darryl',
        surName: 'Ng Eng Soon',
        secondName: 'Eddie'
    };

    //4. copy originalMember to member. memebr will be bind to a form
    $scope.member = angular.copy($scope.originalMember);

    //5. create submitStudentForm() function. This will be called when user submits the form
    $scope.submitMemberForm = function () {
        var onSuccess = function (data, status, headers, config) {
            alert('Member saved successfully.');
        };

        var onError = function (data, status, headers, config) {
            alert('Error occured.');
        }

        $http.post('/member/submitData', { student:$scope.member })
            .success(onSuccess)
            .error(onError);
    };

    //6. create resetForm() function. This will be called on Reset button click.
    $scope.resetForm = function () {
        $scope.member = angular.copy($scope.originalMember);
    };
});
```

Validation – 1
Directives

- client side validation

Directive	Description
ng-required	Sets required attribute on an input field
ng-minlength	Sets minlength attribute on an input field
ng-maxlength	Sets maxlength attribute on an input field. Setting the attribute to a negative or non-numeric value, allows view values of any length.
ng-pattern	Sets pattern validation error key if the ngModel value does not match the specified RegEx expression

Validation – 2
State Properties

Property	Description
$error	$error object contains all the validation attributes applied to the specified element.
$pristine	Returns true if the user has not interacted with control yet else returns false
$valid	Returns true if the model is valid
$invalid	Returns true if the model is invalid
$dirty	Returns true if user changed the value of model at least once
$touched	Returns true if the user has tabbed out from the control
$untouched	Returns true if the user has not tabbed out from the control

Validation – 3
Validated HTML page (memberApp.html)

```
<!DOCTYPE html>
<html ng-app="memberApp">
. . .
    <form name="memberForm" ng-submit="submitMemberForm()" novalidate>
        <label for="firstName">First Name: </label><br />
        <input type="text" id="firstName" ng-model="member.firstName"
                  ng-required="true"/>
        <span ng-show=" memberForm.firstName.$touched && memberForm.firstName.$error.required">
         First name is required.</span><br /><br />
        <label for="surName">Surname: </label><br />
        <input type="text" id="surName" ng-model="member.surName"
                  ng-required="true" ng-minlength="3" ng-maxlength="20" />
        <span ng-show="memberForm.surName.$touched &&
                  memberForm.surName.$error.minlength">min 3 chars.</span>
        <span ng-show="memberForm.surName.$touched &&
                  memberForm.surName.$error.maxlength">Max 10 chars.</span><br /><br />
        <label for="secondname" >Second Name:</label><br />
        <input type="text" id="secondName" ng-model="member.secondName"
                   ng-required="true" ng-minlength="20" ng-maxlength="40" /> <br /><br />
        <input type="submit" value="Submit" />
        <input type="reset" ng-click="resetForm()" value="Reset" />
    </form>
. . .
</html>
```

Routing - 1

- **ngRoute** routing module
- Routing rules
  - $Location service
  - $Route servcie
- url based
  - http://localhost/
  - http://localhost/member.
  - http://localhost/member/{id}
  - http://localhost/member/{firstname}

Routing – 2
Defining route

- Route Provider

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

memberApp.config(function ($routeProvider) {
            $routeProvider.when('/', {
                templateUrl: '/login.html',
                controller: 'loginController'
            }).when('/memberApp', {
                templateUrl: '/memberApp.html',
                controller: 'memberController'
            }).otherwise({
                redirectTo: "/"
            });
```

$\left.\vphantom{\begin{matrix}a\\a\\a\\a\\a\\a\\a\\a\\a\\a\\a\\a\end{matrix}}\right\}$ **app.js**

Routing - 3

loginController.js

```
memberApp.controller("loginController", function ($scope, $location)
{

            $scope.authenticate = function (username) {
                // write authentication code here..

                $location.path('/student/' + username)
            };

        });
```

Routing – 4
Corresponding HTML (login.html)

