<http://www.tutorialsteacher.com/angularjs/angularjs-tutorials>

AngularJS Tutorials

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.

These tutorials will help you learn the essentials of AngularJS starting from the basics to an advanced level. These tutorials are broken down into sections, where each section contains a number of related topics that are packed with easy to understand explanations, real-world examples, useful tips, informative notes and a "points to remember" section.

Try it:

Each tutorial includes practical examples. You can edit and see the result real time with *Code Editor*. Click on 'Try it' button at the bottom of each example to edit and see the actual result in the browser.

For Whom?

These tutorials are designed for beginners and professionals who want to learn AngularJS step by step.

Prerequisites:

Basic knowledge of HTML, JavaScript, CSS and web application is required.

# 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.

http://www.tutorialsteacher.com/Content/images/tips.pngAngularJS 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/)

[](http://www.tutorialsteacher.com/Content/images/ng/angular-website.png)AngularJS Official Website

As you can see in the above angularjs.org website, you can download AngularJS 1 library by clicking on the **Download AngularJS 1** link. AngularJS 2 is in the beta version as of this writing. This tutorials is using AngularJS 1.

Angular is an open source framework. Click on **View on GitHub link** to see the source code.

## 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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-1)

The above example is plain HTML code with couple of AngularJS directives (attributes) such as ng-app, ng-model, and ng-bind.

The same task can be accomplished using jQuery with more lines of code, as shown below.

jQuery Example:

<!DOCTYPE html>

<html>

<head>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

</head>

<body>

Enter Your Name: <input type="text" id="txtName" /> <br />

Hello <label id="lblName"></label>

<script>

$(document).ready( function< () {

$('#txtName').keyup(function () {

$('#lblName').text($('#txtName').val());

});

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-2)

Thus, AngularJS includes built-in attributes using which we can increase the productivity.

## 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.
9. Easy to Unit test.
10. REST friendly.

Let's setup AngularJS development environment in the next section.

# 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.

[](http://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>

## Editor:

AngularJS is eventually HTML and JavaScript code. So you can install any good editor/IDE as per your choice.

The following editors are recommended:

* [Sublime Text](http://www.sublimetext.com/)
* [Aptana Studio 3](http://www.aptana.com/)
* [Ultra Edit](http://www.ultraedit.com/)
* [Eclipse](https://eclipse.org/)
* [Visual Studio](https://www.visualstudio.com/)

## Online Editor:

You can also use the following online editors for learning purpose.

* [plnkr.co](http://plnkr.co/)
* [jsbin.com](http://jsbin.com/)

We are using our own online code editor for all the AngularJS examples in these tutorials.

## Web server:

Use any web server such as IIS, apache etc., locally for development purpose.

## Browser:

You can install any browser of your choice as AngularJS supports cross-browser compatibility. However, it is recommended to use [Google Chrome](http://www.google.com/chrome/) while developing an application.

## Angular Seed:

Use Angular seed project to quickly get started on AngularJS application. The Angular-seed is an application skeleton for a typical AngularJS web application. You can use it to quickly bootstrap your angular webapp projects and development environment for your project.

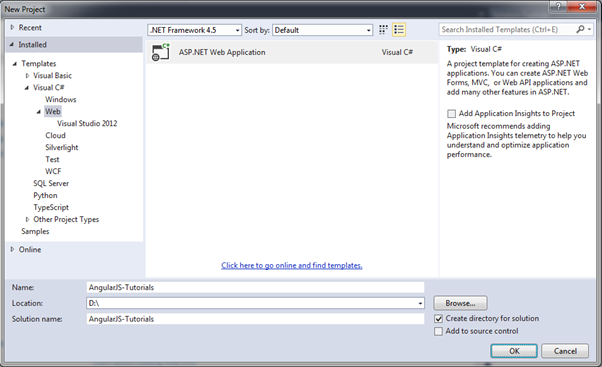
Download angular-seed from [GitHub](https://github.com/angular/angular-seed)

Let's setup Angular project in Visual Studio 2013 for web.

## Setup AngularJS project in Visual Studio:

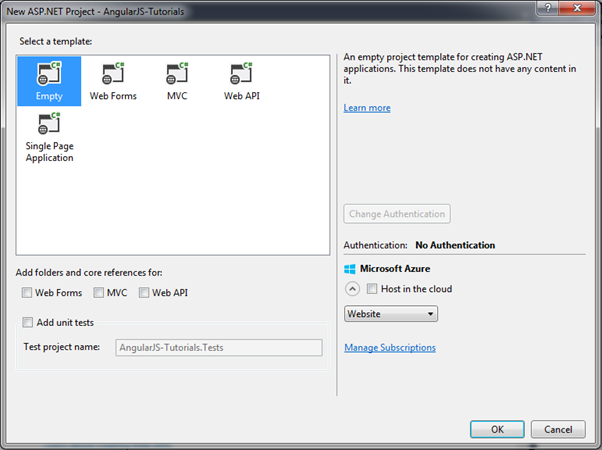
You can create AngularJS application in any version of Visual Studio. Here, we will use Visual Studio 2013 for web.

First, create new project by clicking on New Project link on start page. This will open New Project dialog box, as shown below.

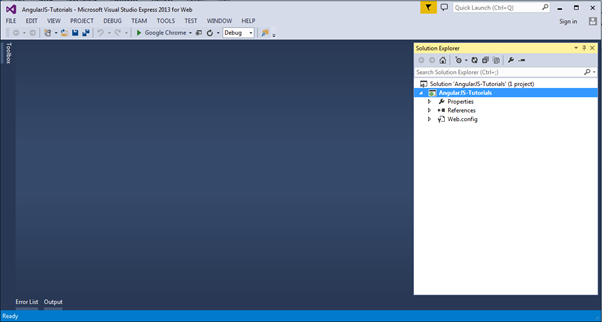
[](http://www.tutorialsteacher.com/Content/images/ng/setup-env-2.png)Development Environment

Select Web in the left pane and ASP.NET Web Application in the middle pane and then click OK.

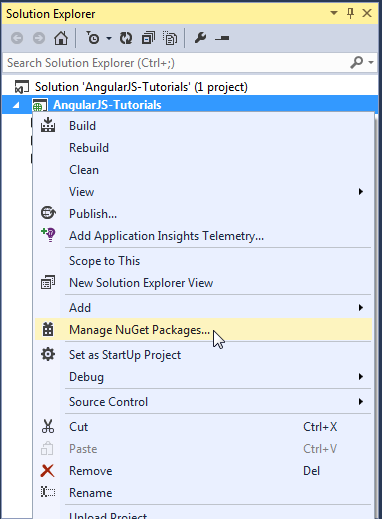
In the New ASP.NET Project dialog box, select Empty template and then click OK.

[](http://www.tutorialsteacher.com/Content/images/ng/setup-env-3.png)Development Environment

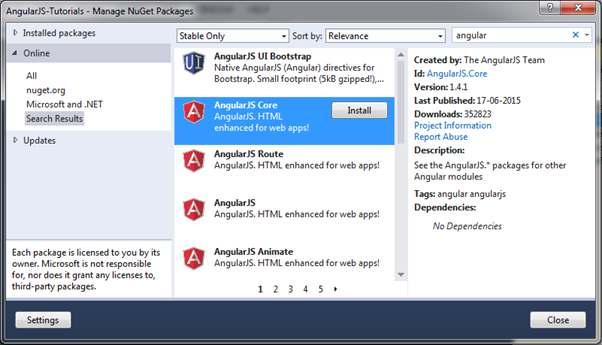
This will create an empty website project in Visual Studio.

[](http://www.tutorialsteacher.com/Content/images/ng/setup-env-4.png)Development Environment

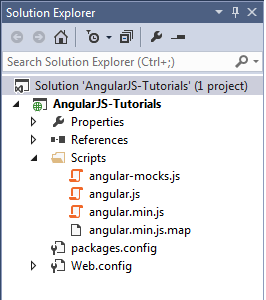
Now, install AngularJS library from NuGet package manager. Right click on the project in Solution Explorer and select Manage NuGet Packages..

