1. **What is AJAX ?**

AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. It is an extended javascript used for creating the interactive and most faster web applications. It just referes the component instead of loading the page every time. It is a technology helps to communicate the data with the server. It is used to make the Asynchronous call with the server.

1. **Can we make the synchronous call with AJAX ?**

When we use ajax with JQuery we jave an option to make the synchronous ajax call by setting the flag **async: false**. But synchronous ajax calls not prefred , since the browser will wait unitl the request gets completed.

**EX:**

function getRemote() {

//var remote;

return $.ajax({

type: "GET",

url: remote\_url,

async: false,

success : function(data) {

remote = data;

}

});

//return remote;

}

**Angular JS**

It is an open source JavaScript framework used to build the web applications. It is freely used to change modify and shared by anyone.

**Advantages over other java script framework: (why Angukar JS)**

1. **Dependency Injection :** It the design pattern used to give the dependencies to the component instead of hardcoding the dependencies into the component
2. **Data Binding:** it supports two way databinding between model and view components
3. **MVC Support :** it supports MVC framework
4. **Testing :**  it provides support for testing the components through unit testing or end to end testing
5. **Other features:** it supports directives, filters, modules and routes etc.

**Building Angular JS application:**

Angular JS is the mix of HTML and javascript. To build the angular is application , we need to include the angular js library in the <head> tag.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">**

**</script>**

**</head>**

**<body ng-app="myapp">**

**<div ng-controller="mycontoller">**

**<p>First Angular J S Application </p>**

**<h1>{{welcome.msg}} </h1>**

**</div> ---- View part**

**</body>**

**<script>**

**angular.module("myapp", []).controller("mycontoller",function($scope){**

**$scope.welcome={};**

**$scope.welcome.msg=" Hi "**

**});**

**</script> --- controller part**

**</html>**

**Angular JS Databinding:**

Databinding is an useful and powerful feature acts as an bridge between the view and business logic of the application. It binds data between model and view component of the angular js application.

There are 2 types of data binding:

1. **One-way databinding:**

In this way, data is bind from model to view . there is no way to bind the data from view to model

1. **Two-way databinding:**

Angular JS supports 2 way data binding , it supports the synchronization of dat between model and view components. The data changes in model automatically reflected into the view and vice versa.

**Ex: 1**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">**

**</script>**

**</head>**

**<body ng-app="" ng-init="firstName='Vasantha'">**

**<div>**

**<p>First Angular J S Application </p>**

**<h1>{{firstName}} </h1>**

**</div>**

**</body>**

**</html>**

**Ex: 2 Bind the view data to Model**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">**

**</script>**

**</head>**

**<body ng-app="" >**

**<div>**

**<p>First Angular J S Application </p>**

**<input type="text" ng-model="firstName" />**

**<h1>you typed :{{firstName}} </h1>**

**</div>**

**</body>**

**</html>**

{{firstName}} -> is the angular JS data binding expression

**Ex: 3**

**<!DOCTYPE html>**

**<html>**

**<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>**

**<body>**

**<div data-ng-app="" data-ng-init="quantity=1;price=20">**

**<h2>Cost Calculator</h2>**

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

**Price: <input type="number" ng-model="price">**

**<p><b>Total in rupees:</b> {{quantity \* price}}</p>**

**</div>**

**</body>**

**</html>**

**Angular JS Expressions:**

Angular JS expressions are used to bind the application data into the html. The expressions can contains literals , operators and variable names. Angular JS resolves the expressions and returns the result.

The expressions can be written inside {{}} or it can be written using the ng-bind directive ng-bind=”expression”

**Ex: 1**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">

</script>

</head>

<body **ng-app**>

**<p>A simple Angular JS expression exmaple :{{5+10}}</p>**

</body>

</html>

**Using the ng-bind directive**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">

</script>

</head>

<body ng-app>

**<p>A simple Angular JS expression exmaple :<span ng-bind="3+4"/> </p>**

</body>

</html>

**Angular JS Numbers , Strings, Arrays and Objects:**

The angular js numbers , string arrays and objects are treated similar to java script.

