**Angular JS**

* AngularJS is a JavaScript framework. It can be added to an HTML page with a <script> tag.
* AngularJS extends HTML attributes with Directives, and binds data to HTML with Expressions.
* AngularJS is a JavaScript framework. It is a library written in JavaScript.
* AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:
* 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.
* 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.
* 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 Modules
* 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.
* 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, and textarea) to application data.

* 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:

<div ng-app="" ng-init="names=[  
{name:'Jani',country:'Norway'},  
{name:'Hege',country:'Sweden'},  
{name:'Kai',country:'Denmark'}]">  
  
<ul>  
  <li ng-repeat="x in names">  
    {{ x.name + ', ' + x.country }}  
  </li>  
</ul>  
  
</div>

You can invoke a directive by using:

* Element name
* Attribute
* Class
* Comment

You can restrict your directives to only be invoked by some of the methods.

By adding a restrict property with the value "A", the directive can only be invoked by attributes:

var app = angular.module("myApp", []);  
app.directive("w3TestDirective", function() {  
    return {  
        restrict : "A",  
        template : "<h1>Made by a directive!</h1>"  
    };  
});

The legal restrict values are:

* **E for Element name**
* **A for Attribute**
* **C for Class**
* **M for Comment**

By default the value is EA, meaning that both Element names and attribute names can invoke the directive.

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**

* AngularJS Data Binding

Data binding in AngularJS is the synchronization between the model and the view. AngularJS applications usually have a data model. The data model is a collection of data available for the application.

* Two-way Binding

Data binding in AngularJS is the synchronization between the model and the view. When data in the model changes, the view reflects the change, and when data in the view changes, the model is updated as well. This happens immediately and automatically, which makes sure that the model and the view is updated at all times.

* AngularJS Controllers

AngularJS controllers control the data of AngularJS applications.

AngularJS controllers are regular JavaScript Objects.

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.

<div ng-app="myApp" ng-controller="myCtrl">  
  
First Name: <input type="text" ng-model="firstName"><br>  
Last Name: <input type="text" ng-model="lastName"><br>  
<br>  
Full Name: {{firstName + " " + lastName}}  
  
</div>  
  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.firstName = "John";  
    $scope.lastName = "Doe";  
});  
</script>

Application explained:

The AngularJS application is defined by  **ng-app="myApp"**. The application runs inside the <div>.

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).

## 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.

When you make a controller in AngularJS, you pass the $scope object as an argument:

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.

## 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.

Note : 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.

<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 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.

Filters can be added to expressions by using the pipe character |, followed by a filter.

Filters are added to directives, like ng-repeat, by using the pipe character |, followed by a filter:

Example

The orderBy filter sorts an array:

<div ng-app="myApp" ng-controller="namesCtrl">  
  
<ul>  
  <li ng-repeat="x in names | orderBy:'country'">  
    {{ x.name + ', ' + x.country }}  
  </li>  
</ul>  
  
</div>

By adding the ng-click directive on the table headers, we can run a function that changes the sorting order of the array:

Example

<div ng-app="myApp" ng-controller="namesCtrl">  
  
<table border="1" width="100%">  
  <tr>  
    <th ng-click="orderByMe('name')">Name</th>  
    <th ng-click="orderByMe('country')">Country</th>  
  </tr>  
  <tr ng-repeat="x in names | orderBy:myOrderBy">  
    <td>{{x.name}}</td>  
    <td>{{x.country}}</td>  
  </tr>  
</table>  
  
</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'}  
  ];  
  $scope.orderByMe = function(x) {  
    $scope.myOrderBy = x;  
  }  
});  
</script>

## Custom Filters

You can make your own filters by registering a new filter factory function with your module:

Example

Make a custom filter called "myFormat":

<ul ng-app="myApp" ng-controller="namesCtrl">  
    <li ng-repeat="x in names">  
        {{x | myFormat}}  
    </li>  
</ul>  
  
<script>

var app = angular.module('myApp', []);  
app.filter('myFormat', function() {  
    return function(x) {  
        var i, c, txt = "";  
        for (i = 0; i < x.length; i++) {  
            c = x[i];  
            if (i % 2 == 0) {  
                c = c.toUpperCase();  
            }  
            txt += c;  
        }  
        return txt;  
    };  
});  
app.controller('namesCtrl', function($scope) {  
    $scope.names = ['Jani', 'Carl', 'Margareth', 'Hege', 'Joe', 'Gustav', 'Birgit', 'Mary', 'Kai'];  
});

</script>

## AngularJS Services

In AngularJS you can make your own service, or use one of the many built-in services.

## What is a Service?

In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application.

AngularJS has about 30 built-in services. One of them is the $location service.

The $location service has methods which return information about the location of the current web page:

Example

Use the $location service in a controller:

var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $location) {  
    $scope.myUrl = $location.absUrl();  
});

Note that the $location service is passed in to the controller as an argument. In order to use the service in the controller, it must be defined as a dependency.

## Why use Services?

For many services, like the $location service, it seems like you could use objects that are already in the DOM, like the window.location object, and you could, but it would have some limitations, at least for your AngularJS application.

AngularJS constantly supervises your application, and for it to handle changes and events properly, AngularJS prefers that you use the $location service instead of the window.location object.

## The $http Service

The $http service is one of the most common used services in AngularJS applications. The service makes a request to the server, and lets your application handle the response.

### Example

Use the $http service to request data from the server:

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("welcome.htm").then(function (response) {  
        $scope.myWelcome = response.data;  
    });  
});

## The $timeout Service

The $timeout service is AngularJS' version of the window.setTimeout function.

