Ayisha Farhath Abbas
Assistant Professor
Department of Computer Science & IT

FILTERS

FILTER

Filter is used to format the value of data. The pipe sign (|) indicates that filter is used. The proper syntax of filter looks like this:

Value | filter

Let`s try to understand the filers one by one.

UPPERCASE FILTER

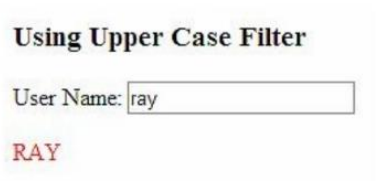
Value | uppercase

The uppercase filter changes the text to upper case. Suppose a user writes a text in lower case (e.g. ray) or title case (e.g. Ray) or in mixed case (e.g. rAy or RaY or rAY etc.), and you want the upper case result, then you will have to use upper case filter.

Program

```
<!DOCTYPE html>

<html >

<head>

<title>AngularJSfor beginners</title> <script src="js\angular.min.js">

</script> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/> </head>

<body>

<h3>Using Upper Case Filter</h3> <div ng-app="" ng-init="Username= 'ray' "> <p>User Name: <input type="text" ng-model = "Username"></p>

<p style="color:red" ng-bind="Username | uppercase"></p> </div>

</body>

</html>
```

Output:



“Username |uppercase” changes the value of “Username” to uppercase.

In the above program, the default value of ray is in lower case, but the result becomes upper case (RAY).

LOWERCASE FILTER

Value | lowercase

The lowercase filter changes the text to lower case. Suppose a user writes a text in upper case (e.g. RAY YAO) or title case (e.g. Ray Yao) or in mixed case (e.g. rAy or RaY or rAY etc.), and you want the lower case result, then you will have to use lower case filter.

Program

<html >

<head>

<title>AngularJSfor beginners</title> <script src="js\angular.min.js">

</script> </head>

<body>

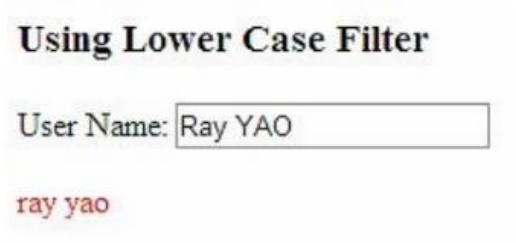
<h3>Using Lower Case Filter</h3> User Name: <input type="text" ng-model="Username">

</body>

</html>

Open the notepad and paste the above mentioned code with .html extension.

Output:



"Username | lowercase": changes the value of "Username" to lowercase.

In the above Program, when I enter text (Ray YAO) in upper case, but the result become lower case (ray yao).

ORDERBY FILTER

OrderBy filer is used to display values in ascending order or descending order.

The syntax of “orderBy” looks like this:

Value | orderBy: ’value’ //for ascending order

Value | orderBy: ’-value’ //for descending order

Let`s take a program for better understanding.

Program

```
<!DOCTYPE html>

<html >

<head>

<title>AngularJSfor beginners</title> <script src="js\angular.min.js">

</script> </head>

<body>

<h1>Using OrderBy filter</h1> <div ng-app="" ng-init="StudentsResult=[{name: 'Tienq', marks:81}, {name: 'Svbrf', marks:70},{name: 'Yaito', marks:90}, {name: 'Pewfn', marks:63}, {name:'Riet', marks:98}]"> <table border="1" > <tr>

<th>Student Name</th> <th>Mathematics’ Result</th> </tr>

<tr ng-repeat="x in StudentsResult | orderBy:'-marks' "> <td ng-bind="x.name "></td> <td ng-bind="x.marks "></td> </tr>

</table>

</div>

</body></html>
```

Output

Using OrderBy filter

Student Name	Mathematics’ Result
Riet	98
Yaito	90
Tienq	81
Svbrf	70
Pewfn	63

StudentsResult | orderBy:'-marks' displays the values of StudentsResult in descending order.

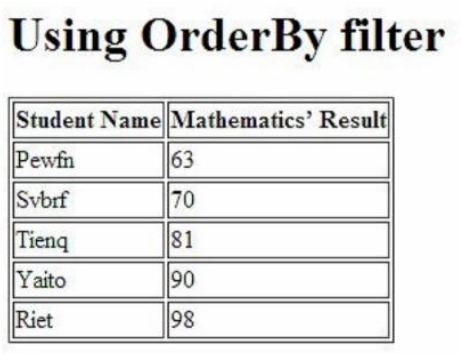
You can see that the highest mark is on top and the lowest mark is on bottom by using (value | orderBy:'-marks').

