**1) What is AngularJS?**

AngularJS is a javascript framework used for creating single web page applications .It extends HTML DOM with additional attributes and makes it more responsive to user actions.

**2) Explain what are the key features of AngularJS ?**

The key features of AngularJS are

* Scope
* Controller
* Model
* View
* Services
* Data Binding
* Directives
* Filters
* Testable

**3) Explain what is scope in AngularJS ?**

Scope is a special javascript object which plays the role of joining controller with the views. Scope contains the model data. In controllers, model data is accessed via $scope object.

<script>

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

mainApp.controller("shapeController", function($scope) {

$scope.message = "In shape controller";

$scope.type = "Shape";

});

</script>

Following are the important points to be considered in above example.

* $scope is passed as first argument to controller during its constructor definition.
* $scope.message and $scope.type are the models which are to be used in the HTML page.
* We've set values to models which will be reflected in the application module whose controller is shapeController.
* We can define functions as well in $scope.

Scope Inheritance

Scope are controllers specific. If we defines nested controllers then child controller will inherit the scope of its parent controller.

<script>

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

mainApp.controller("shapeController", function($scope) {

$scope.message = "In shape controller";

$scope.type = "Shape";

});

mainApp.controller("circleController", function($scope) {

$scope.message = "In circle controller";

});

</script>

Following are the important points to be considered in above example.

* We've set values to models in shapeController.
* We've overridden message in child controller circleController. When "message" is used within module of controller circleController, the overridden message will be used.
* <html>
* <head>
* <title>Angular JS Forms</title>
* </head>
* <body>
* <h2>AngularJS Sample Application</h2>
* <div ng-app = "mainApp" ng-controller = "shapeController">
* <p>{{message}} <br/> {{type}} </p>
* <div ng-controller = "circleController">
* <p>{{message}} <br/> {{type}} </p>
* </div>
* <div ng-controller = "squareController">
* <p>{{message}} <br/> {{type}} </p>
* </div>
* </div>
* <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
* <script>
* var mainApp = angular.module("mainApp", []);
* mainApp.controller("shapeController", function($scope) {
* $scope.message = "In shape controller";
* $scope.type = "Shape";
* });
* mainApp.controller("circleController", function($scope) {
* $scope.message = "In circle controller";
* });
* mainApp.controller("squareController", function($scope) {
* $scope.message = "In square controller";
* $scope.type = "Square";
* });
* </script>
* </body>
* </html>

## 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.

<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>

Try it Yourself »

Output:

The rootScope's favorite color:

# blue

The scope of the controller's favorite color:

# red

The rootScope's favorite color is still:

# blue

Notice that controller's color variable does not overwrite the rootScope's color value.

**4) Explain what is services in AngularJS ?**

In AngularJS services are the singleton objects or functions that are used for specific tasks.  It holds some business logic.

AngularJS provides many inbuilt services for example, $http, $route, $window, $location etc.

Exp:

$location service:

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

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

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_services)

## The $http Service

The AngularJS $http service makes a request to the server, and returns a response.

<div ng-app="myApp" ng-controller="myCtrl">   
  
<p>Today's welcome message is:</p>  
<h1>{{myWelcome}}</h1>  
  
</div>  
  
<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 .get method is a shortcut method of the $http service. There are several shortcut methods:

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

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http({  
        method : "GET",  
        url : "welcome.htm"  
    }).then(function mySuccess(response) {  
        $scope.myWelcome = response.data;  
    }, function myError(response) {  
        $scope.myWelcome = response.statusText;  
    });  
});

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.

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;   
    });  
});

## The $timeout Service

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

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.

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

There are two ways to create a service.

* factory
* service

## Using factory method

Using factory method, we first define a factory and then assign method to it.

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

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b

}

return factory;

});

## Using service method

Using service method, we define a service and then assign method to it. We've also injected an already available service to it.

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

Exp:

<html>

<head>

<title>Angular JS Services</title>

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

</head>

<body>

<h2>AngularJS Sample Application</h2>

<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", []);

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b

}

return factory;

});

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

mainApp.controller('CalcController', function($scope, CalcService) {

$scope.square = function() {

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

}

});

</script>

</body>

</html>

**5) Explain what is Angular Expression? Explain what is key difference between angular expressions and JavaScript expressions?**

Expressions are used to bind application data to html. Expressions are written inside double braces like {{ expression}}. Expressions behaves in same way as ng-bind directives. AngularJS application expressions are pure javascript expressions and outputs the data where they are used.

