AngularJS Programming

For Beginners
Learn Coding Fast!

Ray Yao

AngularJS Programming

For Beginners
Learn Coding Fast!

Ray Yao

AngularJs Programming

For Beginners
Learn Coding Fast!

Ray Yao

About the Author

Certified PHP engineer by Zend, USA

Certified JAVA programmer by Sun, USA

Certified SCWCD developer by Oracle, USA

Certified A+ professional by CompTIA, USA

Certified ASP**.** NET expert by Microsoft, USA

Certified MCP professional by Microsoft, USA

Certified TECHNOLOGY specialist by Microsoft, USA

Certified NETWORK+ professional by CompTIA, USA

Recommended Books

Advanced C++ Programming in 8 Hours

Advanced JAVA Programming in 8 Hours

AngularJS Programming for Beginners

C# Interview & Certification Exam

C# Programming for Beginners

C# Programming. Include 100 Q & A

C++ Interview & Certification Exam

C++ Programming for Beginners

Coding Interview,1000 Q & A

Django Programming in 8 Hours

Go Programming in 8 Hours

HTML CSS Interview & Certification Exam

HTML CSS Programming for Beginners

JAVA Interview & Certification Exam

JAVA Programming for Beginners

JAVA Programming. Include 100 Q & A

JavaScript 50 Useful Programs

JavaScript Interview & Certification Exam

JavaScript Programming for Beginners

JQuery Interview & Certification Exam

JQuery Programming for Beginners

Kotlin Programming in 8 Hours

Linux Command Line

Linux Interview & Certification Exam

Linux Shell Programming in 8 Hours

MySQL Programming for Beginners

Node . js Programming in 8 Hours

PHP Interview & Certification Exam

PHP MySQL Programming for Beginners

PowerShell Programming in 8 Hours

Python Interview & Certification Exam

Python Programming for Beginners

Python Programming. Include 100 Q & A

R Programming for Beginners

Ruby Programming for Beginners

Rust Programming in 8 Hours

Scala Programming in 8 Hours

Visual Basic Interview & Certification Exam

Visual Basic Programming for Beginners

Preface

This book covers all essential AngularJS knowledge**.** You can learn complete primary skills of AngularJS fast and easily**.**
The book includes more than 80 practical examples for beginners and includes tests & answers for college exam, engineer certification exam and job interview exam**.**

Source Code for Download

This book provides source code for download; you can download the source code for better study, or copy the source code to your favorite editor to test the programs**.**

Source Code Download Link:

https://forms . aweber . com/form/31/1924478131 . htm

Note:

This book is only for AngularJs beginners, it is not suitable for experienced programmers**.**

Table of Contents

Chapter 1
Introduction to AngularJS

What is AngularJS?

The AngularJS is a framework of JavaScript**.** It can use HTML as a template language and can extend HTML's sentence structure to state an application's components plainly and briefly**.**

The syntax of AngularJS looks like this:

```
<div ng-app=" ">
……
</div>
```

We know that **div** is an html tag, but **ng-app** is a directive of Angular JS, which is used in div tag like an attribute**.** (We will discuss about directive latter)

Prerequisite to learn AngularJS

Before learning the Angular JS, you should have basic knowledge of HTML and JAVASCRIPT, because AngularJS works with HTML and JavaScript**.** You can review HTML, CSS and JavaScript in Appendix1 or Appendix 2**.**

(Note**:** The appendix of this book includes “HTML, CSS and JavaScript Review”)

Download Angular JS

Downloading the AngularJS is not a big deal; you just go to AngularJS site https://AngularJS **.** org and click on “Download” button**.** After downloading the AngularJS file “angular**.** min**.** js”, you can put it on **myFolder\js** folder (Namely, put html files and JavaScript files to **myFolder** folder, and put “angular**.** min**.** js” to **myFolder\js** folder), then write the path of this “angular**.** min**.** js” file in the html page`s head section like this:

<html>

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

……

……

……

</html>

In the above example, you can see the path of AngularJS like “<script src="js\ angular**.** min**.** js">”**.** Now AngularJS framework has been added in our application**.** So we can use the AngularJS framework from now on**.**

Benefits of AngularJS

AngularJS has the capability to bind data to HTML.
AngularJS provides Single Page Application (SAP).
Angular JS's View is totally written in HTML, the Controller is written in JavaScript, and Model is written in AngularJS. It is an MVC framework.
You can achieve more functionality with just a few lines of code.

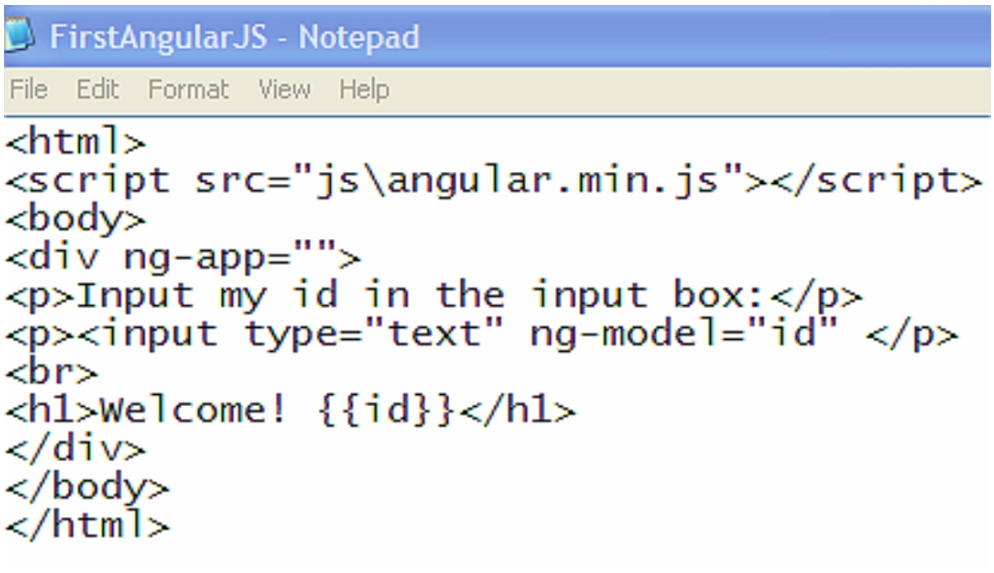
AngularJS provides the reusability for codes and components.

First AngularJS Script

Example 1.1

Open the Notepad, write code as follows. (Figure 1)

```
<html>
<script src="js\angular.min.js"></script>
<body>
<div ng-app="">
<p>Input my id in the input box:</p>
<p><input type="text" ng-model="id" </p>
<br>
<h1>Welcome! {{id}}</h1>
</div>
</body>
</html>
```



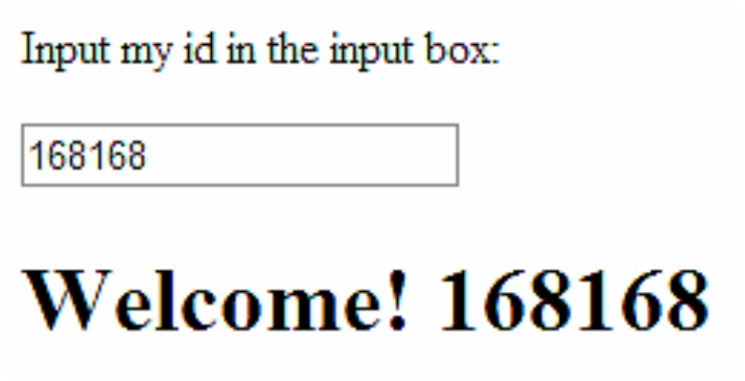
(Figure 1)

Please save the file with name “FirstAngularJS. html” at myFolder. Note: make sure to use **“.html”** extension name.

Double click “FirstAngularJS. html” file, “FirstAngularJS. html” will be run in a browser, and see the output. (Figure 2)

(Assume that I entered id “168168”.)

Output:



(Figure 2)

Explanation:

“<script src="js\angular. min. js"></script>” adds AngularJS framework into the current application.

“ ” indicates you can write AngularJS applications in here.

“ng-model=”id”” binds the inputted data from HTML controls (input, select, and textarea) to application data “id”.

{{id}} is synchronized with the “id” in ng-model="id"

{{id}}: displays the value of “id”.

If you want to edit the codes, right click the file “FirstAngularJS. html” > open with > Notepad.

Hello World!

<html ng-app>…</html>

<html ng-app>… </html> indicates that AngularJS script is used in the tags.

Open the Notepad, write code as follows.

Example 1.2

<html ng-app>

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

<body>

<label>Name: </label>

<input type = "text" ng-model = "yourName" placeholder = "Please enter your name. ">

<h1>Hello {{yourName}}!</h1>

</body>

</html>

Please save the file with name “Hello. html” at myFolder. Note: make sure to use **“.html”** extension name. Double

Click “Hello. html” file, “Hello. html” will be run by a browser, input “would” into the text field, and see the output.

Output:

Explanation:

<html ng-app>…</html> indicates that AngularJS script is used in the tags.

"<script src="js\angular. min. js"></script>" adds AngularJS framework into the current application.

"ng-model = "yourname" " binds the inputted value from the HTML control to "Username".

"placeholder = …" specifies a short hint that describes the expected value of an input text.

{{yourname}} is synchronized with the "yourname" in ng-model="yourname"

{{yourname}}: displays the value of "yourname".

data-ng-app

<html data-ng-app>…</html>

<html data-ng-app>…</html> indicates that AngularJS script is used in the **html5** tags**.**

Example 1.3

<!DOCTYPE html>

<html data-ng-app=""> // compatible in html5

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

<body>

<label>Name: </label>

<input type = "text" ng-model = "yourName" placeholder = "Please enter your name**.** ">

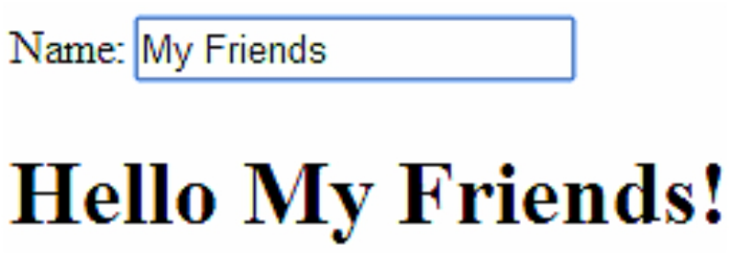
<h1>Hello {{yourName}}!</h1>

</body>

</html>

Please input “My Friends” into the text field**.**

Output:



Explanation:

<html data-ng-app>…</html> indicates that AngularJS script is used in the **html5** tags**.**

“<script src="js\angular**.** min**.** js"></script>” adds AngularJS framework into the current application**.**

“ng-model = “yourname” ” binds the inputted value from the HTML control to “Username”**.**

“placeholder = …” specifies a short hint that describes the expected value of an input text**.**

{{yourname}} is synchronized with the “yourname” in ng-model="yourname"

{{yourname}}: displays the value of “yourname”**.**

Summary

In this chapter, we learnt what AngularJS is, how to download the AngularJS and what the benefits of AngularJS are.

“<script src="js\angular. min. js"></script>” adds AngularJS framework into the current application so that we can use AngularJS framework now.

<html ng-app>…</html> indicates that AngularJS script is used in the tags.

<html data-ng-app>…</html> indicates that AngularJS script is used in the **html5** tags.

If you want to edit the codes, please right click the file “FirstAngularJS. html” > open with > Notepad.

Exercises

Show what you inputted

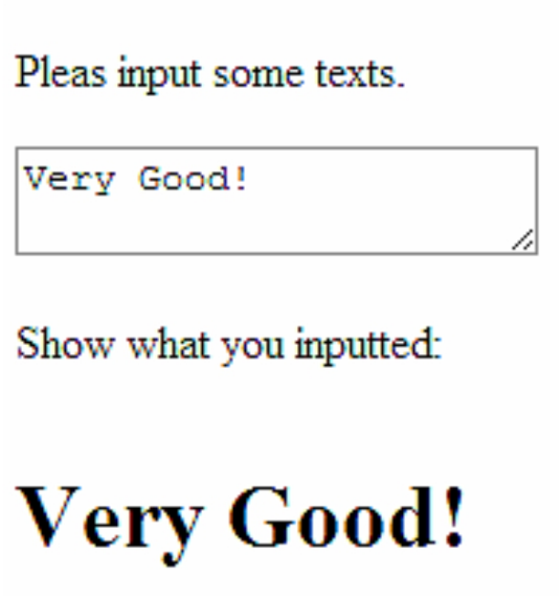
Open Notepad, write AngularJS codes:

```
<!DOCTYPE html>
<html>
<script src="js\angular. min. js"></script>
<body>
<div ng-app="">
<p>Pleas input some texts.</p>
<textarea ng-model="texts"></textarea>
<p>Show what you inputted :< /p>
<H1><p ng-bind="texts"></p><H1>
</div>
</body>
</html>
```

Please save the file with name “ShowInputted. html” at myFolder. Note: make sure to use **“.html”** extension name.

Double click “ShowInputted. html” file, “ShowInputted. html” will be run in a browser, please input some texts in the text area, and see the output.

Output:



Explanation:

“<script src="js\angular. min. js"></script>” adds AngularJS framework into the current application.

“ ” indicates you can write AngularJS applications in here.

“ng-model="texts"” binds the inputted data from HTML controls (textarea) to application data “texts”.

“ ” can “output” the value of “texts” in specified HTML element .

