**AngularJS Tutorial**

AngularJS is a JavaScript Framework, like JS, it can be added with the tag <script>. The newest version of AngularJS is 1.6.6. You can use either online resource or local JS file like JQuery.

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

AngularJS extends HTML attributes with Directives, and binds data to HTML with Expressions.

**Example 1:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Document</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body>

<div ng-app="">

<p>Text: <input type="text" ng-model="text"></p>

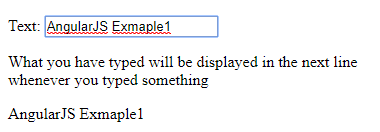
<p>What you have typed will be displayed in the next line<br />whenever you typed something</p>

<p ng-bind="text"></p>

</div>

</body>

</html>



In the example codes, ng-app, ng-model are ng-bind directives.

<div ng-app=""> tells the AngularJS that this element needs to execute as an AngularJS application. <input type="text" ng-model="text"> binds the input field with the application variable “text”. <p ng-bind="text"></p> binds the innerHTML of the <p> element with the AngularJS application variable name.

**Notice:** <p ng-bind="text"></p>is equivalent to <p>**{{**text**}}**</p>. Use data-ng- instead of ng- if you want your html page pass the HTML5 validators but both work if you don’t care.

**AngularJS Expressions:**

There are two types of expression in AngularJS. {{ variable }} double curly bracket and ng-bind="variable".

**Example 2:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Example 2</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body>

<div ng-app="">

<input ng-model="firstName" ng-init="firstName='First'" /><br />

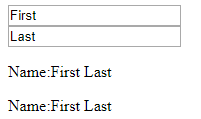
<input ng-model="lastName" ng-init="lastName='Last'" /> <br />

<span>Name:**{{** firstName + " " + lastName **}}**</span><br />

Name:<span ng-bind="firstName+' '+lastName"></span> </div>

</body>

</html>



ng-init="firstName='First'" to initialize the value. However, this is not very common to initialize value by ng-init. See more details later. **{{** firstName + " " + lastName **}}** is equivalent to ng-bind="firstName+' '+lastName".

**AngularJS Object and arrary:**

AngularJS can declare an object with attributes. Use {} for object and [] for array.

**Example 3:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Example 3</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body>

<div ng-app="" ng-init="student={firstName:'Sally',lastName:'Baker',studentID:'123456'}">

<p>

First Name: <span ng-bind="student.firstName"></span><br />

Last Name: **{{**student.lastName**}}**<br />

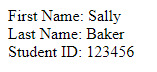
Student ID: **{{**student.studentID**}}**

</p>

</div>

</body>

</html>



student={firstName:'Sally',lastName:'Baker',studentID:'123456'} is a student object, use student.firstName to get the variable like other program language.

**Module and Controller:**

A module is a collection of services, directives, controllers, filters, and configuration information. A controller always belongs to a module. Using angular.module("applicationName", []); to create an AngularJS module. If there is only one parameter, it will retrieve existing module. If there are more than one parameters it will create a new module. Using $scope instead of ng-init to initialize values.

**Example 4:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Example 4</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body>

<div ng-app="studentApp" ng-controller="studentCtrl">

First Name: **{{**firstName**}}** <br />

Last Name: **{{**lastName**}}** <br />

Student ID: **{{**studentID**}}** <br />

</div>

<script>

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

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

$scope.firstName = "Sally";

$scope.lastName = "Bill";

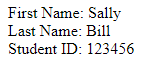
$scope.studentID = "123456";

});

</script>

</body>

</html>



In addition, you can put the script in a js file and link this file in your html page.

first.js

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

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

$scope.firstName = "Sally";

$scope.lastName = "Bill";

$scope.studentID = "123456";

});

Then link <script src="JS Files/first.js"></script> still works.