If you want reverser the order, you can remove the “-“sign”.

Program

```
<!DOCTYPE html>

<html >

<head>

<title>AngularJSfor beginners</title> <script src="js\angular.min.js">

</script> </head>

<body>

<h1>Using OrderBy filter</h1> <div ng-app="" ng-init="StudentsResult= [{name: 'Tienq', marks:81}, {name: 'Svbrf', marks:70}, {name: 'Yaito', marks:90}, {name: 'Pewfn', marks:63}, {name: 'Riet',

marks:98}]"> <table border="1" >

<tr>

<th>Student Name</th> <th>Mathematics’ Result</th> </tr>

<tr ng-repeat="x in StudentsResult | orderBy:'marks' "> <td ng-bind="x.name "></td> <td ng-bind="x.marks "></td> </tr>

</table></div>

</body></html>
```

Output



StudentsResult | orderBy: 'marks' displays the values of the StudentsResult in ascending order.

CURRENCY FILTER

Value | currency

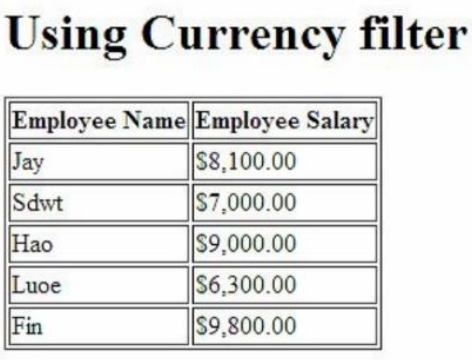
The currency filter is used to display the result in currency format.

Program

```
<!DOCTYPE html>

<html >

<head>

<title>AngularJSfor beginners</title> <script src="js\angular.min.js">

</script> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/> </head>

<body>

<h1>Using Currency filter</h1> <div ng-app="" ng-init ="Employees_Monthly_Salary=[{name:
'Jay', salary:8100}, {name: 'Sdwt', salary:7000}, {name: 'Hao', salary:9000}, {name:
'Luoe',salary:6300}, {name: 'Fin', salary:9800}]"> <table border="1" > <tr>

<th>Employee Name</th> <th>Employee Salary</th> </tr>

<tr ng-repeat="x in Employees_Monthly_Salary "> <td ng-bind="x.name"></td> <td ng-
bind="x.salary | currency "></td> </tr>

</table>

</div></body></html>
```

Open the notepad and paste the above mentioned code with .html extension.

Output:



Using Currency filter

Employee Name	Employee Salary
Jay	$8,100.00
Sdwt	$7,000.00
Hao	$9,000.00
Luoe	$6,300.00
Fin	$9,800.00

"x.salary | currency " converts the salary to currency format.

In the above program, there are two columns in the table, the first column is Employee Name and the second is Employee Salary. The salary column displays the salary in currency format.

ARRAY FILTER

Array | filter:input

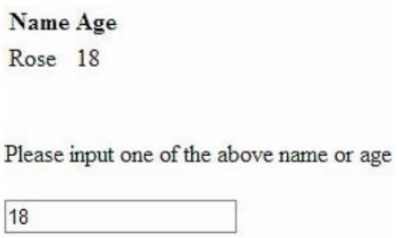
“Array | filter:input” can filter the array elements based on the user input.

Program

```
<!DOCTYPE html>
<html ng-app="">
<head>
<script src="js\angular.min.js"></script> <meta charset="utf-8"> </head>
<body>
<div ng-init="students = // define an array “students”
[{name:'Andy', age:'19'},
{name:'Rose', age:'18'},
{name:'Jony', age:'17'},
{name:'Judy', age:'16'},
{name:'Tomy', age:'15'},
{name:'Lily', age:'14'}]"> </div>
<table>
<tr><th>Name</th><th>Age</th></tr> <tr ng-repeat="person in students |
filter:myList" > // filter the array “students” according to the input value
<td>{{person.name}}</td> <td>{{person.age}}</td> </tr>
</table>
<br><br>
<label>Please input one of the above name or age <br><br>
<input ng-model="myList"> </label> // user input </body>
</html>
```

Please try to input a number 18 to text field.

Output:



"<div ng-init="students =...." defines an array "students".

"person in students | filter:myList" filters the array "students" according to the input value "myList".

<input ng-model="myList"> accepts the user input, and store the input value to "myList".