Ex:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">

</script>

</head>

**<body ng-app="" ng-init="quantity=4;price=3;firstName='vasantha';lastName='periyasamy';arr=[1,2,3,4,5,6,9];person={firstName:'sakthi',lastName:'Mugesh'}">**

**<p>Total Cost:{{quantity\*price}} </p>**

**<p> My full Name is : <span ng-bind="firstName+' ' +lastName" /> </p>**

**<p> Arrays 3rd element : {{arr[4]}} </p>**

**<p> Access object value (firstName f person object) : {{person.firstName}} </p>**

</body>

</html>

**Similarity between AngularJS Expressions and Javascript Expressions:**

Both expressions contains literals, operators and variables

**Difference between AngularJS expressions and Javascript expressions:**

1. Angular JS expressions are written inside HTML, but java script expressions cannot.
2. Angular JS expression support filters but java script expressions are not.
3. Angular Js expressions cannot support loops, conditionals and exceptions but java script expressions can support.

**Angular JS Directives:**

Angular JS facilitate to extend the HTML with special attributes. These special attributes are called directives. Directives are special attributes which offers functionality to the Angular JS application. The directives are starts with “ng-“ prefix.

The commonly used directives are,

1. **ng-app :** This is the directive to start the angular JS application. The ng-app specifies the root element of the angular JS application. We should have only one ng-app directive in the html document . if we have multiple directive , then the first appeared one will be used.
2. **ng-init:** it is used to initialize the application data . Which gives the default value to the angular JS application
3. **ng-model:** it is used to define the model or variable to be used in the angular JS application.
4. **ng-repeat:** it is used to repeat the html elements for each item in the collection.

Ex:

<!DOCTYPE html>

<html>

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

<body>

<div ng-app="" ng-init="countries=[{name:'india',locale:'en\_ind'},{name:'australia',locale:'en\_aus'}]">

**<h2> ng-app,ng-init,ng-repeat example </h2>**

<ol>

<li ng-repeat="count in countries">

{{'country : '+count.name +' , Locale:'+count.locale}}

</li>

</ol>

**<h2> ng-model example </h2>**

<input type="text" ng-model="myName" />

My Name is : {{myName}}

</div>

</body>

</html>

1. **ng-bind:** it is used to bind the data. it replaces the html content with a variable value of the value of the expression. If the value of the variable is changed then the html content also changed.

**Ex:**

<!DOCTYPE html>

<html>

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

<body ng-app="myapp">

<div ng-controller="mycontroller">

<h2> ng-bind example </h2>

<input type="text" ng-model="myName" />

My Name is : <span ng-bind="myName" />

</div>

<script>

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

app.controller("mycontroller",function($scope){

$scope.myName="javaTpoint";

});

</script>

</body>

</html>

1. **sdf**

**Angular JS Controllers:**

Angular js controller is used to control the flow of data in angular JS application. The controller is defined using the ng-controller directive. Controller is an java script object which contains the variables and functions. Each controller contains the $scope as a parameter , which referes the module or application that controller used to control.

**Ex:**

<!DOCTYPE html>

<html>

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

<body ng-app="myapp">

<div ng-controller="mycontroller">

<h2> ng-bind example </h2>

<input type="text" ng-model="firstName" />

<input type="text" ng-model="lastName" />

**My Full Name is : {{fullName()}}**

**or**

**My Full Name is using ng-bind: <span ng-bind="fullName()" />**

</div>

<script>

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

app.controller("mycontroller",function($scope){

$scope.firstName="Vasantha";

$scope.lastName="periyasamy";

**$scope.fullName=function(){**

**return $scope.firstName + " " + $scope.lastName;**

**};**

});

</script>

</body>

</html>

**Angular JS controller in external files:**

In larger applications, the controller is saved in external files. (appController.js) . This external file is then used with html by using the <script> tag,

<script src=”appController.js” />

The file contains the following controller part of code,

angular.module("myapp",[]);

controller("mycontroller",function($scope){

$scope.firstName="Vasantha";

$scope.lastName="periyasamy";

$scope.fullName=function(){

return $scope.firstName + " " + $scope.lastName;

};

});

**Angular JS Modules:**

Angular JS modules define an application. Modules are container for the controller , directives, filters and services. The modules are created by angular object module() method. It is like the Main() method of the application.

**Add controller to Module:**

The ng-controller directive is used to add the controller to the application or module.