[](http://www.tutorialsteacher.com/Content/images/ng/setup-env-5.png)Development Environment

Search for "angular" in the Manage NuGet Packages dialog box and install AngularJS Core.

[](http://www.tutorialsteacher.com/Content/images/ng/setup-env-6.png)Development Environment

This will add AngularJS files into Scripts folder such as angular.js, angular.min.js, and angular-mocks.js, as shown below.

[](http://www.tutorialsteacher.com/Content/images/ng/setup-env-7.png)Development Environment

Now, you can start writing AngularJS web application.

Create a simple AngularJS application and understand basic building blocks of AngularJS, in the next section.

# 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>

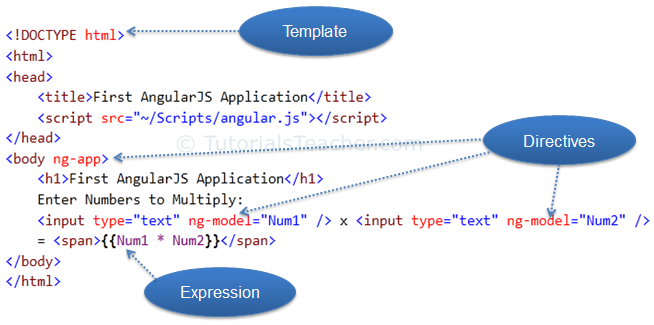
</body>

</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 |

Learn about the ng-app directive in the next section.

# 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.

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>

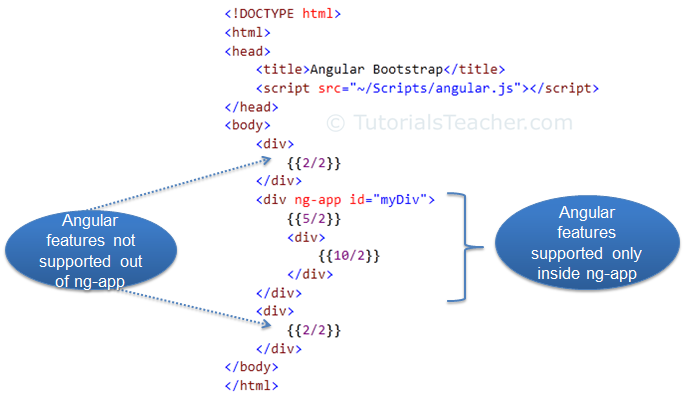
</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-4)

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.

[](http://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](http://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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-5)

In the above example, we have specified a module name using ng-app = 'myAngularApp' in the <body> tag, and then we have created 'myAngularApp' module using angular.module() function inside <script>. Visit [module](http://www.tutorialsteacher.com/angularjs/modules-in-angularjs) section to learn about Angular module in detail.

**Note :**You must create a module with the same name if you have specified it with ng-app directive.

## Manual Bootstrap:

We have learned that the ng-app directive auto initializes an AngularJS framework. However, we can also initialize AngularJS manually without using ng-app directive.

The following example demonstrates manual initialization of Angular.

Example: Manual Bootstrap

<!DOCTYPE html>

<html >

<head>

<title>Angular Bootstrap</title>

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

</head>

<body>

<div>

{{2/2}}

</div>

<div>

{{5/2}}

<div>

{{10/2}}

</div>

</div>

<script>

angular.element(document).ready(function () {

angular.bootstrap(document);

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-6)

In the above example, we call angular.bootstrap() function and specify the root element, which is document object. This will initialize AngularJS and compile all the elements starting from root element i.e. the whole document in this example.

Learn about AngularJS expression in the next section.

# 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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-11)

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](http://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](http://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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-14)

Result:

10500

200

Steve Jobs

# 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)

Visit AngularJS documentation to learn all the built-in [directives](https://docs.angularjs.org/api/ng/directive).

# 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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-31)

Result:

Hello World!

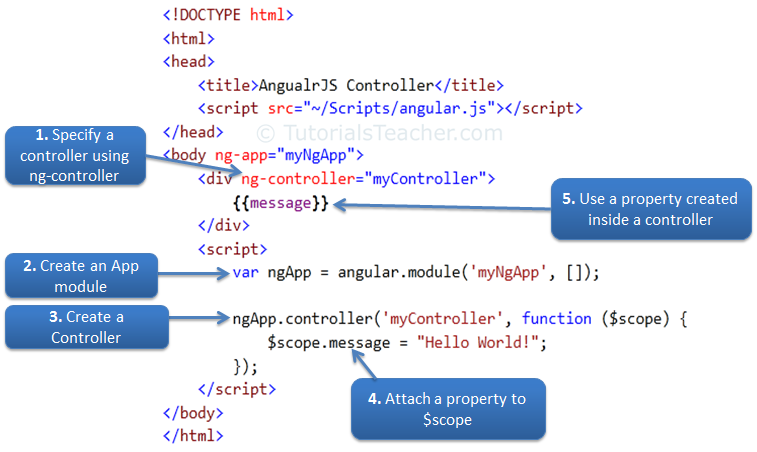
In the above example, ng-controller="myController" directive is applied to the <div> element where "myController" is the name of the controller. Inside div element, we have specified {{message}} expression.

http://www.tutorialsteacher.com/Content/images/tips.pngThe $ 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.

[](http://www.tutorialsteacher.com/Content/images/ng/ng-controller.png)Steps to create an AngularJS Controller

## Attach Behaviors:

You can attach multiple methods to the scope object inside a controller, which can be used as an event handler or for other purposes.

The following example demonstrates handling click event of a button.

Example: Handle Button Click

<!DOCTYPE html>

<html>

<head>

<title>AngualrJS Controller</title>

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

</head>

<body ng-app="myNgApp">

<div ng-controller="myController">

Enter Message: <input type="text" ng-model="message" /> <br />

<button ng-click="showMsg(message)">Show Message</button>

</div>

<script>

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

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

$scope.message = "Hello World!";

$scope.showMsg = function (msg) {

alert(msg);

};

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-32)

In the above example, we have attached showMsg() function to the scope object. The showMsg() method is called on button click. The ng-click directive is used to handle click event in AngularJS application.

Note that the properties and methods attached to the scope object inside a particular controller is only available to the HTML elements and its child elements where ng-controller directive is applied.

Example: Controller

<!DOCTYPE html>

<html>

<head>

<title>AngualrJS Controller</title>

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

</head>

<body ng-app="myNgApp">

<div id="div1" ng-controller="myController">

Message: {{message}} <br />

<div id="div2">

Message: {{message}}

</div>

</div>

<div id="div3">

Message: {{message}}

</div>

<div id="div4" ng-controller="anotherController">

Message: {{message}}

</div>

<script>

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

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

$scope.message = "This is myController";

});

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

$scope.message = "This is anotherController";

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-33)

Result:

Message: This is myController

Message: This is myController

Message:

Message: This is anotherController

In the above example, the "message" property is defined inside myController, so it will only be available to div1 and div2 but not div3 and div4. The same way, message property defined inside anotherController will only be available to div4. The div3 element does not come under any controller, so "message" property will be null or undefined.

## Attach Complex object:

You can also attach an object to the $scope inside controller and display value of its properties in HTML.

Example: Attach an object

<!DOCTYPE html>

<html>

<head>

<title>AngualrJS Controller</title>

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

</head>

<body ng-app="myNgApp">

<h2>Student Information:</h2>

<div ng-controller="myController">

First Name: {{student.firstName}} <br />

Last Name: {{student.lastName}}

</div>

<script>

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

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

$scope.student = { firstName: 'James', lastName: 'Bond' };

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-34)

Result:

First Name: James

Last Name: Bond

As you can see in the above example, a student object is attached to the $scope and its properties and methods can be accessed using an expression, ng-model, or ng-bind directives with dot notation.

## Nested Controllers:

Angular allows nested controllers. The following example demonstrates multiple controllers.