Chapter 2
Directives

The directives of AngularJS

The AngularJS allows us to extend HTML in a very simple way using attributes**.** The attributes are basically directives**.** There are different types of directives which can play different roles in the Application**.** They are App Directive, Model Directive, Bind Directive, Init Directive, and Repeat Directive**.** Let`s discuss one by one in detail**.**

App Directive

ng-app= “ ”

The app directive defines the area of AngularJS application**.** The syntax of app directive is **ng-app = “ ”** ; In here, **ng** is the namespace of AngularJS and **app** is the application area of Angular JS**.**

Example 2.1

```
<!DOCTYPE html>
<html >
<head>
    <title>AngularJS for beginners</title>
    <script src="js\angular. min. js"></script>
</head>
<body>
    <div ng-app=" ">
        The AngularJSapplication has been started.
    </div>
</body>
</html>
```

Output:

The AngularJSapplication has been started**.**

Explanation:

In the head section (**<script src = "js\angular.min.js"> </script>**), the framework of AngularJSis loaded, which means that we can use AngularJS framework now**.**

In the body section (**),** AngularJS application can be written in here**.**

In last tag (** **), the AngularJSapplication is ended**.**

Model Directive

ng-model = “data”

The model directive is used to bind the inputted value from HTML controls (input, checkbox and select etc**.**) to application data**.** The **ng-model = “data”** is the syntax of model directive**.** Let`s take an example for better understanding**.**

Example 2.2

```
<!DOCTYPE html>
<html >
<head>
    <title>AngularJSfor beginners</title>
    <script src="js\angular. min. js"></script>
</head>
<body>
<div ng-app="">
<p>User Name: <br>
<input type="text" ng-model = "Username"></p>
</div>
</body>
</html>
```

Output:

Username:

Explanation:

“ng-model = “Username”” binds the inputted value from the HTML control to “Username”**.**

Now you cannot see the result because this program is missing the code ** .**

Bind Directive

```
<p>ng-bind = “data”</p>
```

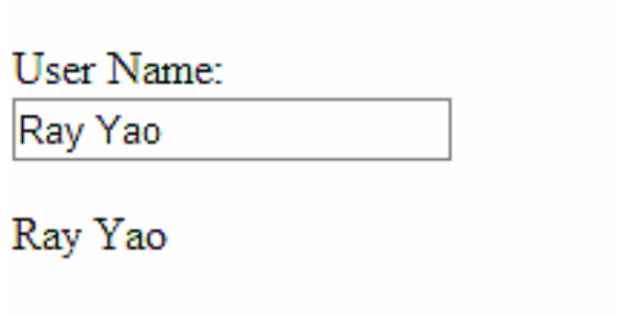
The bind directive is used to bind the data value of an html element ; the syntax of bind directive is ** ng-bind = “data” .** Let`s take an example for better understanding**.**

Example 2.3

```
<!DOCTYPE html>
<html >
<head>
    <title>AngularJSfor beginners</title>
    <script src="js\angular. min. js"></script>
</head>
<body>
<div ng-app="">
<p>User Name: <br>
<input type="text" ng-model = "Username"></p>
<p ng-bind="Username"></p>
</div>
</body>
</html>
```

Open the notepad and paste the above mentioned code with a **.** html extension, and type username “Ray Yao” in the input box**.**

Output:



Explanation:

“ ng-bind="Username" ” binds the value of “Username” to tag, and “shows” its value**.**

In the above example, (**)** can update the value of “Username” and writes it to tag**.**

ng-bind="data" can “output” the value of “data” in specified HTML element **.**

ng-model vs ng-bind

They both use “bind”**.**

(1)

ng-model = “data” : binds the inputted value from Html controls (textarea, checkbox, select, input and radio etc**.**) to “data”**.**

For example:

“ng-model = “Username”” binds the inputted value to “Username”**.**

(Just like to assign the inputted value to “Username”**.**)

(2)

** ng-bind= “data” :** binds the “data” to an html element **.**

For example:

 binds the data “Username” to element and shows its value**.**

(Just like to display the value of the “Username” in **.**)

Init Directive

ng-init = "data = ‘value’”

The init directive is used to initialize the data with a value**.** The syntax of init directive is **ng-init = "data = ‘value’”.** Let`s take an example for better understanding**.**

Example 2.4

```
<!DOCTYPE html>
<html >
<head>
    <title>AngularJSfor beginners</title>
    <script src="js\angular. min. js"></script>
</head>
<body>
<div ng-app=""   ng-init="Username= 'Andy Smith' ">
   <p>User Name: <input type="text" ng-model = "Username"></p>
   <p ng-bind="Username"></p>
</div>
</body>
</html>
```

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

Output:

Explanation:

ng-init="Username= ‘Andy Smith’” initializes “Username” with a value “Andy Smith”**.**

In the above example, (< div ng-app="" **ng-init = "Username= 'Andy Smith'** ">) initializes the value of User Name**.** When the page is loaded completely, the value of User Name “**Andy Smith** ” is displayed**.**

Repeat Directive

ng-repeat = “variable in array”

The repeat directive works like a loop**.** The **ng-repeat** directive repeats to get the value of an array**.**

Example 2.5

```
<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>
<div ng-app=""  ng-init = "ColorName = ['Pink', 'Red', 'Green', 'Blue',
'Black', 'White', 'Yellow', 'Gray']">
<p style="color:green; font-weight:bold">Colours Name:</p>
<ol>
     <li ng-repeat="x in ColorName">
         <p ng-bind="x"></p>
     </li>
</ol>
</div>
</body>
</html>
```

Output:

Colours Name:

1. Pink
2. Red
3. Green
4. Blue
5. Black
6. White
7. Yellow
8. Gray

Explanation:

ng-repeat = "x in ColorName" repeats to get the value of the array "ColorName", and assigns all array values to "x" variable.

 repeats to output the value of "x".

Valid Directive

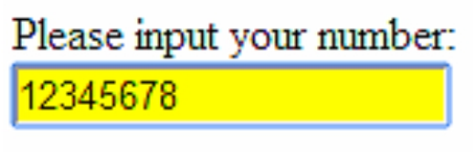
input. ng-valid

ng-valid directive can change the status of a specified element when an input is valid.

Example 2.6

```
<!DOCTYPE html>
<html ng-app="">
<head>
<script src="js\angular. min. js"></script>
<meta charset="utf-8">
<style>
input.ng-valid {    // check number if valid
background-color: yellow;
}
</style>
</head>
<body>
<form>
Please input your number:<br>
<input text="mynumber" ng-model="psw" required>
</form>
</body>
</html>
```

Output:



Explanation:

“input**.** ng-valid” change the input element status when the input is valid**.**

<input text="mynumber" ng-model="psw" required> is used to input numbers**.** ng-model= “psw” need a property of “required”, otherwise this element cannot change status**.**

When you input some valid numbers to a text field, the text field background becomes yellow**.**

Check Email Address

$error. email

“$error. email” is used to check email address validity; and return an error message if wrong email address format is being inputted.

Example 2.7

```
<!DOCTYPE html>
<html ng-app="">
<head>
<script src="js\angular. min. js"></script>
<meta charset="utf-8">
</head>
<body>
<form name="iForm">
Please input your email address: <br><br>
<input type="email" name="eMailAddr" ng-model="text"> <br><br>
<div ng-show="iForm. eMailAddr.$error.email "> Invalid email address!
</div>
</form>
```

Output:

Please input your email address:

123456#yahoo.com

Invalid email address!

Explanation:

“$error**.** email” is used to check email address validity; and return an error message if wrong email address format is being inputted**.**

When you input a wrong format email address, you will see an alert: Invalid email address!

Summary

In this chapter, we learnt ng-app directive, ng-model directive, ng-bind directive and ng-repeat directive and their functionalities with proper examples and outputs.

ng-app directive indicates you can write AngularJS application codes in here.

ng-init directive initializes application data.

ng-model directive binds the inputted value from HTML controls (input, select, and textarea) to an application data.

ng-bind directive binds an application data to a specified HTML element and outputs the value of the data.

ng-repeat directive repeats to get the values of an array.

ng-valid directive can change the status of a specified element when an input is valid.

“$error. email” is used to check email address validity; and return an error message if wrong email address format is being inputted.

Exercises

Repeat Output

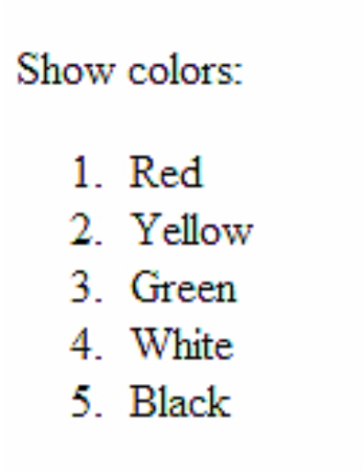
Open Notepad, write AngularJS codes:

```
<!DOCTYPE html>
<html>
<script src="js\angular. min. js"></script>
<body>
<div data-ng-app=""
data-ng-init="colors=
['Red','Yellow','Green','White','Black']">
  <p>Show colors:</p>
  <ol type = "1">
    <li data-ng-repeat="show in colors">
      {{ show }}
    </li>
  </ol>
</div>
</body>
</html>
```

Please save the file with name “RepeatOutput. html” at myFolder. Note: make sure to use **“.html”** extension name.

Double click “RepeatOutput. html” file, “RepeatOutput. html” will be run in a browser, and see the output.

Output:



Explanation:

“data-ng-app” is almost the same as “ng-app”, but “data-ng-app” is more suitable for HTML5**.**

“ ” indicates you can write AngularJS application in here**.**

“data-ng-init” is almost the same as “ng-init”, but “data-ng-init” is more suitable for HTML5**.**

“data-ng-init="colors=…” initializes array “colors” with some values **.**

“data-ng-repeat = "show in colors” repeats to get the value of array “colors”, and assigns all array values to “show” variable**.**

{{ show }} displays the value of “show”**.**

Chapter 3
Filters

What is the Filter?

A 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 filters one by one**.**

Uppercase Filter

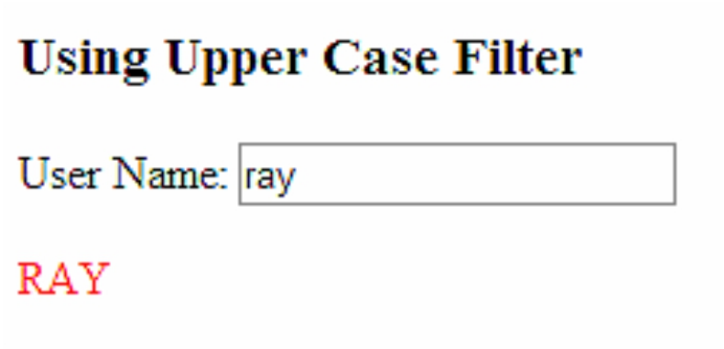
Value | uppercase

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 uppercase filter.

Example 3.1

```
<!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:



Explanation:

"Username | uppercase" changes the value of "Username" to uppercase.

In the above example, I set the default value (**ray**) in lower case, but the result becomes upper case (RAY).

Lowercase Filter

Value | lowercase

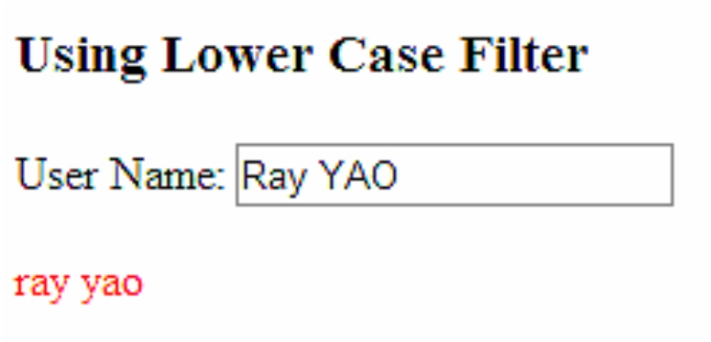
The lowercase filter changes the text to lower case. Suppose a user writes a text in upper case (e. g. RAY YAO) or title case (e. g. Ray Yao) or in mixed case (e. g. rAy or RaY or rAY etc.), and you want the lower case result, then you will have to use a lower case filter.

Example 3.2

```
<html >
<head>
    <title>AngularJSfor beginners</title>
    <script src="js\angular. min. js"></script>
</head>
<body>
<h3>Using Lower Case Filter</h3>
<div ng-app=""  ng-init="Username= 'Ray YAO' ">
<p>User Name: <input type="text" ng-model="Username"></p>
<p style="color:red" ng-bind="Username | lowercase"></p>
</div>
</body>
</html>
```

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

Output:



Explanation:

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

In the above example, when I enter text (**Ray YAO**) in upper case, but the result becomes lower case (ray you)**.**

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
```

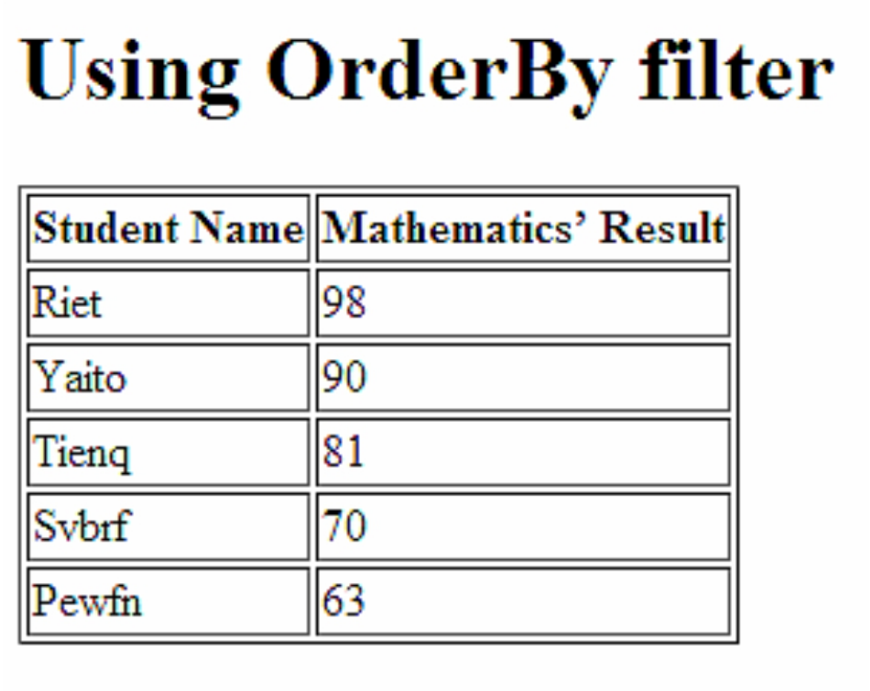
Let`s take an example for better understanding**.**

