**AngularJS**

AngularJS is an open source JavaScript MVC framework for web application or web sites. It extends the HTML and makes it dynamic. AngularJS can be used to create Single Page Applications.

What is AngularJS?

AngularJS is a client side JavaScript MVC framework to develop a dynamic web application. AngularJS was originally started as a project in Google but now, it is open source framework.

AngularJS is entirely based on HTML and JavaScript, so there is no need to learn another syntax or language.

AngularJS is also called just "Angular".

AngularJS changes static HTML to dynamic HTML. It extends the ability of HTML by adding built-in attributes and components and also provides an ability to create

custom attributes using simple JavaScript.AngularJS website - [https://angularjs.org](https://angularjs.org/)

[](https://www.tutorialsteacher.com/Content/images/ng/angular-website.png)

## AngularJS Example

The following is a simple AngularJS example that changes a label to whatever you type in the textbox.

AngularJS Example:

<!DOCTYPE html>

<html>

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body **ng-app**>

Enter Your Name: <input type="text" **ng-model="name"** /> <br />

Hello <label **ng-bind="name"**></label>

</body>

</html>

## Advantages of AngularJS

1. Open source JavaScript MVC framework.
2. Supported by Google
3. No need to learn another scripting language. It's just pure JavaScript and HTML.
4. Supports separation of concerns by using MVC design pattern.
5. Built-in attributes (directives) makes HTML dynamic.
6. Easy to extend and customize.
7. Supports Single Page Application.
8. Uses Dependency Injection.

# Setup AngularJS Development Environment

We need the following tools to setup a development environment for AngularJS:

1. AngularJS Library
2. Editor/IDE
3. Browser
4. Web server

## AngularJS Library

To download AngularJS library, go to [angularjs.org](https://angularjs.org/) -> click download button, which will open the following popup.

[](https://www.tutorialsteacher.com/Content/images/ng/setup-env-1.png)Download AngularJS Library

Select the required version from the popup and click on download button in the popup.

CDN: You can include AngularJS library from CDN url - <https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js>

First AngularJS Application

Let's create a simple AngularJS web application step by step and understand the basic building blocks of AngularJS.

1. First, create an HTML document with <head> and <body> elements, as show below.

Example: HTML Template

<!DOCTYPE html>

<html>

<head>

</head>

<body>

</body>

</html>

2. Include angular.js file in the head section (you have learned how to download angular library in the previous section). You can take a reference from the CDN also. (all the examples in this tutorials will use CDN reference.)

Example: Include AngularJS Library

<!DOCTYPE html>

<html>

<head>

<title>First AngularJS Application</title>

<script src= "~/Scripts/angular.js"></script>

</head>

<body>

</body>

</html>

3. Here, we will be creating a simple multiplier application which will multiply two numbers and display the result. User will enter two numbers in two separate textboxes and the result will be displayed immediately, as shown below.

[](https://www.tutorialsteacher.com/Content/images/ng/first-ng-app-ui.png)First AngularJS Application

The following is the HTML code with AngularJS for the above multiplier example.

Example: First AngularJS Application

<!DOCTYPE html>

<html>

<head>

<title>First AngularJS Application</title>

<script src= "~/Scripts/angular.js"></script>

</head>

<body **ng-app** >

<h1>First AngularJS Application</h1>

Enter Numbers to Multiply:

<input type="text" **ng-model="Num1"** /> x <input type="text" **ng-model="Num2"** />

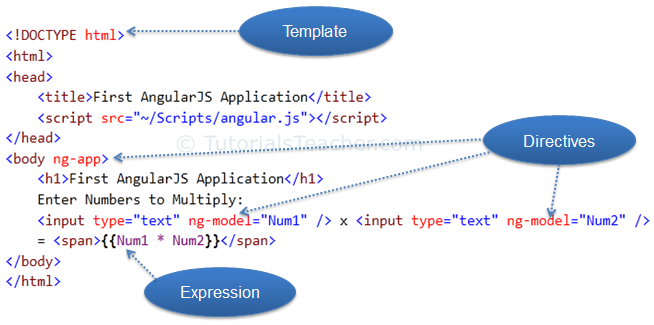
= <span>**{{Num1 \* Num2}}**</span>

</body>

</html>

The above example is looks like HTML code with some strange attributes and braces such as ng-app, ng-model, and {{ }}. These built-in attributes in AngularJS are called directives.

The following figure illustrates the AngularJS building blocks in the above example.

[](https://www.tutorialsteacher.com/Content/images/ng/first-ng-app.png)First AngularJS Application

## Template

In AngularJS, a template is HTML with additional markups. AngularJS compiles templates and renders the resultant HTML.

## Directive

Directives are markers (attributes) on a DOM element that tell AngularJS to attach a specific behavior to that DOM element or even transform the DOM element and its children.

