ANGULARJS 1

by Harsha Vardhan (UI Expert)

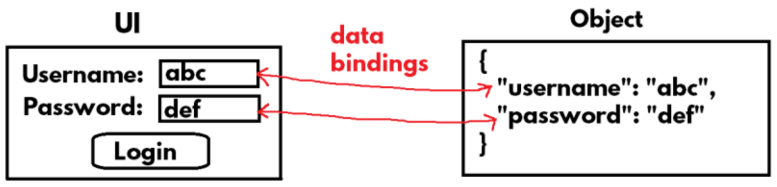
Contents

AngularJS 1

ANGULARJS 1

Introduction to AngularJS

- "AngularJS" is a "Client side framework", which is used to create "data bindings" in web pages.
- "Data binding" is the relationship between "UI" and "Object". When the data is modified in the object, it will be automatically updated in "UI" and vice versa.



- AngularJS was developed by Google in 2010.
- AngularJS supports "clean separation between html code and javascript code". So that the html code and javascript code can be developed individually by two different programmers / teams.
- AngularJS is open source.
- AngularJS is free-to-use.

AngularJS - Browser Compatibility

Browser		AngularJS Version
• IE 8 and earlier	:	AngularJS 1.2 or older
• IE 9+	:	AngularJS 1.3+
• Opera 15+	:	Any
• Google Chrome (any version)	:	Any
• Mozilla Firefox (any version)	:	Any
• Safari (any version)	:	Any
• Microsoft Edge (any version)	:	Any

Downloading AngularJS

- Open any browser.
- Go to "https://angular.org".
- Click on "Download AngularJS".
- Click on "1.6.x (latest)" - "Zip".
- Click on "Download".
- Download the "angular-1.6.4.zip" file.

- Right click on the "angular-1.6.4.zip" file and click on "Extract All".
- You will get extracted folder called "angular-1.6.4".

AngularJS Library Files

1. angular.js
2. angular.min.js
3. angular-animate.js
4. angular-animate.min.js
5. angular-route.js
6. angular-route.min.js
7. angular-mocks.js

1. angular.js
  - This file contains core (primary) functionality angularjs framework.
  - All other library files works based on this file only.
  - Size: 1.19+ MB

2. angular.min.js
  - This file contains minified (compressed) code of "angular.js" file.
  - Minified / compressed: No comments, no line breaks, no spaces, no long variable names.
  - Size: 163 KB
  - Recommended in "production" (to upload to server).

3. angular-animate.js
  - This file contains essential code for creating "animations" in angularjs.
  - Animations are used to show / hide the elements attractively.
  - Size: 147+ KB

4. angular-animate.min.js
  - This file contains minified (compressed) code of "angular-animate.js".
  - Size: 25+ KB

5. angular-route.js
  - This file contains essential code for creating "routing" (page navigation) in angularjs.
  - Size: 43+ KB

6. angular-route.min.js
  - This file contains minified (compressed) code of "angular-route.js".
  - Size: 5+ KB

7. angular-mocks.js
  - This file contains code for performing unit testing of angularjs controllers.
  - Size: 120 KB

AngularJS - First Example

- Create "c:\angularjs" folder.
- Copy and paste the "angular.js" file from the "extracted folder (angular-1.6.4)" into the "application folder (c:\angularjs)".
- Create "first.html" in "c:\angularjs" folder.
- You will have two files in "c:\angularjs" folder.

c:\angularjs

- **angular.js**
- **first.html**

c:\angularjs\first.html

```html
<!DOCTYPE html>
<html>
   <head>
      <title>AngularJS - First Example</title>
      <script src="angular.js"></script>
   </head>
   <body>
      <div ng-app>
         {{10+20}}
      </div>
   </body>
</html>
```

- Go to "c:\angularjs" folder and double click on "first.html".
- Output:

```
30
```

Built-in Directives

- "AngularJS directives" are the "attributes", which are provided by AngularJS.
- AngularJS directives specify additional behavior of the html tag.
- All the angularjs directives start with "ng-".