Example 3.3

```
<!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:



Explanation:

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 to reverse the order, you can remove the **“-“** sign”**.**

Example 3.4

```
<!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
Pewfn	63
Svbrf	70
Tienq	81
Yaito	90
Riet	98

Explanation:

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

Currency Filter

Value | currency

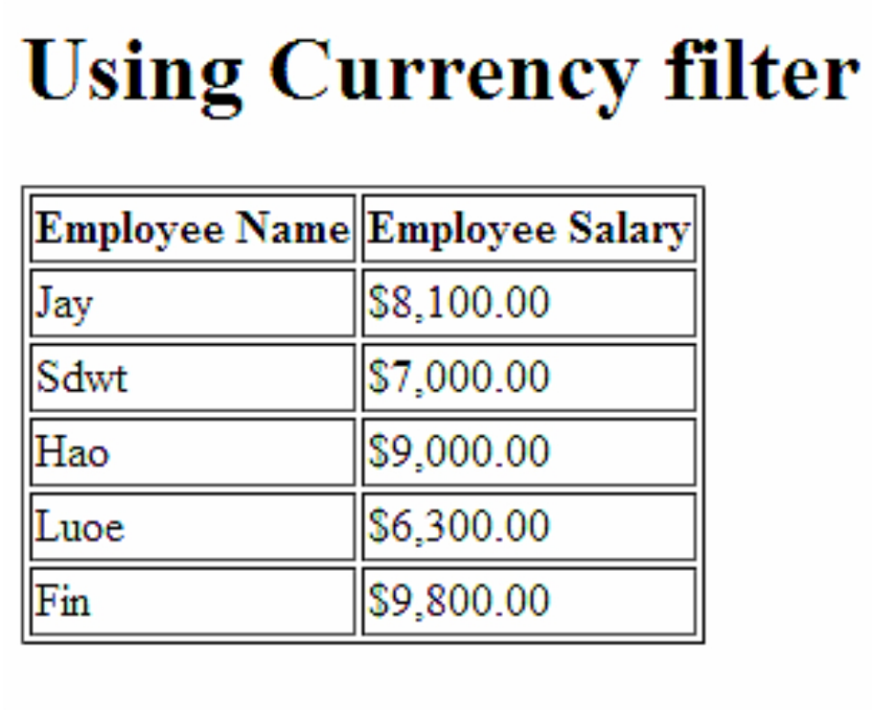
The currency filter is used to display the result in currency format.

Example 3.5

```
<!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:



Explanation:

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

In the above example, 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**.**

Example 3.6

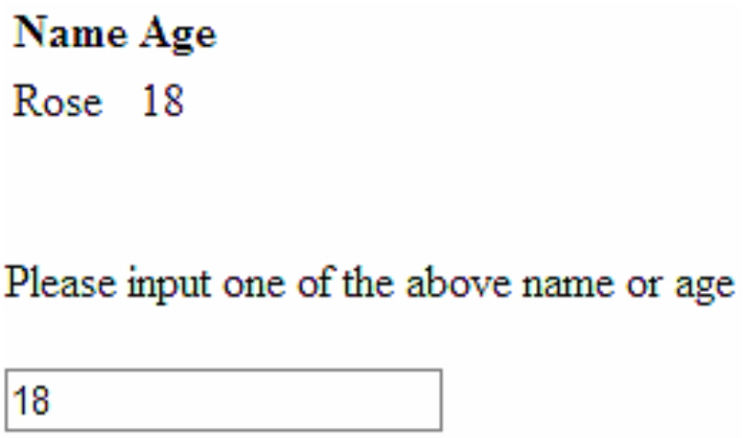
```
<!DOCTYPE html>
<html ng-app="">
<head>
<script src="js\angular. min. js"></script>
<meta charset="utf-8">
</head>
<body>
<div ng-init="students =        // define an array “students”
[{name:'Andy', age:'19'},
{name:'Rose', age:'18'},
{name:'Jony', age:'17'},
{name:'Judy', age:'16'},
{name:'Tomy', 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:



Explanation:

"<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 a text field, the output shows "Rose 18".

Summary

In this chapter, we learnt what a filter is. And different types of filter like Upper Case filter, Lower Case filter, Order By filter and Currency filter. I have tried to explain each topic of this chapter with proper example, output and explanation.

“Value | uppercase” changes the value to uppercase.

“Value | lowercase” changes the value to lower case.

Value | orderBy: ’value’ shows the value in ascending order

Value | orderBy: ’- value’ shows the value in descending order

“Value | currency” converts the value to currency format.

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

Exercises

Uppercase Filter

Open Notepad, write AngularJS codes:

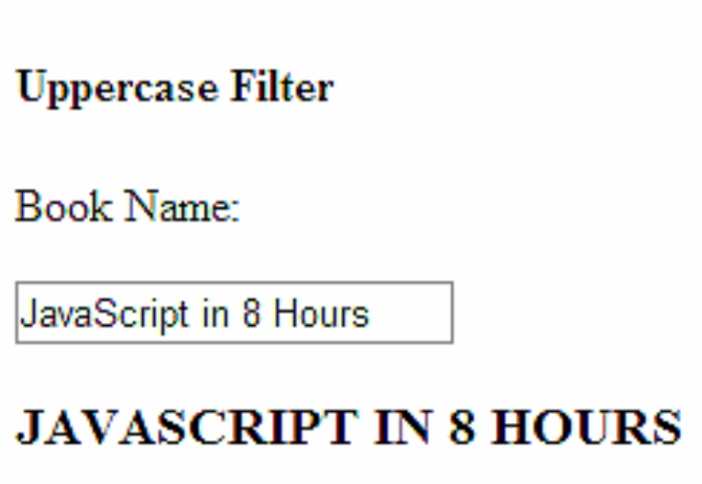
```
<!DOCTYPE html>
<script src="js\angular. min. js"></script>
<body>
<h4>Uppercase Filter</h4>
<div ng-app="">
<p>Book Name: </p>
<p><input type="text" ng-model = "Book"></p>
<H3><p ng-bind="Book|uppercase"></p><H3>
</div>
</body>
</html>
```

Please save the file with name “UppercaseFilter. html” at myFolder.

Note: make sure to use **“.html”** extension name.

Double click “UppercaseFilter. html” file, “UppercaseFilter. html” will be run by a browser, input the book name “JavaScript in 8 Hours” in the text box, and see the output.

Output:



Explanation:

"** ** " changes the value of Book to upper case**.**

Chapter 4
Directive in DOM

Show Directive

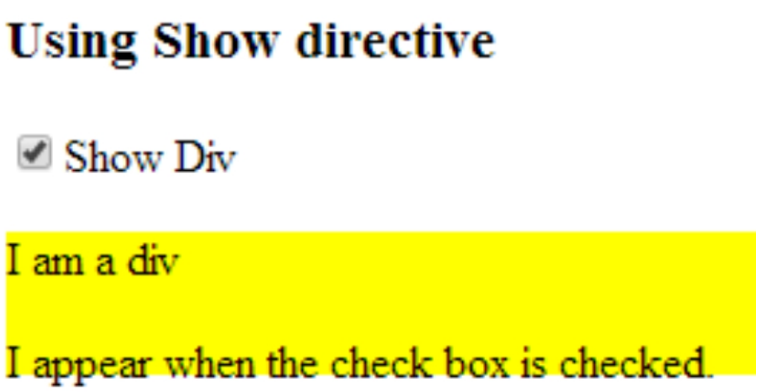
```
<p ng-show="true"> visible. </p>
<p ng-show="false">invisible. </p>
```

Show directive is used to show or hide the text in <p> elements of HTML.
if ng-show is true, the text “visible” will be shown.
if ng-show is false, the text “invisible” will be hidden.

Example 4.1

```
<!DOCTYPE html>
<html >
<head>
    <title>AngularJS for beginners</title>
    <script src="js/angular. min. js"></script>
</head>
<body>
<h3>Using Show directive</h3>
<div ng-app="" >
<input type="checkbox" ng-model="showHideDiv">Show Div
<div ng-show="showHideDiv" style="background-color:yellow;height:50px;">
<p>I am a div</p>
<p>I appear when the check box is checked.</p>
</div>
</div>
</body>
</html>
```

Output:



Explanation:

“ng-model="showHideDiv” binds the check box value with “showHideDiv”**.**
“ng-show="showHideDiv"” shows the text when the value of “showHideDiv” is true**.**

In the above example, at first time when the page is loaded, the div (yellow area of the page) is not shown, but when I check the **check box** (show Div), the value of “showHideDiv” is true; the div (yellow area of the page) is appeared**.**

Hide Directive

```
<p ng-hide="true"> invisible. </p>
<p ng-hide="false">visible. </p>
```

Hide directive is used to hide or show the text in <p> elements of HTML.

if ng-hide is true, the text “invisible” will be hidden.

if ng-hide is false, the text “visible” will be shown.

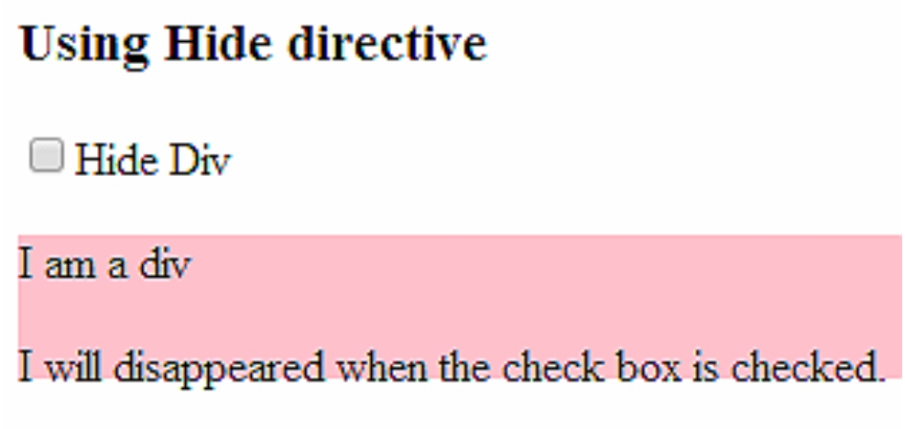
Example 4.2

```
<!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 Hide directive</h3>
<div ng-app="" >
<input type="checkbox" ng-model="HideShowDiv">Hide Div
<div ng-hide="HideShowDiv" style="background-
color:pink;height:50px;">
<p>I am a div</p>
<p>I will disappear when the check box is checked. </p>
</div>
</div>
```

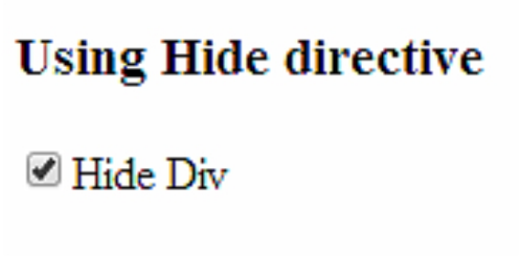
```
</body>
</html>
```

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

Output:



Please chick the Check Box, and see the result**.**



Explanation:

“ng-model="showHideDiv” binds the check box value with “showHideDiv”**.**
“ng-hide="showHideDiv"” hides the text when the value of “showHideDiv” is true**.**

In the above example, at the first time when the page is loaded, the div (pink area of the page) is shown**.**

But when I click **check box** (Hide Div), the value of “showHideDiv” is true; the div (pink area of the page) disappeared**.**

Disable Directive

```
<input type="button" ng-disabled=true value="myButton”>
<input type="button" ng-disabled=false  value="myButton”>
```

Disable directive is used to disable or enable the element of HTML.

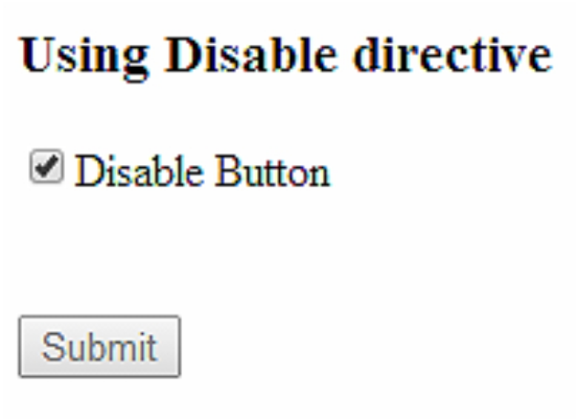
if ng-disable is true, myButton will be disabled.

if ng-disable is false, myButton will be enabled.

Example 4.3

```
<!DOCTYPE html>
<html >
<head>
    <title>AngularJS for beginners</title>
    <script src="js/angular. min. js"></script>
</head>
<body>
<h3>Using Disable directive</h3>
<div ng-app="" >
<input type="checkbox" ng-model="DisableEnable">Disable Button<br><br><br>
<input type="button" ng-disabled="DisableEnable" value="Submit">
</div>
</body>
</html>
```

Output:



Explanation:

“ng-model="DisableEnable"” binds the checkbox value with “DisableEnable”**.**

<input type="button" **ng-disabled** ="DisableEnable" value="Submit"> disables the “submit” button when “DisableEnable” is true**.**

In the above example, at the first time when the page is loaded, the **button** (Submit) is enabled, but when I click **check box,** which makes the “DisableEnable” true, the **button** (Submit) is disabled**.**

Click Directive

ng-click = “expression”

The click directive is used to define an event, and evaluates the expression.