## Using numbers

<p>Expense on Books : {{cost \* quantity}} Rs</p>

## Using strings

<p>Hello {{student.firstname + " " + student.lastname}}!</p>

## Using object

<p>Roll No: {{student.rollno}}</p>

## Using array

<p>Marks(Math): {{marks[3]}}</p>

Exp:

<div ng-app="" ng-init="myCol='lightblue'">  
  
<input style="background-color:{{myCol}}" ng-model="myCol" value="{{myCol}}">  
  
</div>

The key difference between the JavaScript expressions and Angular expressions

* **Context :** In Angular, the expressions are evaluated against a scope object, while the Javascript expressions are evaluated against the global window
* **Forgiving:** In Angular expression evaluation is forgiving to null and undefined, while in Javascript undefined properties generates TypeError or ReferenceError
* **No Control Flow Statements:** Loops, conditionals or exceptions cannot be used in an angular expression
* **Filters:** To format data before displaying it you can use filters

**6) With options on page load how you can initialize a select box ?**

You can initialize a select box with options on page load by using **ng-init** directive

* <div ng-controller = “ apps/dashboard/account ” ng-switch
* On = “! ! accounts” ng-init = “ loadData ( ) ”>

**7) Explain what are directives ? Mention some of the most commonly used directives in AngularJS application ?**

AngularJS directives are extended HTML attributes with the prefix ng-.

The ng-app directive initializes an AngularJS application.

The ng-init directive initializes application data.

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

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

The ng-app directive also tells AngularJS that the <div> element is the "owner" of the AngularJS application.

## Create New Directives

In addition to all the built-in AngularJS directives, you can create your own directives.

New directives are created by using the .directive function.

To invoke the new directive, make an HTML element with the same tag name as the new directive.

When naming a directive, you must use a camel case name, w3TestDirective, but when invoking it, you must use -separated name, w3-test-directive:

### **Example**

<body ng-app="myApp">  
  
<w3-test-directive></w3-test-directive>  
  
<script>  
var app = angular.module("myApp", []);  
app.directive("w3TestDirective", function() {  
    return {  
        template : "<h1>Made by a directive!</h1>"  
    };  
});  
</script>  
  
</body>

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_tagname)

You can invoke a directive by using:

* Element name
* Attribute
* Class
* Comment

The examples below will all produce the same result:

Element name

<w3-test-directive></w3-test-directive>

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_element)

Attribute

<div w3-test-directive></div>

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_attribute)

Class

<div class="w3-test-directive"></div>

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_class)

Comment

<!-- directive: w3-test-directive -->

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_comment)

## Restrictions

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

### **Example**

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>"  
    };  
});

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_restrictions)

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.

For use these directive in comment:

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>

<body ng-app="myApp">

<!-- directive: w3-test-directive -->

<script>

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

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

return {

restrict : "M",

replace : true,

template : "<h1>Made by a directive!</h1>"

};

});

</script>

<p><strong>Note:</strong> We've added the <strong>replace</strong> property in this example, otherwise the comment would be invisible.</p>

<p><strong>Note:</strong> You must add the value "M" to the <strong>restrict</strong> property to be able to invoke the directive from a comment.</p>

</body>

</html>

**Q=What is the Module?**

Ans=The module is a container for the different parts of an application.

Controllers always belong to a module.

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

**8) Mention what are the advantages of using AngularJS ?**

AngularJS has several advantages in web development.

* AngularJS supports MVC pattern
* Can do two ways data binding using AngularJS
* It has per-defined form validations
* It supports both client server communication
* It supports animations

**9) Explain what Angular JS routes does ?**

## What is Routing in AngularJS?

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.