  Syntax: `<tag directive="value">`
  `</tag>`

List of AngularJS Directives

- ng-app
- ng-init
- ng-bind
- ng-model
- ng-show
- ng-hide
- ng-disabled
- ng-if
- ng-src
- ng-class
- ng-click
- ng-dblclick
- ng-focus
- ng-blur
- ng-keyup
- ng-keypress

- ng-mouseover
- ng-mouseout
- ng-mousemove
- ng-change
- ng-repeat
- ng-view

ng-app

- The "ng-app" attribute is used to create "angularjs application", in which you can use any angularjs concepts such as expressions, filters etc.
- **Syntax:**

```
<div ng-app>
  …
</div>
```

Expressions

- Expressions are used to display the value of a property in the angularjs application.
- **Syntax:** {{property}}
- **Example:** {{x}}

ng-init

- The "ng-init" directive is used to initialize properties in angularjs.
- **Syntax:**

```
<div ng-app ng-init="property=value; property=value; …">
</div>
```

Example on ng-init:

```
<html>
  <head>
     <title>ng-init</title>
     <script src="angular.js"></script>
  </head>
  <body>
     <div ng-app ng-init="empid=101; empname='Scott'; salary=7000">
        EmpID: {{empid}}<br>
        EmpName: {{empname}}<br>
        Salary: {{salary}}
     </div>
  </body>
</html>
```

Example 2 on ng-init:

```
<html>
  <head>
     <title>JSON</title>
     <script src="angular.js"></script>
  </head>
  <body>
```

```
        <div ng-app ng-init="emp = { empid:101, empname:'Scott', salary:7000 }; stu = { rollno:123, studentname: 'Allen',
  marks: 80 }">
            EmpID: {{emp.empid}}<br>
            EmpName: {{emp.empname}}<br>
            Salary: {{emp.salary}}<hr>
            Student RollNo: {{stu.rollno}}<br>
            Student Name: {{stu.studentname}}<br>
            Marks: {{stu.marks}}
        </div>
    </body>
</html>
```

ng-bind

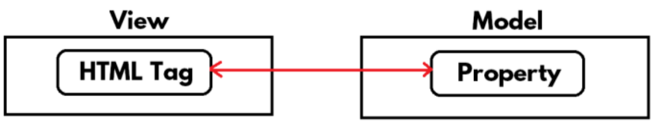
- The "ng-bind" directive is used to display the value of a property.
- It replaces the inner html of the tag with the value of the property.
- **Advantage:** The property name is not visible to the user, while loading the web page.
- **Disadvantages:**
  - The property can't be mixed with other content.
  - "ng-bind" can't be used in another attribute.
- **Syntax:** <tag ng-bind="property"></tag>
- **Example:** <p ng-bind="empname"></p>

Example on ng-bind:

```
<html>
    <head>
        <title>ng-bind</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <div ng-app ng-init="x='John' ">
            Student Name:
            <p ng-bind="x"></p>
        </div>
    </body>
</html>
```

Data Bindings

- "Data Binding" is the relationship between "property" of the model and "html tag" in the view.

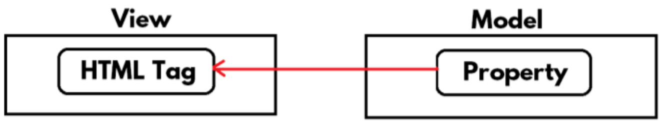


Types of Data Bindings:

- AngularJS supports three data bindings:
  1. One Way Data Binding
  2. Two Way Data Binding
  3. Event Binding

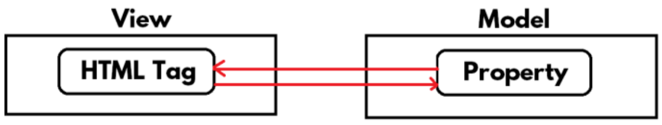
1. One Way Data Binding:

- Every time when the model property value is changed, the same value will be automatically updated in the view.
- **Syntax:** {{property}}



2. Two Way Data Binding:

- Every time when the model property value is changed, the same value will be automatically updated in the view; and vice versa.
- **Syntax:** <tag ng-model="property">
  </tag>
- The "ng-model" attribute is applicable only for <input> tag and <select> tag only.
- After creating two-way data binding, we have to set / get the value only through the model property, instead of accessing the tag directly.



ng-model

- The "ng-model" directive (attribute) is used to create "two way data binding".
- Every time when the model property value is changed, the same value will be automatically updated in the view; and vice versa.
- It is applicable only for <input> and <select> tag.
- **Syntax:** <tag ng-model="property"></tag>
- **Example:** <input ng-model="empname">

Example on ng-model:

```
<html>
    <head>
        <title>ng-model</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <div ng-app ng-init="x='John' ">
            Student Name:
            <p ng-bind="x"></p>
            Student Name:
            <input type="text" ng-model="x">
        </div>
    </body>
</html>
```

ng-show

- The "ng-show" directive shows / hides the element in the web page.
- ng-show="true" : Displays the element.