### Example

Display a new message after two seconds:

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $timeout) {  
    $scope.myHeader = "Hello World!";  
    $timeout(function () {  
        $scope.myHeader = "How are you today?";  
    }, 2000);  
});

## The $interval Service

The $interval service is AngularJS' version of the window.setInterval function.

### Example

Display the time every second:

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $interval) {  
    $scope.theTime = new Date().toLocaleTimeString();  
    $interval(function () {  
        $scope.theTime = new Date().toLocaleTimeString();  
    }, 1000);  
});

## Create Your Own Service

To create your own service, connect your service to the module:

Create a service named hexafy:

app.service('hexafy', function() {  
    this.myFunc = function (x) {  
        return x.toString(16);  
    }  
});

To use your custom made service, add it as a dependency when defining the filter:

### Example

Use the custom made service named hexafy to convert a number into a hexadecimal number:

app.controller('myCtrl', function($scope, **hexafy**) {  
    $scope.hex = **hexafy**.myFunc(255);  
});

<div ng-app="myApp">

Convert the number 255, using a custom made service inside a custom made filter:

<h1>{{255 | myFormat}}</h1>

</div>

<script>

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

app.service('hexafy', function() {

this.myFunc = function (x) {

return x.toString(16);

}

});

app.filter('myFormat',['hexafy', function(hexafy) {

return function(x) {

return hexafy.myFunc(x);

};

}]);

</script>

# AngularJS AJAX - $http

**$http** is an AngularJS service for reading data from remote servers.

<script>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("welcome.htm")  
    .then(function(response) {  
        $scope.myWelcome = response.data;  
    });  
});

</script>

The example above uses the .get method of the $http service.

The .get method is a shortcut method of the $http service. There are several shortcut methods:

* .delete()
* .get()
* .head()
* .jsonp()
* .patch()
* .post()
* .put()

The methods above are all shortcuts of calling the $http service:

<script>

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

app.controller('myCtrl', function($scope, $http) {

$http({

method : "GET",

url : "welcome.htm"

}).then(function mySucces(response) {

$scope.myWelcome = response.data;

}, function myError(response) {

$scope.myWelcome = response.statusText;

});

});

</script>

The example above executes the $http service with an object as an argument. The object is specifying the HTTP method, the url, what to do on success, and what to do on failure.

## Properties

The response from the server is an object with these properties:

* .config the object used to generate the request.
* .data a string, or an object, carrying the response from the server.
* .headers a function to use to get header information.
* .status a number defining the HTTP status.
* .statusText a string defining the HTTP status.

### Example

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("welcome.htm")  
    .then(function(response) {  
        $scope.content = response.data;  
        $scope.statuscode = response.status;  
        $scope.statustext = response.statustext;   
    });  
});

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("wrongfilename.htm")  
    .then(function(response) {  
        //First function handles success  
        $scope.content = response.data;  
    }, function(response) {  
        //Second function handles error  
        $scope.content = "Something went wrong";  
    });  
});

## JSON

The data you get from the response is expected to be in JSON format.

JSON is a great way of transporting data, and it is easy to use within AngularJS, or any other JavaScript.

Example: On the server we have a file that returns a JSON object containing 15 customers, all wrapped in array called records.

[Take a look at the JSON object.](javascript:void(0))

### Example

The ng-repeat directive is perfect for looping through an array:

<div ng-app="myApp" ng-controller="customersCtrl">   
  
<ul>  
  <li ng-repeat="x in myData">  
    {{ x.Name + ', ' + x.Country }}  
  </li>  
</ul>  
  
</div>  
  
<script>

var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $http) {  
    $http.get("customers.php").then(function(response) {  
        $scope.myData = response.data.records;  
    });  
});

</script>

Application explained:

The application defines the customersCtrl controller, with a $scope and $http object.

$http is an **XMLHttpRequest object** for requesting external data.

$http.get() reads **JSON data** from <http://www.w3schools.com/angular/customers.php>.

On success, the controller creates a property, myData, in the scope, with JSON data from the server.

# AngularJS Tables

## Displaying with CSS Style

<!DOCTYPE html>

<html>

<style>

table, th , td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

table tr:nth-child(odd) {

background-color: #f1f1f1;

}

table tr:nth-child(even) {

background-color: #ffffff;

}

</style>

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

<body>

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

<table>

<tr ng-repeat="x in names">

<td>{{ x.Name }}</td>

<td>{{ x.Country }}</td>

</tr>

</table>

</div>

<script>

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

app.controller('customersCtrl', function($scope, $http) {

$http.get("http://www.w3schools.com/angular/customers.php")

.then(function (response) {$scope.names = response.data.records;});

});

</script>

</body>

</html>

## Using $even and $odd

<!DOCTYPE html>

<html>

<style>

table, td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

</style>

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

<body>

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

<table>

<tr ng-repeat="x in names">

<td ng-if="$odd" style="background-color:#f1f1f1">

{{ x.Name }}</td>

<td ng-if="$even">

{{ x.Name }}</td>

<td ng-if="$odd" style="background-color:#f1f1f1">

{{ x.Country }}</td>

<td ng-if="$even">

{{ x.Country }}</td>

</tr>

</table>

</div>

<script>

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

app.controller('customersCtrl', function($scope, $http) {

$http.get("http://www.w3schools.com/angular/customers.php")

.then(function (response) {$scope.names = response.data.records;});

});

</script>

</body>

</html>

## Cross-Site HTTP Requests

Requests for data from a different server (than the requesting page), are called **cross-site** HTTP requests.

Cross-site requests are common on the web. Many pages load CSS, images, and scripts from different servers.

In modern browsers, cross-site HTTP requests **from scripts** are restricted to **same site** for security reasons.

The following line, in our PHP examples, has been added to allow cross-site access.