When you input 18 to text field, the output shows "Rose 18".

EVENTS

EVENT

Events are associated with different HTML elements. e.g. the click event is associated with button element; similarly, keypress event is associated with text box or text area element. AngularJS provides multiple events which are associated with HTML control.

CLICK EVENT

ng-click = "expression"

ng-click = "expression" defines a click event.

When a button is clicked, an event occurs, and evaluates the expression. The click event normally works on button.

Program

<!DOCTYPE html>

<html >

<head>

<title>AngularJSfor beginners</title> <script src="js\angular.min.js">

</script> </head>

<body>

<h3>Add Two Numbers Using Click Event</h3> First Number : <input type="number" ng-model = "firstNumber">

Second Number:<input type="number" ng-model = "secondNumber"> <button ng-click="Result=firstNumber + secondNumber"> Add Numbers </button>

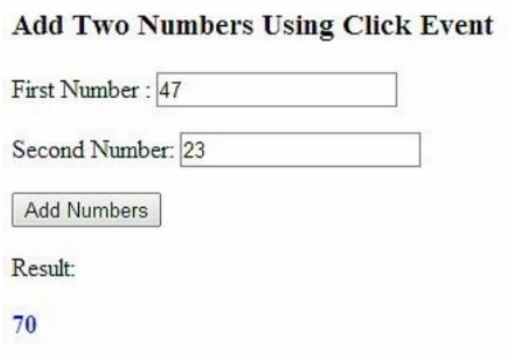
Result:

</body>

</html>

Open the notepad and paste the above mentioned code with .html extension.

Output



"<button ng-click="Result=firstNumber + secondNumber"> Add Numbers

</button>": when the button is clicked, an event occurs. "firstNumber" adds "secondNumber", and assigns the result to "Result".

In the above program, at first time when the page is loaded, the first text box has a number 47, and the second text box has a number 23, but there is no result. When I click on button (Add Numbers), the result displaying in Result area is 70.

DOUBLE CLICK EVENT

ng-dblclick = "expression"

ng-dblclick = "expression" defines a double click event. When a button is double clicked, an event occurs, and evaluates the expression.

Double click event normally works on button.

Program

<!DOCTYPE html>

<html >

<head>

<title>AngularJS for beginners</title> <script src="js/angular.min.js">

</script> </head>

<body>

<h4>Add Two Numbers Using Double Click Event</h4> First Number: <input type="number" ng-model="firstNumber">

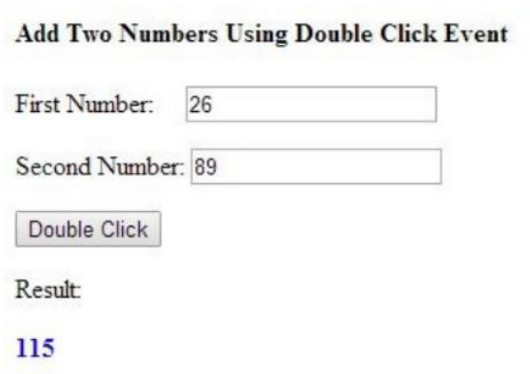
 Second Number: <input type="number" ng-model="SecondNumber"> <button ng-dblclick="Result=firstNumber + SecondNumber">Double Click</button> Result:

</body>

</html>

Open the notepad and paste the above mentioned code with .html extension.

Output:



"<button ng-dblclick="Result=firstNumber + secondNumber"> Add Numbers </button>": when the button is double clicked, an event occurs. "firstNumber" adds "secondNumber", and assigns the result to "Result".

In the above program, at first time when the page is loaded, the first text box has a number 26, and the second text box has a number 89, but there is no result.

When I double click on the button (Add Numbers), the result displaying in Result area is 115.

MOUSE MOVE EVENT

ng-mousemove = "expression"

ng-mousemove = "expression" defines a mouse move event. When the mouse moves, an event occurs, and evaluates the expression.

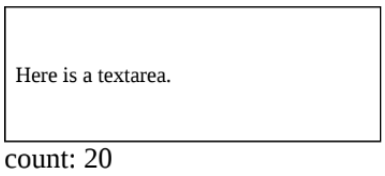
Mouse move event normally works on div, body and specific area or element.

Program

```
<!doctype html>
<html>
<head>
<script src="js\angular.min.js"> </script>
</head>
<body ng-app="">
<br><br>
<textarea ng-mousemove="count = count + 1" ng-init="count=0">
Here is a textarea
</textarea>
<br><br>
<h2>count: {{count}}</h2> </body>
</html>
```

(Assume you move the mouse on the textarea for 20 times.)

