ANGULARJS

NOTES AND TUTORIALS
LAB BY: JONATHAN MCCARTHY

INTRO TO ANGULAR

SECTION 1

HISTORY OF ANGULAR

- AngularJS was created by Misko Hevery and Adam Abrons in 2009. Abrons left the project.
- Misko Hevery worked at Google and continued working on Angular. Misko offered Angular as a possible solution for a problem they were having on a project named Google Feedback.
- Misko estimated that the entire project could be re-written in two weeks using Angular. If took 3 weeks and the lines of code went from 17,000 to 1,500.
  - Ref: O'Reilly, AngularJS (Book)

OVERVIEW

- AngularJS is a JavaScript Framework.
- AngularJS is a structural framework used to develop dynamic web applications.
- It uses HTML as the basis for its templating and provides the ability to extend HTML to work easily with Angular.
- Angular uses data binding and dependency injection to add content to the application.
- Everything happens within the browser, eliminating and dependency on server side technologies.

OVERVIEW

- Angular can interact with the DOM and make AJAX requests from the server.
- Angular follows a well defined structure and follows an opinionated convention on how certain tasks should be performed.
- Angular provides the following functionality to create simple CRUD apps out of the box:
  - Data-binding
  - Templates
  - form validation
  - deep-linking
  - reusable components
  - dependency injection

MVC

- Trygve Reenskaug when working at the Xerox Palo Alto Research Laboratory (PARC), created the pattern known as MVC in 1979.
- Model–view–controller (MVC) is a software design pattern.
- It is an architecture pattern to separate the representation of information from the user's interaction with it.
- The business logic of the web application resides on the model.
- The controller acts as a mediator between the models and views.
- The views primary role is the presentation of information to the end users.
- Angular follows the MVC pattern (MVC \ MVVM \ MV*)

WHY USE MVC?

- MVC implements Separation of Concerns (SoC).
- Separating the business logic from the presentation code allows for better testing of the application. (TDD)
- By separating functionality between the Model, View, and Controller allows each section to be tested independently.
- Cleaner presentation code (HTML, CSS, JavaScript), with minimal code nuggets.
- MVC offers structure to a web application.
- By using this structure in a team environment allows everyone to work off "the same hymn sheet".
- It provides common features and functionality to developers in an environment with a common structure.
- MVC model is using REST based URLs instead of specific file names (eg. default.aspx, index.php)

WHY USE ANGULARJS

- Angular helps simplify the development process by offering a higher level of abstraction for the basic functionality and structure required to develop an application.
- As with all libraries and frameworks, they are only suitable for certain types of projects.
- Applications that need CRUD functionality with no specific requirement for complex DOM manipulation, then AJAX will be a good fit.

ANGULAR PHILOSOPHY

- Decouple DOM manipulation from business logic
- This makes it easier to test the application
- Distinct separation of client side and server side
- Framework offers common functionality to developers in a structured manner
- Most of the tedious tasks of manipulating the DOM have been taken care of by Angular.

ANGULAR TERMINOLOGY

SECTION 2

TEMPLATES

- Standard web apps generate a html response that comprises of the data in the html structure.
- Angular passes the template and the data to the browser for assembly.
- The template is the html document with some extensions that are added to the syntax. An example of an extension is the addition of new attributes for data binding etc (see next slide)
- Data can be passed to the page for processing and display.

DIRECTIVES

- Angular uses HTML as the template.
- It includes a DOM transformation engine that is used to extend the HTML syntax.
- A directive is a marker that is added to a DOM element. Eg:
- Element
- Attribute
- This is used by the DOM transformation engine to attach a specific behaviour to a DOM element.
- Angular provides a set of pre-defined directives.

DIRECTIVES

- Angular provides a set of pre-defined directives. Eg:
  - ngApp
  - ngBind
  - ngModel
  - ngController
  - ngClass
  - ngRepeat
- It is possible to create your own directives

DIRECTIVES (NG-APP)

- The ng-app directive tells angular which element block of the page it will be managing. Eg. If we wanted Angular to manage the entire page the template would contain the following starting html element:
- <html ng-app>

- If we wanted Angular to manage a subset of the page, the ng-app attribute could be placed on a different element on the page.

DIRECTIVES (NG-BIND)

- Ng-bind tells angular what data or expression should be used to replace the existing content of a given element. This can also be updated when the data or expression changes. Double curley brackets can be used to bind data {{ }}. Ng-bind is preferred, as this in invisible as part of a page load, the double curley brackets may show!!