Example: Nested Controllers

<!DOCTYPE html>

<html>

<head>

<title>AngualrJS Controller</title>

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

</head>

<body ng-app="myNgApp">

<div ng-controller="parentController">

Message: {{message1}}

<div ng-controller="childController">

Parent Message: {{message1}} </br>

Child Message: {{message2}}

</div>

Child Message: {{message2}}

</div>

<script>

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

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

$scope.message1 = "This is parentController";

});

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

$scope.message2 = "This is childController";

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-35)

Result:

Message: This is parentController

Parent Message: This is parentController

Child Message: This is childController

Child Message:

As you can see in the above example, a child controller can access properties and methods attached in parent controller function, whereas parent controller cannot access properties and methods attached in child controller.

## Minification Syntax:

All the script files in AngularJS application should be minified in the production environment.

The minification process shortens parameter and function names. As mentioned before, AngularJS controller function may include $scope or other parameters. If minification process changes the parameter names then AngularJS application will break because Angular framework needs the same parameter name for built-in objects such as $scope. Use the following syntax so that minification will not change the parameter name.

Example: Controller Syntax for minification

<!DOCTYPE html>

<html >

<head>

<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', ['$scope', function ($scope) {

$scope.message = "Hello World!";

}]);

</script>

</body>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-36)

In the above example, we have given name of the parameter and controller function in the square bracket []. Parameter name will be first and last member will be action controller function in the square brackets. This will tell minifier not to convert $scope name in the minification process.

# 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!

As per the above example, properties added in $rootScope are available in all the controllers.

The $scope object contains various methods. The following table lists important methods of $scope object.

| **Method** | **Description** |
| --- | --- |
| $new() | Creates new child scope. |
| $watch() | Register a callback to be executed whenever model property changes. |
| $watchGroup() | Register a callback to be executed whenever model properties changes. Here, specify an array of properties to be tracked. |
| $watchCollection() | Register a callback to be executed whenever model object or array property changes. |
| $digest() | Processes all of the watchers of the current scope and its children.Â |
| $destroy() | Removes the current scope (and all of its children) from the parent scope. |
| $eval() | Executes the expression on the current scope and returns the result. |
| $apply() | Executes an expression in angular outside the angular framework. |
| $on() | Register a callback for an event. |
| $emit() | Dispatches the specified event upwards till $rootScope. |
| $broadcast() | Dispatches the specified event downwards to all child scopes. |

## $watch:

Angular scope object includes $watch event which will be raised whenever a model property is changed.

Example:$watch()

<!DOCTYPE html>

<html>

<head>

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

</head>

<body ng-app="myNgApp">

<div ng-controller="myController">

Enter Message: <input type="text" ng-model="message" /> <br />

New Message: {{newMessage}} <br />

Old Message: {{oldMessage}}

</div>

<script>

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

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

$scope.message = "Hello World!";

$scope.$watch('message', function (newValue, oldValue) {

$scope.newMessage = newValue;

$scope.oldMessage = oldValue;

});

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-43)

As you can see in the above example, $watch registers a callback, which will get called whenever the specified model property "message" changes.

# 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-change |
| ng-click |
| ng-dblclick |
| ng-focus |
| ng-keydown |
| ng-keyup |
| ng-keypress |
| ng-mousedown |
| ng-mouseenter |
| ng-mouseleave |
| ng-mousemove |
| ng-mouseover |
| ng-mouseup |

Let's take a look at some of the important event directives.

## ng-click:

The ng-click directive is used to provide event handler for click event.

Example: ng-click

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

Enter Password: <input type="password" ng-model="password" /> <br />

<button ng-click="DisplayMessage(password)">Show Password</button

</div>

<script>

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

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

$scope.DisplayMessage = function (value) {

$window.alert(value)

}

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-91)

In the above example, ng-click directive is used to call a DisplayMessage() function with the 'password' parameter when a user clicks a button. A 'password' is a model property defined using ng-model directive in the input box. The DisplayMessage() function is attached to a $scope object in myController, so it will be accessible from button click as button comes under myController. The[$window](http://www.tutorialsteacher.com/angularjs/angularjs-window-service) service is used to display an alert.

## Mouse events:

The following example demonstrates important mouse event directives - ng-mouseenter and ng-mouseleave.

Example: Mouse events

<!DOCTYPE html>

<html>

<head>

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

<style>

.redDiv {

width: 100px;

height: 100px;

background-color: red;

padding:2px 2px 2px 2px;

}

.yellowDiv {

width: 100px;

height: 100px;

background-color: yellow;

padding:2px 2px 2px 2px;

}

</style>

</head>

<body ng-app>

<div ng-class="{redDiv: enter, yellowDiv: leave}" ng-mouseenter="enter=true;leave=false;" ng-mouseleave="leave=true;enter=false">

Mouse <span ng-show="enter">Enter</span> <span ng-show="leave">Leave</span>

</div>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-92)

In the above example, the ng-class directive includes map of CSS classes, so redDiv will be applied if enter=true and yellowDiv will be applied if leave=true. The ng-mouseenter directive sets 'enter' to true, which will apply redDiv class to the <div> element. In the same way, ng-mouseleave will set leave to true, which will apply yellowDiv class.

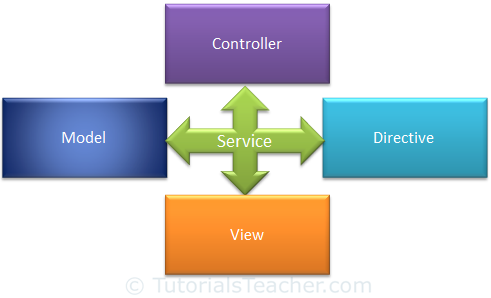
# AngularJS Service:

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.

Learn some of the important built-in services in the next section.

# $http Service:

The $http service is used to send or receive data from the remote server using browser's XMLHttpRequest or JSONP.

$http is a service as an object. It includes following shortcut methods.

| **Method** | **Description** |
| --- | --- |
| $http.get() | Perform Http GET request. |
| $http.head() | Perform Http HEAD request. |
| $http.post() | Perform Http POST request. |
| $http.put() | Perform Http PUT request. |
| $http.delete() | Perform Http DELETE request. |
| $http.jsonp() | Perform Http JSONP request. |
| $http.patch() | Perform Http PATCH request. |

Let's look at some of the important methods of $http.

## $http.get():

$http.get() method sends http GET request to the remote server and retrieves the data.

Signature: HttpPromise $http.get(url)

$http.get() method returns HttpPromise object, which includes various methods to process the response of http GET request.

The following example demonstrates the use of $http service in a controller to send HTTP GET request.

Example: $http.get()

<!DOCTYPE html>

<html>

<head>

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

</head>

<body ng-app ="myApp">

<div>

<div ng-controller="myController">

Response Data: {{data}} <br />

Error: {{error}}

</div>

</div>

<script>

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

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

var onSuccess = function (data, status, headers, config) {

$scope.data = data;

};

var onError = function (data, status, headers, config) {

$scope.error = status;

}

var promise = $http.get("/demo/getdata");

promise.success(onSuccess);

promise.error(onError);

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-51)

In the above example, 'myController' controller includes $http parameter, so that it can be used to send GET request. AngularJS automatically injects $scope parameter at runtime. The $http.get() method returns HttpPromise which includes methods like success() and error(). The success() method registers a callback method which is called when a request completes successfully. The error() method registers a callback method which is called when a request fails and returns an error.

The onSuccess() method above, attaches the response data to the $scope. The onError() method attaches status property to the $scope. These methods can be called in chain, as shown below.

Example: $http.get()

<!DOCTYPE html>

<html>

<head>

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

</head>

<body ng-app ="myApp">

<div>

<div ng-controller="myController">

Response Data: {{data}} <br />

Error: {{error}}

</div>

</div>

<script>

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

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

var onSuccess = function (data, status, headers, config) {

$scope.data = data;

};

var onError = function (data, status, headers, config) {

$scope.error = status;

}

var promise = $http.get("/demo/getdata").success(onSuccess).error(onError);

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-52)

## $http.post:

The $http.post() method sends Http POST request to the remote server to submit and retrieve the data.