Output:



count: 20

“ng-mousemove="count = count + 1"” : when mouse moves on the textarea,

“count” increases 1.

“ng-init="count=0"” initializes the “count” value as 0.

{{count}} displays the value of “count”.

“count : 20” means that mouse moves for twenty times.

MOUSE OVER EVENT

ng-mouseover = “expression”

ng-mouseover = “expression” defines a mouse over event. When the mouse overs over, an event occurs, and evaluates the expression.

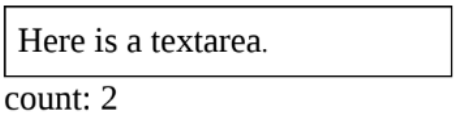
Mouse over event normally works on div, body and specific area or element.

Program

```
<!doctype html>
<html>
<script src="js\angular.min.js"></script> <body ng-app="">
<br><br>
<textarea ng-mouseover="count = count + 1" ng-init="count=0">
Here is a textarea.
</textarea>
<br><br>
<h2>count: {{count}}</h2> </body>
</html>
```

(Assume you move the mouse over the textarea for 2 times.)

Output:



count: 2

“ng-mouseover="count = count + 1"” : when mouse moves over the textarea,

“count” increases 1.

“ng-init="count=0"” initializes the “count” value as 0.

{{count}} displays the value of “count”.

“count : 2” means that mouse moves over the textarea for two times.

KEY UP EVENT

ng-keyup = “expression”;

ng-keyup = “expression” defines a key up event. When the key is up in a specified element, an event occurs, and evaluates the expression.

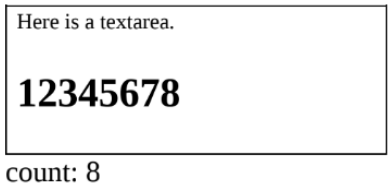
Key up event normally works on text box and text area.

Program

```
<!doctype html>
<html>
<head>
<script src="js\angular.min.js"> </script>
</head>
<body ng-app="">
<br><br>
<textarea ng-keyup="count = count + 1" ng-init="count=0">
Here is a textarea
</textarea>
<h2>count: {{count}}</h2> <body>
</html>
```

(Assume that you type 12345678 in the textarea.)

Output:



“ng-keyup="count = count + 1"” : when typing something and key up on the textarea, “count” increases 1.

“ng-init="count=0"” initializes the “count” value as 0.

{{count}} displays the value of “count”.

“count : 8” means that the typing makes key up 8 times.

KEY DOWN EVENT

ng-keydown = “expression”;

ng-keyup = “expression” defines a key down event.

When the key is down in a specified element, an event occurs, and evaluates the expression.

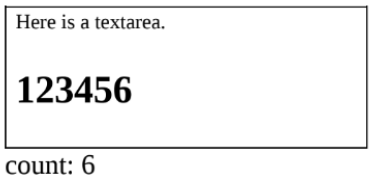
Key down event normally works on text box and text area.

Program

```
<!doctype html>
<html>
<head>
<script src="js\angular.min.js"> </script>
</head>
<body ng-app="">
<br><br>
<textarea ng-keydown="count = count + 1" ng-init="count=0">
Here is a textarea
</textarea>
<br><br>
<h2>count: {{count}}</h2> <body>
</html>
```

(Assume that you type 123456 in the textarea.)

Output:



“ng-keydown="count = count + 1"” : when typing something and key down on the textarea, “count” increases 1.

“ng-init="count=0"” initializes the “count” value as 0.

{{count}} displays the value of “count”.

“count : 6” means that the typing makes key down 6 times.

Ayisha Farhath Abbas

Assistant Professor

Department of Computer Science & IT

UNIT V

EXPRESSION

{{Expression}}

{{Expression}} is used to bind the value with html element and displays the value. It works same as ng-bind directive. {{Expression}} is written within two curly brackets. The {{expression}} is basically pure JavaScript expression.

There are different kinds of expression, let`s talk one by one.

STRING EXPRESSION

We know that string is collection of characters. In AngularJS the string expression looks like this.