**Important Notes:**

1. In the AngularJS part, you can use

app.controller("studentCtrl", function ($scope){ //function }

And inline array annotation

app.controller("studentCtrl", ['$scope', function ($scope) { /\* some code \*/}]); From the AngularJS official website dependency annotation part, the second format is preferred as “ If you plan to minify your code, your service names will get renamed and break your app when you use the first style. ”. (<https://docs.angularjs.org/guide/di>)

1. If the script part is easy enough, you can use

app.module('studentApp', []).

controller('studentCtrl', ['$scope', function ($scope) {

/\* code \*/

}]);

If you need call your module more than once, then declare a variable is better.

**Add your own directives:**

In AngularJS, we can add our own directives. It can be used to define our own tag.

**Example 5:**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8" />

<title>Example 5</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body ng-app="myDirectives">

<my-first-directive></my-first-directive>

<div my-first-directive></div>

<div class="my-first-directive"></div>

<!-- directive: my-first-directive -->

<script>

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

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

return {

restrict: "ECAM",

replace: true,

template: "<h1>You must use camel case name in definition.<br/>But use lower case with - when invoke</h1>"

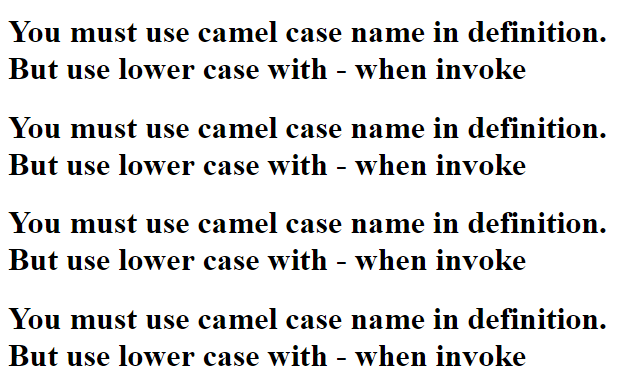
};

});

</script>

</body>

</html>



restrict: "ECAM" can specify what kind of invoke. E for element (tag), C for class, A for attribute and M for comment. In addition, use replace: true when you want your comment replaced by your created directive. Otherwise, it will be invisible.

Note: Use attribute and tag are prefer and default restrict is EA.

**Email Validation:**

AngularJS offers easy way to check if the input is an email address on client-side.

**Example 6:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Example 5</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body>

<form ng-app="" name="myForm">

<input type="email" ng-model="myEmail" name="emailInput">

<p>**{{**myForm.emailInput.$valid**}}**</p>

</form>

</body>

</html>

**Note:** It’s true before you type anything. There are more validation types for fields.

**$untouched**: true if the field has not been touched

**$touched**: true if the field has been touched

**$pristine**: true if the field has not been modified

**$dirty**: true if the field has been modified

**$invalid**: true if the field is invalid

**Filters**

AngularJS has some built-in filters which we can use to filter data. We can also add our own filters. The format to use a filter is add | after the AngularJS expression. For example {{expression | filter1 | filter2 | filter3}}. Therefore, you can use more than one filter in one bracket.

**Example 7: Built-in filters**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example 7</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body ng-app="myFilter" ng-controller='myCt'>

<ul>

<li ng-repeat="x **in** students ">**{{**'Name: '+x.name+' Student ID:'+x.studentId | uppercase**}}**</li>

</ul>

<script>

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

app.controller('myCt', ['$scope', function ($scope) {

$scope.students = [

{ name: 'Sally', studentId: '123456' },

{ name: 'Baker', studentId: '123457' },

{ name: 'Brand', studentId: '123458' },

{ name: 'Coke', studentId: '123459' },

{ name: 'Grady', studentId: '123460' },

{ name: 'Tony', studentId: '123461' }

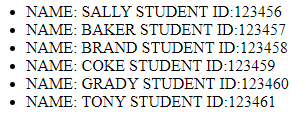
];

}]);

</script>

</body>

</html>



All the outputs in the {{}} will be uppercase as we use an uppercase filter here. AngularJS has some built in filters such as **lowercase, currency, date, orderBy, limitTo, number and json**. Now we create a custom filter for our application.