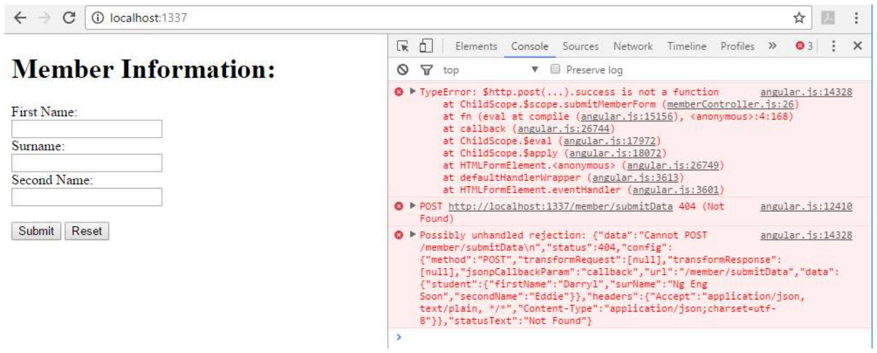
```
<form class="form-horizontal" role="form" name="loginForm" novalidate>
    <div class="form-group" >
        <div class="col-sm-3"></div>
        <div class="col-sm-6">
            <input type="text" id="userName" name="userName" placeholder="User Name" class="form-control" ng-
model="userName" required />
            <span class="help-block" ng-show="loginForm.userName.$touched && loginForm.userName.$invalid">Please enter
User Name.</span>
        </div>
        <div class="col-sm-3"></div>
    </div>
    <div class="form-group" >
        <div class="col-sm-3"></div>
        <div class="col-sm-6">
            <input type="password" id="password" name="password" placeholder="Password" class="form-control" ng-
model="password" required />
            <span ng-show="loginForm.password.$touched && loginForm.password.$error.required">Please enter
Password.</span>
        </div>
        <div class="col-sm-3">
        </div>
    </div>
    <input type="submit" value="Login" class="btn btn-primary col-sm-offset-3" ng-click="authenticate(userName)" />
</form>
```

Routing – 4
Steps

- include angular.js, angular-route.js, and bootstrap.css
- apply
  - ng-app directive
  - ng-view directive to <div> or other elements where you want to inject another child view
- create an application module and specify 'ngRoute' as a dependency module [app.js]
- config() method to configure the routing rules that need to compile before any other module of an application [app.js]
  - $routingProvider object

Exception Handling

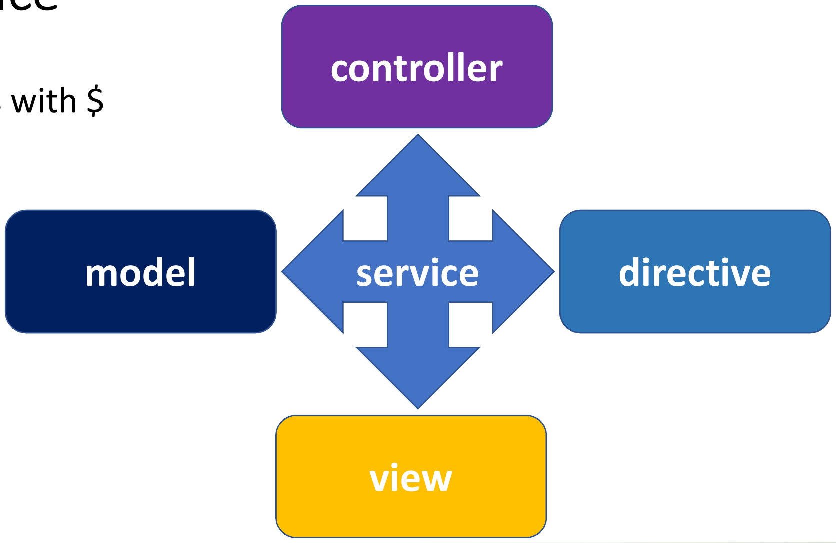
- built-in **`$exceptionHandler`** service
- Exception logs into the browser console



```
TypeError: $http.post(...).success is not a function        angular.js:14328
    at ChildScope.$scope.submitMemberForm (memberController.js:26)
    at fn (eval at compile (angular.js:15156), <anonymous>:4:168)
    at callback (angular.js:26744)
    at ChildScope.$eval (angular.js:17972)
    at ChildScope.$apply (angular.js:18072)
    at HTMLFormElement.<anonymous> (angular.js:26749)
    at defaultHandlerWrapper (angular.js:3613)
    at HTMLFormElement.eventHandler (angular.js:3601)
POST http://localhost:1337/member/submitData 404 (Not        angular.js:12410
Found)
Possibly unhandled rejection: {"data":"Cannot POST        angular.js:14328
/member/submitData\n","status":404,"config":
{"method":"POST","transformRequest":[null],"transformResponse":
[null],"jsonpCallbackParam":"callback","url":"/member/submitData","data":
{"student":{"firstName":"Darryl","surName":"Ng Eng
Soon","secondName":"Eddie"}},"headers":{"Accept":"application/json,
text/plain, */*","Content-Type":"application/json;charset=utf-
8"}},"statusText":"Not Found"}
```