<element> {{First String + Second String}}

</element>

Program

<!DOCTYPE html>

<html >

<head>

<title>AngularJS for beginners</title> <script src="js/angular.min.js">

</script> </head>

<body>

<h4>Combine Two String Using String Expression</h4>

First String   : <input type="text" ng-model="firstString"/>

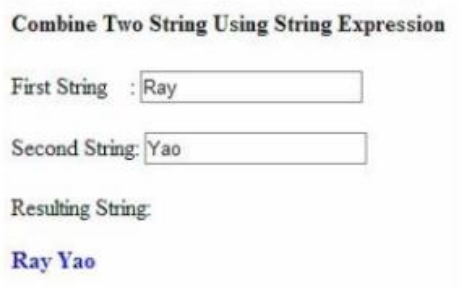
 Second String: <input ng-model="secondString"/>

 Resulting String: {{firstString +" "+secondString}}

</body>

</html>

Output:



“{{firstString +" "+secondString}}” joins two strings together.

{{ expression}} displays the value of expression.

In the above Program, the text Ray is written in the first text box and Yao in the second text box, but in the resulting string area, Ray string and Yao string are combined due to use of string expression {{ }}.

NUMBER EXPRESSION

In AngularJSyou can perform different mathematic operation by using Number Expression.

<element> {{First Number + Second Number}}

</element>

Program

<!DOCTYPE html>

<html >

<head>

<title>AngularJS for beginners</title> <script src="js/angular.min.js">

</script> </head>

<body>

<h4>Multiply Two Number Using Number Expression</h4> First Number   : <input type="number" ng-model="firstNumber"/>

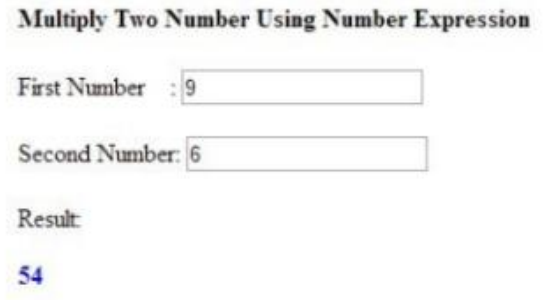
 Second Number: <input type="number" ng-model="secondNumber"/>

 Result: {{firstNumber * secondNumber}}

</body>

</html>

Output:



"{{firstNumber * secondNumber}}" multiplies the firstNumber and the secondNumber.

{{ expression}} displays the value of expression.

In the above Program, the number 9 is written in the first text box and 6 in the second text box, and 54 is the result of multiplication of 9 and 6. You can perform any arithmetic operation by using Number Expression.

OBJECT EXPRESSION

AngularJSobject works like a JavaScript object. The syntax looks like this:

object = {property: value}

Program

<html >

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

<body>

<h4>Object Expression</h4> <div ng-app="" ng-init="EmployeeObject =

{Emp_name: 'Jay Smith', Emp_Month: 'June.15 2015', Emp_salary:

'$8000'}"> Employee Name : {{EmployeeObject.Emp_name}}

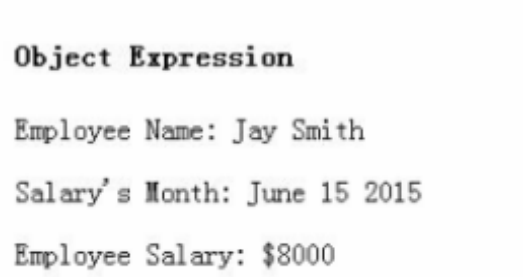
Salary's Month: {{EmployeeObject.Emp_Month}} Employee

Salary: {{EmployeeObject.Emp_salary}}

</body>

</html>

Output:



Explanation:

“EmployeeObject” is an object.

“Emp_name” is a property.

“Emp_Month” is a property.

“Emp_salary” is a property.

{{ object.property }} displays the value of the property.

ARRAY EXPRESSION

The array expression of AngularJS works like JavaScript array. The syntax looks like this:

Array=[val1, val2, val3,]

Program

<!DOCTYPE html>

<html >

<head>

<title>AngularJS for beginners</title> <script src="js/angular.min.js">

</script> </head>

<body>

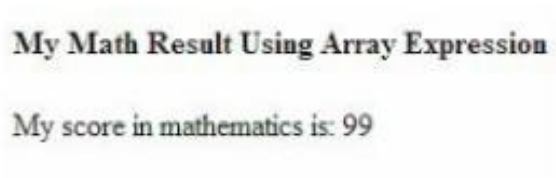
<h4>My Math Result Using Array Expression</h4> <div ng-app="" ng-

init="MyArray=[98,96,93,90,99]"> My score in mathematics is:{{MyArray[4]}}

</body>

</html>

Output:



“MyArray=[98,96,93,90,99]” is an array.