Example 4.4

```
<!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 Click directive</h3>
<div ng-app=""  ng-init="increaseNumber=0">
<button ng-click="increaseNumber = increaseNumber + 1">Click Me!
</button>
<p ng-bind="increaseNumber" style="color:blue"></p>
</div>
</body>
</html>
```

Output:

Explanation:

“ng-click="increaseNumber = increaseNumber + 1"” updates the value “increaseNumber” when click the button**.**

“ ”: binds the “increaseNumber” value with and displays its value**.**

In the above example, at the first time when the page is loaded, the count shows **0,** it means that I do not click on the button (**Click Me!),** but when I click the button, it increases one by one, which signifies the number of clicks**.**

If Directive

ng-if="result"

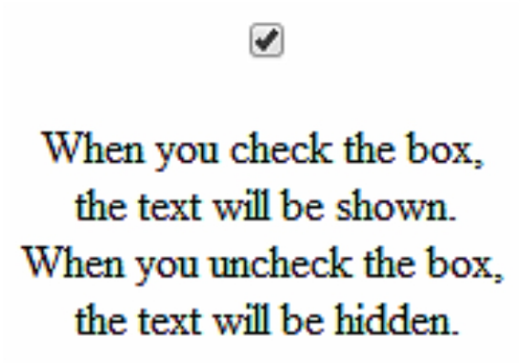
“ng-if directive” is used to add or remove an element in Dom.
If the result of “ng-if” is true, an element will be added.
If the result of “ng-if” is false, an element will be removed.

Example 4.5

```
<!DOCTYPE html>
<html>
<head>
<script src="js\angular. min. js"></script>
<meta charset="utf-8">
</head>
<body ng-app="" align = "center"><br><br>
<input type="checkbox" ng-init="result = true" ng-model="result" >
<div ng-if="result" >       // check true or false
<p>When you check the box, <br>
the text will be shown. <br>
When you uncheck the box, <br>
the text will be hidden. </p>
</div>
</body>
</html>
```

Please check or uncheck the box.

Output:



Explanation :

ng-if="result" checks the result if true or false**.**

When the result of “ng-if” is true, an element will be added**.**

When the result of “ng-if” is false, an element will be removed**.**

Summary

“ng-show="showHideDiv"” shows the text when the value of “showHideDiv” is true.

“ng-hide="showHideDiv"” hides the text when the value of “showHideDiv” is true.

“ng-disabled="DisableEnable"” disables the element of HTML when “DisableEnable” is true.

ng-click = “expression or function( )” updates the value in express or run the function( ) when clicking the button.

“ng-if directive” is used to add or remove an element in Dom.

Exercises

Click Event

Open Notepad, write AngularJS codes:

<!DOCTYPE html>

<html >

<head>

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

</head>

<body>

Click Directive

** **

<button ng-click="result=result+100">

Click Here!</button>

<H1> </H1>

** **

</body>

</html>

Please save the file with name “ClickEvent**.** html” at myFolder**.** Note**:** make sure to use **“.html”** extension name**.**

Double click “ClickEvent**.** html” file, “ClickEvent**.** html” will be run in a browser, click the button “Click Here!”, and see the output**.**

Output:

Explanation:

“**ng-click="result=result+100"** ” evaluates the expression when clicking the button**.**

Chapter 5
Events

Event

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

Click event

ng-click = “expression”

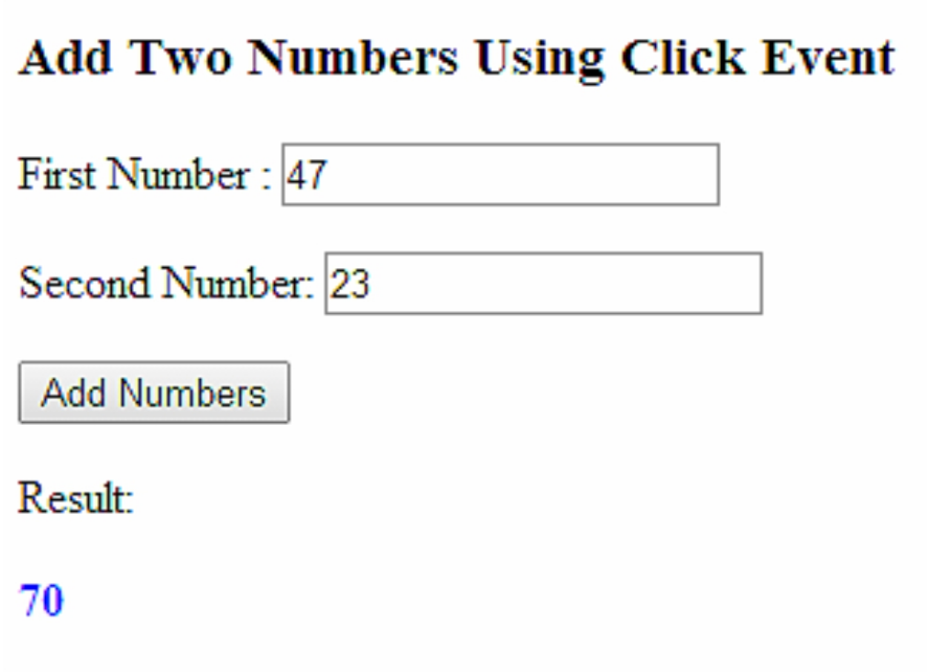
ng-click = “expression” defines a click event**.** When a button is clicked, an event occurs, and evaluates the expression**.** The click event normally works on button**.**

Example 5.1

```
<!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>
<div ng-app=""  ng-init="firstNumber=47; secondNumber=23">
<p>First Number :  <input type="number" ng-model =
"firstNumber"></p>
<p>Second Number: <input type="number" ng-model =
"secondNumber"></p>
<button ng-click="Result=firstNumber + secondNumber"> Add
Numbers </button>
<p>Result:<p style="font-weight:bold; color:blue" ng-bind =
"Result"> </p></p>
</div>
</body>
</html>
```

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

Output:



Explanation:

“<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 example, 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**.** (Note: these numbers are initialized in ng-init directive, but you can enter your own numbers in both text boxes)**.** When I click on the button (Add Numbers), the result displaying in Result area is **70 .**

Double Click event

ng-dblclick = “expression”

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

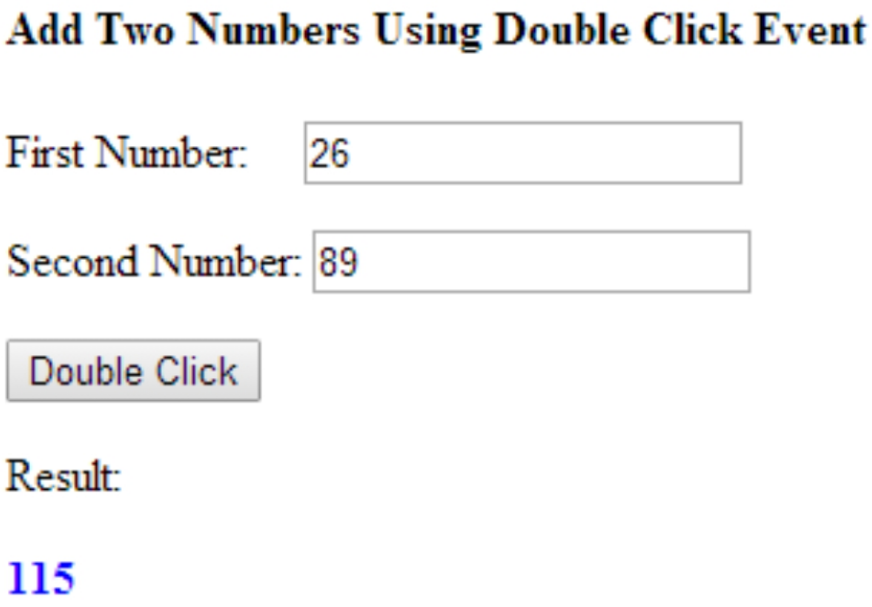
A double click event normally works on button**.**

Example 5.2

```
<!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>
<div ng-app=""  ng-init="firstNumber=26;SecondNumber=89">
<p>First Number:      <input type="number" ng-model="firstNumber"></p>
<p>Second Number: <input type="number" ng-model="SecondNumber"></p>
<button ng-dblclick="Result=firstNumber + SecondNumber">Double Click</button>
<p>Result:<p style="font-weight:bold;color:blue" ng-bind="Result"></p>
</div>
</body>
</html>
```

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

Output:



Explanation:

“<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 example, 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**.**

The mouse move event normally works on div, body and specific area or element**.**

Example 5.3

```
<!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:

Here is a textarea.

count: 20

Explanation:

“ng-mousemove="count = count + 1"”: when the 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 hovers over, an event occurs, and evaluates the expression.

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

Example 5.4

```
<!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:

Here is a textarea.

count: 2

Explanation:

“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**.**

Mouse Leave event

```
ng-mouseleave = “expression”;
```

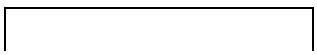
ng-mouseleave = “expression” defines a mouse leave event**.** When the mouse leaves a specified element, an event occurs, and evaluates the expression**.**

Example 5.5

```
<!doctype html>
<html>
<head>
<script src="js\angular. min. js">
</script>
</head>
<body ng-app="">
<br><br>
<textarea ng-mouseleave="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 and leave the textarea for 10 times**.**)**.**

Output:



Here is a textarea.

count: 10

Explanation:

“ng-mouseleave="count = count + 1"”: when the mouse leaves the texture, “count” increases 1.

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

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

“count : 10” means that mouse leaves ten 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 the text box and text area**.**

Example 5.6

```
<!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>
<br><br>
<h2>count: {{count}}</h2>
<body>
</html>
```

(Assume that you type 12345678 in the textarea**.**)

Output:

Here is a textarea.
12345678

count: 8

Explanation:

“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-keydown = “expression” defines a key down event**.** When the key is down in a specified element, an event occurs, and evaluates the expression**.**

The key down event normally works on the text box and text area**..**

Example 5.7

```
<!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:

Here is a textarea.
123456

count: 6

Explanation:

“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.

Copy & Cut event

ng-copy= "result1=true" ng-cut= "result2=true"

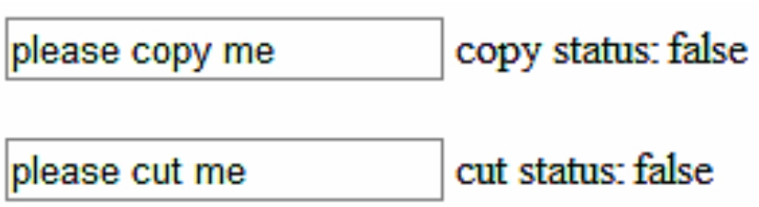
When copy a specified text, the result 1 will be true.

When cut a specified text, the result 2 will be true.

Example 5.8

```
<!doctype html>
<html ng-app>
<head>
<script src="js\angular. min. js"></script>
<meta charset="utf-8">
</head>
<body>
<br>
<input ng-copy= "result1=true"  ng-init= "result1=false;
value1='please copy me'"  ng-model= "value1" >
copy status: {{result1}}</br><br>
<input ng-cut= "result2=true"  ng-init= "result2=false;
value2='please cut me'"  ng-model= "value2" >
cut status: {{result2}}
</body>
</html>
```

Output:



Explanation:

ng-copy= "result1=true" : When copy a specified text, the result 1 will be true**.**

ng-cut= "result2=true" : When cut a specified text, the result 2 will be true**.**

{{result1}} will show true or false**.**

{{result2}} will show true or false**.**

Summary

In this chapter, we learnt different events like Click event, Double Click event, Mouse Move event, Mouse Over event, Mouse Leave event, Key Up event and Key Down event.

“ng-click” defines an AngularJS click event.

“ng-dblclick” defines an AngularJS double click event.

“ng-mousemove” defines an AngularJS mouse move event.

“ng-mouseover” defines an AngularJS mouse over event.

“ng-mouseleave” defines an AngularJS mouse leave event.

“ng-keyup” defines an AngularJS key up event.

“ng-keydown” defines an AngularJS key down event.

ng-copy= "result1=true" : When copy a specified text, the result 1 will be true.

ng-cut= "result2=true" : When cut a specified text, the result 2 will be true.

Exercises

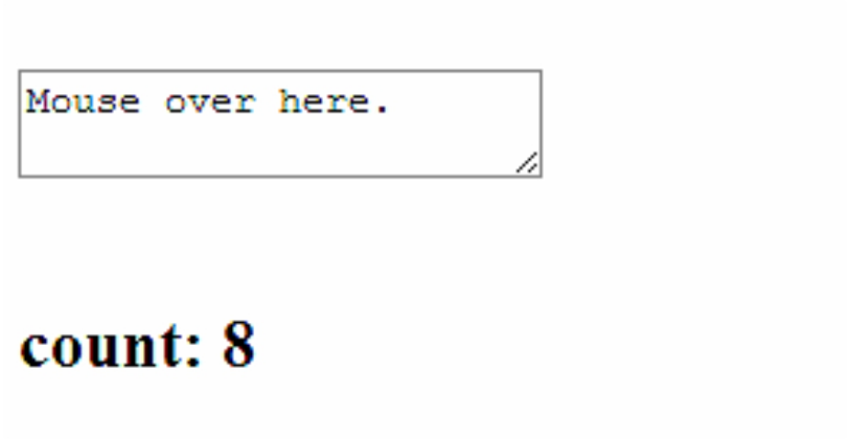
Mouse Over Event

Open Notepad, write AngularJS codes:

```
<!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">
Mouse over here.
</textarea>
<br><br>
<h2>count: {{count}}</h2>
</body>
</html>
```

Please save the file with name “MouseOverEvent. html” at myFolder, run the “MouseOverEvent. html” file, put the mouse over the textarea for 8 times, and see the output.