**header("Access-Control-Allow-Origin: \*");**

## AngularJS Routing

The ngRoute module helps your application to become a Single Page Application.

If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

The **ngRoute** module *routes* your application to different pages without reloading the entire application.

<script>

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

app.config(function($routeProvider) {

$routeProvider

.when("/", {

templateUrl : "main.htm"

})

.when("/red", {

templateUrl : "red.htm"

})

.when("/green", {

templateUrl : "green.htm"

})

.when("/blue", {

templateUrl : "blue.htm"

});

});

</script>

With the $**routeProvider** you can also define a controller for each "view".

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";  
});

## Template

In the previous examples we have used the templateUrl property in the $routeProvider.when method.

You can also use the template property, which allows you to write HTML directly in the property value, and not refer to a page.

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>"  
    });  
});

## AngularJS Directives

|  |  |
| --- | --- |
| Directive | Description |
| [ng-app](http://www.w3schools.com/angular/ng_ng-app.asp) | Defines the root element of an application. |
| [ng-bind](http://www.w3schools.com/angular/ng_ng-bind.asp) | Binds the content of an HTML element to application data. |
| [ng-bind-html](http://www.w3schools.com/angular/ng_ng-bind-html.asp) | Binds the innerHTML of an HTML element to application data, and also removes dangerous code from the HTML string. |
| [ng-bind-template](http://www.w3schools.com/angular/ng_ng-bind-template.asp) | Specifies that the text content should be replaced with a template. |
| [ng-blur](http://www.w3schools.com/angular/ng_ng-blur.asp) | Specifies a behavior on blur events. |
| [ng-change](http://www.w3schools.com/angular/ng_ng-change.asp) | Specifies an expression to evaluate when content is being changed by the user. |
| [ng-checked](http://www.w3schools.com/angular/ng_ng-checked.asp) | Specifies if an element is checked or not. |
| [ng-class](http://www.w3schools.com/angular/ng_ng-class.asp) | Specifies CSS classes on HTML elements. |
| [ng-class-even](http://www.w3schools.com/angular/ng_ng-class-even.asp) | Same as ng-class, but will only take effect on even rows. |
| [ng-class-odd](http://www.w3schools.com/angular/ng_ng-class-odd.asp) | Same as ng-class, but will only take effect on odd rows. |
| [ng-click](http://www.w3schools.com/angular/ng_ng-click.asp) | Specifies an expression to evaluate when an element is being clicked. |
| [ng-cloak](http://www.w3schools.com/angular/ng_ng-cloak.asp) | Prevents flickering when your application is being loaded. |
| [ng-controller](http://www.w3schools.com/angular/ng_ng-controller.asp) | Defines the controller object for an application. |
| [ng-copy](http://www.w3schools.com/angular/ng_ng-copy.asp) | Specifies a behavior on copy events. |
| [ng-csp](http://www.w3schools.com/angular/ng_ng-csp.asp) | Changes the content security policy. |
| [ng-cut](http://www.w3schools.com/angular/ng_ng-cut.asp) | Specifies a behavior on cut events. |
| [ng-dblclick](http://www.w3schools.com/angular/ng_ng-dblclick.asp) | Specifies a behavior on double-click events. |
| [ng-disabled](http://www.w3schools.com/angular/ng_ng-disabled.asp) | Specifies if an element is disabled or not. |
| [ng-focus](http://www.w3schools.com/angular/ng_ng-focus.asp) | Specifies a behavior on focus events. |
| ng-form | Specifies an HTML form to inherit controls from. |
| [ng-hide](http://www.w3schools.com/angular/ng_ng-hide.asp) | Hides or shows HTML elements. |
| [ng-href](http://www.w3schools.com/angular/ng_ng-href.asp) | Specifies a url for the <a> element. |
| [ng-if](http://www.w3schools.com/angular/ng_ng-if.asp) | Removes the HTML element if a condition is false. |
| [ng-include](http://www.w3schools.com/angular/ng_ng-include.asp) | Includes HTML in an application. |
| [ng-init](http://www.w3schools.com/angular/ng_ng-init.asp) | Defines initial values for an application. |
| ng-jq | Specifies that the application must use a library, like jQuery. |
| [ng-keydown](http://www.w3schools.com/angular/ng_ng-keydown.asp) | Specifies a behavior on keydown events. |
| [ng-keypress](http://www.w3schools.com/angular/ng_ng-keypress.asp) | Specifies a behavior on keypress events. |
| [ng-keyup](http://www.w3schools.com/angular/ng_ng-keyup.asp) | Specifies a behavior on keyup events. |
| [ng-list](http://www.w3schools.com/angular/ng_ng-list.asp) | Converts text into a list (array). |
| [ng-maxlength](http://www.w3schools.com/angular/ng_ng-maxlength.asp) | Specifies the maximum number of characters allowed in the input field. |
| [ng-minlength](http://www.w3schools.com/angular/ng_ng-minlength.asp) | Specifies the minimum number of characters allowed in the input field. |
| [ng-model](http://www.w3schools.com/angular/ng_ng-model.asp) | Binds the value of HTML controls to application data. |
| [ng-model-options](http://www.w3schools.com/angular/ng_ng-model-options.asp) | Specifies how updates in the model are done. |
| [ng-mousedown](http://www.w3schools.com/angular/ng_ng-mousedown.asp) | Specifies a behavior on mousedown events. |
| [ng-mouseenter](http://www.w3schools.com/angular/ng_ng-mouseenter.asp) | Specifies a behavior on mouseenter events. |
| [ng-mouseleave](http://www.w3schools.com/angular/ng_ng-mouseleave.asp) | Specifies a behavior on mouseleave events. |
| [ng-mousemove](http://www.w3schools.com/angular/ng_ng-mousemove.asp) | Specifies a behavior on mousemove events. |
| [ng-mouseover](http://www.w3schools.com/angular/ng_ng-mouseover.asp) | Specifies a behavior on mouseover events. |
| [ng-mouseup](http://www.w3schools.com/angular/ng_ng-mouseup.asp) | Specifies a behavior on mouseup events. |
| [ng-non-bindable](http://www.w3schools.com/angular/ng_ng-non-bindable.asp) | Specifies that no data binding can happen in this element, or its children. |
| [ng-open](http://www.w3schools.com/angular/ng_ng-open.asp) | Specifies the open attribute of an element. |
| [ng-options](http://www.w3schools.com/angular/ng_ng-options.asp) | Specifies <options> in a <select> list. |
| [ng-paste](http://www.w3schools.com/angular/ng_ng-paste.asp) | Specifies a behavior on paste events. |
| ng-pluralize | Specifies a message to display according to en-us localization rules. |
| [ng-readonly](http://www.w3schools.com/angular/ng_ng-readonly.asp) | Specifies the readonly attribute of an element. |
| [ng-repeat](http://www.w3schools.com/angular/ng_ng-repeat.asp) | Defines a template for each data in a collection. |
| [ng-required](http://www.w3schools.com/angular/ng_ng-required.asp) | Specifies the required attribute of an element. |
| [ng-selected](http://www.w3schools.com/angular/ng_ng-selected.asp) | Specifies the selected attribute of an element. |
| [ng-show](http://www.w3schools.com/angular/ng_ng-show.asp) | Shows or hides HTML elements. |
| [ng-src](http://www.w3schools.com/angular/ng_ng-src.asp) | Specifies the src attribute for the <img> element. |
| [ng-srcset](http://www.w3schools.com/angular/ng_ng-srcset.asp) | Specifies the srcset attribute for the <img> element. |
| [ng-style](http://www.w3schools.com/angular/ng_ng-style.asp) | Specifies the style attribute for an element. |
| [ng-submit](http://www.w3schools.com/angular/ng_ng-submit.asp) | Specifies expressions to run on onsubmit events. |
| [ng-switch](http://www.w3schools.com/angular/ng_ng-switch.asp) | Specifies a condition that will be used to show/hide child elements. |
| ng-transclude | Specifies a point to insert transcluded elements. |
| [ng-value](http://www.w3schools.com/angular/ng_ng-value.asp) | Specifies the value of an input element. |