Signature: HttpPromise $http.post(url, dataToSubmit);

The following example demonstrates $http.post() method.

Example: $http.post()

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

Response Data: {{data}} <br />

Error: {{error}}

</div>

<script>

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

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

var onSuccess = function (data, status, headers, config) {

$scope.data = data;

};

var onError = function (data, status, headers, config) {

$scope.error = status;

}

$http.post('/demo/submitData', { myData: 'Hello World!' })

.success(onSuccess)

.error(onError);

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-53)

## $http():

You can use construction function of $http service to perform http request, as shown below.

Example: $http()

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

Response Data: {{data}} <br />

Error: {{error}}

</div>

<script>

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

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

var onSuccess = function (data, status, headers, config) {

$scope.data = data;

};

var onError = function (data, status, headers, config) {

$scope.error = status;

}

var getReq = {

method: 'GET',

url: '/demo/getdata'

};

$http(getReq).success(onSuccess).error(onError);

var postReq = {

method: 'POST',

url: '/demo/submitData',

data: { myData: 'test data' }

};

$http(postReq).success(onSuccess).error(onError);

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-54)

Thus, you can use $http service to send AJAX request to the remote server.

# $log Service:

AngularJs includes logging service $log, which logs the messages to the browser's console.

The $log service includes different methods to log the error, information, warning or debug information. It can be useful in debugging and auditing.

Example: $log

<!DOCTYPE html>

<html>

<head>

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

</head>

<body ng-app="myApp" >

<div ng-controller="myController">

<p>Please check the browser console for the logging information.</p>

</div>

<script>

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

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

$log.log('This is log.');

$log.error('This is error.');

$log.info('This is info.');

$log.warn('This is warning.');

$log.debug('This is debugging.');

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-61)

In the above example, controller function includes $log parameter which will be supplied by AngularJS framework.

# $interval Service:

AngularJS includes $interval service which performs the same task as setInterval() method in JavaScript. The $interval is a wrapper for setInterval() method, so that it will be easy to override, remove or mocked for testing.

The $interval service executes the specified function on every specified milliseconds duration.

Signature: $interval(fn, delay, [count], [invokeApply], [Pass]);

The following example demonstrates $interval service that displays a counter on each 1000 milliseconds.

Example: $interval

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

{{counter}}

</div>

<script>

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

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

$scope.counter = 0;

var increaseCounter = function () {

$scope.counter = $scope.counter + 1;

}

$interval(increaseCounter, 1000);

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-62)

In the above example, $interval service calls increaseCounter() function on every 1000 milliseconds. The increaseCounter() function increases the $scope.counter property by 1. Thus, counter increases on every milliseconds.

## Specify count:

The $interval service also executes the specified function for the specified number of times as count parameter.

Example: $interval

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

{{counter}}

</div>

<script>

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

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

$scope.counter = 0;

var increaseCounter = function () {

$scope.counter = $scope.counter + 1;

}

$interval(increaseCounter, 1000, 10);

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-63)

In the above example, increaseCounter() method will be executed on each 1000 milliseconds but not more than 10 times.

## Cancel execution:

The $interval service returns an object of HttpPromise which can be used to stop the counter by using $interval.cancel(promise) method.

Example: $interval.cancel()

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div>

<div ng-controller="myController">

{{counter}} <br /><br />

<button ng-click="cancel()">Cancel</button>

</div>

</div>

<script>

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

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

$scope.counter = 0;

var increaseCounter = function () {

$scope.counter = $scope.counter + 1;

}

var promise = $interval(increaseCounter, 1000);

$scope.cancel = function () {

$interval.cancel(promise);

$scope.counter = "Cancelled!";

};

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-64)

# $window Service:

AngularJs includes $window service which refers to the browser window object.

In the JavaScript, window is a global object which includes many built-in methods like alert(), prompt() etc.

The $window service is a wrapper around window object, so that it will be easy to override, remove or mocked for testing. It is recommended to use $window service in AngularJS instead of global window object directly.

Example: $window

<!DOCTYPE html>

<html>

<head>

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

</head>

<body ng-app="myApp" ng-controller="myController">

<button ng-click="DisplayAlert('Hello World!')">Display Alert</button>

<button ng-click="DisplayPrompt()">Display Prompt</button>

<script>

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

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

$scope.DisplayAlert = function (message) {

$window.alert(message);

}

$scope.DisplayPrompt = function () {

var name = $window.prompt('Enter Your Name');

$window.alert('Hello ' + name);

}

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-71)

# 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.

Example: Number filter

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app >

Enter Amount: <input type="number" ng-model="amount" /> <br />

100000 | number = {{100000 | number}} <br />

amount | number = {{amount | number}} <br />

amount | number:2 = {{amount | number:2}} <br />

amount | number:4 = {{amount | number:4}} <br />

amount | number = <span ng-bind="amount | number"></span>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-81)

## Currency Filter:

The currency filter formats a number value as a currency. When no currency symbol is provided, default symbol for current locale is used.

{{ expression | currency : 'currency\_symbol' : 'fraction'}}

Example: Currency filter

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

Default currency: {{person.salary | currency}} <br />

Custom currency identifier: {{person.salary | currency:'Rs.'}} <br />

No Fraction: {{person.salary | currency:'Rs.':0}} <br />

Fraction 2: <span ng-bind="person.salary| currency:'GBP':2"></span>

</div>

<script>

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

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

$scope.person = { firstName: 'James', lastName: 'Bond', salary: 100000}

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-86)

Output:

Default currency: $100,000.00

Custom currency identifier: Rs.100,000.00

No Fraction: Rs.100,000

Fraction 2: GBP100,000.00

In the above example, we have applied currency filter to person.salary, which is a numeric property. It can be displayed with different currency symbols and fractions.

## Date filter:

Formats date to string based on the specified format.

{{ date\_expression | date : 'format'}}

Example: date filter

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app>

<div ng-init="person.DOB = 323234234898">

Default date: {{person.DOB| date}} <br />

Short date: {{person.DOB| date:'short'}} <br />

Long date: {{person.DOB | date:'longDate'}} <br />

Year: {{person.DOB | date:'yyyy'}} <br />

</div>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-82)

Output:

Default date: Mar 30, 1980

short date: 3/30/80 8:47 AM

long date: March 30, 1980

Year: 1980

Visit Angular documentation for more information on [date](https://docs.angularjs.org/api/ng/filter/date) filter.

## Uppercase/lowercase filter:

The uppercase filter converts the string to upper case and lowercase filter converts the string to lower case.

Example: uppercase & lowercase filters

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app>

<div ng-init="person.firstName='James';person.lastName='Bond'">

Lower case: {{person.firstName + ' ' + person.lastName | lowercase}} <br />

Upper case: {{person.firstName + ' ' + person.lastName | uppercase}}

</div>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-83)

Output:

Lower case: james bond

Upper case: JAMES BOND

## Filter:

Filter selects items from an array based on the specified criteria and returns a new array.

{{ expression | filter : filter\_criteria }}

Example: filter

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

Enter Name to search: <input type="text" ng-model="searchCriteria" /> <br />

Result: {{myArr | filter:searchCriteria}}

</div>

<script>

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

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

$scope.myArr = ['Steve', 'Bill', 'James', 'Rob', 'Ram', 'Moin']

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-84)

In the above example, searchCriteria contains a text entered in the input box, which will be used to filter items of an array myArr using filter:searchCriteria expression.

## orderBy filter:

The orderBy filter sorts an array based on specified expression predicate.