**Example 8: Create Customer Filter**

<!DOC<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example 8</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body ng-app="myFilter" ng-controller='myCt'>

<input type="text" ng-model='yourInput'>

<p>

What you have typed will be reversed.<br />

**{{**yourInput|reverse**}}**

</p>

<script>

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

app.controller('myCt', ['$scope', function ($scope) {

$scope.yourInput = "CS555 AngularJS";

}]);

app.filter('reverse', function () {

return function (input) {

var out = '';

for (var i = input.length; i > -1; i--) {

out += input.charAt(i);

}

return out;

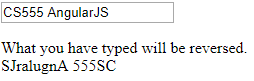
};

});

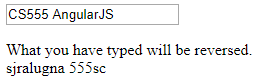
</script>

</body>

</html>



In this example, we create a filter which can reverse the input. If you add one more lowercase filter **{{**yourInput|reverse|lowercase**}}**. The output will be reversed and convert to lower case.



**Table**

Display data in a table in AngularJS is very easy especially for data from database. Whenever the database can retrieve data and return JSON format to AngularJS application page, you can use very few code to display all the data in a table.

**Example 9:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example 9</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

<style>

table, td, th {

border: 1px solid black;

border-collapse: collapse;

padding: 10px;

}

</style>

</head>

<body ng-app="myTable" ng-controller='myCt'>

<table>

<tr>

<th>Name:</th>

<th>Student ID</th>

</tr>

<tr ng-repeat="x **in** students">

<td>**{{**x.name**}}**</td>

<td>**{{**x.studentId**}}**</td>

</tr>

</table>

<script>

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

app.controller('myCt', ['$scope', function ($scope) {

$scope.students = [/\* data may come from other source like DB \*/

{ name: 'Sally', studentId: '123456' },

{ name: 'Baker', studentId: '123457' },

{ name: 'Brand', studentId: '123458' },

{ name: 'Coke', studentId: '123459' },

{ name: 'Grady', studentId: '123460' },

{ name: 'Tony', studentId: '123461' }

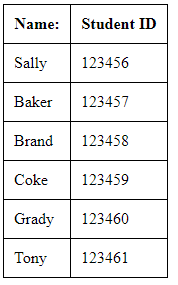
];

}]);

</script>

</body>

</html>



You can either use style css in the head or AngularJS built-in functions with inline style.

<style>

table tr:nth-child(even) {

background-color: #666666;

}

</style>

And

<tr ng-repeat="x **in** students">

<td ng-if="$even" style="background-color: #666666;">**{{**x.name**}}**</td>

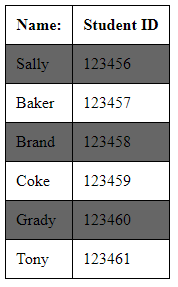
<td ng-if="$odd">**{{**x.name**}}**</td>

<td ng-if="$even" style="background-color: #666666;">**{{**x.studentId**}}**</td>

<td ng-if="$odd">**{{**x.studentId**}}**</td>

</tr>

Have the same css style.



<tr ng-repeat="x **in** students"> can access each object in students.

**Form:**

AngularJS also supports form elements ant they are very similar with html form. In AngularJS, you can use either value attribute or ng-model to initialize data.

**Example 10:**

<form action="">

<input type="text" ng-model="firstname" value="Greetings!" /> <br />

</form>

<script>

angular.module('myText', []).

controller('myCt', ['$scope', function ($scope) {

$scope.firstname = 'CS555';

}]);

</script>

If you delete ng-model="firstname" , the text box displays “Greetings!” Otherwise, it shows the value of firstname which default is empty.



**Checkbox** is easy to use with ng-show directive.

**Example 11:**

<form action="">

Check to display message<input type="checkbox" ng-model="check1"/>

<pre ng-show="check1">

If you can see this message, it means you have checked the checkbox.

</pre>

</form>





**Radio button** in AngularJS is almost same as in standard html. In html, we group few radio buttons with same name. But in AngularJS, we use the same ng-model value.