Service

- starts with $



Common Services

$anchorScroll	$exceptionHandler	$interval	$rootScope
$animate	$filter	$locale	$sceDelegate
$cacheFactory	$httpParamSerializer	$location	$sce
$templateCache	$httpParamSerializerJQLike	$log	$templateRequest
$compile	$http	$parse	$timeout
$controller	$httpBackend	$q	$window
$document	$interpolate	$rootElement	

$http Service - 1

Method	Description
$http.get()	Perform Http GET request.
$http.head()	Perform Http HEAD request.
$http.post()	Perform Http POST request.
$http.put()	Perform Http PUT request.
$http.delete()	Perform Http DELETE request.
$http.jsonp()	Perform Http JSONP request.
$http.patch()	Perform Http PATCH request.

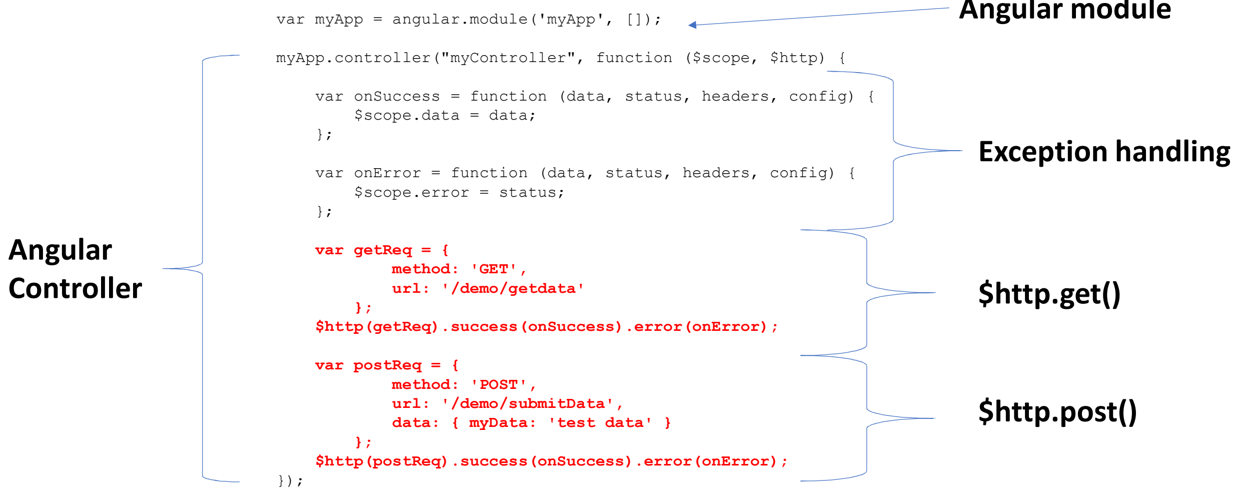
$http Service - 2

- $http.get() method
  - sends http GET request to the remote server and retrieves the data
  - returns HttpPromise object
    - includes various methods to process the response of http GET request
  - Signature: HttpPromise $http.get(url)
- $http.post() method
  - sends Http POST request to the remote server to submit and retrieve the data
  - Signature: HttpPromise $http.post(url, dataToSubmit);

