**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.

[](http://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>

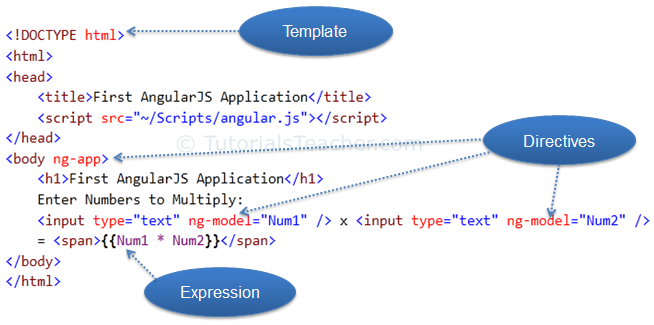
</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-3)

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.

[](http://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](http://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 following table lists all the important concepts in AngularJS.

| **Concept** | **Description** |
| --- | --- |
| Template | HTML with additional markup |
| Directives | Extends the HTML with custom attributes and elements |
| Model | The data shown to the user in the view and with which the user interacts |
| Scope | A context where the model is stored so that controllers, directives and expressions can access it |
| Expressions | Executes JavaScript code inside brackets {{ }}. |
| Compiler | Parses the template and instantiates directives and expressions |
| Filter | Formats the value of an expression for display to the user |
| View | what the user sees (the DOM) |
| Data Binding | Sync data between the model and the view |
| Controller | Maintains the application data and business logic |
| Module | a container for different parts of an app including controllers, services, filters, directives which configure the Injector |
| Service | Reusable business logic, independent of views |
| Dependency Injection | Creates and wires objects and functions |
| Injector | Dependency injection container |

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.

http://www.tutorialsteacher.com/Content/images/tips.pngCompiling 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.

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>

**AngularJS Directives:**

We used directives in our [first AngularJS application](http://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.

The following table lists the important built-in AngularJS directives.

| **Directive** | **Description** |
| --- | --- |
| ng-app | Auto bootstrap AngularJS application. |
| ng-init | Initializes AngularJS variables |
| ng-model | Binds HTML control's value to a property on the $scope object. |
| ng-controller | Attaches the controller of MVC to the view. |
| ng-bind | Replaces the value of HTML control with the value of specified AngularJS expression. |
| ng-repeat | Repeats HTML template once per each item in the specified collection. |
| ng-show | Display HTML element based on the value of the specified expression. |
| ng-readonly | Makes HTML element read-only based on the value of the specified expression. |
| ng-disabled | Sets the disable attribute on the HTML element if specified expression evaluates to true. |
| ng-if | Removes or recreates HTML element based on an expression. |
| ng-click | Specifies custom behavior when an element is clicked. |

**ng-app:**

The ng-app directive initializes AngularJS and makes the specified element a root element of the application. Visit [ng-app](http://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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-21)

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](http://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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-22)

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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-23)

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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-24)

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.

**ng-if:**

The ng-if directive creates or removes an HTML element based on the Boolean value returned from the specified expression. If an expression returns true then it recreates an element otherwise removes an element from the HTML document.

**ng-readonly:**

The ng-readonly directive makes an HTML element read-only, based on the Boolean value returned from the specified expression. If an expression returns true then the element will become read-only, otherwise not.

**ng-disabled:**

The ng-disabled directive disables an HTML element, based on the Boolean value returned from the specified expression. If an expression returns true the element will be disabled, otherwise not.

The following example demonstrates ng-if, ng-readonly, and ng-disabled directives.

Example: ng-if, ng-readonly, ng-disabled

<!DOCTYPE html>

<html>

<head>

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

<style>

div {

width: 100%;

height: 50px;

display: block;

margin: 15px 0 0 10px;

}

</style>

</head>

<body ng-app ng-init="checked=true" >

Click Me: <input type="checkbox" ng-model="checked" /> <br />

<div>

New: <input ng-if="checked" type="text" />

</div>

<div>

Read-only: <input ng-readonly="checked" type="text" value="This is read-only." />

</div>

<div>

Disabled: <input ng-disabled="checked" type="text" value="This is disabled." />

</div>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-25)

**Directive Syntax:**

AngularJS directives can be applied to DOM elements in many ways. It is not mandatory to use ng- syntax only.

Directive can start with x- or data-, for example ng-model directive can be written as data-ng-model or x-ng-model.

Also, the - in the directive can be replaced with : or \_ or both. For example, ng-model can be written as ng\_model or ng:model. It can also be a mix with data- or x-.

The following example demonstrates all the rules of a directive syntax.

Example: Directives syntax variation

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app>

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

data-ng-bind: <span data-ng-bind="name"></span><br />

data-ng:bind: <span data-ng:bind="name"></span><br />

data:ng:bind: <span data:ng:bind="name"></span><br />

x:ng:bind: <span x:ng:bind="name"></span><br />

ng:bind: <span ng:bind="name"></span><br />

x-ng-bind: <span x-ng-bind="name"></span><br />

x\_ng\_bind: <span x\_ng\_bind="name"></span><br />

ng\_bind: <span ng\_bind="name"></span>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-26)

**AngularJS Controller:**

The controller in AngularJS is a JavaScript function that maintains the application data and behavior using [$scope](http://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>

**Scope in AngularJS:**

The $scope in an AngularJS is a built-in object, which contains application data and methods. You can create properties to a $scope object inside a controller function and assign a value or function to it.

The $scope is glue between a controller and view (HTML). It transfers data from the controller to view and vice-versa.