- ng-show="false" : Hides the element.
- **Syntax:** <tag ng-show="property"></tag>
- **Example:** <p ng-show="true">Welcome</p>

Example on ng-show:

```
<html>
   <head>
       <title>ng-show</title>
       <script src="angular.js"></script>
   </head>
   <body>
       <h1>ng-show</h1>
       <div ng-app ng-init="x = true; y = false">
            <div ng-show="x">div 1</div>
            <div ng-show="y">div 2</div>
       </div>
   </body>
</html>
```

ng-hide

- The "ng-hide" directive hides / shows the element in the web page.
- ng-hide="true" : Hides the element.
- ng-hide="false" : Displays the element.
- **Syntax:** <tag ng-hide="property"> </tag>
- **Example:** <p ng-hide="false">Welcome</p>

Example on ng-hide:

```
<html>
   <head>
       <title>ng-hide</title>
       <script src="angular.js"></script>
   </head>
   <body>
       <h1>ng-hide</h1>
       <div ng-app ng-init="x = true; y = false">
            <div ng-hide="x">div 1</div>
            <div ng-hide="y">div 2</div>
       </div>
   </body>
</html>
```

ng-if

- The "ng-if" directive shows / removes the element in the web page.
- ng-if="true" : Displays the element.
- ng-if="false" : Removes the element.
- **Syntax:** <tag ng-if="property"></tag>
- **Example:** <p ng-if="true">Welcome</p>

Example on ng-if:

```
<html>
```

```
    <head>
        <title>ng-if</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-if</h1>
        <div ng-app ng-init="x = true; y = false">
            <div ng-if="x">div 1</div>
            <div ng-if="y">div 2</div>
        </div>
    </body>
</html>
```

ng-disabled

- The "ng-disabled" directive disables / enables the element.
- ng-disabled="true" : Disables the element.
- ng-disabled="false" : Enables the element.
- **Syntax:** <tag ng-disabled="property"></tag>
- **Example:** <input type="button" ng-disabled="true">

Example on ng-disabled:

```
<html>
    <head>
        <title>ng-disabled</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-disabled</h1>
        <div ng-app ng-init="x = true; y = false">
            <input type="button" value="button1" ng-disabled="x">
            <input type="button" value="button2" ng-disabled="y">
        </div>
    </body>
</html>
```

ng-src

- The "ng-src" directive sets "src" attribute value dynamically based on the property.
- Advantage: It avoids sending a request to "{{property}}" address.
- **Syntax:** <img ng-src="{{property}}">
- **Example:** <img ng-src="{{a}}">

Example on ng-src:

```
<html>
    <head>
        <title>ng-src</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-src</h1>
        <div ng-app ng-init="a='img1.jpg' ">
            <img ng-src="{{a}}" width="200px">
        </div>
    </body>
</html>
```

ng-class

- The "ng-class" directive sets css class name to the element dynamically.
- **Syntax:** <tag ng-class="property"></tag>
- **Example:** <p class="a">Welcome</p>

Example on ng-class:

```
<html>
    <head>
        <title>ng-class</title>
        <style>
            .class1
            {
                background-color: darkgreen;
                color: white;
                border: 3px solid red;
            }
            .class2
            {
                background-color: darkred;
                color: white;
                border: 5px dotted #ffcc33;
            }
        </style>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-class</h1>
        <div ng-app ng-init="a='class1' ">
            <p ng-class="a">Hello</p>
        </div>
    </body>
</html>
```

ng-click

- The "ng-click" directive handles "click" event.
- When the user clicks on the element, the "click" event will be executed.
- **Syntax:** <tag ng-click="functionname( )"></tag>
- **Example:** <input type="button" ng-click="fun1( )">

Example on ng-click:

```
<html>
    <head>
        <title>ng-click</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-click</h1>
        <div ng-app ng-init="name='Scott' ">
            <input type="button" value="button1" ng-click="name='Allen' ">
            <br>
            {{name}}
        </div>
    </body>
</html>
```

Example 2 on ng-click:

```
<html>
    <head>
        <title>ng-click with ng-show</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-click with ng-show</h1>
        <div ng-app ng-init="x = true">
            <div ng-show="x">div 1</div>
            <input type="button" value="Hide" ng-click="x = false">
            <input type="button" value="Show" ng-click="x = true">
        </div>
    </body>
</html>
```