AngularJS Directives on HTML Elements

AngularJS modifies the default behavior of some HTML elements.

|  |  |
| --- | --- |
| Element | Description |
| [a](http://www.w3schools.com/angular/ng_a.asp) | AngularJS modifies the <a> element's default behaviors. |
| [form](http://www.w3schools.com/angular/ng_form.asp) | AngularJS modifies the <form> element's default behaviors. |
| [input](http://www.w3schools.com/angular/ng_input.asp) | AngularJS modifies the <input> element's default behaviors. |
| script | AngularJS modifies the <script> element's default behaviors. |
| select | AngularJS modifies the <select> element's default behaviors. |
| [textarea](http://www.w3schools.com/angular/ng_textarea.asp) | AngularJS modifies the <textarea> element's default behaviors. |

AngularJS Filters

|  |  |
| --- | --- |
| Filter | Description |
| [currency](http://www.w3schools.com/angular/ng_filter_currency.asp) | Format a number to a currency format. |
| [date](http://www.w3schools.com/angular/ng_filter_date.asp) | Format a date to a specified format. |
| [filter](http://www.w3schools.com/angular/ng_filter_filter.asp) | Select a subset of items from an array. |
| [json](http://www.w3schools.com/angular/ng_filter_json.asp) | Format an object to a JSON string. |
| [limitTo](http://www.w3schools.com/angular/ng_filter_limitto.asp) | Limits an array, or a string, into a specified number of elements/characters. |
| [lowercase](http://www.w3schools.com/angular/ng_filter_lowercase.asp) | Format a string to lower case. |
| [number](http://www.w3schools.com/angular/ng_filter_number.asp) | Format a number to a string. |
| [orderBy](http://www.w3schools.com/angular/ng_filter_orderby.asp) | Orders an array by an expression. |
| [uppercase](http://www.w3schools.com/angular/ng_filter_uppercase.asp) | Format a string to upper case. |

Filters are explained in [Angular Filters](http://www.w3schools.com/angular/angular_filters.asp).

AngularJS Validation Properties

$dirty

$invalid

$error

Validation is explained in [Angular Validation](http://www.w3schools.com/angular/angular_validation.asp).

AngularJS Global API

Converting

|  |  |
| --- | --- |
| API | Description |
| angular.lowercase() | Converts a string to lowercase |
| angular.uppercase() | Converts a string to uppercase |
| angular.copy() | Creates a deep copy of an object or an array |
| angular.forEach() | Executes a function for each element in an object or array |

Comparing

|  |  |
| --- | --- |
| API | Description |
| angular.isArray() | Returns true if the reference is an array |
| angular.isDate() | Returns true if the reference is a date |
| angular.isDefined() | Returns true if the reference is defined |
| angular.isElement() | Returns true if the reference is a DOM element |
| angular.isFunction() | Returns true if the reference is a function |
| angular.isNumber() | Returns true if the reference is a number |
| angular.isObject() | Returns true if the reference is an object |
| angular.isString() | Returns true if the reference is a string |
| angular.isUndefined() | Returns true if the reference is undefined |
| angular.equals() | Returns true if two references are equal |

JSON

|  |  |
| --- | --- |
| API | Description |
| angular.fromJson() | Deserializes a JSON string |
| angular.toJson() | Serializes a JSON string |

Basic

|  |  |
| --- | --- |
| API | Description |
| angular.bootstrap() | Starts AngularJS manually |
| angular.element() | Wraps an HTML element as an jQuery element |
| angular.module() | Creates, registers, or retrieves an AngularJS module |

1. **What is $injector and $inject?**

$injector is a service which is used to invoke controller functions, service functions, filter functions, and any other function that might need dependencies as parameters. Angular creates only a single $injector object.