Output:



Explanation:

“**ng-mouseover="count = count + 1"** ” evaluates the expression when mouse moves over the text area**.**

Chapter 6
Expression

{{ Expression }}

{{Expression}} is used to bind the value with html element and displays the value**.** It works same as the **ng-bind** directive**.** {{Expression}} is written within two curly brackets**.** The {{expression}} is basically a pure JavaScript expression**.** There are different kinds of expression, let`s talk about it one by one**.**

String Expression

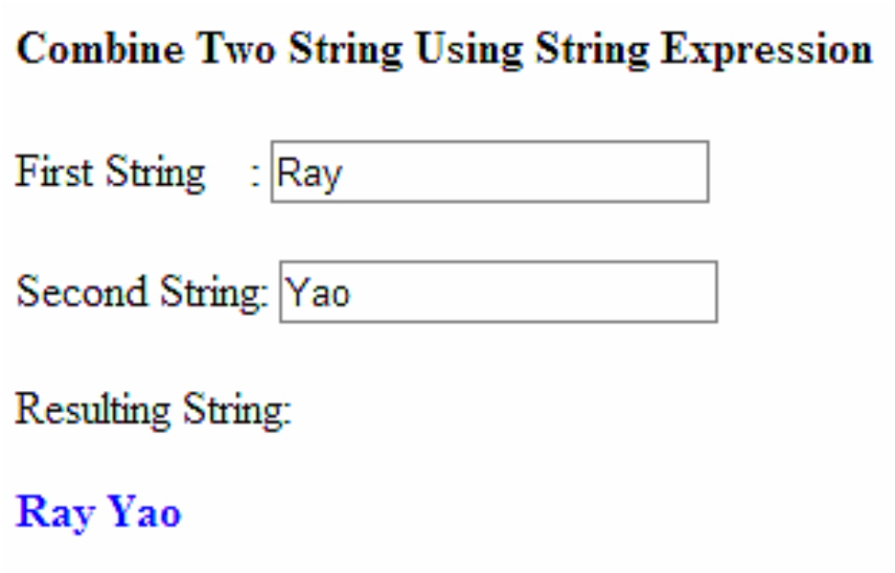
We know that string is a collection of characters. In AngularJSthe string expression looks like this.

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

Example 6.1

```
<!DOCTYPE html>
<html >
<head>
    <script src="js/angular. min. js"></script>
</head>
<body>
<h4>Combine Two String Using String Expression</h4>
<div ng-app="" >
First String    : <input type="text" ng-model="firstString"/><br><br>
Second String: <input ng-model="secondString"/><br><br>
Resulting String:<p style="color:blue;font-weight:bold;">{{firstString +" "+secondString}}</p>
</div>
</body>
</html>
```

Output:



Explanation:

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

{{ expression}} displays the value of expression**.**

In the above example, 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 the string expression {{ }}**.** Note: The plus + sign is used for string concatenation**.**

Number Expression

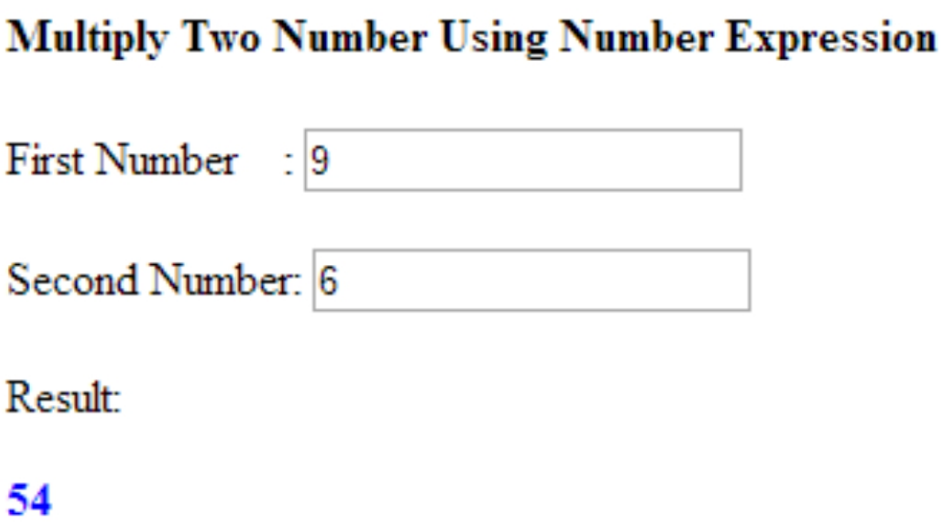
In AngularJS, you can perform the different mathematical operation by using Number Expression**.**

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

Example 6.2

```
<!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>
<div ng-app="" ng-init="firstNumber=9;secondNumber=6">
First Number    : <input type="number" ng-model="firstNumber"/><br><br>
Second Number: <input type="number" ng-model="secondNumber"/><br><br>
Result:<p style="color:blue;font-weight:bold;">{{firstNumber * secondNumber}}</p>
</div>
</body>
</html>
```

Output:



Explanation:

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

{{ expression}} displays the value of expression**.**

In the above example, 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

AngularJS object works like a JavaScript object**.** The syntax looks like this:

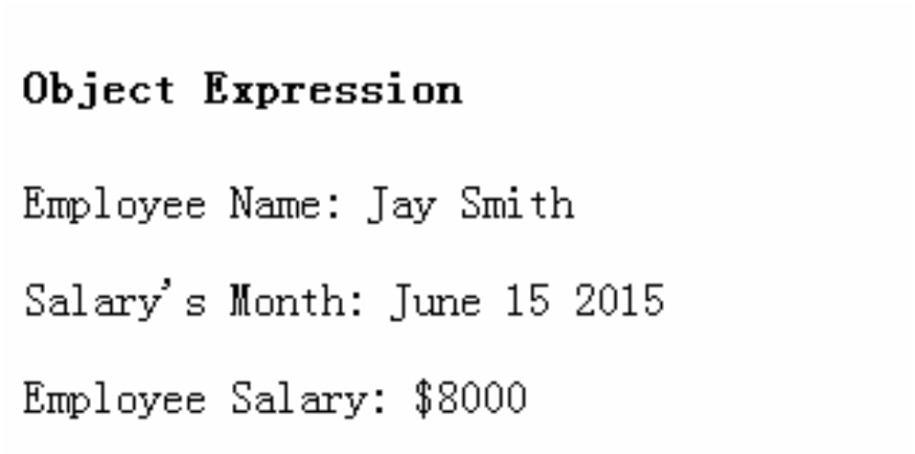
```
object = {property: value}
```

“object = {property: value}” defines an object**.**

Example 6.3

```
<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'}">
<p>Employee Name :  {{EmployeeObject.Emp_name}}</p>
<p>Salary's Month:  {{EmployeeObject.Emp_Month}}</p>
<p>Employee Salary:  {{EmployeeObject.Emp_salary}}</p>
</div>
</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:

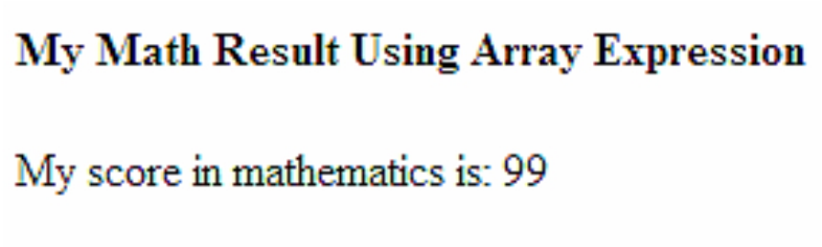
```
Array=[val1, val2, val3,]
```

“Array=[val1, val2, val3,]” defines an array.

Example 6.4

```
<!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]">
<p>My score in mathematics is:  {{MyArray[4]}}</p>
</div>
</body>
</html>
```

Output:



Explanation:

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

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

Using Expression

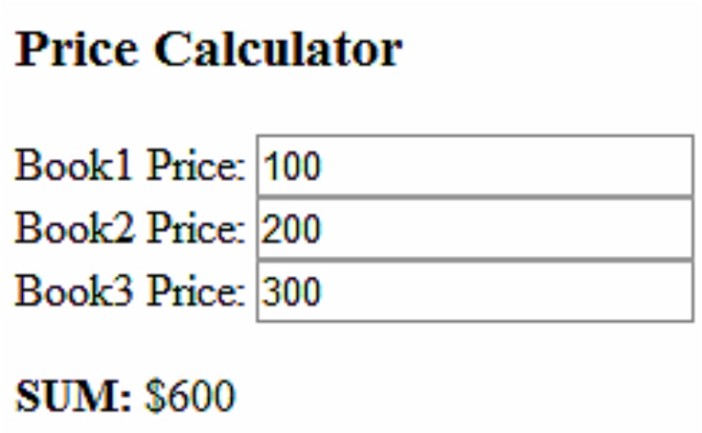
{{expression}}

{{expression}} can calculate the value of the expression, and show the result.

Example 6.5

```
<!DOCTYPE html>
<html data-ng-app="">
<head>
<script src="js\angular. min. js"></script>
<meta charset="utf-8">
</head>
<body>
<h3>Price Calculator</h3>
Book1 Price: <input type="number" ng-model="price1"><br>
Book2 Price: <input type="number" ng-model="price2"><br>
Book3 Price: <input type="number" ng-model="price3"><br>
<p><b>SUM:</b> ${{price1+price2+price3}}</p>
</div>
</body>
</html>
```

Output:



Explanation:

{{expression}} can calculate the value of the expression**.**

${{price1+price2+price3}} can calculate all the value input by the user, and show the result**.**

Summary

In this chapter, we learnt about different expressions like string expression, number expression, object expression and array expression.

{{ expression}} displays the value of expression.

{{First String + Second String}} joins two strings together.

{{First Number + Second Number}} calculates the arithmetic expression.

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

{{ Array [index] }} displays the value of the array.

{{expression}} can calculate the value of the expression, show the result.

Exercises

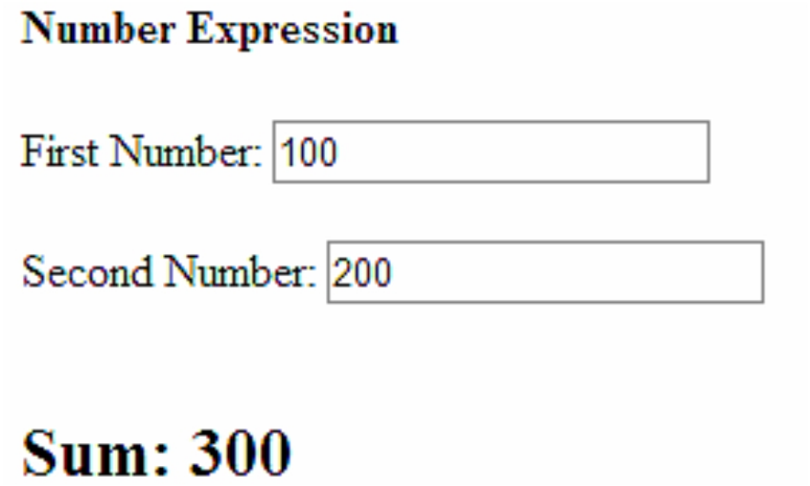
Expression Sample

Open Notepad, write AngularJS codes:

```
<!DOCTYPE html>
<html >
<head>
<title>Expression Sample</title>
<script src="js\angular. min. js"></script>
</head>
<body>
<h4>Number Expression</h4>
<div ng-app="" >
First Number:
<input type="number" ng-model="firstNumber"/>
<br><br>
Second Number:
<input type="number" ng-model="secondNumber"/>
<br><br>
<h2>Sum:  {{firstNumber + secondNumber}} </h2>
</div>
</body>
</html>
```

Please save the file with name “ExpressionSample. html” at myFolder. Double click “ExpressionSample. html” file, and see the output.

Output:



Explanation:

{{firstNumber + secondNumber}} shows the sum of the firstNumber and secondNumber**.**

Chapter 7
Controller & Scope

What is a Controller?

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

How to define 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 the controller is as follows:

```
<div ng-app=""  ng-controller="controllerName">
```

“ ” defines a controller**.**

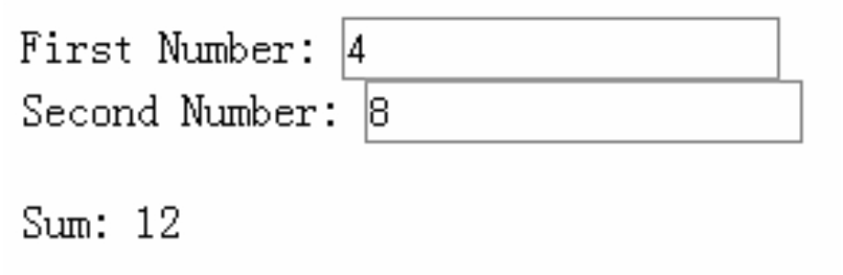
Example 7.1

```
<html>
<script src= "js\angular. min. js"></script>
<body>
<div ng-app="Calculation" ng-controller="myController">