DIRECTIVES (NG-MODEL)

- The ngModel directive binds user input to a property on the scope using NgModelController.
- User Input can be an input,select, textarea (or custom form control) etc…

MODEL

- The model is responsible for managing application data. It responds to the request from view and to the instructions from controller to update itself.

CONTROLLER

- In Angular, the controller is our JavaScript classes.
- JavaScript functions will perform the role of the controller.

- The controller can request functionality from the model.

- The controller can send and receive to\from the views.

VIEW

- What can we do in a View?
  - Display Data
  - Loop through a result set
  - Get user input
  - Display messages
  - Links to certain functionality
  - Structuring Layouts

SECTION 3

EXAMPLES

HELLO WORLD EXAMPLE

- The following example will be explained in class!!
- If you have missed class please follow the online tutorials that have been added to Moodle.

HELLO WORLD EXAMPLE

- Create a new page named hello.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
<h1>Angular Hello World!!</h1>
</body>

</html>
```

ANGULAR SCRIPT

- Add Angular to the page from the Google CDN
- <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
- Add the above script to the head section.

ADD OUR OWN JS FILE

- Create a new assets folder in your project folder
- In the new assets folder, create a new js folder.
- Create a new file named controller.js in the js folder.

- Add the following to the head section:
- <script src="/assets/js/controller.js"></script>

ADD THE ANGULAR MARKUP TO THE PAGE

- Update your page to look as follows:

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

  <head>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

    <script src="assets/js/controller.js"></script>
  </head>

  <body>
    <div ng-controller="helloController">
      <p>{{greeting.text}}, world </p>

      <p><span ng-bind="greeting.text"></span> Jono, hope this works for you!!</p>
    </div>
  </body>

</html>
```

CONTROLLER.JS

- Add the following code to controller.js

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

myFirstApp.controller('helloController', ['$scope',
  function($scope) {
    $scope.greeting = { text: 'Hello' };
}]);
```

BIG PICTURE – HELLO WORLD

View

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

  <head>
   <script src="https://ajax.googleapis.com/
ajax/libs/angularjs/1.3.14/
angular.min.js"></script>

    <script src="assets/js/controller.js"></
script>
  </head>

  <body>
    <div ng-controller="helloController">
      <p>{{greeting.text}}, world </p>

      <p><span ng-bind="greeting.text"></
span> Jono, hope this works for you!!</p>
    </div>
  </body>

</html>
```

Controller

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

myFirstApp.controller('helloController', ['$scope',
  function($scope) {
    $scope.greeting = { text: 'Hello' };
}]);
```

Model

```
// Business logic, if we had some!!
```

EXAMPLE 2 – STUDENT LIST

- Create a page to display a students information and their grades for semester 2.
  - Firstname
  - Lastname
  - Student Number
  - Email
  - Grades
- Note:
  - The following example will be explained in class!!
  - If you have missed class please follow the online tutorials that have been added to Moodle.

STUDENTS – HTML FILE

- Create a new file named students.html

```
<html ng-app="myStudentApp">
<head>
  <title>Your Shopping Cart</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

    <script src="assets/js/students.js"></script>
</head>
<body ng-controller='studentController'>
<h1>Student Details</h1>
<p>{{student.firstname}}</p>
<p>{{student.lastname}}</p>
<p>{{student.studentNum}}</p>
<p>{{student.email}}</p>

<h2>Grades</h2>
<div ng-repeat='i in grades'>
  <span>{{i.module}} - {{i.grade}}</span>
  <button ng-click="remove($index)">Remove</button>
   <input ng-model='i.grade'>
</div>

</body>
</html>
```

STUDENTS – STUDENTS.JS

- Create a new file named students.js in the assets/js folder

```
var myStudentApp = angular.module('myStudentApp', []);
myStudentApp.controller('studentController', ['$scope',
  function($scope) {
    $scope.student = { firstname: 'Johnny',
                       lastname: 'McCarthy',
                       studentNum: 123456,
                      email: 'jon@email.ie' };

    $scope.grades = [
      {module: 'Advanced Client Side Development', grade: 60},
      {module: 'Cloud Computing', grade: 67},
      {module: 'Programming 101', grade: 80}
    ];

    $scope.remove = function(index) {
      $scope.grades.splice(index, 1);
    };
}]);
```

ADDITIONAL LAB WORK

- See Moodle for additional lab work!!