$inject is property which is used to inject the dependencies of a function as an array of strings.

MyController['$inject'] = ['$scope', 'greeter']; //inject dependencies as an array of strings

1. **What is Dependency Injection in AngularJS?**

Dependency Injection (DI) is a software design pattern that deals with how components get hold of their dependencies. AngularJS comes with a built-in dependency injection mechanism. You can divide your AngularJS app into multiple different types of components which AngularJS can inject into each other.

1. **What is Routing in AngularJS?**

AngularJS Routing helps you to divide your app into multiple views and bind different views to Controllers. The magic of Routing is taken care by an AngularJS service $**routeProvider**. $**routeProvider** service provides method when() and otherwise() to define the routes for your app. Routing has dependency on ngRoute module.

1. **What is difference between $interval and window. setInterval in AngularJS?**

$interval is an Angular service which wraps the browser's window. setInterval() function. It is also used to call a JavaScript function repeatedly within a time interval.

1. **What is difference between $timeout and window.setTimeout in AngularJS?**

$timeout is an Angular service which wraps the browser's window.setTimeout() function into a try/catch block and delegates any exceptions to $exceptionHandler service. It is used to call a JavaScript function after a given time delay. The $timeout service only schedules a single call to the function.

1. **What is Restangular?**

Restangular is an Angular service specifically designed simply to fetch data from the rest of the world. To use Restangular you need to link the restangular.js file and include restangular resource as a dependency within your angular app.

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

app.controller('MainController',function ($scope, Restangular) { //TO DO: });

1. **What is $q service and when to use it?**

$q is a service that helps you to run functions asynchronously, and use their return values when they have done processing.

$q service is said to be inspired by Chris Kowal's Q library which allow users to monitor asynchronous progress by providing a "promise" as a return from a call.

It is good when you need to process a number of asynchronous activities simultaneously. The $q.all() function lets you trigger several callbacks at the same time, and use a single then function to join them all together.

1. **What methods $resource service object support?**

The $resource service object supports the following methods:

1. get() 2. query() 3. save() 4. remove() 5. delete()
2. **How to enable caching in $http service?**

You can enable caching in $http service by setting configuration property cache to true. When cache is enabled, $http service stores the response from the server in local cache. In this way, next time the response will be served from the cache without sending request to server.

$http.get("http://server/myserviceapi",

{ cache:true }).sucess(function()

{ //TO DO: })

1. **What methods $http service support? Ans. The $http service supports the following methods?**

1. $http.get() 2. $http.head() 3. $http.post() 4. $http.put() 5. $http.delete() 6. $http.jsonp() 7. $http.patch()

**11. What is the difference between $http and $resource?**

$http service is a core Angular service which allows you to make AJAX requests by using GET, HEAD, POST, PUT, DELETE, JSONP and PATCH methods. It is very much like the $.ajax() method in jQuery. It can be used with RESTful and Non-RESTful server-side data sources.

$http is good for quick retrieval of server-side data that doesn’t really need any specific structure or complex behaviors.

$resource warps $http and allows you to interact with RESTful server-side data sources. It requires the ngResource module to be installed which exist in angular-resource.js

$http is good for retrieval of RESTful server-side data sources that might need any specific structure or complex behaviors.

1. **What is difference between value and constant?**

Value and Constant are simple objects which are used to share data globally with in a module.

Value - A value can be a number, string, date-time, array or object. You can also register a function as a value. Values are typically used as configuration which is injected into factories, services or controllers.

//define module var app = angular.module('app', []);

//define value app.value("numberValue", 100); app.value("stringValue", "dotnet-tricks.com"); app.value("objectValue", { name: "dotnet-tricks.com", totalUsers: 120000 });

Constant - A constant is like as value. The difference between a value and a constant service is that constant service can be injected into a module configuration function i.e. config() but value service cannot be.

//define module var app = angular.module('app', []);

//define constant app.constant("mynumberValue", 200); app.constant("mystringValue", "webgeekschool.com");

//module configuration function app.config(function (mynumberValue) { //here value objects can't be injected console.log("Before:" + mynumberValue); mynumberValue = "New Angular Constant Service"; console.log("After:" + mynumberValue); });

**What is the difference between Factory, Service and Provider?**

1. **Factory** - A factory is a simple function which allows you to add some logic before creating the object. It returns the created object

When to use: It is just a collection of functions like a class. Hence, it can be instantiated in different controllers when you are using it with constructor function.

1. **Service** - A service is a constructor function which creates the object using new keyword. You can add properties and functions to a service object by using this keyword. Unlike factory, it doesn’t return anything.

When to use: It is a singleton object. Use it when you need to share a single object across the application.

1. **Provider** - A provider is used to create a configurable service object. It returns value by using $get() function.

When to use: When you need to provide module-wise configuration for your service object before making it available.

**What are different ways to create service in AngularJS?**

There are five ways to create a service as given below:

1. Service 2. Factory 3. Provider 4. Value 5. Constant

**What is Service in AngularJS?**

A service is a reusable singleton object which is used to organize and share code across your app. A service can be injected into controllers, filters, directives.

AngularJS offers several built-in services (like $http, $provide, $resource, $window, $parse) which always start with $ sign.

**What are different states of a form in AngularJS?**

The AngularJS form goes to the following states, starting from the form rendering and when the user has finished the filling of form.