**Example 12:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example 12</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body ng-app="">

<form action="">

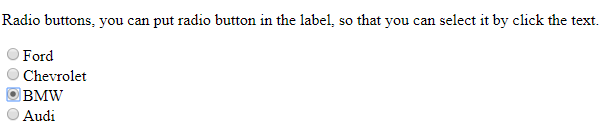
<input type="radio" ng-model="radio1" value="Ford" ng-checked="true"/>Ford<br />

<input type="radio" ng-model="radio1" value="Chevrolet"/>Chevrolet

</form>

</body>

</html>



**Select** can build a dropdown list for selecting.

**Example 13:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example 13</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body ng-app="">

<form>

Select a topic:

<select ng-model="myVar">

<option value=""/>

<option value="dogs"/>Dogs

<option value="cats"/>Cats

<option value="tigers"/>Tigers

<option value="wolves"/>Wolves

</select>

</form>

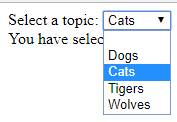
<div>

You have selected **{{**myVar**}}**

</div>

</body>

</html>





We can also use built-in directive ng-options and ng-repeat to create a dropdown list. However, ng-repeat has its limitation that the selected value must be a string. In ng-options, we can select an item and pass the whole object.

**Example 14:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example 14</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

</head>

<body ng-app="mySelect" ng-controller="myCt">

<select ng-model="mySelected1">

<option ng-repeat="x **in** students">**{{**x.name**}}**</option>

</select>

<p>

{

**{{**mySelected1**}}**<br />

**{{**mySelected1.name**}}**<br />

**{{**mySelected1.studentId**}}**

}

</p>

<select ng-model="mySelected2" ng-options="x.name for x **in** students">**{{**x.name**}}**</select>

<p>

{

**{{**mySelected2**}}**<br />

**{{**mySelected2.name**}}**<br />

**{{**mySelected2.studentId**}}**

}

</p>

<script>

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

app.controller('myCt', ['$scope', function ($scope) {

$scope.students = [/\* data may come from other source like DB \*/

{ name: 'Sally', studentId: '123456' },

{ name: 'Baker', studentId: '123457' },

{ name: 'Brand', studentId: '123458' },

{ name: 'Coke', studentId: '123459' },

{ name: 'Grady', studentId: '123460' },

{ name: 'Tony', studentId: '123461' }

];

}]);

</script>

</body>

</html>

Focus on

<select ng-model="mySelected1">

<option ng-repeat="x **in** students">**{{**x.name**}}**</option>

</select>

<p>

{

**{{**mySelected1**}}**<br />

**{{**mySelected1.name**}}**<br />

**{{**mySelected1.studentId**}}**

}

</p>

And

<select ng-model="mySelected2" ng-options="x.name for x **in** students">**{{**x.name**}}**</select>

<p>

{

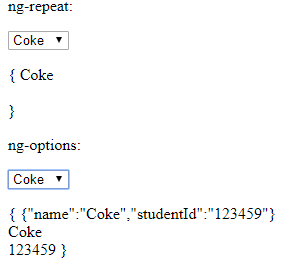
**{{**mySelected2**}}**<br />

**{{**mySelected2.name**}}**<br />

**{{**mySelected2.studentId**}}**

}

</p>

If you use ng-options, you don’t need a select-option nest tag. 

You can see that ng-options return the object as json format and ng-repeat only return the string. Choose either one depends on your situation.

**Animation:**

To use animation, you need also include AngularJS animation library and you must add ngAnimate as a dependency for you AngularJS application. If your application already has a ng-app,

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular-animate.js"></script>

**Example 15:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example 15</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular-animate.js"></script>

<style>

div {

background-color: lightpink;

border: 1px solid black;

text-align: center;

width: 200px;

height: 100px;

padding: 10px;

transition: all linear 1s;

}

.ng-hide {

background-color: lightgreen;

}

</style>

</head>

<body ng-app="ngAnimate">

<label> <input type="checkbox" ng-model="checkBox1">Hide the div.</label>

<div ng-hide="checkBox1">

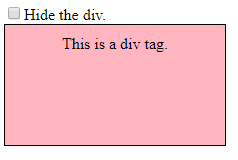
This is a div tag.

</div>

</body>

</html>

At first, it will display the div and the background color is light pink.



After you click the checkbox, the background color turn to be light green and div tag will be hide after 1 second.