**Add Directive to Module:**

**Ex:**

<!DOCTYPE html>

<html>

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

<body >

<div ng-app="myApp" w3-test-directive></div>

<script>

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

app.directive("w3TestDirective", function() {

return {

template : "This is a directive constructor. "

};

});

</script>

</body>

</html>

**Angular JS Modules in external files:**

In angular JS, modules and controllers are stored in external files with .js extension.

**Angular JS Scopes:**

The scope is an object specified as binding part between view (HTML) and Controller (Javascript). It joins the controller with views. It is applicable for both view and controller.

To make the controller in Angular js , we have to pass the @scope object as parameter.

**Ex: Refer previous example**

**Angular JS Dependency Injection:**

Dependency injection is an design pattern which describes how the components get hold of their dependencies. Angular JS has the built in design pattern which help the components to given their dependencies instead of creating the dependencies in the component.

The following are the types of objects and components which are injected into each other by using the dependency injection mechanism.

1. **Value :** it is an simple object which contains the numbers , strings and java script objects. It can be injected into controllers , services and factories. It is created using the value() of module object.
2. **Factory:** it is an function which is used to return the value based on some calculation. It is created using the factory() method of module object. It can be injected into services and controllers.
3. **Services :** it is an java script object which has set of functions to perform certain tasks. It is created using the service() of module object.
4. **Provider:** it is the special type of factory method which has get() function which returns the factory. It internally creates factory and services during the config phase.
5. **Constant:** it is used to see the constant value during the config time.

All these are injected into each other by using their names passing as a parameter into the respective function.

**Ex:**

<!DOCTYPE html>

<html>

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

<body>

<div ng-app = "mainApp" ng-controller = "CalcController">

<p>Enter a number: <input type = "number" **ng-model** = "number" /></p>

<button ng-click = **"square()">**X<sup>2</sup></button>

<p>Result: **{{result}}</**p>

</div>

<script>

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

//**Factory is created during the config time**

mainApp.config(function($provide) { 🡪 **config method is invoked at the time of page load**

$provide.provider('MathCalcFactory', function() { 🡪 **'MathCalcFactory' –Factory name**

this.$get = function() { 🡪 Get method definition

var factory = {};

factory.multiply = function(a, b) { **🡪method definition in factory**

return a \* b;

}

return factory;

};

});

});

mainApp.value("defaultInput", 10); 🡪 Injecting the number value

**//Factory without provider**

/\* mainApp.factory('MathCalcFactory', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

}); \*/

mainApp.service('CalcService', function(MathCalcFactory){ 🡪 **define the service, injecting the factory named MathCalcFactory**

this.square = function(a) {

alert("inside service");

return MathCalcFactory.multiply(a,a); 🡪 **invoking the method in factory**

}

});

mainApp.controller('CalcController', function($scope, CalcService, defaultInput) { **🡪 injecting the service and value**

$scope.number = defaultInput;

$scope.result = CalcService.square($scope.number);

$scope.square = function() {

$scope.result = CalcService.square($scope.number);

}

});

</script>

</div>

</body>

</html>

**Angular JS Filters:**

Filters are used to format the data. The following are the default filters used in Angular JS.

1. **lowercase:** convert the string to lowercase
2. **uppercase:** convert the string to uppercase
3. **date :** format the date to specified format
4. **currency:** format the number in currency format
5. **number:** format the number in number format
6. **orderBy:** sort the data in ascending order
7. **filter:** used to filter the array of data based on given input

Filters can be added to expressions and directives by using the | (pipeline ) character.

Ex: Various usages of filters can be explained in the below example

<!DOCTYPE html>

<html>

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

<body>

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

**<h2> example for lowercase , uppercase , currency, number, date filters and adding the filters to an expression </h2>**