State 1: pristine and invalid - When the form is first time rendered and the user has not interacted with the form yet.

State 2: dirty and invalid - User has interacted with the form, but form validity has not been satisfied, yet.

State 3: dirty and valid - User has finished the filling of form and the entire form validations has been satisfied

**What are different Angular form properties?**

Angular provides properties on form which help you to get information about a form or its inputs and to validate them.

$valid - It is a boolean property that tells whether the form or it's inputs are valid or not. If all containing form and controls are valid, then it will be true, otherwise it will be false.

Syntax

formName.$valid formName.inputFieldName.$valid

$invalid - It is a boolean property that tells whether the form or it's inputs are invalid or not. If at least one containing form and control is invalid then it will be true, otherwise it will be false.

Syntax

formName.$invalid formName.inputFieldName.$invalid

$pristine - It is a boolean property that tells whether the form or it's inputs are unmodified by the user or not. If the form or its inputs are unmodified by the user, then it will be true, otherwise it will be false.

Syntax

formName.inputFieldName.$pristine

$dirty - It is a boolean property that is actually just reverse of pristine i.e. it tells whether the form or it's inputs are modified by the user or not. If the form or its inputs are modified by the user, then it will be true, otherwise it will be false.

Syntax

formName.$dirty formName.inputFieldName.$dirty

$error - This is an object hash which contains references to all invalid controls or forms. It has all errors as keys: where keys are validation tokens (such as required, url or email) and values are arrays of controls or forms that are invalid with given error. For a control, if a validation fails then it will be true, otherwise it will be false.

Syntax

formName.$error formName.inputFieldName.$error

**What is View in AngularJS?**

The view is responsible for presenting your models data to end user. Typically it is the HTML markup which exists after AngularJS has parsed and compiled the HTML to include rendered markup and bindings.

**How to apply validation in AngularJS?**

AngularJS provides you built-in validation directives to validate form client side. This makes your life pretty easy to handle client-side form validations without adding a lot of extra effort. AngularJS form validations are based on the HTML5 form validators.

AngularJS directives for form validation. Here is a list of AngularJS directive which can be applied on an input field to validate its value.

**How to share information between controllers in AngularJS?**

**OR**

**What are the ways to communicate between controllers in AngularJS?**

There are various different ways to share data between controllers in an AngularJS app. The most commonly used are **Scope**, **Service**, **Factory** and **Providers.**

**Does AngularJS support MVC?**

AngularJS is a MVC framework. It does not implement MVC in the traditional way, but rather something closer to MVVM Model-View-ViewModel).

**What is Model in AngularJS?**

Models are plain old JavaScript objects that represent data used by your app. Models are also used to represent your app's current state**.**

**What is ViewModel in AngularJS?**

A view model is an object that provides specific data and methods to maintain specific views. Basically, it is a $scope object which lives within your AngularJS app's controller. A view model is associated with a HTML element with the ng-model and **ng-bind** directives.

**What is Controller in AngularJS?**

The controller defines the actual behavior of your app. It contains business logic for the view and connects the model to view with the help of $scope. A controller is associated with a HTML element with the **ng-controller** directive.

AngularJS supports Single Page Application via multiple views on a single page. To do this AngularJS has provided ng-view and ng-template directives and $routeProvider services.

**ng-view**

ng-view tag simply creates a place holder where a corresponding view (html or ng-template view) can be placed based on the configuration.

**ng-template**

ng-template directive is used to create an html view using script tag. It contains "id" attribute which is used by $routeProvider to map a view with a controller.

**Custom directives:**

Custom directives are used in AngularJS to extend the functionality of HTML. Custom directives are defined using "directive" function. A custom directive simply replaces the element for which it is activated. AngularJS application during bootstrap finds the matching elements and do one time activity using its compile () method of the custom directive then process the element using link () method of the custom directive based on the scope of the directive. AngularJS provides support to create custom directives for following type of elements.

* **Element** directives − Directive activates when a matching element is encountered.
* **Attribute** − Directive activates when a matching attribute is encountered.
* **CSS** − Directive activates when a matching css style is encountered.
* **Comment** − Directive activates when a matching comment is encountered.

**Advanced AngularJS Interview Questions with Answers**

a

* Many newcomers has a question “[What is AngularJS?](http://jharaphula.com/introduction-to-angularjs-for-beginners)”.
* In replay I can say AngularJS is a MVC framework to design web applications. It helps to enrich HTML for a great extend. Using AngularJS you can build structural & [high performance based web applications](http://jharaphula.com/how-to-improve-performance-of-a-web-application). Directive is one of the most powerful [feature in AngularJS](http://jharaphula.com/category/programming-solutions/learn-angularjs-with-examples). AngularJS comes with many in-built directives. Even you can Create your own. AngularJS is powered by Google. Due this is a newly introduced technology, there are lot of vacancies available for AngularJS developers. Are you one among them who is [Seeking a Job](http://jharaphula.com/category/career-job-seekers) in AngularJS. If so, before going to attained interview read our selected Angularjs Interview Questions.
* **What are the key features of AngularJS?**
* AngularJS is a latest technology introduced by Google. The key features of AngularJS are like Scope, Controller, View, Model, Directives, Validation, Filters, Data Binding, Services & Testable. AngularJS supports MVC & MVVM design pattern. Using AngularJS we can do massively Parallel Development. AngularJS Supports Single Page Applications design.
* **Does Angular use the JQuery library?**
* Yes. AngularJS is an advanced language. It supports [JQuery](http://jharaphula.com/top-jquery-interview-questions-with-answers" \t "_blank) library.
* **How to use ng-repeat in AngularJS?**
* Ng-repeat is a per-defined directive in AngularJS. Look at the Example below. Here I used ng-repeat to bind data from JavaScript. In sample data I have two fields name & designation. I am showing these records using ng-repeat in a HTML UL element.