{{ expression | orderBy : predicate\_expression : reverse}}

Example: orderBy filter

<!DOCTYPE html>

<html>

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

<select ng-model="SortOrder">

<option value="+name">Name (asc)</option>

<option value="-name">Name (dec)</option>

<option value="+phone">Phone (asc)</option>

<option value="-phone">Phone (dec)</option>

</select>

<ul ng-repeat="person in persons | orderBy:SortOrder">

<li>{{person.name}} - {{person.phone}}</li>

</ul>

</div>

<script>

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

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

$scope.persons = [{ name: 'John', phone: '512-455-1276' },

{ name: 'Mary', phone: '899-333-3345' },

{ name: 'Mike', phone: '511-444-4321' },

{ name: 'Bill', phone: '145-788-5678' },

{ name: 'Ram', phone: '433-444-8765' },

{ name: 'Steve', phone: '218-345-5678' }]

$scope.SortOrder = '+name';

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-85)

The above example displays a list of person names and phone numbers in a particular order specified using orderBy:SortOrder filter. SortOrder is a model property and will be set to the selected value in the dropdown. Therefore, based on the value of SortOrder, ng-repeat directive will display the data.

# 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>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-101)

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.

## Modules in separate files:

In the controller example shown above, we created application module and controller in the same HTML file. However, we can create separate JavaScript files for each module as shown below.

Example:Angular Modules in Separate Files

<!DOCTYPE html>

<html >

<head>

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

</head>

<body ng-app="myApp">

<div ng-controller="myController">

{{message}}

</div>

<script src="app.js" ></script>

<script src="myController.js" ></script>

</body>

</html>

app.js

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

myController.js

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

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

});

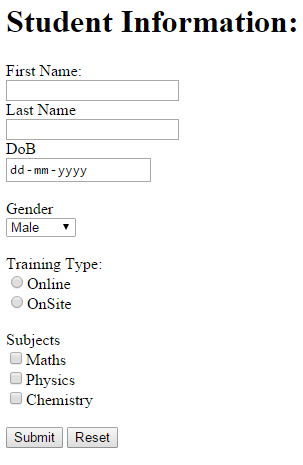
Visit AngularJS documentation for more information of [Module](https://docs.angularjs.org/api/ng/type/angular.Module).

# AngularJS Forms:

The HTML form is a collection of input controls where user can enter the data. Here, you will learn how to display AngularJS form and submit the data.

## An AngularJS Form Example:

We will create following Student Information form with submit and reset functionality.

[](http://www.tutorialsteacher.com/Content/images/ng/angular-form.png)Sample AngularJS Form

The following is the code of the above form.

Example: AngularJS Form

<!DOCTYPE html>

<html ng-app="studentApp">

<head>

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

</head>

<body ng-controller="studentController">

<h1>Student Information:</h1>

<form **ng-submit="submitStudnetForm()"** >

<label for="firstName" >First Name: </label><br />

<input type="text" id="firstName" ng-model="student.firstName" /> <br />

<label for="lastName">Last Name</label><br />

<input type="text" id="lastName" ng-model="student.lastName" /> <br />

<label for="dob" >DoB</label><br />

<input type="date" id="dob" ng-model="student.DoB" /> <br /><br />

<label for="gender" >Gender</label> <br />

<select id="gender" ng-model="student.gender">

<option value="male">Male</option>

<option value="female">Female</option>

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

<span>Training Type:</span><br />

<label><input value="online" type="radio" name="training" ng-model="student.trainingType" />Online</label><br />

<label><input value="onsite" type="radio" name="training" ng-model="student.trainingType" />OnSite</label> <br /><br />

<span>Subjects</span><br />

<label><input type="checkbox" ng-model="student.maths" />Maths</label> <br />

<label><input type="checkbox" ng-model="student.physics" />Physics</label> <br />

<label><input type="checkbox" ng-model="student.chemistry" />Chemistry</label><br /><br />

<input type="submit" value="Submit" />

<input type="reset" ng-click="resetForm()" value="Reset" />

</form>

<script>

//1. create app module

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

//2. create controller

studentApp.controller("studentController", function ($scope, $http) {

//3. attach originalStudent model object

$scope.originalStudent = {

firstName: 'James',

lastName: 'Bond',

DoB: new Date('01/31/1980'),

gender: 'male',

trainingType: 'online',

maths: false,

physics: true,

chemistry: true

};

//4. copy originalStudent to student. student will be bind to a form

$scope.student = angular.copy($scope.originalStudent);

//5. create submitStudentForm() function. This will be called when user submits the form

$scope.submitStudnetForm = function () {

var onSuccess = function (data, status, headers, config) {

alert('Student saved successfully.');

};

var onError = function (data, status, headers, config) {

alert('Error occured.');

}

$http.post('/student/submitData', { student:$scope.student })

.success(onSuccess)

.error(onError);

};

//6. create resetForm() function. This will be called on Reset button click.

$scope.resetForm = function () {

$scope.student = angular.copy($scope.OriginalStudent);

};

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-111)

The following is a step by step explanation of the above example:

1. Create an HTML page and wrap all the necessary input controlls into <form> tag.
2. Create the AngularJS application module in the <script> tag.
3. Create studentController in application module.
4. Create originalStudent object and attach to the $scope with required properties. This will stay unchanged during entire life cycle.
5. Create new student object and attach to the $scope and copy all the properties and values from originalStudent. This student object will be bound to the form using ng-model directive. Therefore, if user changes form values then sudent object will also get changed.
6. Create submitStudnetForm function which will get called when user submits the form using Submit button. Here, send http POST request to the remote server to submit the data using[$http service](http://www.tutorialsteacher.com/angularjs/angularjs-service-http).
7. Create resetForm() function, which will reset the form values to the originalStudent values by copying it to student object.
8. Apply ng-app, ng-controller directives.
9. Apply ng-model directives to each HTML input element to bind appropriate properties of student object.
10. Apply ng-submit directive to form which will call submitStudentForm() on the form submit event.
11. Apply ng-click directive to reset button which will call resetForm() on the button click event.

An AngularJS form can be submitted using either ng-submit or ng-click directive but not both.

**Ng-submit:** Binds angular expression to onsubmit event when form does not include action attribute.

**Ng-click:** Binds angular expression to onclick event.

**Note :**The angular form can be submitted using ng-submit directive on the form tag or using ng-click directive on <input type="submit" /> element. Use either ng-submit or ng-click directive but not both to submit the form. The form will be submitted twice if both ng-submit and ng-click directives are used.

# Apply Bootstrap CSS to AngularJS Form:

In the previous section, we created AngularJS form but did not apply CSS classes. Let's apply simple bootstrap CSS classes to make it responsive.

We will be applying following bootstrap CSS classes:

* **container:** Top level bootstrap class with fixed width and left margin.
* **form-horizontal:** Places labels to the left of input/select controls.
* **form-group:** wraps labels and controls in div for optimum spacing.
* **control-label:** groups labels and input/select controls.
* **form-control:** styles for input/textarea/select elements.
* **radio:** styles for radio element.
* **checkbox:** styles for checkbox element.
* **btn, btn-primary:** styles for button element.

The following is a student form with bootstrap styling.