<p>The name in lower case is {{ firstName | lowercase }}</p>

<p>The name in upper case is {{lastName | uppercase}} </p>

<p> Thumber filter is {{salary|number}}

<p> Thumber filter is {{hra|number}}

<p> Thumber filter is {{price|currency}}

<p> Thumber filter is {{todate|date}}

**<h2> example for orderBy filter and adding the filters to an directives </h2>**

<ul>

<li ng-repeat="x in names|orderBy:'name'" >

{{x.name}}

</li>

</ul>

**<h2>example for Filter filter </h2>**

<ul>

<li ng-repeat="val in filterValues|filter:'e'">

{{val}}

</li>

</ul>

**<h2> example for filter an array based on user input </h2>**

<input type="text" ng-model="filterVal" />

<ul>

<li ng-repeat="val in filterValues | filter:filterVal">

{{val}}

</li>

</ul>

**<h2> example for customer filter or sort an array based on user input</h2>**

<table border="1" >

<tr>

<th ng-click="orderByMe('stuName')" >Name</th>

<th ng-click="orderByMe('mark')" > Mark </th>

</tr>

<tr ng-repeat="student in students|orderBy:myOrder" >

<td> {{student.stuName}} </td>

<td> {{student.mark}} </td>

</tr>

</table>

</div>

<script>

angular.module('myApp', []).controller('filterCtrl', function($scope) {

$scope.firstName = "Sonoo";

$scope.lastName = "Jaiswal";

$scope.salary="15000000";

$scope.hra="16657.23";

$scope.price="1523";

$scope.todate="1982/24/5";

**// orderBy filter values**

$scope.names=[{name:'stella'},{name:'preethi'}, {name:'asha'},{name:'bella'},{name:'jothi'}] ;

**// filter Filter values**

$scope.filterValues=['stella','preethi','asha','bella','jothi'];

**// Cutom filter values**

$scope.students=[{stuName:'Vasantha',mark:'35'},{stuName:'Suja',mark:'75'},{stuName:'Meena',mark:'56'},{stuName:'karthi',mark:'78'},{stuName:'jovi',mark:'100'}];

$scope.orderByMe=function(columnName){

$scope.myOrder=columnName;

alert("my order is :"+myOrder);

}

});

</script>

</body>

</body>

</html>

**Angular JS Tables:**

Angular JS tables are easily created by ng-repeat directive.

**Ex:**

<!DOCTYPE html>

<html>

<head>

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

<style>

table, th , td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

table tr:nth-child(odd) {

background-color: #f2f2f2;

}

table tr:nth-child(even) {

background-color: #ffffff;

}

</style>

</head>

<body>

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

<h2> example for angular js tables with CSS </h2>

**<table>**

**<tr>**

**<th ng-click="orderByMe('stuName')" >Name</th>**

**<th ng-click="orderByMe('mark')" > Mark </th>**

**</tr>**

**<tr ng-repeat="student in students|orderBy:myOrder" >**

**<td> {{student.stuName}} </td>**

**<td> {{student.mark}} </td>**

**</tr>**

**</table>**

</div>

<script>

angular.module('myApp', []).controller('filterCtrl', function($scope) {

$scope.firstName = "Sonoo";

$scope.lastName = "Jaiswal";

$scope.salary="15000000";

$scope.hra="16657.23";

$scope.price="1523";

$scope.todate="1982/24/5";

$scope.students=[{stuName:'Vasantha',mark:'35'},{stuName:'Suja',mark:'75'},{stuName:'Meena',mark:'56'},{stuName:'karthi',mark:'78'},{stuName:'jovi',mark:'100'}];

$scope.orderByMe=function(columnName){

$scope.myOrder=columnName;

alert("my order is :"+myOrder);

}

});

</script>

</body>

</body>

</html>

**AngularJS Select:**

The select box or drop down list is created using the ng-options directive. This directive returns the value as object. The values are populated from an object or from an array.

**Ex:**

<!DOCTYPE html>

<html>

<head>

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

</head>

<body>

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

<h2> example for angular js Select or Drop Down box from an array</h2>

**<select ng-model="selectedValue" ng-options="x for x in dropDownValues" >**

</select>

selected value is : {{selectedValue}}

<h2> example for angular js Select or Drop Down box from an object</h2>

**<select ng-model="selectedStudent" ng-options="x.stuName for x in students" >**

</select><br/>

selected student name is : {{selectedStudent.stuName}} <br/>

selected student mark is : {{selectedStudent.mark}}

</div>

<script>

angular.module('myApp', []).controller('selectCtrl', function($scope) {

$scope.dropDownValues=['stella','preethi','asha','bella','jothi'];

//drop down values as an object

$scope.students=[{stuName:'Vasantha',mark:'35'},{stuName:'Suja',mark:'75'},{stuName:'Meena',mark:'56'},{stuName:'karthi',mark:'78'},{stuName:'jovi',mark:'100'}];

});

</script>

</body>

</html>

The same drop down values are created by using the ng-repeat directive.