Example 3 on ng-click:

```
<html>
    <head>
        <title>ng-click</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-click</h1>
        <div ng-app ng-init="a=null; b=null; c=null ">
            Price: <input type="text" ng-model="a"><br>
            Discount: <input type="text" ng-model="b"><br>
            <input type="button" value="Add" ng-click="c = a - b"><br>
            Net Price: <input type="text" ng-model="c">
        </div>
    </body>
</html>
```

ng-dblclick

- The "ng-dblclick" directive handles "dblclick" event.
- When the user double clicks on the element, the "dblclick" event will be executed.
- **Syntax:** <tag ng-dblclick="functionname( )"></tag>
- **Example:** <input type="button" ng-dblclick="fun1( )">

Example on ng-dblclick:

```
<html>
<head>
    <title>AngularJS - dblclick</title>
    <style type="text/css">
        #div1
        {
            background-color: #00ccff;
            width: 200px;
```

```
                height: 100px;
            }
        </style>
        <script src="angular.js"></script>

        <script>
            var app = angular.module("mymodule", []);

            app.controller("mycontroller", fun1);
            function fun1($scope)
            {
                $scope.f1 = function()
                {
                    alert("hello");
                };
            }
        </script>
    </head>
    <body>
        <h1>AngularJS - dblclick</h1>
        <div ng-app="mymodule">
            <div ng-controller="mycontroller">
                <div id="div1" ng-dblclick="f1()">
                    double click me
                </div>
            </div>
        </div>
    </body>
    </html>
```

ng-focus

- The "ng-focus" directive handles "focus" event.
- The "focus" event executes when the cursor enters into the element.
- **Syntax:** <tag ng-focus="functionname( )"></tag>
- **Example:** <input type="text" ng-focus="fun1( )">

ng-blur

- The "ng-blur" directive handles "blur" event.
- The "blur" event executes when the cursor go out of the element.
- **Syntax:** <tag ng-blur="functionname( )"></tag>
- **Example:** <input type="text" ng-blur="fun1( )">

Example on ng-focus and ng-blur:

```
<html>
    <head>
        <title>ng-focus and ng-blur</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-focus and ng-blur</h1>
        <div ng-app ng-init="b=false;">
```

```
            Email:
            <input type="text" ng-focus="b=true" ng-blur="b=false">
            <span ng-show="b">Use gmail only</span>
        </div>
    </body>
</html>
```

ng-keyup

- The "ng-keyup" directive handles "keyup" event.
- The "keyup" event executes when the user presses any key on the keyboard.
- **Syntax:** <tag ng-keyup="functionname( )"></tag>
- **Example:** <input type="text" ng-keyup="fun1( )">

Example on ng-keyup:

```
<html>
    <head>
        <title>ng-keyup</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-keyup</h1>
        <div ng-app ng-init="a=''; b=''; c='' ">
            Price:
            <input type="text" ng-model="a" ng-keyup="c = a - b"><br>
            Discount:
            <input type="text" ng-model="b" ng-keyup="c = a - b"><br>
            Net Price: <input type="text" ng-model="c">
        </div>
    </body>
</html>
```

ng-keypress

- The "ng-keypress" directive handles "keypress" event.
- The "keypress" event executes when the user presses any key on the keyboard, before accepting currently pressed character.
- **Syntax:** <tag ng-keypress="functionname( )"></tag>
- **Example:** <input type="text" ng-keypress="fun1( )">

Example on ng-keypress:

```
<html>
<head>
    <title>AngularJS - keypress</title>
    <script src="angular.js"></script>
</head>
<body>
    <h1>AngularJS - keypress</h1>
    <div ng-app ng-init="a=null; b=null">
        Source text:
        <input type="text" ng-model="a" ng-keypress="b=a"><br>
        Destination text:
        <input type="text" ng-model="b"><br>
    </div>
```

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

ng-mouseover

- The "ng-mouseover" directive handles "mouseover" event.
- The "mouseover" event executes when the user places mouse pointer on the element.
- **Syntax:** <tag ng-mouseover="functionname( )"></tag>
- **Example:** <div ng-mouseover="fun1( )"></div>

ng-mouseout

- The "ng-mouseout" directive handles "mouseout" event.
- The "mouseout" event executes when the user moves the mouse pointer out of the element.
- **Syntax:** <tag ng-mouseout="functionname( )"></tag>
- **Example:** <div ng-mouseout="fun1( )"></div>