Example: AngularJS form with bootstrap CSS classes

<!DOCTYPE html>

<html ng-app="myApp">

<head>

<link href="~/Content/bootstrap.CSS" rel="stylesheet" />

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

</head>

<body ng-controller="studentController" class="container" > <br />

<form class="form-horizontal" ng-submit="submitStudnetForm()" role="form">

<div class="form-group">

<label for="firstName" class="col-sm-3 control-label">First Name</label>

<div class="col-sm-6">

<input type="text" id="firstName" class="form-control" ng-model="student.firstName" />

</div>

<div class="col-sm-3"></div>

</div>

<div class="form-group">

<label for="lastName" class="col-sm-3 control-label">Last Name</label>

<div class="col-sm-6">

<input type="text" id="lastName" class="form-control" ng-model="student.lastName" />

</div>

<div class="col-sm-3"></div>

</div>

<div class="form-group">

<label for="dob" class="col-sm-3 control-label">DoB</label>

<div class="col-sm-2">

<input type="date" id="dob" class="form-control" ng-model="student.DoB" />

</div>

<div class="col-sm-7"></div>

</div>

<div class="form-group">

<label for="gender" class="col-sm-3 control-label">Gender</label>

<div class="col-sm-2">

<select id="gender" class="form-control" ng-model="student.gender">

<option value="male">Male</option>

<option value="female">Female</option>

</select>

</div>

<div class="col-sm-7"></div>

</div>

<div class="form-group">

<div class="col-sm-3"></div>

<div class="col-sm-2">

<span><b>Training Location</b></span>

<div class="radio">

<label><input value="online" type="radio" name="training" ng-model="student.trainingType" />Online</label>

</div>

<div class="radio">

<label><input value="onsite" type="radio" name="training" ng-model="student.trainingType" />OnSite</label>

</div>

</div>

<div class="col-sm-7">

<span><b>Main Subjects</b></span>

<div class="checkbox">

<label><input type="checkbox" ng-model="student.maths" />Maths</label>

</div>

<div class="checkbox">

<label><input type="checkbox" ng-model="student.physics" />Physics</label>

</div>

<div class="checkbox">

<label><input type="checkbox" ng-model="student.chemistry" />Chemistry</label>

</div>

</div>

</div>

<input type="submit" value="Save" class="btn btn-primary col-sm-offset-3" />

<input type="reset" value="Reset" ng-click="resetForm()"

</form>

<script>

//1. create app module

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

//2. create controller

studentApp.controller("studentController", function ($scope, $http) {

//3. attach originalStudent model object

$scope.originalStudent = {

firstName: 'James',

lastName: 'Bond',

DoB: new Date('01/31/1980'),

gender: 'male',

trainingType: 'online',

maths: false,

physics: true,

chemistry: true

};

//4. copy originalStudent to student. student will be bind to a form

$scope.student = angular.copy($scope.originalStudent);

//5. create submitStudentForm() function. This will be called when user submits the form

$scope.submitStudnetForm = function () {

var onSuccess = function (data, status, headers, config) {

alert('Student saved successfully.');

};

var onError = function (data, status, headers, config) {

alert('Error occured.');

}

$http.post('/student/submitData', { student:$scope.student })

.success(onSuccess)

.error(onError);

};

//6. create resetForm() function. This will be called on Reset button click.

$scope.resetForm = function () {

$scope.student = angular.copy($scope.OriginalStudent);

};

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-121)

You can create separate JS files for the main application module and controllers to organize application files.

Next, learn how to implement client side validation in AngularJS form.

# Validation in AngularJS:

We created an HTML form in the previous section. Here, we will implement client side validation in AngularJS form.

AngularJS includes the following validation directives.

| **Directive** | **Description** |
| --- | --- |
| ng-required | Sets required attribute on an input field. |
| ng-minlength | Sets minlength attribute on an input field. |
| ng-maxlength | Sets maxlength attribute on an input field. Setting the attribute to a negative or non-numeric value, allows view values of any length. |
| ng-pattern | Sets pattern validation error key if the ngModel value does not match the specified RegEx expression. |

Let's implement validation in the student form which contains First Name, Last Name and Email fields.

Example: Form Validation

<!DOCTYPE html>

<html>

<head>

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

</head>

<body ng-app >

<form name="studentForm" novalidate>

<label for="firstName">First Name: </label> <br />

<input type="text" name="firstName" ng-model="student.firstName" ng-required="true" />

<span ng-show="studentForm.firstName.$touched && studentForm.firstName.$error.required">First name is required.</span><br /><br />

<label for="lastName">Last Name</label><br />

<input type="text" name="lastName" ng-minlength="3" ng-maxlength="10" ng-model="student.lastName" />

<span ng-show="studentForm.lastName.$touched && studentForm.lastName.$error.minlength">min 3 chars.</span>

<span ng-show="studentForm.lastName.$touched && studentForm.lastName.$error.maxlength">Max 10 chars.</span><br /><br />

<label for="dob">Email</label><br />

<input type="email" id="email" ng-model="student.email" name="email" />

<span ng-show="studentForm.email.$touched && studentForm.email.$error.email">Please enter valid email id.</span><br /><br />

<input type="submit" value="Submit" />

</form>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-131)

Let's understand the above example step by step:

1. Apply novalidate attribute in <form> tag. The novalidate attribute will disable the browser's default validation.
2. Set the name attribute in <form> and other elements, which will be used to obtain a reference of the elements.
3. Now, set ng-required="true" on the input element of First Name. Also, set name attribute to the name of model property, "firstName" in this case.
4. Create <span> element to specify an error message with every input filed where the validation directive is applied.
5. Set ng-show directives to <span> element to an expression "studentForm.firstName.$touched && studentForm.firstName.$error.required". This expression will return true if a user tabbed out without entering FirstName.
6. The same way set ng-minlength & ng-maxlength directives to last name. Also, set ng-show directive to "studentForm.lastName.$touched && studentForm.lastName.$error.minlength" expression to <span> element adjacent to LastName input field.
7. Create another <span> for maxlength validation message.
8. Email will be validated automatically with input type=email. Also, create <span> for email validation message.

We have applied an expression "studentForm.firstName.$touched && studentForm.firstName.$error.required" to the <span>, in the above example. $touched & $error are built-in properties which return the state of the specified input controls and form. Let's learn about the state properties.

## State Properties:

Angular includes properties which return the state of form and input controls. The state of the form and control changes based on the user's interaction and validation errors. These built-in properties can be accessed using form name or input control name. To check the form status useformName.propertyName, and to check the state of input control, useformName.inputFieldName.propertyName.

The following table lists the state properties.

| **Property** | **Description** |
| --- | --- |
| $error | $error object contains all the validation attributes applied to the specified element. |
| $pristine | Returns true if the user has not interacted with control yet else returns false. |
| $valid | Returns true if the model is valid |
| $invalid | Returns true if the model is invalid |
| $dirty | Returns true if user changed the value of model at least once |
| $touched | Returns true if the user has tabbed out from the control. |
| $untouched | Returns true if the user has not tabbed out from the control. |

The following example demonstrates the state properties.

Example: State Properties

<!DOCTYPE html>

<html>

<head>

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

</head>

<body ng-app>

<form name="studentForm" novalidate>

<p>

First Name Status: <br />

Pristine: {{studentForm.firstName.$pristine}} <br />

Touched: {{studentForm.firstName.$touched}}<br />

Untouched: {{studentForm.firstName.$untouched}}<br />

Valid: {{studentForm.firstName.$valid}} <br />

Invalid: {{studentForm.firstName.$invalid}} <br />

Dirty: {{studentForm.firstName.$dirty}} <br />

Error: {{studentForm.firstName.$error}} <br />

</p>

<label for="firstName">First Name: </label> <br />

<input type="text" name="firstName" ng-model="student.firstName" ng-required="true" />

<span ng-show="studentForm.firstName.$touched && studentForm.firstName.$error.required">First name is required.</span><br /><br />

<label for="lastName">Last Name</label><br />

<input type="text" name="lastName" ng-minlength="3" ng-maxlength="10" ng-model="student.lastName" /> <br />

<span ng-show="studentForm.lastName.$error.minlength">min 3 chars.</span>

<span ng-show="studentForm.lastName.$error.maxlength">Max 10 chars.</span> <br />

<input type="submit" value="Save" />

</form>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-132)

Learn about built-in validation css classes in AngularJS in the next section.

# AngularJS Validation CSS Classes:

AngularJS includes following CSS classes to allow styling of form and input controls based on the state of form field.

| **CSS Class** | **Description** |
| --- | --- |
| ng-valid | Angular will set this CSS class if the input field is valid without errors. |
| ng-invalid | Angular will set this CSS class if the input does not pass validations. |
| ng-pristine | Angular will set this CSS class if a user has not interacted with the control yet. |
| ng-dirty | Angular will set this CSS class if the value of form field has been changed. |
| ng-touched | Angular will set this CSS class if a user tabbed out from the input control. |
| ng-untouched | Angular will set this CSS class if a user has not tabbed out from the input control. |
| ng-submitted | Angular will set this CSS class if the form has been submitted. |

Note that you must provide implementation of these CSS classes and include in your CSS file. AngularJS automatically includes these classes based on the current state of input controls.