Ex:

<!DOCTYPE html>

<html>

<head>

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

</head>

<body>

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

<h2> example for angular js Select or Drop Down box from an array</h2>

**<select ng-model="selectedValue" >**

**<option ng-repeat="x in dropDownValues" > {{x}}**

**</select>**

selected value is : {{selectedValue}}

<h2> example for angular js Select or Drop Down box from an object</h2>

**<select ng-model="selectedStudent" >**

**<option ng-repeat="x in students" /> {{x.stuName}}**

**</select><br/>**

selected student name is : {{selectedStudent}} <br/>

</div>

<script>

angular.module('myApp', []).controller('selectCtrl', function($scope) {

$scope.dropDownValues=['stella','preethi','asha','bella','jothi'];

//drop down values as an object

$scope.students=[{stuName:'Vasantha',mark:'35'},{stuName:'Suja',mark:'75'},{stuName:'Meena',mark:'56'},{stuName:'karthi',mark:'78'},{stuName:'jovi',mark:'100'}];

});

</script>

</body>

</html>

The difference between ng-options and ng-repeat is , ng-repeat return the selected value as string but the ng-option return as object. The most preferable way is using the ng-option directive to create the drop down.

**Angular JS DOM:**

Angular JS provides an feature to bind the application data to attributes of the HTML DOM elements. The following directives used for this,

1. ng-disabled : it disable the given html control
2. ng-show: it shows the given control
3. ng-hide: it hide the given control
4. ng-click: it represents the click event

**Ex:**

<!DOCTYPE html>

<html>

<head>

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

</head>

<body>

<div ng-app >

<h2> example for angular js Select or Drop Down box from an array</h2>

<input type = "checkbox" ng-model = "enableDisableButton">Disable Button

<button **ng-disabled** = "enableDisableButton">Click Me!</button>

<br/>

<input type = "checkbox" ng-model = "showHide1">Show Button

<button **ng-show** = "showHide1">Click Me!</button>

<br/>

<input type = "checkbox" ng-model = "showHide2">Hide Button

<button **ng-hide** = "showHide2">Click Me!</button>

<br/>

<p>Total click: {{ clickCounter }}</p>

<button **ng-click** = "clickCounter = clickCounter + 1">Click Me!</button>

</div>

</body>

</html>

**Angular JS Forms & form Validation:**

Angular JS provides an effective way of creating the forms with data binding and validation on input controls. Data binding can be done by using the ng-model directive.

The following are the various input controls used in forms:

* input elements
* textarea elements
* Select elements
* Button elements

There are certain events associated with input controls are,

* ng-click
* ng-dbl-click
* ng-mouseentered
* ng-mouseleave
* ng-mouseover
* ng-mouseup
* ng-mousedown
* ng-mousemove
* ng-keypress
* ng-keydown
* ng-keyup
* ng-change

**Validation:**

Angular JS supports client side validation . It check the state of the form and input controls and return the current state to the user.

These are the directives used to report the errors in angular js forms.

* $invalid – indicates that the value entered is invalid
* $dirty – indicates that the value has been changed
* $error- indicates the actual error text

Ex:

<!DOCTYPE html>

<html>

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

<body>

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

<form name="myForm" novalidate>

<table>

<tr>

<td>

Enter user name :

</td>

<td>

<input name="userName" type="text" ng-model="uName" required /> **<span style="color:red" ng-show="myForm.userName.$dirty && myForm.userName.$invalid">**

**<span ng-show="myForm.userName.$error.required"> User Name is required</span>**

</span>

</td>

</tr>

<tr>

<td>

Enter user email :

</td>

<td>

<input name="userEmail" type="email" ng-model="uEmail" required /> <span style="color:red" ng-show="myForm.userEmail.$dirty && myForm.userEmail.$invalid">

<span ng-show="myForm.userEmail.$error.required"> Email is required</span>

</span>

</td>

</tr>

<tr>

<td> <button ng-disbled="myForm.userName.$dirty && myForm.userName.$invalid" ng-click="submit()">Submit </button>

</td>

<td> <button ng-click="reset()">Clear </button>

</td>

</tr>

</table>

</form>

</div>

<script>

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

app.controller("myCtrl",function($scope){

$scope.uName="vasantha";

$scope.reset=function(){

$scope.uName="";

};

});

</script>

</body>

</html>

**Angular AJAX:**

Angular JS supports Ajax by using the $http service. AJAX call is used to fetch records from remote servers.