Example on ng-mouseover and ng-mouseout:

```
<html>
<head>
    <title>AngularJS - mouseover and mouseout</title>
  <style>
    #div1
    {
      background-color: #0094ff;
      font-size: 30px;
    }
  </style>
    <script src="angular.js"></script>
</head>
<body>
    <h1>AngularJS - mouseover and mouseout</h1>
    <div ng-app ng-init="a='Mouse over me';">
        <div id="div1" ng-mouseover="a='Thanx';" ng-mouseout="a='Mouse over me';">
            {{a}}
        </div>
    </div>
</body>
</html>
```

ng-mousemove

- The "ng-mousemove" directive handles "mousemove" event.
- The "mousemove" event executes when the user moves the mouse pointer on the element.
- **Syntax:** <tag ng-mousemove="functionname( )"></tag>
- **Example:** <div ng-mousemove="fun1( )"></div>

Example on ng-mousemove:

```
<html>
<head>
    <title>AngularJS - mousemove</title>
    <style type="text/css">
        #div1
        {
            background-color: #00ccff;
            width: 200px;
            height: 100px;
        font-size: 30px;
        }
    </style>
    <script src="angular.js"></script>
</head>
<body>
    <h1>AngularJS - mousemove</h1>
    <div ng-app ng-init="a=null;">
        <div id="div1" ng-mousemove="a=$event.pageX + ',' + $event.pageY">
            {{a}}
        </div>
    </div>
</body>
</html>
```

ng-change

- The "ng-change" directive handles "change" event.
- The "change" event executes when the user changes the value of an element (checkbox / radio button / dropdownlist).
- **Syntax:** <tag ng-change="functionname( )"></tag>
- **Example:** <input type="checkbox" ng-change="fun1( )">

Example on ng-change with CheckBox:

```
<html>
    <head>
        <title>ng-change - checkbox</title>
        <script src="angular.js"></script>
    </head>
    <body>
        <h1>ng-change - checkbox</h1>
        <div ng-app ng-init="a=true; b=false">
            <input type="checkbox" ng-model="a" ng-change="b=!a">
            I accept license agreement
            <br>
            <input type="submit" ng-disabled="b"><br>
            {{a}}
        </div>
    </body>
</html>
```

Example on ng-change with RadioButton:

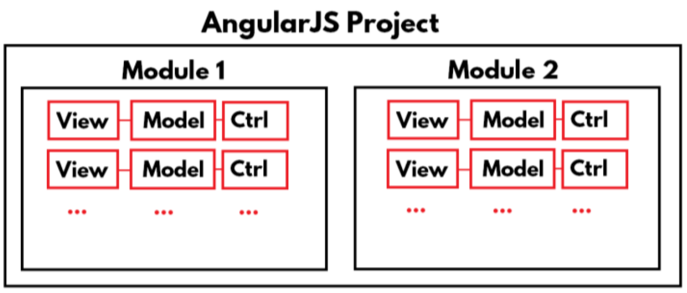
```
<html>
<head>
    <title>AngularJS - change - radiobutton</title>
    <script src="angular.js"></script>
</head>
<body>
    <h1>AngularJS - change - radiobutton</h1>
    <div ng-app ng-init="gender=null; msg=null;">
        Gender:
        <input type="radio" id="rb1" name="gender" ng-model="gender" value="M" ng-change="msg='Male selected'">
        <label for="rb1">Male</label>
        <input type="radio" id="rb2" name="gender" ng-model="gender" value="F" ng-change="msg='Female
selected'">
        <label for="rb2">Female</label>
        <br>{{gender}}<br>{{msg}}
    </div>
</body>
</html>
```

Example on ng-change with DropDownList:

```
<html>
<head>
    <title>AngularJS - change - dropdownlist</title>
    <script src="angular.js"></script>
</head>
<body>
    <h1>AngularJS - change - dropdownlist</h1>
    <div ng-app ng-init="country=null; msg=null">
        Country:
      <select ng-model="country" ng-change="msg=country + ' is selected' ">
            <option>India</option>
            <option>UK</option>
            <option>US</option>
            <option>Canada</option>
            <option>Japan</option>
        </select>
        <br>{{country}}<br>{{msg}}
    </div>
</body>
</html>
```

Modules

- The "angularjs project" is divided into multiple parts; each part is called as "module".
- "Module" is a collection of views, models and controllers.
- **Syntax:** angular.module("module name", [list of modules to import]);
- **Example:** angular.module("module1", [ ]);



Bootstrapping

- Bootstrapping is a process of loading module into the <div> tag.
- **Syntax:**

```
angular.element(document).ready(functionname);
function  functionname( )
{
```

```
            angular.bootstrap(document.getElementById("id of div"), [ "module name" ] );
          }
```