First Number: <input type="number" ng-model="firstNumber"><br>
Second Number: <input type="number" ng-model="secondNumber">
<br>
<br>
Sum: {{firstNumber + secondNumber}}
</div>
<script>
var app = angular.module('Calculation', [ ]);
app.controller('myController', function($scope) {
    $scope.firstNumber = 4;
    $scope.secondNumber = 8;
});
</script>
</body>
```

</html>

Output:



Explanation:

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

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

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

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

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

In the above example, at the first time when the page is loaded, you see there are two text boxes 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. An AngularJS module defines an application, which will be discussed in later chapters in detail.

What is 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 example for understanding**.**

Example 7.2

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

Explanation:

In the above example, 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 are 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 example:

$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 the controller to view.

Module Basic

app = angular. module()

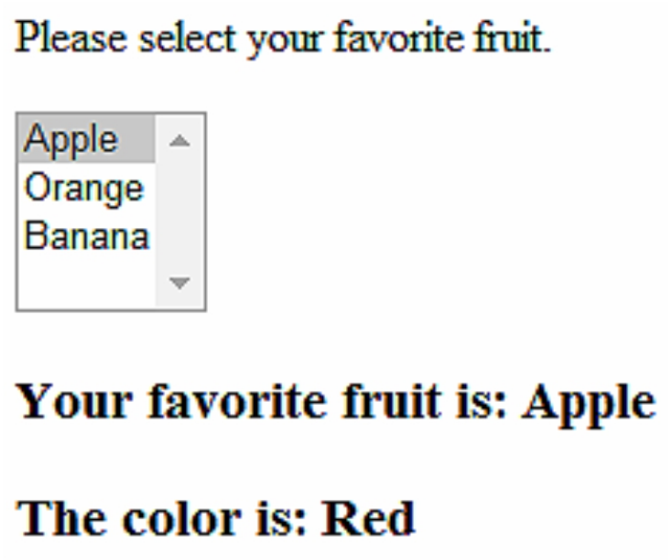
“app = angular. module()” creates a new module.

Example 7.3

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="js\angular. min. js"></script>
</head>
<body>
<div ng-app="myFruit" ng-controller="iController">
<p>Please select your favorite fruit. </p>
<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>
</div>
<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:



Explanation:

 defines a controller “iController”.

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

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

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

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

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

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

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

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

“function($scope)” defines a controller constructor function, and also defines an object $scope. When the 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.

Summary

In this chapter, we learnt what controller is, how to define controller, what MVC and Scope are.

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

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

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

“function ($scope) { }” defines a controller function and an object $scope.

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

  displays data, which is a View.

“ ” defines a Controller, and calls function ($scope) { } when the web page is loaded.

Exercises

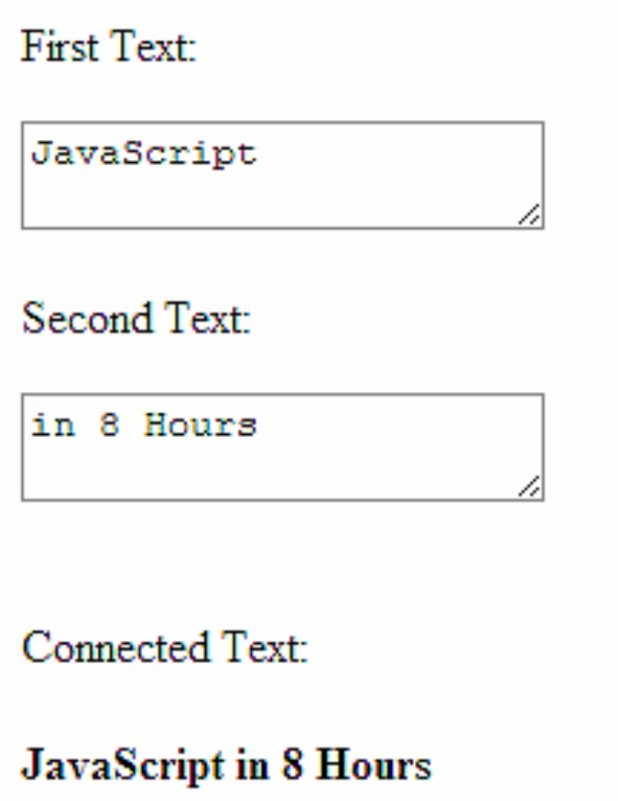
Controller Sample

Open Notepad, write AngularJS codes:

```
<!DOCTYPE html>
<html>
<script src= "js\angular. min. js"></script>
<body>
<div ng-app="myApplication"
ng-controller="myController">
First Text:
<p><textarea ng-model="firstText"></textarea></p>
Second Text:
<p><textarea ng-model="SecondText"></textarea></p>
<br>
Connected Text:
<h4><p>{{firstText + " " + SecondText}}</p></h4>
</div>
<script>
var app = angular.module('myApplication', [ ]);
app.controller('myController', function($scope) {
    $scope.firstText = "JavaScript";
    $scope.SecondText = "in 8 Hours";
});
</script>
</body>
</html>
```

Please save the file with name “ControllerSample. html” at myFolder. Double click “ControllerSample. html” file, input texts, and see the output.

Output:



Explanation:

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

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

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

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

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

Chapter 8
Module & API

What is 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 creating a module looks like this:

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

Example 8.1

```
<html>
<script src= "js\angular. min. js"></script>
<body>
<div ng-app="myApplication" ng-controller="myController">
{{ firstNum + lastNum }}
</div>
<script>
var myModule = angular.module("myApplication", [ ]);
myModule.controller("myController", function($scope) {
    $scope.firstNum = 100;
    $scope.lastNum = 200;
});
</script>
</body>
</html>
```

Output:

300

Explanation:

 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.

What is 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.

Example 8.2

```
<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

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.

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

lowercase( )

angular. lowercase( ) converts a string to lowercase.

Example 8.3

```
<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

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.

“$scope. text2 = angular. lowercase($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.

Example 8.4

```
<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

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.

“$scope. text2 = angular. isString($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.

Example 8.5

```
<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 see if it is a Date object, returns true if the value is a Date Object.

Example 8.6

```
<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

Explanation:

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.

Example 8.7

```
<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

Explanation:

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 an HTML element, returns true if the value is an HTML element.

Example 8.8

```
<!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

Explanation:

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.

Example 8.9

```
<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

Explanation:

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.

Example 8.10

```
<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

Explanation:

“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**.**

Example 8.11

```
<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

Explanation:

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

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

Summary

“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.

angular. uppercase( ) converts a string to uppercase.

angular. lowercase( ) converts a string to lowercase.

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

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

isDate( ) tests a value to see if it is a Date object, returns true if the value is a Date Object.

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

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

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

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

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

Exercises

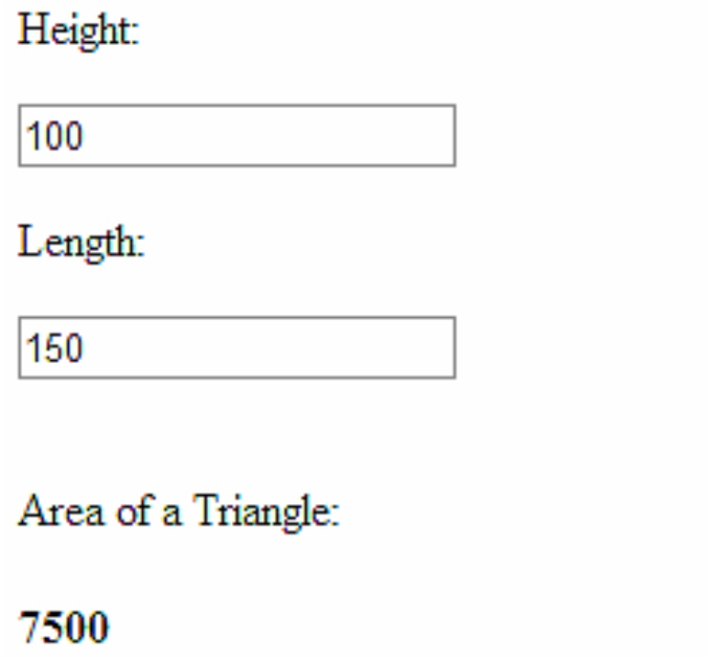
Module Sample

Open Notepad, write AngularJS codes:

```
<!DOCTYPE html>
<html>
<script src= "js\angular. min. js"></script>
<body>
<div ng-app="myApplication"
ng-controller="myController">
Height:
<p><input type="number" ng-model="height"></p>
Length:
<p><input type="number" ng-model="length"></p>
<br>
Area of a Triangle:
<h4><p>{{(height * length)/2}}</p></h4>
</div>
<script>
var app = angular.module('myApplication', [ ]);
app.controller('myController', function($scope) {
    $scope.height = 0;
    $scope.length = 0;
});
</script>
</body>
</html>
```

Please save the file with name “ModuleSample. html” at myFolder. Double click “ModuleSample. html” file, input height and length of a triangle, and see the output.

Output:



Explanation:

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

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

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

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

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

Appendix 1
Ajax Basic

What is Ajax?

AJAX stands for “**A** synchronous **J** avaScript **A** nd **X** ML”**.** AJAX is a new technique for creating better, faster, and more interactive web scripts with XML, HTML, CSS, PHP, ASP and Java Script by the server**.**

AJAX is used to update parts of a web page, without reloading the whole page**.**

That means with Ajax, you can communicate with the server to renew the information, without refreshing the web page**.**

By jQuery and JavaScript, you can easily implement Ajax in your web pages**.**

To run Ajax, you need a server**.**

The following will talk about how to set up a server on your own computer, and then how to use Ajax**.**

Set up a Server

If you want to run Ajax, you need to set up a server on your own computer**.**

"**AppServ** " is free software to setup a server; it can run Apache, PHP, MySQL, PhpMyAdmin, Ajax, JavaScript, JQuery……

Step 1:

Download "**AppServ** " form link: (Figure 1)

http://www **.** appservnetwork **.** com/

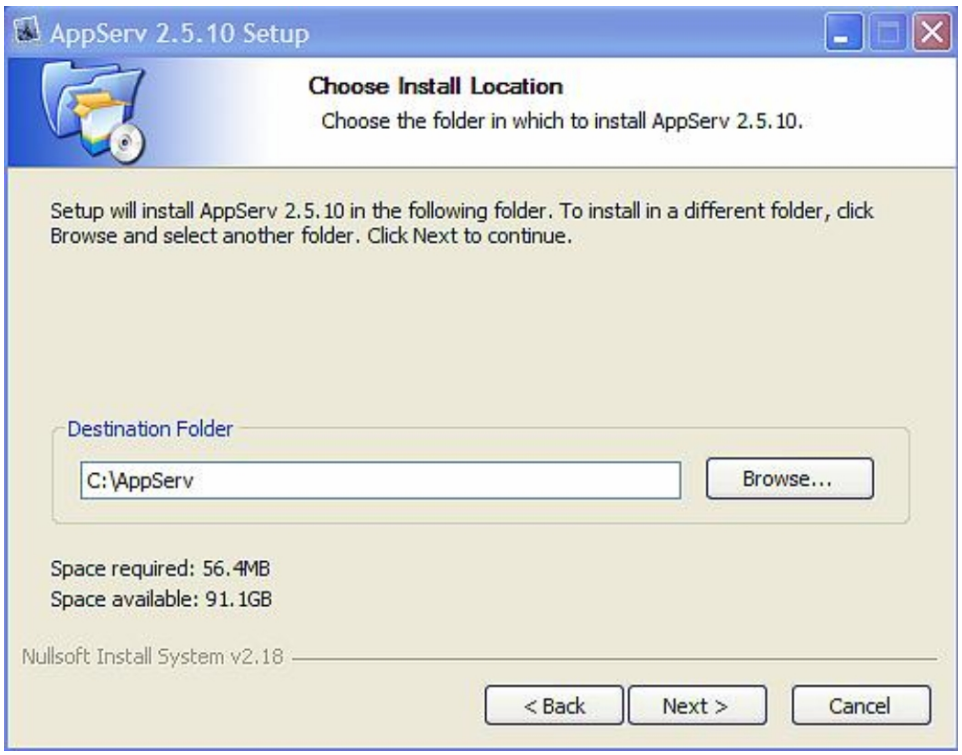
(Figure 1)

Step 2:

Install AppServ to local computer**.**

C:\AppServ

(Figure 2)



(Figure 2)

Step 3:

Please check all box to install anything**.** (Figure 3)

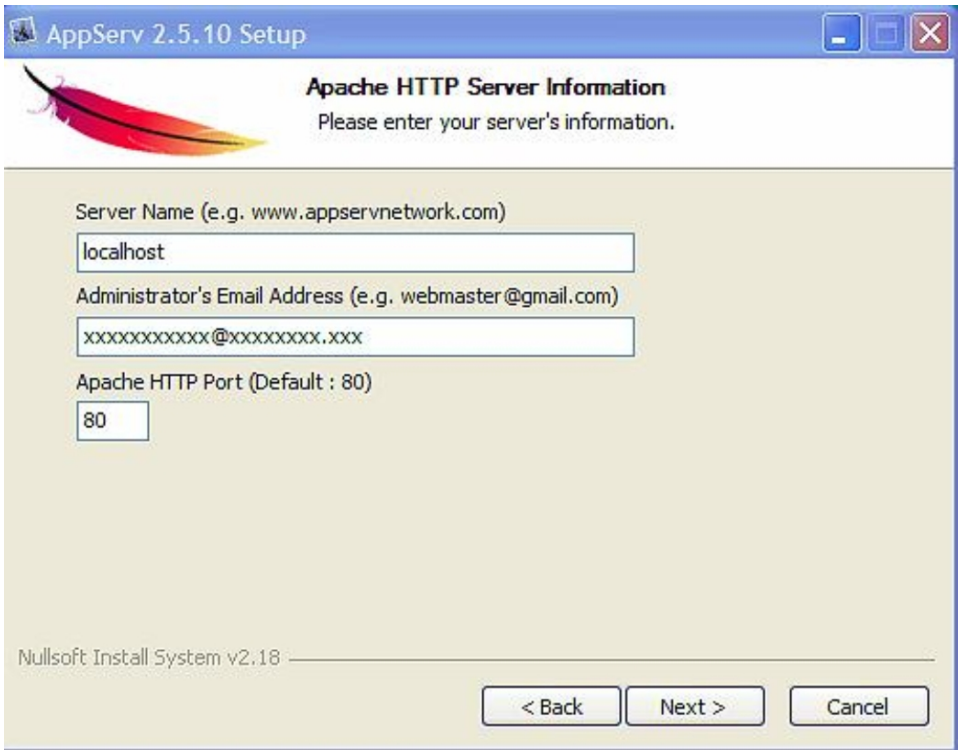
(Figure 3)

Step 4:

Server Name: localhost

Email Address: xxxxxxxx@xxxxx**.** xxx

Apache HTTP Port: 80 (Figure 4)



(Figure 4)

Step 5:

Enter root password: **12345** (Figure 5)

(Default user name: **root**)

(Figure 5)

Step 6:

Check the box, Start Apache, Start MySQL

Click Finish button**.** (Figure 6)

(Figure 6)

Step 7:

Test to see if AppServ installation is successful, please run a browser, enter the address:

http://localhost

If you can see the page like this: (Figure 7)

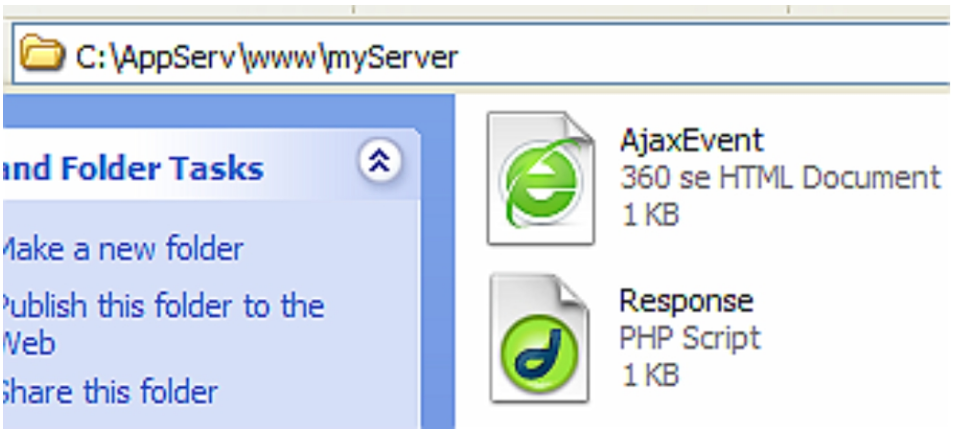
(Figure 7)

Congratulation! Your own server runs successfully**.**

Step 8:

How to use server?

Open folder **C:\AppServ\www** , please create a folder named “myServer”, which can work as a workplace**.** You can put all your program files to “myServer” folder**.** From now on, you are able to run Ajax files, PHP files and HTML files on **C:\AppServ\www\myServer.** (Figure 8)



(Figure 8)

Note:

Please use "**http://localhost/.....** "to run any files on the server, for example:

If you want to run AjaxEvent**.** html, enter an address:

http://localhost/myServer/ AjaxEven**.** html

If you want to run Response**.** php, enter an address:

http://localhost/myServer/ Response**.** php

How to use Ajax?

Note: Ajax needs a server support.

If your computer can work as a server, please prepare a text file “data. txt” which contains the words “*Hello Ajax World!”*, and put “data. txt” in the same folder with the following Ajax file.

Sample 1

Example B1