### **Example:**

Navigate to "red.htm", "green.htm", and "blue.htm":

<body ng-app="myApp">  
  
<p><a href="#/">Main</a></p>  
  
<a href="#red">Red</a>  
<a href="#green">Green</a>  
<a href="#blue">Blue</a>  
  
<div ng-view></div>  
  
<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>

## What do I Need?

To make your applications ready for routing, you must include the AngularJS Route module:

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

Then you must add the ngRoute as a dependency in the application module::

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

Now your application has access to the route module, which provides the $routeProvider.

Use the $routeProvider to configure different routes in your application:

app.config(function($routeProvider) {  
  $routeProvider  
  .when("/", {  
    templateUrl : "main.htm"  
  })  
  .when("/red", {  
    templateUrl : "red.htm"  
  })  
  .when("/green", {  
    templateUrl : "green.htm"  
  })  
  .when("/blue", {  
    templateUrl : "blue.htm"  
  });  
});

## Where Does it Go?

Your application needs a container to put the content provided by the routing.

This container is the ng-view directive.

There are three different ways to include the ng-view directive in your application:

### **Example:**

<div ng-view></div>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_routing_view1)

Or

<ng-view></ng-view>

Or

<div class="ng-view"></div>

Applications can only have one ng-view directive, and this will be the placeholder for all views provided by the route.

## $routeProvider

With the $routeProvider you can define what page to display when a user clicks a link.

### **Example:**

Define a $routeProvider:

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"  
    });  
});

Define the $routeProvider using the config method of your application. Work registered in the config method will be performed when the application is loading.

## Controllers

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

### **Example:**

Add controllers:

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";  
});

The "london.htm" and "paris.htm" are normal HTML files, which you can add AngularJS expressions as you would with any other HTML sections of your AngularJS application.

The files looks like this:

[london.htm](http://www.w3schools.com/angular/london.htm)

<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>

[paris.htm](http://www.w3schools.com/angular/paris.htm)

<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>

## 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.

### **Example:**

Write templates:

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>"  
    });  
});

## The otherwise method

In the previous examples we have used the when method of the $routeProvider.

You can also use the otherwise method, which is the default route when none of the others get a match.

### **Example:**

If neither the "Banana" nor the "Tomato" link has been clicked, let them know:

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>"  
    });  
});

for more :

<http://viralpatel.net/blogs/angularjs-routing-and-views-tutorial-with-example/>

**10)  Explain what is data binding in AngularJS ?**

Ans=Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.

## One-Way Data Binding

You can use the ng-bind directive, which will bind the innerHTML of the element to the specified model property:

### **Example**

<p ng-bind="firstname"></p>

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_databinding)

You can also use double braces {{ }} to display content from the model:

### **Example**

<p>First name: {{firstname}}</p>

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_databinding_braces)

## Two-Way Data 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.

Use the ng-model directive to bind data from the model to the view on HTML controls (input, select, textarea)

### **Example**

<input ng-model="firstname">

[Try it Yourself »](https://www.w3schools.com/angular/tryit.asp?filename=try_ng_databinding_ng-model)

The ng-model directive provides a two-way binding between the model and the view.

# **Q=**[**Can an AngularJS controller inherit from another controller in the same module?**](https://stackoverflow.com/questions/18461263/can-an-angularjs-controller-inherit-from-another-controller-in-the-same-module)

Ans=Yes, it can but you have to use the $controller service to instantiate the controller instead:-

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

app.controller('ParentCtrl', function($scope) {

// I'm the sibling, but want to act as parent

});

app.controller('ChildCtrl', function($scope, $controller) {

$controller('ParentCtrl', {$scope: $scope}); //This works

});

<https://stackoverflow.com/questions/18461263/can-an-angularjs-controller-inherit-from-another-controller-in-the-same-module>

# Q=[**Angular Js difference between ng-init and data-ng-init**](https://stackoverflow.com/questions/36027539/angular-js-difference-between-ng-init-and-data-ng-init)?

**OR difference between n-app and data-n-app?**

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

Ideally there is no difference between the two in terms of functionality but with only validation.

After the inception of HTML5, the code editor like Visual Studio, highlights the 'ng-', as something which is invalid. But actually it's valid, so there is a way to make the code editor understand that the attribute of AngularJS is a valid one by prefixing it with 'data-ng-\*'.

So when used the prefix, in any HTML5 code editor, it doesn't underlines the attributes and treats them as valid.