Example on Modules

```
 <html>
    <head>
        <title>Modules</title>
        <style>
            #div1,#div2,#div3
            {
                margin: 5px;
                padding: 10px;
                background-color: #0099ff;
                height: 100px;
            }
        </style>
        <script src="angular.js"></script>
        <script>
            var app1 = angular.module("module1", [ ]);
            var app2 = angular.module("module2", [ ]);
            var app3 = angular.module("module3", [ ]);

            angular.element(document).ready(myfunction);
        function myfunction()
        {
    angular.bootstrap( document.getElementById("div1"), ["module1"] );
    angular.bootstrap( document.getElementById("div2"), ["module2"] );
    angular.bootstrap( document.getElementById("div3"), ["module3"] );
        }
    </script>
  </head>
  <body>
    <h1>Modules, Models, Controllers and Views</h1>
    <div id="div1">
        {{10+20}}
    </div>
    <div id="div2">
        {{10+20}}
    </div>
    <div id="div3">
        {{10+20}}
    </div>
  </body>
</html>
```

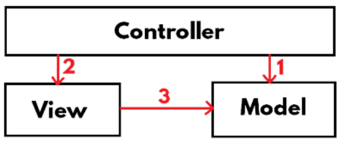
MVC

- MVC stands for "Model – View – Controller".
- "MVC" is an architecture, which allows you to divide the code into three parts:
    1. Model
    2. View
    3. Controller

- **Model:**
    - Model is a JSON object, which stores the data.

- **View:**
  - View is a "set of html tags", which displays model data.
- **Controller:**
  - Controller is a JavaScript function, that loads data into model and passes model to view; and also manipulates model.

MVC Architecture



Principles of MVC

1. Controller calls Model.
2. Controller calls View.
3. View calls Model.

Execution Flow of MVC

1. Model will be created automatically. Model is "an empty javascript object", by default.
2. Controller receives the model as "$scope". Then the controller executes. Controller adds data to model ($scope).
3. After controller's execution is finished, view executes. "Data bindings" will be executed next.

Example on MVC:

```
<html>
   <head>
      <title>Models, Controllers and Views</title>
      <style>
         body
         {
            font-size: 25px;
         }
         #div1,#div2,#div3
         {
            margin: 5px;
            padding: 10px;
            background-color: #0099ff;
            height: 150px;
         }
         #div1 div,#div2 div,#div3 div
         {
            margin: 10px;
            padding: 10px;
            width: 300px;
            float: left;
            background-color: #33cc66;
            height: 90px;
         }
      </style>
      <script src="angular.js"></script>
```

```
    <script>
        var app1 = angular.module("module1", []);
        var app2 = angular.module("module2", []);
        var app3 = angular.module("module3", []);

        app1.controller("c1", fun1);
        function fun1($scope)
        {
            $scope.a = 10;
        }

        app1.controller("c2", fun2);
        function fun2($scope)
        {
            $scope.b = 20;
        }

        app2.controller("c3", fun3);
        function fun3($scope)
        {
            $scope.c = 30;
        }

        app2.controller("c4", fun4);
        function fun4($scope)
        {
            $scope.d = 40;
        }

        app3.controller("c5", fun5);
        function fun5($scope)
        {
            $scope.e = 50;
        }

        app3.controller("c6", fun6);
        function fun6($scope)
        {
            $scope.f = 60;
        }

        angular.element(document).ready(myfunction);
        function myfunction()
        {
            angular.bootstrap( document.getElementById("div1"), ["module1"] );
            angular.bootstrap( document.getElementById("div2"), ["module2"] );
            angular.bootstrap( document.getElementById("div3"), ["module3"] );
        }
    </script>
</head>
<body>
    <h1>Models, Controllers and Views</h1>
    <div id="div1">
        <div ng-controller="c1">{{a}}</div>
        <div ng-controller="c2">{{b}}</div>
        </div>
        <div id="div2">
            <div ng-controller="c3">{{c}}</div>
            <div ng-controller="c4">{{d}}</div>
        </div>
        <div id="div3">
            <div ng-controller="c5">{{e}}</div>
            <div ng-controller="c6">{{f}}</div>
        </div>
    </body>
</html>
```

$rootScope

- $rootScope represents the model, which is accessible for entire module.
- Every module has its own $rootScope.
- $rootScope is commonly accessible commonly in all controllers and views of the same module.