|  |
| --- |
| <html ng-app=”nameDesgApp”>  <head>  <script src=”<https://ajax.googleapis.com/ajax/libs/angularjs/1.0.1/angular.min.js>”></script>  <script type=”text/javascript”>  var nameDesgApp = angular.module(‘nameDesgApp’, []);  nameDesgApp.controller(‘NameDesgCrtl’, function ($scope) {    $scope.employes = [  {‘name’: ‘Swatio Rao’,  ‘designation’: ‘Project Manager’},  {‘name’: ‘Biswabhusan Panda’,  ‘designation’: ‘Module Lead’},  {‘name’: ‘Kumar Abhishek’,  ‘designation’: ‘Senior Team Leader’}  ];  });  </script>  </head>  <body ng-controller=”NameDesgCrtl”>  <ul>  <li ng-repeat=”emp in employes”>{{emp.name}}  <p>{{emp.designation}}</p>  </li>  </ul>  </body>  </html> |

* **How to use Filter in AngularJS?**
* Filter is like Search in AngularJS. In the following code I have 3 records & a search box. If I am doing search filter keyword help to purify the results according to the Query.

|  |
| --- |
| <html ng-app=”nameDesgApp”>  <head>  <script src=”<https://ajax.googleapis.com/ajax/libs/angularjs/1.0.1/angular.min.js>”></script>  <script type=”text/javascript”>  var nameDesgApp = angular.module(‘nameDesgApp’, []);  nameDesgApp.controller(‘NameDesgCrtl’, function ($scope) {  $scope.employes = [  {‘name’: ‘Swatio Rao’,  ‘designation’: ‘Project Manager’},  {‘name’: ‘Biswabhusan Panda’,  ‘designation’: ‘Module Lead’},  {‘name’: ‘Kumar Abhishek’,  ‘designation’: ‘Senior Team Leader’}  ];  });  </script>  </head>  <body ng-controller=”NameDesgCrtl”>  Search: <input ng-model=”query”>  <ul class=”phones”>  <li ng-repeat=”emp in employes | filter:query”>{{emp.name}}  <p>  {{emp.designation}}</p>  </li>  </ul>  </body>  </html> |

* **In AngularJS how will you initialize a select box with options on page load?**
* Using AngularJS ng-init directive. Look at the example below.

|  |
| --- |
| <div ng-app="" ng-init="empName='Raghav'">  <p>Name: <input type="text" ng-model="empName"></p>  <p>You wrote: {{ empName }}</p>  </div> |

* **What are the advantages of using AngularJS?**
  + AngularJS is a latest Technology introduced by Google. It has several advantages in web development. AngularJS supports MVC pattern. Using AngularJS we can do 2 ways data binding. AngularJS has per-defined form validations. AngularJS supports both client & server communication. AngularJS supports Animations.
* **How to use Controller in AngularJS?**
  + Controller is a JavaScript constructor function. When Creating an application we need to provide initial value to the $Scope object of a Controller. We can add property to $Scope object. Before to create controller we have to create a module. Look at the example below how I created Controller in AngularJS.

|  |
| --- |
| var myApp = angular.module('myApp',[]);    myApp.controller('myController', ['$scope', function($scope) {  $scope.welcome = 'Hello!';  }]); |

* The property of a controller we can assign to the DOM object using ng-controller.

|  |
| --- |
| <div ng-controller="myController">  {{ welcome }}  </div> |

* **What are the form Validations AngularJS provides?**
* Client Validations made easy using AngularJS. All input fields can have Required field, Minimum Length, Maximum Length, Matches a Pattern, Email, Number, URL & Custom Validations. Look at the Examples below.
* *Required Field Validation*
* Using Required field validation we can prevent form submission with null value. I mean input fields need to filled by the user. The syntax for required filed validation is as follows.

|  |
| --- |
| <input type="text" required /> |

* *Minimum & Maximum field Length Validations*
* To prevent the input field from less or excess number of characters we use Minimum & Maximum length validation. The AngularJS directive used for Minimum & Maximum length validations are ng-minlength & ng-maxlength. Look at the example below.

|  |
| --- |
| <input type="text" ng-minlength=5 />  <input type="text" ng-maxlength=10 /> |

* *Matches Pattern Validation*
* AngularJS provides ng-pattern directive to validate regular expressions. Look at the example below.

|  |
| --- |
| <input type="text" ng-pattern="[a-zA-Z]" /> |

* *Email Validations*
* To valid an email id AngularJS provides ng-model directive. Using the following line of code we can validate an email id from any input field.

|  |
| --- |
| <input type="email" name="email" ng-model="user.email" /> |

* *Number Validation*
* To validate an input against Number we can use ng-model directive from AngularJS. Look at the example below.

|  |
| --- |
| <input type="number" name="personage" ng-model="user.age" /> |

* *URL Validation*
* To validate an input field for URL we can use the following syntax in AngularJS.

|  |
| --- |
| <input type="url" name="weblink" ng-model="user.facebook\_url" /> |

* **How to create Directive using AngularJS?**
* Directives are designed to use as markers with DOM elements. Markers can be attribute, element name, comment or CSS class. AngularJS provides many in-built directives. Few of them are ngModel, ngView or ngBind. The naming standard directive follows is camel-case. Much like controllers directives are registered on Module. To register a directive we can use module.directive. The time we create directive by default it restricted to attribute. In order to create directive which will triggered by element or class name we need to use restrict. The options are as follows.
* A – only matches attribute name  
  E – only matches element name  
  C – only matches class name
* These restrictions can be combined as required.
* AEC – it matches either attribute or element or class name
* Let us look at the below example [how to create a directive using AngularJS](http://jharaphula.com/fundamental-of-using-directives-in-angularjs-module).