Most of the directives in AngularJS are starting with **ng**. It stands for Angular. We have applied ng-app and ng-model directive in the above example.

**ng-app**: The ng-app directive is a starting point. If AngularJS framework finds ng-app directive anywhere in the HTML document then it bootstraps (initializes) itself and compiles the HTML template.

**ng-model**: The ng-model directive binds HTML element to a property on the [$scope](https://www.tutorialsteacher.com/angularjs/angularjs-scope) object. You will learn about this model later but for now let us consider this as a model property.

In the above example, we have included ng-model directive to both the textboxes with different names Num1 and Num2. AngularJS framework will create two properties called Num1 and Num2 in the scope and will assign a value that we type into textboxes.

## Expression

An expression is like JavaScript code which is usually wrapped inside double curly braces such as {{ expression }}. AngularJS framework evaluates the expression and produces a result. In the above example, {{ Num1 \* Num2}} will simply display the product of Num1 and Num2.

The ng-app Directive

The **ng-app** directive is a starting point of AngularJS Application. It initializes the AngularJS framework automatically. AngularJS framework will first check for ng-app directive in a HTML document after the entire document is loaded and if ng-app is found, it bootstraps itself and compiles the HTML template.

Compiling HTML in AngularJS means attaching event listeners to the HTML to make it interactive.

Typically ng-app directives should be placed at the root of an HTML document e.g. <html> or <body> tag, so that it can control the entire DOM hierarchy. However, you can place it in any DOM element.

The AngularJS framework will only process the DOM elements and its child elements where the ng-app directive is applied. Consider the following example.

Example: ng-app placement

<!DOCTYPE html>

<html>

<head>

<title>ng-app Directive</title>

<script src="../Scripts/angular.min.js"></script>

</head>

<body >

<div>

{{2/2}}

</div>

<div id="myDiv" **ng-app**>

{{5/2}}

<div>

{{10/2}}

</div>

</div>

<div>

{{2/2}}

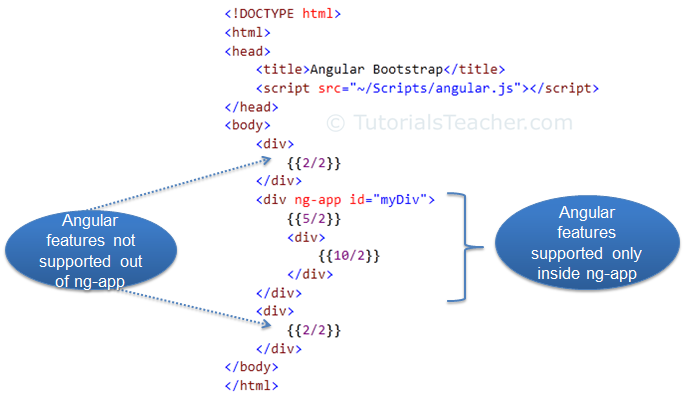
</div>

</body>

</html>

In the above example, ng-app directive is placed in the div element whose id is "myDiv". Therefore, AngularJS will only compile myDiv and its child elements. It will not compile the parent or sibling elements of myDiv.

The following figure illustrates the above example.

[](https://www.tutorialsteacher.com/Content/images/ng/ng-app.png)Bootstrap

Note that multiple ng-app directives are **NOT** allowed in a single HTML document.

## ng-app with Module name

The ng-app directive can also specify an application [module](https://www.tutorialsteacher.com/angularjs/modules-in-angularjs) name. This application module separates different parts of your application such as controllers, services, filters etc.

Example: ng-app with App Module

<!DOCTYPE html>

<html>

<head>

<title>ng-app Directive</title>

<script src="~/Scripts/angular.js"></script>

</head>

<body ng-app="myAngularApp">

<div>

{{2/2}}

</div>

<div>

{{5/2}}

<div>

{{10/2}}

</div>

</div>

<script>

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

</script>

</body></html>

AngularJS Expression

AngularJS expression is like JavaScript expression surrounded with braces - {{ expression }}. AngularJS evaluates the specified expression and binds the result data to HTML.

AngularJS expression can contain literals, operators and variables like JavaScript expression. For example, an expression {{2/2}} will produce the result 1 and will be bound to HTML.

Example: Expression

<!DOCTYPE html>

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body >

<h1>AngularJS Expression Demo:</h1>

<div ng-app>

2 + 2 = {{2 + 2}} <br />

2 - 2 = {{2 - 2}} <br />

2 \* 2 = {{2 \* 2}} <br />

2 / 2 = {{2 / 2}}

</div>

</body>

</html>

Result:

2 + 2 = 4

2 - 2 = 0

2 \* 2 = 4

2 / 2 = 1

AngularJS expression is like JavaScript code expression except for the following differences:

1. AngularJS expression cannot contain conditions, loops, exceptions or regular expressions e.g. if-else, ternary, for loop, while loop etc.
2. AngularJS expression cannot declare functions.
3. AngularJS expression cannot contain comma or void.
4. AngularJS expression cannot contain return keyword.

