AngularJS

Introduction

- AngularJS is a JavaScript framework written in JavaScript.
- AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
```

What is AngularJs

- AngularJS is an open source web application framework. It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google. Its latest version is 1.4.3.
- Definition of AngularJS as put by its official documentation is as follows −
- AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly. Angular's data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

Dependency Injection can exist between two objects, for instance, **a flashlight and a battery**. The flashlight needs the battery to function. However, any changes made to the battery, such as switching it with another brand/set of batteries, does not mean the dependent object (flashlight) also needs to be changed

In object-oriented programming (OOP) software design, dependency injection (DI) is **the process of supplying a resource that a given piece of code requires**. The required resource, which is often a component of the application itself, is called a dependency.

Features of Angular js

- AngularJS is a powerful JavaScript based development framework to create RICH Internet Application(RIA).
- AngularJS provides developers options to write client side application (using JavaScript) in a clean MVC(Model View Controller) way.
- Application written in AngularJS is cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
- AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.
- Overall, AngularJS is a framework to build large scale and high performance web application while keeping them as easy-to-maintain.

Core Features of Angular JS

- **Data-binding –** It is the automatic synchronization of data between model and view components.
- **Scope –** These are objects that refer to the model. They act as a glue between controller and view.
- **Controller –** These are JavaScript functions that are bound to a particular scope.
- **Services –** AngularJS come with several built-in services for example $https: to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
- **Filters –** These select a subset of items from an array and returns a new array.
- **Directives –** Directives are markers on DOM elements (such as elements, attributes, css, and more). These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives (ngBind, ngModel...)
- **Templates –** These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".
- **Routing –** It is concept of switching views.
- **Model View Whatever –** MVC is a design pattern for dividing an application into different parts (called Model, View and Controller), each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
- **Deep Linking –** Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
- **Dependency Injection –** AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.

Advantages of AngularJs

- AngularJS provides capability to create Single Page Application in a very clean and maintainable way.
- AngularJS provides data binding capability to HTML thus giving user a rich and responsive experience
- AngularJS provides reusable components.
- With AngularJS, developer write less code and get more functionality.
- In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.
- On top of everything, AngularJS applications can run on all major browsers and smart phones including Android and iOS based phones/tablets.

Disadvantages of AngularJs

- Though AngularJS comes with lots of plus points but same time we should consider the following points −
- **Not Secure** − Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
- **Not degradable** − If your application user disables JavaScript then user will just see the basic page and nothing more.

Component of Angular JS

- The AngularJS framework can be divided into following three major parts –
- **ng-app –** This directive defines and links an AngularJS application to HTML.
- **ng-model –** This directive binds the values of AngularJS application data to HTML input controls.
- **ng-bind –** This directive binds the AngularJS Application data to HTML tags.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/
angular.min.js"></script>
<body>
<div ng-app >
  <p >Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>
<div>
  <p >Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>
</body> </html>
```

LECT-21\examples\ng-directives.html

- AngularJS Extends HTML
- AngularJS extends HTML with **ng-directives**.
- The **ng-app** directive defines an AngularJS application.
- The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
- The **ng-bind** directive binds application data to the HTML view.

- As you have already seen, AngularJS directives are HTML attributes with an **ng** prefix.
- The **ng-init** directive initializes AngularJS application variables.
-

  The name is

- You can use **data-ng-**, instead of **ng-**, if you want to make your page HTML valid.

LECT-21\examples\ng-init.html

Angular Expression

- AngularJS Expressions
- AngularJS expressions are written inside double braces: **{{ expression }}**.
- AngularJS will "output" data exactly where the expression is written:

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

    My first expression: {{ 5 + 5 }}

  </body>
  </html>
  ```

LECT-21\examples\AJS_expression_1.html

LECT-21\examples\AJS_expression_2.html
  ```

Expression can be Written anywhere

- You can write expressions wherever you like, AngularJS will simply resolve the expression and return the result.
- Example: Let AngularJS change the value of CSS properties.
- Change the color of the input box below, by changing its value:

```
<div ng-app="" ng-init="myCol='lightblue'">

    <input style="background-color:{{myCol}}" ng-model="myCol"
    value="{{myCol}}">

    </div>
```

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

    My first expression: {{ 5 + 5 }}

  </body>
  </html>
  ```
- Removing **ng-app** will print content as it is without solving
  ```

AngularJS Expressions

- AngularJS expressions can be written inside double braces: {{ *expression* }}.
- AngularJS expressions can also be written inside a directive: ng-bind="*expression*".
- AngularJS will resolve the expression, and return the result exactly where the expression is written.
- **AngularJS expressions** are much like **JavaScript expressions:** They can contain literals, operators, and variables.
- Example {{ 5 + 5 }} or {{ firstName + " " + lastName }}

- AngularJS Applications
- AngularJS **modules** define AngularJS applications.
- AngularJS **controllers** control AngularJS applications.
- The **ng-app** directive defines the application, the **ng-controller** directive defines the controller.
- Example Ang2.html

AngularJS Expressions vs. JavaScript Expressions

- Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
- Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.
- AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
- AngularJS expressions support filters, while JavaScript expressions do not.