There are various methods supported by $http service as

* Get
* Post
* Put
* delete
* head
* jsonp
* patch

The ajax call can be invoked as

Function myController($scope,$http){

Var url=” some url”;

$http.get(url).success(function(response){

//process the response here

});

}

The response is an object which has different states

* status – http status code of the response
* statusText – http status text of the response
* data – actual response data
* headers - response headers
* config - object used to generate the request

**Angular JS Animation:**

Angular JS supports animation by including the following library,

**<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-animate.js"></script>**

Then , refer the ngAnimate module in our application module as like,

<div ng-app=”ngAnimate” >

<div>

Or refer the ngAnimate module as dependency in our application module as like,

Angular.module(“myApp”,[‘ngAnimate’])

**ngAnimate :-**  is used to add or remove the classes. It doesn’t animate the html elements . it wil notice certain events and according to that attach certain predefined classes into html elements which take care of the animation

The directives which can add or removes classes,

* ng-hide
* ng-show
* ng-view
* ng-class
* ng-if
* ng-switch
* ng-repeat
* ng-include

**Angular Includes:**

Angular Js has the feature to include the html content from external file into the current page by using the ng-include directive.

**Ex:**

<body ng-app=””>

<div ng-include=”test.html” />

</body>

**Angular JS Routing:**

Angular JS supports the feature called routing by providing the ngRoute module. Routing is the feature to navigate to the different pages in the application without reloading the current page. It helps to creating the single page application. ngRoute module routes to the different pages in the application without reloading the entire application.

To use the routing feature in our application, the routing library should be included,

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-route.js"></script>**

Then ,incude the ngRoute module to one of the dependencies in our currernt module as like,

Angular.module(“myApp” [‘ngRoute’])

We need a container to place the content provided by routing. The container is provided by the ng-view directive.

There are 3 ways we can include the ng-view directive in our application,

1. <div ng-view ></div>
2. <ng-view> </ng-view>
3. <div class=”ng-view” </div>

The ngRoute module provides $routeProvider ,which helps to configure the different routes of the application. The $routeProvider uses the config method of the application , the work registered in the config method is done when the application is loading.

Ex:

<!DOCTYPE html>

<html>

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

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-route.js"></script>

<body ng-app="myApp">

<p><a href="#/">Main</a></p>

<a href="#london">City 1</a>

<a href="#paris">City 2</a>

<p>Click on the links to read about London and Paris.</p>

<div ng-view></div>

<script>

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

**app.config(function($routeProvider) {**

**$routeProvider**

**.when("/", {**

**templateUrl : "main.htm"**

**})**

**.when("/london", {**

**templateUrl : "london.htm"**

**})**

**.when("/paris", {**

**templateUrl : "paris.htm"**

**});**

});

</script>

</body>

</html>

**Define Controller for each view:**

We can define the controller for each view.

**Ex :**

**london.html**

<h1>London</h1>  
<h3>London is the capital city of England.</h3>  
<p>It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>  
<p>{{msg}}</p> 🡪 expression get message from its own controller

**Paris.html**

<h1>Paris</h1>  
<h3>Paris is the capital city of France.</h3>  
<p>The Paris area is one of the largest population centers in Europe, with more than 12 million inhabitants.</p>  
<p>{{msg}}</p>🡪 expression get message from its own controller

<!DOCTYPE html>

<html>

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

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-route.js"></script>

<body ng-app="myApp">

<p><a href="#/">Main</a></p>

<a href="#london">City 1</a>

<a href="#paris">City 2</a>

<p>Click on the links.</p>

<p>Note that each "view" has its own controller which each gives the "msg" variable a value.</p>

<div ng-view></div>

<script>

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

app.config(function($routeProvider) {

$routeProvider

.**when("/", {**

**templateUrl : "main.htm",**

**})**

**.when("/london", {**

**templateUrl : "london.htm",**

**controller : "londonCtrl"**

**})**

**.when("/paris", {**

**templateUrl : "paris.htm",**

**controller : "parisCtrl"**

});

});

app.controller("londonCtrl", function ($scope) {

**$scope.msg = "I love London";**

});

app.controller("parisCtrl", function ($scope) {

**$scope.msg = "I love Paris";**

});