AngularJS expression contains literals of any data type.

Example: Expression

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body >

<h1>AngularJS Expression Demo:</h1>

<div ng-app>

{{"Hello World"}}<br />

{{100}}<br />

{{true}}<br />

{{10.2}}

</div>

</body>

</html>

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=ng-12)

Result:

Hello World

100

True

10.2

AngularJS expression can contain arithmetic operators which will produce the result based on the type of operands, similar to JavaScript:

Example: Expression

<!DOCTYPE html>

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body >

<div ng-app>

{{"Hello" + " World"}}<br />

{{100 + 100 }}<br />

{{true + false}}<br />

{{10.2 + 10.2}}<br />

</div>

</body>

</html>

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=ng-13)

Result:

Hello World

200

1

20.4

AngularJS expression can contain variables declared via ng-init directive. The ng-init directive is used to declare AngularJS application variables of any data type.

Example: Expression

<!DOCTYPE html>

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body >

<div ng-app ng-init="greet='Hello World!'; amount= 10000;rateOfInterest = 10.5; duration=10; myArr = [100, 200]; person = { firstName:'Steve', lastName :'Jobs'}">

{{ (amount \* rateOfInterest \* duration)/100 }}<br />

{{myArr[1]}} <br />

{{person.firstName + " " + person.lastName}}

</div>

</body>

</html>

Result:

10500

200

Steve Jobs

# AngularJS Directives

We used directives in our [first AngularJS application](https://www.tutorialsteacher.com/angularjs/first-angularjs-application) section. Here, we will learn directives in detail.

Directives are markers on a DOM element that tell AngularJS to attach a specified behavior to that DOM element or even transform the DOM element and its children. In short, it extends the HTML.

Most of the directives in AngularJS are starting with ng- where ng stands for Angular. AngularJS includes various built-in directives. In addition to this, you can create custom directives for your application.

## ng-app

The ng-app directive initializes AngularJS and makes the specified element a root element of the application. Visit [ng-app](https://www.tutorialsteacher.com/angularjs/angularjs-ng-app-directive) section for more information.

## ng-init

The ng-init directive can be used to initialize variables in AngularJS application.

The following example demonstrates ng-init directive that initializes variable of string, number, array, and object.

Example: ng-init

<!DOCTYPE html>

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body >

<div ng-app ng-init="greet='Hello World!'; amount= 100; myArr = [100, 200]; person = { firstName:'Steve', lastName :'Jobs'}">

{{amount}} <br />

{{myArr[1]}} <br />

{{person.firstName}}

</div>

</body>

</html>

Result:

100

200

Steve

In the above example, we initialized variables of string, number, array and object. These variables can be used anywhere in the DOM element hierarchy where it is declared e.g variables in the above example cannot be used out of <div> element.

## ng-model

The ng-model directive is used for two-way data binding in AngularJS. It binds <input>, <select> or <textarea> elements to a specified property on the [$scope](https://www.tutorialsteacher.com/angularjs/angularjs-scope) object. So, the value of the element will be the value of a property and vica-versa.

Example: ng-model

<!DOCTYPE html>

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body ng-app>

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

<div>

Hello {{name}}

</div>

</body>

</html>

The property set via ng-model can be accessed in a controller using $scope object. We will look at it in the next section.

Note : Variables initialized in ng-init are different from the properties defined using ng-model directive. The variables initialized in ng-init are not attached to $scope object whereas ng-model properties are attached to $scope object.

## ng-bind

The ng-bind directive binds the model property declared via $scope or ng-model directive or the result of an expression to the HTML element. It also updates an element if the value of an expression changes.

Example: ng-bind

<!DOCTYPE html>

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body ng-app="">

<div>

5 + 5 = <span ng-bind="5 + 5"></span> <br />

Enter your name: <input type="text" ng-model="name" /><br />

Hello <span ng-bind="name"></span>

</div>

</body>

</html>

In the above example, ng-bind directive binds a result of an expression "5 + 5" to the <span>. The same way, it binds a value of a model property "name" to the <span>. The value of "name" property will be the value entered in a textbox.

## ng-repeat

The ng-repeat directive repeats HTML once per each item in the specified array collection.

Example:

<!DOCTYPE html>

<html>

<head>

<script src="~/Scripts/angular.js"></script>

<style>

div {

border: 1px solid green;

width: 100%;

height: 50px;

display: block;

margin-bottom: 10px;

text-align:center;

background-color:yellow;

}

</style>

</head>

<body ng-app="" ng-init="students=['Bill','Steve','Ram']">

<ol>

<li ng-repeat="name in students">

{{name}}

</li>

</ol>

<div ng-repeat="name in students">

{{name}}

</div>

</body>

</html>

In the above example, ng-repeat is used with students array. It creates <li> element for each item in the students array. Using the same way it repeats the <div> element.