```
1. <script Language = “JavaScript”>
2. var ajaxObject = new XMLHttpRequest( );
3. ajaxObject. open( “GET”, “data. txt”, true);
4. ajaxObject. onreadystatechange = function( ) {
5.        if (ajaxObject. readyState == 4) {
6.       alert( ajaxObject. responseText);
7. }
8. }
9. ajaxObject. send( );
10. </script>
```

Save the file as “**B1.html** ”, and double click it.

Output:

Hello Ajax World!

Explanation:

Line 2:

“new XMLHttpRequest( )” creates an XMLHttpRequest object named “ajaxObject”, which will be used for Ajax scripts.

Line 3:

"ajaxObject**.** open("GET", "data**.** txt", true)" initializes the request of XMLHttpRequest object**.**

"GET" specifies a request method**.** "POST" or "PUT" can also be used**.**

"data**.** txt" is an URL which supplies with data resource**.**

In here, the content of datat**.** txt is "*Hello Ajax World!"* **.**

"true" indicates using Asynchronous technique**.**

Line 4:

"ajaxObject**.** onreadystatechange = function( ) { … }" specifies a function to run when the server's response has been received**.** The server's response is 0 to 4:

readyState	**explanation**
0	uninitialized
1	initialized
2	send data
3	receive data
4	complete**.** returned data is available**.**

"ajaxObject**.** onreadystatechange" is waiting for the server's response**.**

Line 5:

"if (ajaxObject**.** readyState == 4)" means if the server's response is 4, the returned data is available**.**

Line 6:

"alert(ajaxObject**.** responseText)" display the retrieved text from data**.** txt**.**

Output: "*Hello Ajax World!"* **.**

Line 9:

" ajaxObject**.** send( )**"** sends the request to the server**.**

Sample 2

Example B2

```
1. <input type = “button” value = "try" onClick = “ajaxRequest ( );”/>
2. <script type = “text/javascript”>
3. function ajaxRequest ( ) {
4. if (window. XMLHttpRequest){
5. ajaxObject = new XMLHttpRequest ( );
6. }
7. else if (window. ActiveXObject){
8. ajaxObject = new ActiveXObject(“Microsoft . XMLHTTP ”);
9. }
10. ajaxObject. open( “POST”, “data. txt”, true);
11. ajaxObject. onreadystatechange = function( ) {
12.      if (ajaxObject. readyState == 4) {
13.            if(ajaxObject. status == 200) {
14.            alert ( ajaxObject. responseText);
15.            }
16.      }
17. }
19. ajaxObject. send( );
20. }
21. </script>
```

Save the file as “**B2.html** ”, and double click it.

Output:

Hello Ajax World!

Explanation:

Line 1:

"onClick = "ajaxRequest ( )" calls the function ajaxRequest( ) when clicking the button.

Line 3-9:

Those codes create an XMLHttpRequest object named "ajaxObject", which will be used for Ajax scripts.

If the browser is NON-IE, it uses "new XMLHttpRequest ( )" to creates "ajaxObject".

If the browser is IE, it uses "new ActiveXObject(…)" to create "ajaxObject".

Line 10:

"ajaxObject. open("POST", "data. txt", true)" initialize the request of XMLHttpRequest object.

"POST" specifies a request method. "GET" or "PUT" can also be used.

"data. txt" is an URL which supplies with data resource.

In here, the content of datat. txt is "*Hello Ajax World!"* .

"true" indicates using Asynchronous technique.

Line 11:

"ajaxObject. onreadystatechange = function( ) { … }" specified a function to run when the server's response has been received. The server's response is 0 to 4:

readyState	**explanation**
0	uninitialized
1	initialized
2	send data
3	receive data
4	complete. returned data is available.

"ajaxObject. onreadystatechange" is waiting for the server's response.

Line 12:

“if (ajaxObject**.** readyState == 4)” means if the server’s response is 4, the returned data is available**.**

Line 13:

“if(ajaxObject**.** status == 200){ }” means if the server’s returned http status code is 200, the request succeeded**.**

The following chart is about “http status”:

There four http status, 200, 401, 403, 404**.**

http status	**explanation**
200	the request succeeded
401	unauthorized
403	forbidden
404	not found

Line 14:

“alert(ajaxObject**.** responseText)” display the retrieved text from data**.** txt**.**

Output: “*Hello Ajax World!”* **.**

Line 15:

“ ajaxObject**.** send( )**”** sends the request to the server**.**

Result:

Run the whole script, a dialogue box will pop up showing “*Hello Ajax World! ”***.**

Sample 3

The codes in sample 3 are almost the same as sample 2.
But the codes in sample 2 can be modified and become sample 3 as follows:

Example B3

14. document. getElementById(“display”). innerHTML =
“A message from server is: ” + ajaxObject. responseText;
22.

B3.html as follows:

```
<input type = “button” value = "try" onClick = “ajaxRequest ( );”/>
<script type = “text/javascript”>
function ajaxRequest ( ) {
if (window. XMLHttpRequest){
ajaxObject = new XMLHttpRequest ( );
}
else if (window. ActiveXObject){
ajaxObject = new ActiveXObject(“Microsoft. XMLHTTP”);
}
ajaxObject. open( “POST”, “data. txt”, true);
ajaxObject. onreadystatechange = function( ) {
if (ajaxObject. readyState == 4) {
if(ajaxObject. status == 200) {
document. getElementById( “display” ). innerHTML = “A message from
server is: ” + ajaxObject. responseText;
}
}
}
```

ajaxObject. send( );

}

</script>

Save the file as “**B3.html** ”, and double click it.

Output:

A message from server is: Hello Ajax World!

Explanation:

Line 14:

“document. getElementById(“display”). innerHTML” shows the content at the specified location where ID is “display”.

“ajaxObject. responseText” gets data from data. txt.

Output: “*A message from server is: Hello Ajax World! ”*.

Line 22

“ **”** creates a tag whose id is “display”, for the purpose to show the contents.

Result:

Run the whole script.

Output: *“A message from server is: Hello Ajax World! ”*

Note:

“data. txt” can be a data. php, data. asp, data. jsp and data. xml.

Ajax needs the support from the server.

The advantage of Ajax is: Ajax updates a part of a web page without reloading the whole page. The response is very fast!

Ajax Chart

property / method	description
open ( )	initialize a new http request
send ( )	send request to server
onreadystatechange	status event
readyState	request process (0,1,2,3,4)
responseText	return a text from server
status	http status code (200, 404…)

readyState	**explanation**
0	uninitialized
1	initialized
2	send data
3	receive data
4	Complete**.** Returned data is available**.**

http status	**explanation**
200	the request succeeded
401	unauthorized
403	forbidden
404	not found

Appendix 2
Know More AngularJS

Angular Service

Service is a function or an object, which is used to provide with a specified action. In AngularJS, there are about 30 builtin services, such as $http, $location, $interval and $timeout. Sometimes you can build a customized service.

For example, the $timeout service can be used to call another JavaScript function after a given time delay. The following is a $timeout service example:

Example A1

```
<!DOCTYPE html>
<html>
<head>
<script src="js\angular. min. js"></script>
</head>
<body>
<div ng-app="iCode" ng-controller="iController">
<h2>{{message}}</h2>
</div>
<script>
var iModule = angular. module('iCode', []);
iModule. controller('iController', function($scope, $timeout) {
$scope. message = "Please wait..... ";
$timeout(function () {       // $timeout service
$scope. message = "This is a Timeout service example. ";
}, 3000 );      // delay three second
});
</script>
```

</body>
</html>

Output 1:

Please wait……

Output 2:

This is a Timeout service example.

Explanation:

$timeout(function () { } is a AngularJS service**.**

3000 is a parameter: it means to delay three seconds to take action**.**

Angular Http

$Http is an Angular service that is used to access the data from the server.

```
$http. get( ). then( );
$http. post( ). then( );
```

get() or post() can access the data of server, then() returns the message from the server.

Example A2

```
/* Note: the following example needs a server to support .
Assume there is a file httpService . txt in the server, the file has the text
"This is an HTTP service example . " */
<!DOCTYPE html>
<html>
<head>
<script src="js\angular. min. js"></script>
</head>
<body>
<div ng-app="iCode" ng-controller="iController">
<p>A message from Server:</p>
<h2>{{message}}</h2>      <!--show data-->
</div>
<script>
var iModule = angular. module('iCode', []);
iModule. controller('iController', function($scope, $http) {
$http.get("httpService.txt").then(function (feedback) {
// get data from server
$scope. message = feedback. data;   // return data of the file
```

```
});
});
</script>
</body>
</html>
```

Output:

A message from Server:

This is an HTTP service example.

Explanation:

$http. get("httpService. txt"). then (function (feedback) gets data "httpService. txt" in the server, then() returns responding message.

"$scope. message = feedback. data;" returns data of the file.

Angular MySql

AngularJS can access date in MySql database by PHP programs.

Example A3

/*Assume there is a PHP file “angular_mysql . php” with MySQL database table shown like this:

ID	Fruit	Color
001	Apple	Red
002	Banana	Green
003	Orange	Golden

*/

```
<!DOCTYPE html>
<html>
<head>
<script src="js\angular. min. js"></script>
</head>
<body>
<div ng-app="iCode" ng-controller="iController">
<table>
<tr ng-repeat="v in title">
<td>{{ v. ID }}</td>
<td>{{ v. Fruit }}</td>
<td>{{ v. Color }}</td>
</tr>
</table>
```

```
</div>
<script>
var iModule = angular. module('iCode', []);
iModule. controller('iController', function($scope, $http) {
$http.get("angular_mysql.php").then(function (feedback) {
// get data from mysql database
$scope. title = feedback. records;   // return each record of the mysql database
});
});
</script>
</body>
</html>
```

Output:

ID	Fruit	Color
001	Apple	Red
002	Banana	Green
003	Orange	Golden

Explanation:

$http. get("angular_mysql. php"). then(function (feedback) { } gets data from mysql database.

“$scope. title = feedback. records;” returns each record of the mysql database.

<tr ng-repeat="v in title" iterates through each field name.

About PHP & MySQL, please read my other book “PHP & MySQL in 8 Hours” .

Angular Check

$dirty // check that user has inputted
$pristine // check that user has not inputted

Example A4

```
<!DOCTYPE html>
<html>
<head>
<script src="js\angular. min. js"></script>
</head>
<body>
<form ng-app="iCode" ng-controller="iController"
name="iForm" novalidate>
<p>Please input your name in here:<br><br>
<input type="text" name="userName" ng-model="userName" required>
<div style="color:red" ng-show="iForm. userName.!$dirty ">
<!--check input empty-->
<div ng-show="iForm. userName.$error. required">
<!--show error message -->
This field cannot be empty!</div>
</div>
</p>
</form>
<script>
var iModule = angular. module('iCode', []);
iModule. controller('iController', function($scope) {
$scope. userName = '';
```

```
});
</script>
</body>
</html>
```

Output:

Please input your name in here:

This field cannot be empty!

Explanation:

“iForm**.** userName**.** !$dirty” checks the emptiness of the input field**.** You can use **$pristine** or **!$dirty.**

“**.** $error**.** required” shows the error message**.**

Angular Validation