“{{MyArray[4]}}” displays the value whose index is 4 in MyArray.

USING CONTROLLER & SCOPE

CONTROLLER

The controller is basically a JavaScript object that controls the flow of data of an application. So the AngularJS application is controlled by Controller.

DEFINING CONTROLLER

The ng-controller directive is used to define the Controller. We know that Controller is a JavaScript object which contains JavaScript function and properties. The syntax of Controller is as following:

Program

<html>

<script src= "js\angular.min.js"></script> <body>

 First Number: <input type="number" ng-model="firstNumber">
 Second Number: <input type="number" ng-model="secondNumber">

Sum: {{firstNumber + secondNumber}}

<script>

var app = angular.module('Calculation', [ ]); app.controller('myController',

function($scope) {

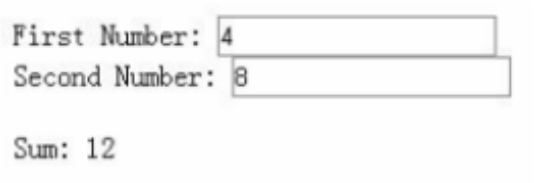
$scope.firstNumber = 4; $scope.secondNumber = 8; });

</script>

</body>

</html>

Output:



“ng-controller="myController"” defines a controller .

“app = angular.module()” creates a new module.

“app.controller()” adds the controller's constructor function to the module.

“function($scope) ” defines a constructor function, and also defines an object $scope. When page is loaded, the function will run.

“$scope.property = value” assigns the value to the property of $scope object.

In the above Program, at the first time when the page is loaded, you see there are two textboxes with different numbers, which are 4 and 8; the result area displays the result 12.

angular. module ( ): AngularJS module is a collection of various part of an application, such as controllers, services, filters, directives, etc.

SCOPE

Scope is a JavaScript object which contains model data.

function ($scope) { }

$scope is a parameter of JavaScript function which is called by a controller.

Let`s take an Program for understanding.

Program

```
<script>
function ($scope) {
$scope.firstNumber = 23;
$scope.secondNumber = 63;
}
</script>
```

In above Program, the $scope is the parameter of the function ($scope) { }. $scope is an object in AngularJS.

$scope.firstNumber and $scope.secondNumber are models used in HTML. Model data is accessed by the $scope object. We assign values to the model with following formula: “$scope.property = value”.

For example:

```
$scope.firstNumber = 23;
$scope.secondNumber = 63;
```

MVC & SCOPE

What is MVC?

MVC stands for Model, View, and Controller.

The Model directly manages the data, logic and rules of the application.

The View displays the data.

The Controller handles the input.

In above Program:

$scope.firstNumber and $scope.secondNumber are models.

  is a view used in HTML

“ng-controller="myController""defines a controller.

The communication between controller and view is called Scope, so we can say

Scope works like a bridge that connects controller to view.

Module Basic

app = angular.module()

“app = angular.module()” creates a new module.

Program

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8"> <script src="js\angular.min.js"></script> </head>

<body>

 Please select your favorite fruit. <select ng-model="choosedFruit" ng-options="obj.fruit for obj in fruits" size = "4"> </select>

<h3>Your favorite fruit is: {{choosedFruit.fruit}}</h3> <h3>The color is:{{choosedFruit.color}}<h3>

<script>

var app = angular.module('myFruit', [ ]); app.controller('iController',

function($scope) {

$scope.fruits = [{fruit : "Apple", color : "Red"}, {fruit : "Orange", color : "Golden"}, {fruit :"Banana", color : "Green"}

];

});

</script>

</body>

</html>

Output:

 defines a controller “iController”.

<select ng-model="choosedFruit" ng-options="obj.fruit for obj in fruits" size ="4"> creates a select manu, which accepts the user input to variable “chooseFruit”.

ng-options="obj.fruit for obj in fruits" create some options “fruit” for select manu. “for...in...” iterates through each options in array fruits, and stores their values to obj.

“size = 4” sets the size of select manu.

{{choosedFruit.fruit}} shows your selected fruit.

{{choosedFruit.color}} shows the fruit color.

var app = angular.module('myFruit', [ ]) creates a new module in application “myFruit”.

The [ ] parameter can be used to define dependent modules.

app.controller('iController', function($scope) adds the controller's constructor function to the module.

“function($scope)” defines a controller constructor function, and also defines an object $scope. When page is loaded, the function will run.

$scope.property = value” assigns a value to the property of $scope object, which is a Model in AngularJS.