The following example demonstrates ng-pristine, ng-touched, ng-valid, and ng-invalid classes to display validity of each form control.

Example: AngularJS Validation CSS Classes

<!DOCTYPE html>

<html>

<head>

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

<style>

input.ng-pristine {

background-color:yellow;

}

input.ng-touched.ng-invalid {

background-color:red;

}

input.ng-touched.ng-valid {

background-color:green;

}

</style>

</head>

<body ng-app>

<form name="studentForm" novalidate class="student-form">

<label for="firstName">First Name: </label> <br />

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

<span ng-show="studentForm.firstName.$touched && studentForm.firstName.$error.required">First name is required.</span><br /><br />

<label for="lastName">Last Name</label><br />

<input type="text" name="lastName" ng-minlength="3" ng-maxlength="10" ng-model="lastName" />

<span ng-show="studentForm.lastName.$touched && studentForm.lastName.$error.minlength">min 3 chars.</span>

<span ng-show="studentForm.lastName.$touched && studentForm.lastName.$error.maxlength">Max 10 chars.</span><br /><br />

<label for="dob">Email</label><br />

<input type="email" id="email" ng-model="email" name="email" />

<span ng-show="studentForm.email.$touched && studentForm.email.$error.email">Please enter valid email id.</span><br /><br />

<input type="submit" value="Save" />

</form>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-141)

In the above example, initially all the controls will have yellow background. If a user enters a valid value then it will change the background color to green, otherwise it will turn to red if there is a validation error.

# Bootstrap CSS Classes for Styling Validation Messages:

In the previous section, we applied AngularJS built-in CSS classes to a form. Here, we will apply bootstrap CSS classes for styling form field error, warning, or validity.

Bootstrap includes following CSS classes for the form validation.

| **Bootstrap CSS Class** | **Description** |
| --- | --- |
| has-error | Set this CSS class when a form field becomes invalid based on applied validation attributes. |
| has-warning | Set this CSS class to display warning for a form field. |
| has-success | Set this CSS class when a form field becomes valid based on applied validation attributes. |

The following is a student form with bootstrap validation classes.

Example: AngularJS form with bootstrap validation CSS classes

<!DOCTYPE html>

<html>

<head>

<link href="~/Content/bootstrap.css" rel="stylesheet" />

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

</head>

<body ng-app class="container">

<br />

<form class="form-horizontal" ng-submit="submitStudnetForm()" role="form" name="studentForm" novalidate>

<div class="form-group" ng-class="{'has-error': studentForm.firstName.$touched && studentForm.firstName.$error.required , 'has-success': studentForm.firstName.$valid }">

<label for="firstName" class="col-sm-3 control-label">First Name</label>

<div class="col-sm-6">

<input type="text" id="firstName" name="firstName" class="form-control" ng-model="firstName" required />

</div>

<div class="col-sm-3">

<span class="help-block" ng-show="studentForm.firstName.$touched && studentForm.firstName.$error.required">Please enter First Name.</span>

</div>

</div>

<div class="form-group" ng-class="{'has-error': studentForm.lastName.$touched && studentForm.lastName.$error.required , 'has-success': studentForm.lastName.$valid}">

<label for=" lastname" class="col-sm-3 control-label">

Last Name

</label>

<div class="col-sm-6">

<input type="text" id="lastName" name="lastName" class="form-control" ng-model="lastName" required />

</div>

<div class="col-sm-3">

<span class="help-block" ng-show="studentForm.lastName.$touched && studentForm.lastName.$error.required">Please enter Last Name.</span>

</div>

</div>

<div class="form-group" ng-class="{'has-error': studentForm.emailId.$touched && studentForm.emailId.$error.email , 'has-success': studentForm.emailId.$valid}">

<label for="dob" class="col-sm-3 control-label">Email</label>

<div class="col-sm-6">

<input type="email" id="dob" name="emailId" class="form-control" ng-model="email" />

</div>

<div class="col-sm-3">

<span class="help-block" ng-show="studentForm.emailId.$touched && studentForm.emailId.$error.email">Please enter valid email.</span>

</div>

</div>

<input type="submit" value="Save" class="btn btn-primary col-sm-offset-3" />

<input type="button" value="Reset" class="btn" />

</form>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-151)

In the above example, we have applied ng-class directive to each <div> element with CSS class name and an expression. If an expression evaluates to true then specified CSS class will be applied.

# AngularJS Routing:

We can build [Single Page Application](https://en.wikipedia.org/wiki/Single-page_application) (SPA) with AngularJS. It is a web app that loads a single HTML page and dynamically updates that page as the user interacts with the web app.

AngularJS supports SPA using routing module ngRoute. This routing module acts based on the url. When a user requests a specific url, the routing engine captures that url and renders the view based on the defined routing rules.

Let's see how to implement simple routing in AngularJS application.

## Routing Example:

We will be building an application, which will display a login page when a user requests for base url -*http://localhost/*. Once the user logs in successfully, we will redirect it to student page*http://localhost/student/{username}* where username would be logged in user's name.

In our example, we will have one layout page - index.html, and two HTML templates - login.html and student.html.

1. Index.html - layout view
2. login.html - template
3. student.html - template

The following is a main layout view - index.html.

Example: Layout view - Index.html

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title></title>

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

<script src="Scripts/angular-route.js"></script>

<link href="Content/bootstrap.css" rel="stylesheet" />

</head>

<body ng-app="ngRoutingDemo">

<h1>Angular Routing Demo</h1>

<div ng-view>

</div>

<script>

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

app.config(function ($routeProvider) {

$routeProvider.when('/', {

templateUrl: '/login.html',

controller: 'loginController'

}).when('/student/:username', {

templateUrl: '/student.html',

controller: 'studentController'

}).otherwise({

redirectTo: "/"

});

app.controller("loginController", function ($scope, $location) {

$scope.authenticate = function (username) {

// write authentication code here..

$location.path('/student/' + username)

};

});

app.controller("studentController", function ($scope, $routeParams) {

$scope.username = $routeParams.username;

});

});

</script>

</body>

</html>

Let's understand the above example step-by-step:

1. The first step is to include angular.js, angular-route.js, and bootstrap.css in the index.html. The angular-route.js includes necessary functions for routing.
2. Apply ng-app directive.
3. Apply ng-view directive to <div> or other elements where you want to inject another child view. AngularJS routing module uses ng-view directive to inject another child view where it is defined. Therefore, Angular will inject login.html or student.html inside this div element.
4. Now, create an application module and specify 'ngRoute' as a dependency module.
5. Now, we need to configure the routing rules that need to compile before any other module of an application. So, use config() method to configure the routing rules using [$routingProvider](https://docs.angularjs.org/api/ngRoute/provider/$routeProvider)object.
6. Use $routeProvider.when(path, route) method to configure routing rules, where the first parameter is request URL and the second parameter is an object which contains controller, template, or other properties. In the above example, we specified that if user request for "/" URL, meaning the base url then inject login.html and loginController. In the same way, if a user requests for "/student/:username" url then inject student.html and studentController. The :username would be url parameter.
7. Use otherwise() method to redirect to base url if user request for the URL other than configured rules.
8. Now, define loginController which attaches authenticate() funtion to the $scope. The authenticate() method redirects to "/student/username/" using $location service.
9. Define studentController which attaches username property to $scope, to display user name in the view. Notice that $routeParams is used to get the value of url parameter supplied from login view.

Create login.html as shown below, which contains username and password input box with validation. Please note that we are using bootstrap.css.