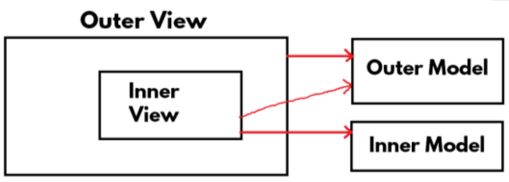
Example on $rootScope

```
<html>
    <head>
        <title>$rootScope</title>
        <style>
            body
            {
                font-size: 25px;
            }
            #div1,#div2,#div3
            {
                margin: 5px;
                padding: 10px;
                background-color: #0099ff;
                height: 150px;
            }
            #div1 div,#div2 div,#div3 div
            {
                margin: 10px;
                padding: 10px;
                width: 400px;
                float: left;
                background-color: #33cc66;
                height: 90px;
            }
        </style>
        <script src="angular.js"></script>

        <script>
            var app1 = angular.module("module1", []);
            var app2 = angular.module("module2", []);
            var app3 = angular.module("module3", []);

            app1.controller("c1", fun1);
            function fun1($scope, $rootScope)
            {
                $scope.a = 10;
                $rootScope.msg = "Message at root scope at module 1";
            }

            app1.controller("c2", fun2);
            function fun2($scope)
            {
                $scope.b = 20;
            }

            app2.controller("c3", fun3);
            function fun3($scope, $rootScope)
            {
                $scope.c = 30;
                $rootScope.msg = "Message at root scope at module 2";
            }

            app2.controller("c4", fun4);
            function fun4($scope)
            {
                $scope.d = 40;
            }

            app3.controller("c5", fun5);
            function fun5($scope, $rootScope)
            {
                $scope.e = 50;
                $rootScope.msg = "Message at root scope at module 3";
            }

            app3.controller("c6", fun6);
            function fun6($scope)
            {
                $scope.f = 60;
            }

            angular.element(document).ready(myfunction);
            function myfunction()
            {
                angular.bootstrap( document.getElementById("div1"), ["module1"] );
                angular.bootstrap( document.getElementById("div2"), ["module2"] );
                angular.bootstrap( document.getElementById("div3"), ["module3"] );
            }
        </script>
    </head>
    <body>
        <h1>$rootScope</h1>
        <div id="div1">
            <div ng-controller="c1">{{a}}<br>{{msg}}</div>
            <div ng-controller="c2">{{b}}<br>{{msg}}</div>
        </div>
        <div id="div2">
            <div ng-controller="c3">{{c}}<br>{{msg}}</div>
            <div ng-controller="c4">{{d}}<br>{{msg}}</div>
        </div>
        <div id="div3">
```

```
            <div ng-controller="c5">{{e}}<br>{{msg}}</div>
            <div ng-controller="c6">{{f}}<br>{{msg}}</div>
        </div>
    </body>
</html>
```

Nested Views

- A view, which is present inside another view is called as "nested view".
- The parent view can be called as "outer view"; and the child view can be called as "inner view".
- The inner view can access the model of "outer view" also.
- When we use an expression in the inner view, first it searches the variable in inner model and then it tries in the outer model.



Example on Nested Views

```
<html>
<head>
    <title>MVC - Nested Views</title>
    <style type="text/css">
        .class1
        {
            background-color: #00ccff;
            height: 1000px;
            font-size: 30px;
            padding: 10px;
            margin: 20px;
        }
        .class2
        {
            background-color: #ff99cc;
            height: 790px;
            font-size: 30px;
            padding: 10px;
            margin: 20px;
        }
        .class3
        {
            background-color: #ccff66;
            height: 430px;
            font-size: 30px;
            padding: 10px;
            margin: 20px;
        }
    </style>

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

    <script>
```

```
        var app = angular.module("mymodule", []);
        app.controller("controller1", fun1);
        app.controller("controller2", fun2);
        app.controller("controller3", fun3);
        function fun1($scope, $rootScope)
        {
            $scope.a = 10;
            $scope.b = 20;
            $rootScope.x = 1000;
            $scope.x = 100;
        }
        function fun2($scope)
        {
            $scope.c = 30;
            $scope.d = 40;
            $scope.x = 200;
        }
        function fun3($scope)
        {
            $scope.e = 50;
            $scope.f = 60;
            $scope.x = 300;
        }
    </script>
</head>
<body>
    <h1>MVC - Nested Views</h1>
    <div ng-app="mymodule">
        <div ng-controller="controller1" class="class1">
            a is: <b>{{a}}</b><br>
            b is: <b>{{b}}</b><br>
            root scope's x is: <b>{{$root.x}}</b><br>
            x is: <b>{{x}}</b>

            <div ng-controller="controller2" class="class2">
                c is: <b>{{c}}</b><br>
                d is: <b>{{d}}</b><br>
                a is: <b>{{a}}</b><br>
                b is: <b>{{b}}</b><br>
                x is: <b>{{x}}</b><br>
                parent x is: <b>{{$parent.x}}</b><br>
                root scope's x is: <b>{{$root.x}}</b><br>

                <div ng-controller="controller3" class="class3">
                    e is: <b>{{e}}</b><br>
                    f is: <b>{{f}}</b><br>
                    c is: <b>{{c}}</b><br>
                    d is: <b>{{d}}</b><br>
                    a is: <b>{{a}}</b><br>
                    b is: <b>{{b}}</b><br>
                    x is: <b>{{x}}</b><br>
                    parent x is: <b>{{$parent.x}}</b><br>
                    parent's parent x is: <b>{{$parent.$parent.x}}</b><br>
                    root scope's x is: <b>{{$root.x}}</b><br>
                </div>
            </div>

        </div>

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

Example on Login