[](http://www.tutorialsteacher.com/Content/images/ng/ng-scope.png)Scope

As we have seen in the controller section, we can attach properties and methods to the $scope object inside controller function. The view can display $scope data using an expression, ng-model, or ng-bind directive, as shown below.

Example: $scope

<!DOCTYPE html>

<html >

<head>

<title>AngualrJS $scope object</title>

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

</head>

<body ng-app="myNgApp">

<div ng-controller="myController">

Message: <br />

{{message}}<br />

<span ng-bind="message"></span> <br />

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

</div>

<script>

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

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

$scope.message = "Hello World!";

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-41)

AngularJS creates and injects a different $scope object to each controller in an application. So, the data and methods attached to $scope inside one controller cannot be accessed in another controller. With the nested controller, child controller will inherit the parent controller's scope object. Therefore, child controller can access properties added in parent controller but parent controller cannot access properties added in child controller.

**Note:** The ng-model directive is used for two-way data binding. It transfers the data from controller to view and vice-versa. An expression and ng-bind directive transfers data from controller to view but not vice-versa.

**$rootScope:**

An AngularJS application has a single $rootScope. All the other $scope objects are child objects.

The properties and methods attached to $rootScope will be available to all the controllers.

The following example demonstrates the $rootScope and $scope object.

Example: $rootScope & $scope

<!DOCTYPE html>

<html>

<head>

<title>AngualrJS Controller</title>

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

</head>

<body ng-app="myNgApp">

<div ng-controller="parentController">

Controller Name: {{controllerName}} <br />

Message: {{message}} <br />

<div style="margin:10px 0 10px 20px;" ng-controller="childController">

Controller Name: {{controllerName}} <br />

Message: {{message}} <br />

</div>

</div>

<div ng-controller="siblingController">

Controller Name: {{controllerName}} <br />

Message: {{message}} <br />

</div>

<script>

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

ngApp.controller('parentController', function ($scope, $rootScope) {

$scope.controllerName = "parentController";

$rootScope.message = "Hello World!";

});

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

$scope.controllerName = "childController";

});

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

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-42)

Result:

Controller Name: parentController

Message: Hello World!

Controller Name: childController

Message: Hello World!

Controller Name:

Message: Hello World!

**AngularJS Events:**

AngularJS includes certain directives which can be used to provide custom behavior on various DOM events, such as click, dblclick, mouseenter etc.

The following table lists AngularJS event directives.

| **Event Directive** |  |  |
| --- | --- | --- |
| ng-blur | ng-keydown | ng-mouseleave |
| ng-change | ng-keyup | ng-mousemove |
| ng-click | ng-keypress | ng-mouseover |
| ng-dblclick | ng-mousedown | ng-mouseup |
| ng-focus | ng-mouseenter |  |

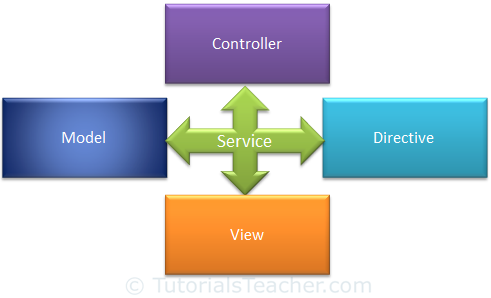
**AngularJS Service:**

AngularJS services are JavaScript functions for specific tasks, which can be reused throughout the application.

http://www.tutorialsteacher.com/Content/images/tips.pngAngularJS built-in services starts with $, same as other built-in objects.

AngularJS includes services for different purposes. For example, $http service can be used to send an AJAX request to the remote server. AngularJS also allows you to create custom service for your application.

Most AngularJS services interact with the controller, model or custom directives. However, some services interact with view (UI) also for UI specific tasks.

[](http://www.tutorialsteacher.com/Content/images/ng/ng-service.png)Services

The following table lists all the built-in AngularJS services.

|  |  |  |  |
| --- | --- | --- | --- |
| $anchorScroll | $exceptionHandler | $interval | $rootScope |
| $animate | $filter | $locale | $sceDelegate |
| $cacheFactory | $httpParamSerializer | $location | $sce |
| $templateCache | $httpParamSerializerJQLike | $log | $templateRequest |
| $compile | $http | $parse | $timeout |
| $controller | $httpBackend | $q | $window |
| $document | $interpolate | $rootElement |  |

All the Angular services are **lazy instantiated** and **singleton**. It means AngularJS framework instantiates a service when an application component depends on it. Also, all the components share the same instance of a service.

**AngularJS Filters:**

AngularJS Filters allow us to format the data to display on UI without changing original format.

Filters can be used with an expression or directives using pipe | sign.

{{expression | filterName:parameter }}

Angular includes various filters to format data of different data types. The following table lists important filters.

| **Filter Name** | **Description** |
| --- | --- |
| Number | Formats a numeric data as text with comma and fraction. |
| Currency | Formats numeric data into specified currency format and fraction. |
| Date | Formats date to string in specified format. |
| Uppercase | Converts string to upper case. |
| Lowercase | Converts string to lower case. |
| Filter | Filters an array based on specified criteria and returns new array. |
| orderBy | Sorts an array based on specified predicate expression. |
| Json | Converts JavaScript object into JSON string |
| limitTo | Returns new array containing specified number of elements from an existing array. |

**Number Filter:**

A number filter formats numeric data as text with comma and specified fraction size.

{{ number\_expression | number:fractionSize}}

If a specified expression does not return a valid number then number filter displays an empty string.

The following example demonstrates how to use number filter with number expression or a model property.

**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.