```
$valid       // check the validity of the input
$invalid     // check  the invalidity of the input
```

Example A5

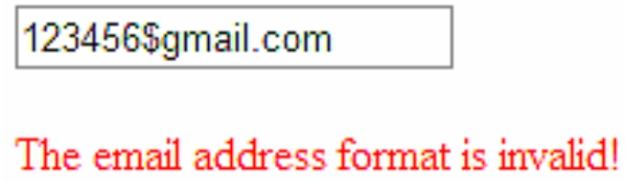
```
<!DOCTYPE html>
<html>
<head>
<script src="js\angular. min. js"></script>
</head>
<body>
<br>
<form ng-app="iCode" ng-controller="iController"
name="iForm" novalidate>
<input type="email" name="email" ng-model="email" required>
<div style="color:red" ng-show="iForm. email.$invalid ">
<!--check invalidity of the email address--> <br>
<div ng-show="iForm. email.$error. email">
<!--show error message -->
The email address format is invalid!</div>
</div>
</form>
<script>
var iModule = angular. module('iCode', []);
iModule. controller('iController', function($scope) {
$scope. email = 'xxxxx@gmail. com';
});
```

</script>

</body>

</html>

Output:



Explanation:

 checks validity of the email address.

 shows error message.

Angular Include

ng-include="'another. txt'"

ng-include="'another. txt'" includes an external file to the current file.

Example A6

/* Note: the following example needs a server to support .

Assume there is a file another . txt in the server, the file has the text “This is an Include Example . ” */

<!DOCTYPE html>

<html ng-app="">

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

<body>

<div **ng-include="'another.txt'"** >

</body>

</html>

Output:

This is an Include Example .

Explanation:

ng-include="'another. txt'" includes an external file to the current file.

Appendix 3
Tests & Answers

Tests

Please choose the correct answer.

(1)

```
<body>
<div ng-app="">
<p>Input my id in the input box:</p>
<p><input type="text" fill in here </p>
// binds the inputted data from HTML controls to “id”
<br>
<h1>Welcome! {{id}}</h1>
</div>
</body>
```

A. bind = “id”

B. ng-bind = “id”

C. ng = “id”

D. ng-model="id"

(2)

```
<body ng-app="">
<br><br>
<textarea ng-keydown="count = count + 1"
ng-init="count=0">
Here is a textarea
</textarea>
<br><br>
```

<h2>count: {{ **fill in here** }}</h2> // displays the value of “count”

<body>

A. show B. display C. count D. variable

(3)

<body>

<script>

var myString = "This is a string";

function myFunction() {

    return "This is inside a function";

}

document. write(**fill in here** . isFunction(myString) + " ");

document. write(**fill in here** . isFunction(myFunction) + " ");

// tests a value to see if it is a function

</script>

</body>

A. ng-test B. angular C. AngularJS D. test

(4)

<body>

<div **fill in here** >

 // indicates you can write AngularJS application from here .

Input my id in the input box:

<input type="text" ng-model="id"

<h1>Welcome! {{id}}</h1>

</body>

A. ng-app="" B. ng-model=”” C. angular D. ng

(5)

<body>

<div ng-app="" **fill in here** ="increaseNumber=0">

// initialize the variable with a value

<button ng-click="increaseNumber = increaseNumber + 1">Click Me!
</button>

</body>

A. init B. initialize C. ng-init D. ng-initalize

(6)

<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) {

  fill in here . firstNumber = 4;

fill in here . secondNumber = 8;

// assigns the value to the property of $scope object .

```
});
</script>
</body>
```

A. object B. angular C. scope D. $scope

(7)

```
<div ng-app="myApplication" ng-controller="myController">
{{ firstNum + lastNum }}
</div>
<script>
var myModule = angular. module("myApplication", [ ]);
fill in here . controller("myController", function($scope) {
// makes myController belongs to myModule .
    $scope. firstNum = 100;
    $scope. lastNum = 200;
});
</script>
```

A. module B. myModule C. $scope D. angular

(8)

```
<body>
<script>
var myCar = "This is my car!";
var carObject = new Object( ) ;
document. write(angular. fill in here Object(myCar) + " ");
document. write(angular. fill in here Object(carObject) + " ");
// tests a value to see if it is an object
</script>
```

```
</body>
```

A. test B. check C. is D. if

(9)

```
<body>
<div ng-app=""  ng-init = "ColorName = ['Pink', 'Red', 'Green', 'Blue',
'Black', 'White', 'Yellow', 'Gray']">
<p style="color:green; font-weight:bold">Colours Name:</p>
<ol>
    <li fill in here = "x in ColorName">
// repeats to get the value of an array
        <p ng-bind="x"></p>
    </li>
</ol>
</div>
</body>
```

A. ng-repeat B. repeat C. foreach D. for

(10)

```
<body>
<div ng-app="Calculation" fill in here = "myController">
// defines a controller
First Number: <input type="number" ng-model="firstNumber"><br>
Second Number: <input type="number" ng-model="secondNumber"><br>
<br>
Sum: {{firstNumber + secondNumber}}
</div>
<script>
```

```
var app = angular. module('Calculation', [ ]);
app. controller('myController', function($scope) {
    $scope. firstNumber = 4;
    $scope. secondNumber = 8;
});
</script>
</body>
```

A. define B. controller C. new D. ng-controller

(11)

```
<html>
<script fill in here ="js\angular. min. js"></script>
// adds AngularJS framework into current application .
<body>
<div ng-app="">
<p>Input my id in the input box:</p>
<p><input type="text" ng-model="id" </p>
<br>
<h1>Welcome! {{id}}</h1>
</div>
</body>
</html>
```

A. import B. url C. src D. add

(12)

```
<body>
<div ng-app="">
<p>User Name: <br>
```

<input type="text" ng-model = "Username">

<p **fill in here** ="Username">

// binds “Username” to tag, and “shows” its value

</body>

A. bind B. ng-bind C. ng-bound D. ng-show

(13)

<body>

<h3>Using Upper Case Filter</h3>

User Name: <input type="text" ng-model = "Username">

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

</body>

A. toUppercase()

B. uppercase()

C. toUppercase

D. uppercase

(14)

<body>

<h3>Using Click directive</h3>

<button **fill in here** ="increaseNumber = increaseNumber + 1">Click Me!
</button>

// updates the value of “increaseNumber” when click the button .

```
<p ng-bind="increaseNumber" style="color:blue"></p>
</div>
</body>
```

A. ng-click B. ng-press C. onClick D. onPress

(15)

```
<body ng-app="">
<br><br>
<textarea fill in here ="count = count + 1"
ng-init="count=0">    // mouse over event
Here is a textarea.
</textarea>
<br><br>
<h2>count: {{count}}</h2>
</body>
```

A. mouseover B. ng-mouseover C. over D. ng-over

(16)

```
<body>
<h4>My Math Result Using Array Expression</h4>
<div ng-app="" ng-init="MyArray=[98,96,93,90,99]">
<p>My score in mathematics is:  {{ fill in here }}</p>
// displays the value whose index is 4 in MyArray .
</div>
</body>
```

A. array[4] B. [4] C. MyArray[4] D. MyArray{4}

(17)

MVC stands for Model, View, and Controller.

“$scope. firstNumber and $scope. secondNumber” is a **fill in here**

  is a **fill in here**

“ng-controller="myController"”defines a **fill in here**

// MVC

A. controller view model

B. view controller model

C. model controller view

D. model view controller

(18)

```
<body>
<script>
var myDate = "Aug/10/2015";
var dateObject = new Date();
document. write(angular. fill in here (myDate) + "  ");
document. write(angular. fill in here (dateObject) + "  ");
// tests a value to see if it is a Date object
</script>
</body>
```

A. check B. test C. isDate D. isObject

(19)

```
<body ng-app="">
<br><br>
<textarea fill in here ="count = count + 1"
ng-init="count=0">   // press key down event
Here is a textarea
```

</textarea>

<h2>count: {{count}}</h2>

</body>

A. keydown B. ng-keydown C. onKeydown D. key()

(20)

<body>

<script>

var futureCar;

var existingCar = "Very Good!";

document. write(angular. **fill in here** (futureCar) + " ");

document. write(angular. **fill in here** (existingCar) + " ");

// tests a value to see if it has been defined

</script>

</body>

A. test B. check C. ng-defined D. isDefined

(21)

<body>

<table border="1" >

<tr>

<th>Student Name</th>

<th>Mathematics' Result</th>

</tr>

<tr ng-repeat="x in StudentsResult | **fill in here** :'-marks' ">

// display values in descending order **.**

<td ng-bind="x**.** name "></td>

<td ng-bind="x**.** marks "></td>

</tr>

</table>

</body>

A**.** order B**.** sort C**.** orderBy D**.** sortBy

(22)

<body>

<input type="checkbox" ng-model="HideShowDiv">Hide Div

<div **fill in here** ="HideShowDiv" style="background-color:pink;height:50px;"> // use hide directive

I am a div

I will disappeared when the check box is checked**.**

</body>

A**.** hide B**.** invisible C**.** ng-hide D**.** ng-invisible

(23)

<script>

function (**fill in here**) {

fill in here **.** firstNumber = 23;

fill in here . secondNumber = 63;
}
</script>
// declare a parameter of JavaScript function, the parameter will be an object in AngularJS .
A. ng-controller B. $scope C. ng-object D. ng-scope

(24)
<body>
<h3>Using Show directive</h3>

<input type="checkbox" ng-model="showHideDiv">Show Div
<div **fill in here** ="showHideDiv" style="background-color:yellow;height:50px;"> // show directive
I am a div
I appear when the check box is checked.

</body>
A. show B. angular C. ng-display D. ng-show

(25)
<body>
<h4>Object Expression</h4>

// use object expression according to context
Employee Name : {{EmployeeObject. Emp_name}}
Salary's Month: {{EmployeeObject. Emp_Month}}

Employee Salary: {{EmployeeObject. Emp_salary}}

</body>
A. EmployeeObject
B. myObject
C. Object
D. ng-Object

(26)

{{ text1 }}
{{ text2 }}

<script>
var myModule = angular. module('myApplication', [ ]);
myModule. controller('myController', function($scope) {
$scope. text1 = "JAVASCRIPT";
$scope. text2 = angular. **fill in here** ($scope. text1);
}); // check “text1” to see if it is a number.
</script>
A. Number B. isNumber C. Digital D. isDigital

(27)
1. <script Language = “JavaScript”>
2. var ajaxObject = new **fill in here** ; // creates an XMLHttpRequest object named “ajaxObject”
3. ajaxObject. open(“GET”, “data. txt”, true);
4. ajaxObject. onreadystatechange = function( ) {

```
5.        if (ajaxObject. readyState == 4) {
6.       alert( ajaxObject. responseText);
7. }
8. }
9. ajaxObject. send( );
10. </script>
```

A. HttpRequest

B. XMLHttpRequest

C. HttpRequest( )

D. XMLHttpRequest( )

(28)

```
1. <input type = “button” value = "try" onClick = “ajaxRequest ( );”/>
2. <script type = “text/javascript”>
3. function ajaxRequest ( ) {
4. if (window. XMLHttpRequest){
5. ajaxObject = new XMLHttpRequest ( );
6. }
7. else if (window. ActiveXObject){
8. ajaxObject = new ActiveXObject(“Microsoft. XMLHTTP”);
9. }
10. ajaxObject. open( “POST”, “data. txt”, true);
11. ajaxObject. onreadystatechange = function( ) {
12.      if (ajaxObject. readyState == 4) {
13.              if(ajaxObject. fill in here == 200) {  // if the server’s returned http status code is 200, the request succeeded
14.              alert ( ajaxObject. responseText);
15.              }
```

16. }

17. }

19. ajaxObject. send( );

20. }

21. </script>

A. respond B. readState C. status D. return

(29)

<input type = “button” value = "try" onClick = “ajaxRequest ( );”/>

<script type = “text/javascript”>

function ajaxRequest ( ) {

if (window. XMLHttpRequest){

ajaxObject = new XMLHttpRequest ( );

}

else if (window. ActiveXObject){

ajaxObject = new ActiveXObject(“Microsoft. XMLHTTP”);

}

ajaxObject. open(“POST”, “data. txt”, true);

ajaxObject. onreadystatechange = function( ) {

if (ajaxObject. readyState == 4) {

if(ajaxObject. status == 200) {

document. getElementById(“display”). **fill in here** = “A message from server is: ” + ajaxObject. responseText; // shows the content at the specified location where ID is “display”

}}}

ajaxObject. send( );

}

</script>

A. show B. innerHTML C. display D. text

(30)

In Ajax, when readyState is 4, it means that request process at the stage is ___?

A. Uninitialized

B. Sending data

C. Received data

D. Returned data available.

(31)

In Ajax, when http status is 200, it means that the result of request is ____?

A. the request succeeded.

B. unauthorized

C. forbidden

D. not found.

(32)

Which following statement is correct about Ajax? (Multiple choice)

A. readyState property is a request status from 0 to 4.

B. Http status returns a number 200, 401, 403 or 404.

C. onreadystatechange property is a status event.

D. responseText property returns a text from server.

Note:

This book is only for beginners, it is not suitable for experienced programmers.

Answers

01.	D	17.	D
02.	C	18.	C
03.	B	19.	B
04.	A	20.	D
05.	C	21.	C
06.	D	22.	C
07.	B	23.	B
08.	C	24.	D
09.	A	25.	A
10.	D	26.	B
11.	C	27.	D
12.	B	28.	C
13.	D	29.	B
14.	A	30.	D
15.	B	31.	A
16.	C	32.	ABCD

Source Code Download Link:

https://forms . aweber . com/form/31/1924478131 . htm

Recommended Books

AngularJS Programming in 8 Hours
C# Programming in 8 Hours
C++ Programming in 8 Hours
Django Programming in 8 Hours
Go Programming in 8 Hours
HTML CSS Programming in 8 Hours
JAVA Programming in 8 Hours
JavaScript Programming in 8 Hours
JQuery Programming in 8 Hours
Kotlin Programming in 8 Hours
Linux Shell Programming in 8 Hours
MySQL Programming in 8 Hours
Node . js Programming in 8 Hours
PHP MySQL Programming in 8 Hours
PowerShell Programming in 8 Hours
Python Programming in 8 Hours
R Programming in 8 Hours
Ruby Programming in 8 Hours
Rust Programming in 8 Hours
Scala Programming in 8 Hours
Visual Basic Programming in 8 Hours