Example: login.html

<form class="form-horizontal" role="form" name="loginForm" novalidate>

<div class="form-group" >

<div class="col-sm-3">

</div>

<div class="col-sm-6">

<input type="text" id="userName" name="userName" placeholder="User Name" class="form-control" ng-model="userName" required />

<span class="help-block" ng-show="loginForm.userName.$touched && loginForm.userName.$invalid">Please enter User Name.</span>

</div>

<div class="col-sm-3">

</div>

</div>

<div class="form-group" >

<div class="col-sm-3">

</div>

<div class="col-sm-6">

<input type="password" id="password" name="password" placeholder="Password" class="form-control" ng-model="password" required />

<span ng-show="loginForm.password.$touched && loginForm.password.$error.required">Please enter Password.</span>

</div>

<div class="col-sm-3">

</div>

</div>

<input type="submit" value="Login" class="btn btn-primary col-sm-offset-3" ng-click="authenticate(userName)" />

</form>

Create student.html with necessary fields as shown below. Visit [Bootsrap Form](http://www.tutorialsteacher.com/angularjs/angularjs-form-with-bootstrap) section to learn how to create bootstrap form in AngularJS.

Example: student.html

<div>

<p>Welcome {{username}}</p>

<a href="/">Log out</a>

</div>

<form class="form-horizontal" ng-submit="submitStudnetForm()" role="form">

<div class="form-group">

<label for="firstName" class="col-sm-3 control-label">First Name</label>

<div class="col-sm-6">

<input type="text" id="firstName" class="form-control" ng-model="student.firstName" />

</div>

<div class="col-sm-3"></div>

</div>

<div class="form-group">

<label for="lastName" class="col-sm-3 control-label">Last Name</label>

<div class="col-sm-6">

<input type="text" id="lastName" class="form-control" ng-model="student.lastName" />

</div>

<div class="col-sm-3"></div>

</div>

<div class="form-group">

<label for="dob" class="col-sm-3 control-label">DoB</label>

<div class="col-sm-2">

<input type="date" id="dob" class="form-control" ng-model="student.DoB" />

</div>

<div class="col-sm-7"></div>

</div>

<div class="form-group">

<label for="gender" class="col-sm-3 control-label">Gender</label>

<div class="col-sm-2">

<select id="gender" class="form-control" ng-model="student.gender">

<option value="male">Male</option>

<option value="female">Female</option>

</select>

</div>

<div class="col-sm-7"></div>

</div>

<div class="form-group">

<div class="col-sm-3"></div>

<div class="col-sm-2">

<span><b>Training Location</b></span>

<div class="radio">

<label><input value="online" type="radio" name="training" ng-model="student.trainingType" />Online</label>

</div>

<div class="radio">

<label><input value="onsite" type="radio" name="training" ng-model="student.trainingType" />OnSite</label>

</div>

</div>

<div class="col-sm-7">

<span><b>Main Subjects</b></span>

<div class="checkbox">

<label><input type="checkbox" ng-model="student.maths" />Maths</label>

</div>

<div class="checkbox">

<label><input type="checkbox" ng-model="student.physics" />Physics</label>

</div>

<div class="checkbox">

<label><input type="checkbox" ng-model="student.chemistry" />Chemistry</label>

</div>

</div>

</div>

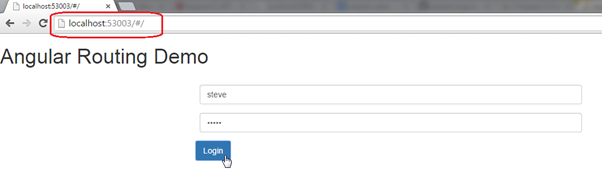
<input type="submit" value="Save" class="btn btn-primary col-sm-offset-3" />

<input type="reset" value="Reset" ng-click="resetForm()"

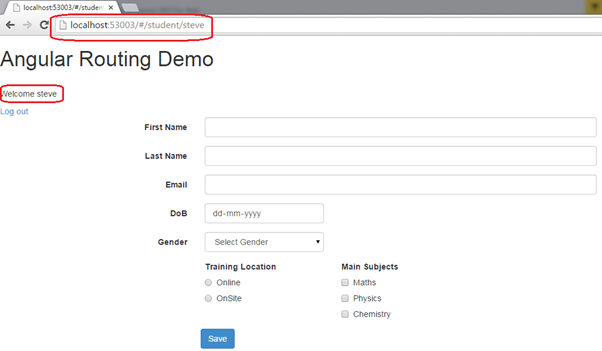
</form>

Notice that login.html and student.html starts from <form> tag, because they are going to be injected in to layout page - index.html. The layout page already contains head and body tag.

Now, when you run the application, it will display a login page as shown below.

[](http://www.tutorialsteacher.com/Content/images/ng/routing-1.png)Routing

Once you enter username and password, it will display student page with supplied username as shown below.

[](http://www.tutorialsteacher.com/Content/images/ng/routing-2.png)Routing

**Note :**AngularJS appends '/#/' to the url to redirect to a particular url using $location service. For example, to redirect to '/student', the url would be http://localhost/#/student.

Thus, you can use routing to create a single page application in AngularJS.

# Exception Handling in AngularJS:

Every application needs proper exception handling mechanism. You can use try, catch, and finally block of JavaScript to handle exceptions in AngularJS modules.

http://www.tutorialsteacher.com/Content/images/tips.png$exceptionHandler does not handle syntax errors.

AngularJS also includes built-in $exceptionHandler service, which handles uncaught exceptions in the application.

The default implementation of $exceptionHandler service logs the exception into the browser console. You can override this service as per your requirement.

The following example demonstrates uncaught exception handling using $exceptionHandler service.

Example: $exceptionHandler

<!DOCTYPE html>

<html ng-app="studentApp">

<head>

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

</head>

<body class="container" ng-controller="studentController">

Status: {{status}} <br />

Data: {{data}} <br />

<input type="button" value="Get Data" ng-click="getStudent()" />

<script>

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

app.config(function ($provide) {

$provide.decorator('$exceptionHandler', function ($delegate) {

return function (exception, cause) {

$delegate(exception, cause);

alert('Error occurred! Please contact admin.');

};

});

});

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

var onSuccess = function (response) {

$scope.status = response.status;

$scope.data = response.data;

};

var onError = function (response) {

$scope.status = response.status;

$scope.data = response.data;

}

$scope.getStudent = function () {

$http.get("/getdata").then(onSuccess, onError);

};

});

</script>

</body>

</html>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=ng-161)

In the above example, we override the $provide service's default behavior using $provide.decorate() method in the app.config() method. The decorate method allow us to override or modify the behavior of the service. So, in the decorate method, we display custom error messages along with logging exception messages to the browser console.

Note that we have used $http service in the studentController. However, we have not included $http service as a parameter in the controller function to raise an exception for demo purpose. Now, the exception will be handled by $exceptionHandler and displays an alert message.

# AngularJS Resources:

## Online Course:

[Learn and Understand AngularJS](http://click.linksynergy.com/fs-bin/click?id=Z8siHqwOs*A&offerid=323058.1906&type=3&subid=0)http://ad.linksynergy.com/fs-bin/show?id=Z8siHqwOs*A&bids=323058.1906&type=3&subid=0

[AngularJS For ASP.NET MVC Developers](http://click.linksynergy.com/link?id=Z8siHqwOs*A&offerid=323058.412952&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fangularjs-for-aspnet-mvc-developers%2F)http://ad.linksynergy.com/fs-bin/show?id=Z8siHqwOs*A&bids=323058.412952&type=2&subid=0

[All You Need To Know About AngularJS - Training On AngularJS](http://click.linksynergy.com/link?id=Z8siHqwOs*A&offerid=323058.177912&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fangularjs%2F)http://ad.linksynergy.com/fs-bin/show?id=Z8siHqwOs*A&bids=323058.177912&type=2&subid=0

[AngularJS Cookbook: Recipes For Common Scenarios](http://click.linksynergy.com/link?id=Z8siHqwOs*A&offerid=323058.599300&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fangularjs-cookbook-recipes-for-common-scenarios%2F)http://ad.linksynergy.com/fs-bin/show?id=Z8siHqwOs*A&bids=323058.599300&type=2&subid=0

[Building an Application with AngularJS](http://click.linksynergy.com/link?id=Z8siHqwOs*A&offerid=323058.420968&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fbuilding-an-application-with-angularjs%2F)