ANGULARJS MODULE

An AngularJS module is a collection of various part of an application, such as controllers, services, filters, directives, etc. An AngularJS module defines an application.

The syntax of create a module looks like this:

var myModule = angular.module("myApplication", [

]);

Program

<html>

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

<body>

{{ firstNum + lastNum }}

<script>

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

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

$scope.firstNum = 100;

$scope.lastNum = 200;

});

</script>

</body>

</html>

Program

<html>

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

<body>

{{ firstNum + lastNum }}

<script>

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

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

$scope.firstNum = 100;

$scope.lastNum = 200;

});

</script>

</body>

</html>

Output: 300

 indicates that this application "myApplication" has one controller “myController”.

{{ firstNum + lastNum }} displays the result of firstNum plus lastNum.

“var myModule = angular.module("myApplication", [ ])” creates a new module “myModule” with “myApplication”.

The [ ] parameter can be used to define dependent modules.

“myModule.controller("myController", function($scope)” adds the controller's constructor to the module using the controller( ) method, makes myController belongs to myModule.

$scope is defined as an object.

The value of “$scope.firstNum” is100.

The value of “$scope.lastNum” is 200.

Therefore, the result of {{ firstNum + lastNum }} is 300.

ANGULARJS API

API means Application Programming Interface.

There are 10 basic functions on API:

angular.uppercase()

angular.lowercase()

angular.isString()

angular.isNumber()

.....

uppercase( )

angular.uppercase( ) converts a string to uppercase.

Program

```
<html>
<script src= "js\angular.min.js"></script>
<body>
<div ng-app="myApplication" ng-controller="myController">
<p>{{ text1 }}</p>
<p>{{ text2 }}</p>
</div>
<script>
var myModule = angular.module('myApplication', [ ]);
myModule.controller('myController', function($scope) {
$scope.text1 = "javascript";
$scope.text2 = angular.uppercase($scope.text1);
});
</script>
</body>
</html>
```

Output:

javascript

JAVASCRIPT

“var myModule = angular.module('myApplication', [ ])” creates a new module “myModule” with “myApplication”.

“myModule.controller('myController', function($scope)” adds the controller's constructor to the module using the controller( ) method, makes myController belongs to myModule.

$scope is defined as an object.

“$scope.text2 = angular.uppercase($scope.text1)” converts the “text1” to uppercase.

lowercase( )

angular.lowercase( ) converts a string to lowercase.

Program

```
<html>
<script src= "js\angular.min.js"></script>
<body>
<div ng-app="myApplication" ng-controller="myController">
<p>{{ text1 }}</p>
<p>{{ text2 }}</p>
</div>
<script>
var myModule = angular.module('myApplication', [ ]);
myModule.controller('myController', function($scope) {
$scope.text1 = "JAVASCRIPT";
$scope.text2 = angular.lowercase($scope.text1);
});
</script>
</body>
</html>
```

Output:

JAVASCRIPT

Javascript

“var myModule = angular.module('myApplication', [ ])” creates a new module “myModule” with “myApplication”.

“myModule.controller('myController', function($scope)” adds the controller's constructor to the module using the controller( ) method, makes myController belongs to myModule.

$scope is defined as an object.

“$scope.text2 = angular.lowercase($scope.text1)” converts the “text1” to lowercase.

isString( )

isString( ) tests a value to see if it is a string, returns true if the value is a string.

Program

```
<html>
```

```
<script src= "js\angular.min.js"></script>
<body>
<div ng-app="myApplication" ng-controller="myController">
<p>{{ text1 }}</p>
<p>{{ text2 }}</p>
</div>
<script>
var myModule = angular.module('myApplication', [ ]);
myModule.controller('myController', function($scope) {
$scope.text1 = "JAVASCRIPT";
$scope.text2 = angular.isString($scope.text1);
});
</script>
</body>
</html>
```

Output:

JAVASCRIPT

True

"var myModule = angular.module('myApplication', [ ])" creates a new module "myModule" with "myApplication".

"myModule.controller('myController', function($scope)" adds the controller's constructor to the module using the controller( ) method, makes myController belongs to myModule.

$scope is defined as an object.

"$scope.text2 = angular.isString($scope.text1)" tests "text1" to see if it is a string.

isNumber( )

isNumber( ) tests a value to see if it is a number, returns true if the value is a number.