AngularJS Strings

- AngularJS objects are like JavaScript objects:
- Example

```
<div ng-app="" ng-init="person={firstName:'John',
lastName:'Doe'}">

<p>The name is {{ person.lastName }}</p>

</div>
```

LECT-21\examples\AJS_string.html

AngularJS Arrays

- AngularJS arrays are like JavaScript arrays:
- Example

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

  <p>The third result is {{ points[2] }}</p>

  </div>
```

- o/p will be The third result is 19

LECT-21\examples\AJS_array.html

An AngularJS module

- An AngularJS module defines an application.
- The module is a container for the different parts of an application.
- The module is a container for the application controllers.
- Controllers always belong to a module.

An AngularJS module

- Creating a Module
- A module is created by using the AngularJS function angular.module

```
<div ng-app="myApp">...</div>

    <script>

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

- The "myApp" parameter refers to an HTML element in which the application will run.
- Now you can add controllers, directives, filters, and more, to your AngularJS application.

Angularjs Controller

- Add a controller to your application, and refer to the controller with the ng-controller directive:
- Example

```
<div ng-app="myApp" ng-controller="myCtrl">
    {{ firstName + " " + lastName }}
    </div>

    <script>

    var app = angular.module("myApp", []);

    app.controller("myCtrl", function($scope) {
        $scope.firstName = "John";
        $scope.lastName = "Doe";
    });

    </script>
```

LECT-21\examples\AJS_module_controller.html

Adding Directives

- AngularJS has a set of built-in directives which you can use to add functionality to your application.
- In addition you can use the module to add your own directives to your applications:
- Example
- <div ng-app="myApp" w3-test-directive>

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

app.directive("w3TestDirective", function() {
    return {
        template : "I was made in a directive constructor!"
    };
});
</script>
```

LECT-21\examples\AJS_module_directives_1.html

LECT-21\examples\AJS_module_directives_2.html

AngularJS Directives

- AngularJS lets you extend HTML with new attributes called **Directives**.
- AngularJS has a set of built-in directives which offers functionality to your applications.
- AngularJS also lets you define your own directives.
- AngularJS directives are extended HTML attributes with the prefix ng-.
- The ng-app directive initializes an AngularJS application.
- The ng-init directive initializes application data.
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

AngularJS Directives

- example two text fields are bound together with two ng-model directives:

- Example
-

  Quantity: <input type="number" ng-model="quantity">
  Costs: <input type="number" ng-model="price">

  Total in dollar: {{ quantity * price }}

Looping HTML

- Repeating HTML Elements
- The ng-repeat directive repeats an HTML element:
- Example
-

    <li ng-repeat="x in names">
      {{ x }}

- The ng-repeat directive actually **clones HTML elements** once for each item in a collection.
- The ng-repeat directive used on an array of objects:

LECT-21\examples\ng-repeat.html

LECT-21\examples\ng-repeat_table.html

Ng-repeat using table

```
<table ng-controller="myCtrl" border="1">
<tr ng-repeat="x in records">
  <td>{{x.Name}}</td>
  <td>{{x.Country}}</td>
</tr>
</table>
```

DROP DOWN USING repeat

LECT-21\examples\dropdown_ng-repeat.html

Directives: ng-app

- The ng-app Directive
- The ng-app directive defines the **root element** of an AngularJS application.
- The ng-app directive will **auto-bootstrap** (automatically initialize) the application when a web page is loaded.

Directives: ng-init

- The ng-init Directive
- The ng-init directive defines **initial values** for an AngularJS application.
- Normally, you will not use ng-init. You will use a controller or module instead.

Directives: ng-model

- The ng-model Directive
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
- The ng-model directive can also:
- Provide type validation for application data (number, email, required).
- Provide status for application data (invalid, dirty, touched, error).
- Provide CSS classes for HTML elements.
- Bind HTML elements to HTML forms.

Directives: ng-model

-
  Name: <input ng-model="name">

  <script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope) {
  $scope.name = "John Doe";
  });
  </script>

Directives: ng-model

- The ng-model directive adds/removes the following classes, according to the status of the form field:

- ng-empty
- ng-not-empty
- ng-touched
- ng-untouched
- ng-valid
- ng-invalid
- ng-dirty
- ng-pending
- ng-pristine

Directives: ng-model

- Application Status

- The ng-model directive can provide status for application data (invalid, dirty, touched, error):

- ```
  <form ng-app="" name="myForm" ng-init="myText = 'post@myweb.com'">
    Email:
    <input type="email" name="myAddress" ng-model="myText" required>
    <h1>Status</h1>
    {{myForm.myAddress.$valid}}
    {{myForm.myAddress.$dirty}}
    {{myForm.myAddress.$touched}}
  </form>
  ```

LECT-21\examples\Ang_Validate_Application_Status.html
  ```

Directives: ng-controller

- AngularJS controllers control the data of AngularJS applications.
- AngularJS controllers are regular JavaScript Objects.
- AngularJS Controllers
- AngularJS applications are controlled by controllers.
- The ng-controller directive defines the application controller.
- A controller is a JavaScript Object, created by a standard JavaScript object constructor.