AngularJS Controller

The controller in AngularJS is a JavaScript function that maintains the application data and behavior using [$scope](https://www.tutorialsteacher.com/angularjs/angularjs-scope) object.

You can attach properties and methods to the $scope object inside a controller function, which in turn will add/update the data and attach behaviours to HTML elements. The $scope object is a glue between the controller and HTML.

The ng-controller directive is used to specify a controller in HTML element, which will add behavior or maintain the data in that HTML element and its child elements.

The following example demonstrates attaching properties to the $scope object inside a controller and then displaying property value in HTML.

Example: AngularJS Controller

<!DOCTYPE html>

<html >

<head>

<title>AngualrJS Controller</title>

<script src="~/Scripts/angular.js"></script>

</head>

<body ng-app="myNgApp">

<div ng-controller="myController">

{{message}}

</div>

<script>

var ngApp = angular.module('myNgApp', []);

ngApp.controller('myController', function ($scope) {

$scope.message = "Hello World!";

});

</script>

</body>

</html>

Result:

Hello World!

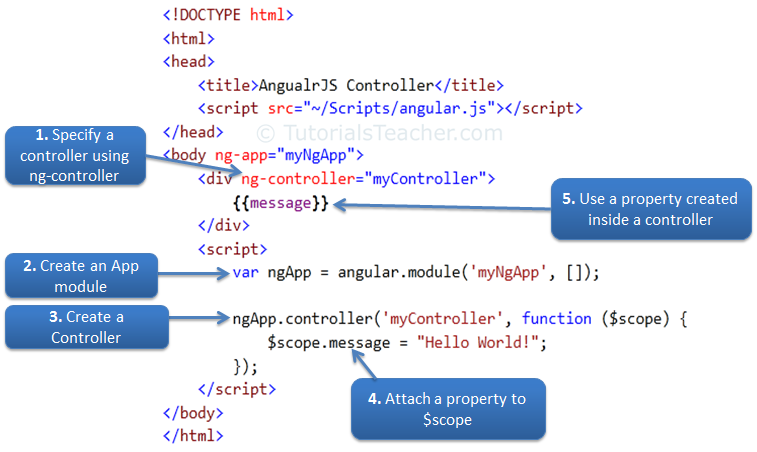
In the above example, ng-controller="myController" directive is applied to the <div> element where "myController" is the name of the controller. Inside div element, we have specified {{message}} expression.

The $ sign is used as prefix in all the built-in objects in AngularJS, so that we can differentiate AngularJS built-in objects and other objects.

Now, to create "myController", we need to create an application module. The module defines an application and keeps its parts like controllers, services etc. out of global scope. (You will learn about the module in the next section.) After creating a module, we add a controller function using the controller() method where the first parameter should be the name of the controller and second parameter should be a function for the controller. The controller function includes $scope parameter which will be injected by AngularJS framework.

Note : AngularJS framework injects $scope object to each controller function. It also injects other services if included as a parameter of controller function.

The following figure illustrates the above example.

[](https://www.tutorialsteacher.com/Content/images/ng/ng-controller.png)

Steps to create an AngularJS Controller

# AngularJS Modules

A module in AngularJS is a container of the different parts of an application such as controller, service, filters, directives, factories etc. It supports separation of concern using modules.

AngularJS stops polluting global scope by containing AngularJS specific functions in a module.

## Application Module

An AngularJS application must create a top level application module. This application module can contain other modules, controllers, filters, etc.

Example: Create Application Module

<!DOCTYPE html>

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body ng-app="myApp">

@\* HTML content \*@

<script>

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

</script>

</body>

</html>

In the above example, the angular.module() method creates an application module, where the first parameter is a module name which is same as specified by ng-app directive.The second parameter is an array of other dependent modules []. In the above example we are passing an empty array because there is no dependency.

Note: The angular.module() method returns specified module object if no dependency is specified. Therefore, specify an empty array even if the current module is not dependent on other module.

Now, you can add other modules in the myApp module.

The following example demonstrates creating controller module in myApp module.

Example:Create Controller Module

<!DOCTYPE html>

<html >

<head>

<script src="~/Scripts/angular.js"></script>

</head>

<body ng-app="myApp">

<div ng-controller="myController">

{{message}}

</div>

<script>

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

myApp.controller("myController", function ($scope) {

$scope.message = "Hello Angular World!";

});

</script>

</body>

</html>

In the above example, we have created a controller named "myController" using myApp.controller() method. Here, myApp is an object of a module, and controller() method creates a controller inside "myApp" module. So, "myController" will not become a global function.