Program

```
<html>
<script src= "js\angular.min.js"></script>
<body>
<div ng-app="myApplication" ng-controller="myController">
<p>{{ text1 }}</p>
<p>{{ text2 }}</p>
```

```
</div>
<script>
var myModule = angular.module('myApplication', [ ]);
myModule.controller('myController', function($scope) {
$scope.text1 = "JAVASCRIPT";
$scope.text2 = angular.isNumber($scope.text1);
});
</script>
</body>
</html>
```

Output:

JAVASCRIPT

False

Explanation:

“var myModule = angular.module('myApplication', [ ])” creates a new module “myModule” with “myApplication”.

“myModule.controller('myController', function($scope)” adds the controller's constructor to the module using the controller( ) method, makes myController belongs to myModule.

$scope is defined as an object.

“$scope.text2 = angular.isNumber($scope.text1)” tests “text1” to see if it is a number.

isDate()

angular.isDate() tests a value to seem if a Date object returns true if the value is a date object.

Program

```
<html>
<script src= "js\angular.min.js"></script>
<body>
<script>
var myDate = "Aug/10/2015";
var dateObject = new Date();
document.write(angular.isDate(myDate) + " ");
document.write(angular.isDate(dateObject) + " ");
</script>
</body>
```

```
</html>
```

Output:

false true

myDate is a string, therefore, angular.isDate(myDate) returns false.

dateObject is an object of Date, therefore, angular.isDate(dateObject) returns true.

isFunction( )

angular.isFunction( ) tests a value to see if it is a function, returns true if the value is a function.

Program

```
<html>
<script src= "js\angular.min.js"></script>
<body>
<script>
var myString = "This is a string";
function myFunction()
{
return "This is inside a function";
}
document.write(angular.isFunction(myString) + " ");
document.write(angular.isFunction(myFunction) + " ");
</script>
</body>
</html>
```

Output:

false true

myString is a string, so the test result is false.

myFuction is a function, so the test result is true.

isElement( )

angular.isElement( ) tests a value to see if it is a Dom element, returns true if the value is a Dom element.

angular.isElement(document.querySelector( )) can access a HTML element, returns true if the value is a HTML element.

Program

```
<!DOCTYPE html>
<html ng-app="">
<head>
<title>Check Element</title>
<script src="js\angular.min.js"></script>
</script>
</head>
<body>
<script>
document.write("<br>");
document.write("Title is a Dom elememt?");
document.write("<br>");
document.write(angular.isElement('title'));
document.write("<br><br>");
document.write("Title is a HTML element?");
document.write("<br>");
var check=angular.isElement(
document.querySelector('title'));
document.write(check);
</script>
</body>
</html>
```

Output:

Title is a Dom element?

false

Title is a HTML element?

true

angular.isElement( ) expects to access a Dom element instead of HTML element. Therefore, angular.isElement('title') returns false.

angular.isElement(document.querySelector('title') can access an html element. Therefore, returns true.

isObject()

angular.isObject() tests a value to see if it is an object, returns true if the value is an object.

Program

```
<html>
<script src= "js\angular.min.js"></script>
<body>
<script>
var myCar = "This is my car!";
var carObject = new Object( ) ;
document.write(angular.isObject(myCar) + " ");
document.write(angular.isObject(carObject) + " ");
</script>
</body>
</html>
```

Output:

false true

myCar is a String. Therefore, angular.isObject(myCar) returns false.

carObject is an object. Therefore, angular.isObject(carObject) returns true.

isDefined( )

angular.isDefined( ) tests a value to see if it has been defined, returns true if the value has been defined.

Program

```
<html>
<script src= "js\angular.min.js"></script>
<body>
<script>
var futureCar;
var existingCar = "Very Good!";
document.write(angular.isDefined(futureCar) + " ");
document.write(angular.isDefined(existingCar) + " ");
</script>
</body>
</html>
```

Output:

false true

“futureCar” has not been defined. Therefore, isDefined(futureCar) returns false.

“existingCar” has been defined. Therefore, isDefined(existingCar) returns true.

isUndefined( )

angular.isUndefined( ) tests a value to see if it has not been defined, returns true if the value has not been defined.

Program

```
<html>
<script src= "js\angular.min.js"></script>
<body>
<script>
var futureCar;
var existingCar = "Very Good!";
document.write(angular.isUndefined(futureCar) + " ");
document.write(angular.isUndefined(existingCar) + " ");
</script>
</body>
</html>
```

Output:

true false

“futureCar” has not been defined. Therefore, isUndefined(futureCar) returns true.

“existingCar” has been defined. Therefore, isUndefined(existingCar) returns false.