Directives: ng-controller

-

  First Name: <input type="text" ng-model="firstName">
  Last Name: <input type="text" ng-model="lastName">

  Full Name: {{firstName + " " + lastName}}

  <script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
  });
  </script>

Directives: ng-controller

- Application explained:
- The AngularJS application is defined by ng-app="myApp". The application runs inside the .
- The ng-controller="myCtrl" attribute is an AngularJS directive. It defines a controller.
- The myCtrl function is a JavaScript function.
- AngularJS will invoke the controller with a $scope object.
- In AngularJS, $scope is the application object (the owner of application variables and functions).
- The controller creates two properties (variables) in the scope (firstName and lastName).
- The ng-model directives bind the input fields to the controller properties (firstName and lastName).

Directives: ng-controller

- **Controller Methods**
- The example above demonstrated a controller object with two properties: lastName and firstName.
- A controller can also have methods (variables as functions):

Directives: ng-controller

-

  First Name: <input type="text" ng-model="firstName">
  Last Name: <input type="text" ng-model="lastName">

  Full Name: {{fullName()}}

  <script>
  var app = angular.module('myApp', []);
  app.controller('personCtrl', function($scope) {
      $scope.firstName = "John";
      $scope.lastName = "Doe";
      $scope.fullName = function() {
          return $scope.firstName + " " + $scope.lastName;
      };
  });
  </script>

AngularJS Scope

- The scope is the binding part between the HTML (view) and the JavaScript (controller).
- The scope is an object with the available properties and methods.
- The scope is available for both the view and the controller.
- How to Use the Scope?
- When you make a controller in AngularJS, you pass the $scope object as an argument:

AngularJS Scope

-

  <h1>{{carname}}</h1>

  <script>

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

  app.controller('myCtrl', function($scope) {
  $scope.carname = "Volvo";
  });

- </script>

AngularJS Scope

- When adding properties to the $scope object in the controller, the view (HTML) gets access to these properties.

- In the view, you do not use the prefix $scope, you just refer to a propertyname, like {{carname}}.
- Understanding the Scope

- If we consider an AngularJS application to consist of:

- View, which is the HTML.
- Model, which is the data available for the current view.
- Controller, which is the JavaScript function that makes/changes/removes/controls the data.

- Then the scope is the Model.

- The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.

AngularJS Scope

- Root Scope
- All applications have a $rootScope which is the scope created on the HTML element that contains the ng-app directive.
- The rootScope is available in the entire application.
- If a variable has the same name in both the current scope and in the rootScope, the application use the one in the current scope.

LECT-21\examples\Ang_RootScope.html

AngularJS Scope

```
<body ng-app="myApp">

<p>The rootScope's favorite color:</p>
<h1>{{color}}</h1>

<div ng-controller="myCtrl">
    <p>The scope of the controller's favorite color:</p>
    <h1>{{color}}</h1>
</div>

<p>The rootScope's favorite color is still:</p>
<h1>{{color}}</h1>

<script>
var app = angular.module('myApp', []);
app.run(function($rootScope) {
    $rootScope.color = 'blue';
});
app.controller('myCtrl', function($scope) {
    $scope.color = "red";
});
</script>
</body>
```

AngularJS Filters

- Filters can be added in AngularJS to format data.
- AngularJS Filters

- AngularJS provides filters to transform data:

- 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 a JSON string.
- limitTo Limits an array/string, into a specified number of elements/characters.
- lowercase Format a string to lower case.
- number Format a number to a string.
- orderBy Orders an array by an expression.
- uppercase Format a string to upper case.

LECT-21\examples\Ang_Filter1.html

LECT-21\examples\Ang_Filter2.html

AngularJS Filters

-

- The name is {{ lastName | uppercase }}

-

AngularJS Filters

- The filter Filter
- The filter filter selects a subset of an array.
- The filter filter can only be used on arrays, and it returns an array containing only the matching items.

AngularJS Filters

-

-
- <li ng-repeat="x in names | filter : 'i'">
- {{ x }}
-
-

-

AngularJS Filters

- Sort an Array Based on User Input

- Click the table headers to change the sort order::

SERVICES

1. TimeOut — LECT-21\examples\10_Ang_Services_1_Timeout.html

2. Convert Number into Hexadecimal — LECT-21\examples\10_Ang_Services_3_T_Hexa.html

3. Interval Time — LECT-21\examples\10_Ang_Services_2_TInterval.html

Ng-disable LECT-21\examples\ng-disable.html

EVENTS

ng-mousemover LECT-21\13_Ang_Event.html
ng-click LECT-21\examples\13_Ang_Event_Function.html

ANIMATE **LECT-21\examples\14_Ang_Animate.html**
ANG_VALIDATION **LECT-21\examples\14_Ang_Validation.html**