</script>

</body>

</html>

templateUrl property of when method is used to specify the routing url. The template property is used to place the html content directly instead of routing to the different page as like,

var app = angular.module("myApp", ["ngRoute"]);  
app.config(function($routeProvider) {  
    $routeProvider  
    .when("/", {  
        template : "<h1>Main</h1><p>Click on the links to change this content</p>"  
    })  
    .when("/banana", {  
        template : "<h1>Banana</h1><p>Bananas contain around 75% water.</p>"  
    })  
    .when("/tomato", {  
        template : "<h1>Tomato</h1><p>Tomatoes contain around 95% water.</p>"  
    });  
});

**Otherwise method:**

$routeProvider ‘s when method is used to route the application to different page based on certain condition but the otherwise method specifies the default routing page .

Ex:

var app = angular.module("myApp", ["ngRoute"]);  
app.config(function($routeProvider) {  
   $routeProvider  
    .when("/banana", {  
        template : "<h1>Banana</h1><p>Bananas contain around 75% water.</p>"  
    })  
    .when("/tomato", {  
        template : "<h1>Tomato</h1><p>Tomatoes contain around 95% water.</p>"  
    })  
    .otherwise({  
        template : "<h1>None</h1><p>Nothing has been selected</p>"  
    });  
});

**AngularJS Training – Birlasoft**

* Latest version is 1.6.4
* Angular js is required to build the Single page application
* Data capturing is done by the angular
* jQuery is the library , but Angular is the framework which supports dependency injection and MVC architecture.
* nginx is the static web server where we can execute our web pages
* download nginx version 1.12.0 and install
* make sure the installation by typing <http://loholhost:portnumbrt(default> is 80) , it should display the installation successful page.
* change the port number as

open config folder ->nginx.conf ->open

line no 36, change the listener port to 9001 or something

restart the ng server by killing the process.

Open run-> type taskkill /F /IM nginx.exe

Make sure the restart process by <http://localhost:9001>

Open Html folder – where we write our angular code

Edit index.html

**Topic covered :**

**Expressions:**

**Directives:**

**ng-bind :**

Alternate to expression to bind the data to view

**ng-app:**

starting point of our angular application

**ng-model:**

* applied to input elements
* data collected from input controls using the ng-model
* ng-model variable placed in the bucket and displayed wherever required
* various events are attached to mg-model
* ng-model returns the type of value mentioned in the input control using the type attribute

**ng-options:**

To work with ng-options ng-model is required. Without ng-model , ng-options wont work.

**ng-route:**

it is used to include the template. Ui-view is used to render the template which acts as container for the template.

**ng-repeat:**

it is used to add data dynamically.

**ng-include:**

* it is used to include the template

**Controller:**

* communications happens via controller
* it is used to interact with the view

There are two ways to interact with the view,

1. using the $scope object
2. using the controller as constructor

**Check list to develop angular application:**

* Always have immediate function
* Create separate module for functionality
* Whenever we create the JS file , we must specify in the html file
* Make sure that spellings are correct

Namespace are required to separate the functionality

Bootswatch -> free themes to bootstrap

Bootswatchcdn – which has link for css

Kirans git hub: <https://github.com/kiranuidev/BSOFTAng.git>

**Services:**

* Which gives functionality to be used in our application
* Services are reusable components
* Services are shared across application
* Plays an important role in architecture
* It can pull data from rest api.

**Promise Patterns:**

* Useful for mocking and unit testing
* It is accessed by $q
* Promise help to mock
* Promise make our DB call and synchronous call

**Datamapping:**

Fetching the data and adding the extra value or data into it.

**Routing:**

* Basic router is ng-router
* Advanced router is angular UI router , which is a third party library
* Angular-ui-router.cdn has thems for ui router
* Ui-router modeul dependency has to be added in the config method

**Controller to controller communication:**

* It happens in multiple way

1. Have a service stored the data inside the service
2. Store the data into the global scope variable $routeScope
3. Borancast and on
4. Chain controller

**$scope :**

* Built in service for the angular
* Help to interact with the view
* Initial version of angular supports only $scope to interact with the view , after that controller is used.
* $rootScope is the global scope for the entire application
* We can attach event to the $rootScope

**Config:**

* It plays an important role in the application lifecycle
* Dependency is loaded first then the main module is loaded