$http Service - 3

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js">
    </script>
</head>
<body ng-app="myHttpDemoApp">
<h1>AngularJS $http Demo: </h1>
    <div ng-controller="myController">
        Response Data: {{data}} <br />
        Error: {{error}}
    </div>
    <script src="/../scripts/myController.js"></script>
</body>
</html>
```

$http Service - 4



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

myApp.controller("myController", function ($scope, $http) {

    var onSuccess = function (data, status, headers, config) {
        $scope.data = data;
    };

    var onError = function (data, status, headers, config) {
        $scope.error = status;
    };

    var getReq = {
            method: 'GET',
            url: '/demo/getdata'
        };
    $http(getReq).success(onSuccess).error(onError);

    var postReq = {
            method: 'POST',
            url: '/demo/submitData',
            data: { myData: 'test data' }
        };
    $http(postReq).success(onSuccess).error(onError);
});
```

$log Service

- logs the messages to the browser's console
- different types of information
  - error
  - information
  - warning
  - debug

```
$log.log('This is log.');
$log.error('This is error.');
$log.info('This is info.');
$log.warn('This is warning.');
$log.debug('This is debugging.');
```

$interval Service - 1

- wrapper methods
  - **`$interval(fn, delay, [count], [invokeApply], [Pass]);`**
  - **`$interval.cancel(promise)`**
- performs the same task as setInterval() method in JavaScript

$interval Service – 2 Simple Example

```
<!DOCTYPE html>
<html >
<head>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app="myApp">
    <div>
        <div ng-controller="myController">
            {{counter}} <br /><br />
            <button ng-click="cancel()">Cancel</button>
        </div>

    </div>
. . .
</body>
</html>
```

```
<script>
  var myApp = angular.module('myApp', []);

  myApp.controller("myController", function ($scope, $interval) {
    $scope.counter = 0;

    var increaseCounter = function () {
      $scope.counter = $scope.counter + 1;
    }

    var promise = $interval(increaseCounter, 1000);

    $scope.cancel = function () {
      $interval.cancel(promise);
      $scope.counter = "Cancelled!";
    };

  });
</script>
```

Filters - 1

- format the data to display on UI without changing original format
- used with an expression or directives using pipe | sign
- `{{ expression | filterName:parameter }}`
- `{{ number_expression | number:fractionSize }}`
- `{{ expression | currency : 'currency_symbol' : 'fraction' }}`
- `{{ date_expression | date : 'format' }}`
- `{{ expression | filter : filter_criteria }}`
- `{{ expression | orderBy : predicate_expression : reverse }}`

Filters - 2

- currency - format a number to a currency format
- date - format a date to a specified format
- filter - select a subset of items from an array
- json - format an object to JSON string
- limitTo - limits an array/string, into a specified number of elements/characters.
- number - format a number to a string
- orderBy - Orders an array by an expression
- lowercase - format a string to lower case
- uppercase - format a string to upper case

Filters – 3
Example 1

```
<div ng-app="myApp" ng-controller="namesCtrl">

<p>Countries:</p>
<ul>
  <li ng-repeat="x in names | orderBy:'country'">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>
```

```
<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Carl',country:'Sweden'},
        {name:'Margareth',country:'England'},
        {name:'Hege',country:'Norway'},
        {name:'Joe',country:'Denmark'},
        {name:'Gustav',country:'Sweden'},
        {name:'Birgit',country:'Denmark'},
        {name:'Mary',country:'England'},
        {name:'Kai',country:'Norway'}
        ];
});
</script>
```

Filters – 4
Example 2

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>

<div ng-app="myApp" ng-controller="namesCtrl">

<p>Type a letter in the input field:</p>

<p><input type="text" ng-model="test"></p>


<ul>
  <li ng-repeat="x in names | filter:test">
    {{ x }}
  </li>
</ul>
</div>
. . .
<p>The list will only consists of names matching the filter.</p>
</body>
</html>
```

```
<script>
angular.module('myApp', []).controller('namesCtrl',
function($scope) {
    $scope.names = [
        'Jani',
        'Carl',
        'Margareth',
        'Hege',
        'Joe',
        'Gustav',
        'Birgit',
        'Mary',
        'Kai'
    ];
});
</script>
```

TERIMA KASIH!

Don't look everywhere and anywhere…

- http://www.angularjshub.com/examples/
- http://www.w3schools.com/angular/angular_controllers.asp
- https://curran.github.io/screencasts/introToAngular/exampleViewer/#/

Find a good wife/husband and one is enough.