|  |
| --- |
| myAngularApp = angular.module("myAngularApp", []);    myAngularApp.directive('div', function() {  var directive = {};    directive.restrict = 'E'; /\* using this we restrict this directive to elements \*/  directive.template = "My first directive: {{ EnterText }}";    return directive;  }); |

* **What is 2 way data binding in AngularJS?**
* In UI Development to create a template we basically do one way data binding. In case of Angular it is a MVC framework. Internally using one way data binding we merge template & model to display in view. But in 2 way data binding there is a continuous communication in between model & view. Where template is just pointing to the view.
* **What is the CDN link for AngularJS?**
* CDN is a permanent Global space where from we can access the various versions of library file to use directly in our web applications. For AngularJS you can refer the following CDN link.
* <https://ajax.googleapis.com/ajax/libs/angularjs/1.3.0-rc.0/angular.min.js>
* **How to bind MySQL data to HTML5 using AngularJS?**
* In server end create a php file with MySQL query for the required data. Format it in the shape of JSON. Here I did this with the file name Customers\_JSON.php.
* *Customers\_JSON.php*

|  |
| --- |
| [  {  "Name" : "Raghav Meheta",  "City" : "Berlin",  "Country" : "Germany"  },  {  "Name" : "Rakesh Srivastab",  "City" : "Lulea",  "Country" : "Sweden"  },  {  "Name" : "Biswabhusan Panda",  "City" : "Mexico D.F.",  "Country" : "Mexico"  },  {  "Name" : "Kumar Abhishek",  "City" : "Graz",  "Country" : "Austria"  },  {  "Name" : "Raveena Tondon",  "City" : "Madrid",  "Country" : "Spain"  }  ] |

* in App.js create a angular module & controller. Inside the controller use $http.get method to fetch the data from JSON formatted php file. Look at the example below.
* *App.js*

|  |
| --- |
| var myApp = angular.module('myApp', []);    myApp.controller('Controller', function($scope,$http) {  $http.get("Customers\_JSON.php").success(function(response) {  $scope.names = response;  });  }); |

* Finally, In your index.html file use ng-repeat over UL element to bind the response data. Sharing the code below.
* *Index.html*

|  |
| --- |
| <!DOCTYPE html>  <html ng-app="myApp">  <head>  <title>How to bind MySql data using AngularJS?</title>  <script type="text/javascript" src="<https://ajax.googleapis.com/ajax/libs/angularjs/1.3.0-rc.0/angular.min.js>"></script>  <script type="text/javascript" src="app.js"></script>  </head>  <body>  <div ng-controller="Controller">  <ul>  <li ng-repeat="x in names">  {{ x.Name + ', ' + x.Country }}  </li>  </ul>  </div>  </body>  </html> |

* **How to use templateURL in the Directive of AngularJS?**
* To implement one line of HTML template is ok for Directive. If we are going to develop a more functional control depending upon HTML for a better professional approach we need to use templateURL. Look at the example below how I used a html page to display in side an AngularJS directive using templateURL.
* *index.html*

|  |
| --- |
| <!DOCTYPE html>  <html ng-app="myApp">  <head>  <title>Using templateURL in the Directive of AngularJS</title>  <script type="text/javascript" src="<https://ajax.googleapis.com/ajax/libs/angularjs/1.3.0-rc.0/angular.min.js>"></script>  <script type="text/javascript" src="app.js"></script>  </head>  <body>  <div ng-controller="templateUrlDemo">    <div template-Url></div>    </div>  </body>  </html> |

* *app.js*

|  |
| --- |
| var myApp = angular.module('myApp', []);    myApp.directive('templateUrl', function() {  return {  templateUrl: 'App-Template.htm'  };  }); |

* *App-Template.htm*

|  |
| --- |
| <!DOCTYPE html>  <html ng-app>  <body>  This is a sample HTML file.  </body>  </html> |

* **Is AngularJS supports MVVM with MVC?**
* Yes. AngularJS is an Advanced JavaScript framework. It supports both MVVM & MVC design patterns.
* **What is routes in AngularJS?**
* In web development single page application designing is more popular than multi page application designing. Let us assume you have 3 pages in your application mainpage, aboutus & contactus. Using older standard of web development you created 3 html pages mainpage.html, aboutus.html & contactus.html. Depending upon the user requests you are redirecting the user to the respective html page. As an framework designer think for a while if we can keep a container or view in main page where you can display the respective pages depending upon the user request, is not this is a better approach to classical web development. AngularJS Routing & Views did the same. Using Routing & Views in AngularJS you can easily create single page web application with no page refresh on a new page redirect. To achieve this technique ng-view is act like the container & routing helps to load each page in the ng-view depending upon the user request. The shape of request is like the url /mainpage or /aboutus. To have a [demo in AngularJS Routing](http://jharaphula.com/example-of-using-angularjs-routing) Look at the Example.
* **What is the difference between compile & link functions in angularjs?**
* Compile function used for Template DOM manipulation. Hence manipulations that apply to all DOM clones of the template associated with the directive.
* Link function used for registering DOM listeners. A $watch() function allows a directive to be notified of instance scope property changes, which allows the directive to render an updated instance value to the DOM by copying content from the instance scope into the DOM.
* **What is difference between Service & Factory?**
* Factory allows us to add some logic before creating the object we require. It differs from service in a way where it allows us to pass the function which factory then invokes & returns the result. To create factory we use module.factory(‘factoryName’, function); & for service we use module.service(‘serviceName’, function);. If you’re using an object, you could use the factory provider. If you’re using a class you could use the service provider