```
<html>
    <head>
        <title>Login</title>
        <script src="angular.js"></script>

        <script>
            //creating module
            var app = angular.module("module1", [] );
            angular.element(document).ready(myfunction);
            function myfunction()
            {
                angular.bootstrap(document.getElementById("div1"), [ "module1" ] );
            }

            //creating controller
            app.controller("mycontroller", fun1);
            function fun1($scope)
            {
                $scope.username = "";
                $scope.password = "";
                $scope.msg = "";

                $scope.login = function()
                {
                    if ($scope.username == "admin" && $scope.password == "manager")
                    {
                        $scope.msg = "Successful login";
                    }
                    else
                    {
                        $scope.msg = "Invalid login";
                    }
                };
            }
        </script>
    </head>
    <body>
        <div id="div1">
            <div ng-controller="mycontroller">
                <form>
                    Username:
                    <input type="text" ng-model="username"><br>
                    Password:
                    <input type="password" ng-model="password">
                    <br>
                    <input type="submit" value="Login" ng-click="login()"><br>
                    {{msg}}
                </form>
            </div>
        </div>
    </body>
</html>
```

Example on Registration

```
<html>
    <head>
        <title>Registration</title>
        <style type="text/css">
            .class1
            {
                color: green;
            }
            .class2
            {
                color: red;
            }
        </style>

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

        <script>
            var app = angular.module("mymodule", []);

            app.controller("mycontroller", fun1);
            function fun1($scope)
            {
                $scope.username = null;
                $scope.password = null;
                $scope.confirmpassword = null;
                $scope.receivenewsletters = false;
                $scope.gender = null;
                $scope.city = null;
                $scope.message = null;
                $scope.classname = null;

                $scope.register = function()
                {
                    if ($scope.password == $scope.confirmpassword)
                    {
                        $scope.message = "Successful registration<br>Username: " + $scope.username + "<br>Password: "
+ $scope.password + "<br>Confirm Password: " + $scope.confirmpassword + "<br>Receive News Letters: " +
this.receivenewsletters + "<br>Gender: " + this.gender + "<br>City: " + this.city;
                        $scope.classname = "class1";
                    }
                    else
                    {
            $scope.message = "Password and confirm password do not match";
                        $scope.classname = "class2";
                    }
                };
            }
        </script>
    </head>
    <body>
        <h1>Register</h1>

        <div ng-app="mymodule">
            <div ng-controller="mycontroller">
                Username:
                <input type="text" ng-model="username"><br>

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

            Confirm Password:
            <input type="password" ng-model="confirmpassword"><br>

            <input type="checkbox" ng-model="receivenewsletters">
            Receive News Letters<br>
            Gender:
            <input type="radio" ng-model="gender" name="gender" value="male">Male
            <input type="radio" ng-model="gender" name="gender" value="female">Female<br>

            City:
            <select ng-model="city">
                <option>Hyderabad</option>
                <option>Pune</option>
                <option>Bangalore</option>
                <option>Mumbai</option>
            </select>
            <br>

            <input type="submit" value="Register" ng-click="register()"><br>
            <span class="{{classname}}">{{message}}</span>
        </div>
    </div>
  </body>
</html>
```

Filters

- Filters are the keywords that can be used in expressions:
  {{variable | filter }}
- Filters are used to format the value of expression.

List of angularjs filters

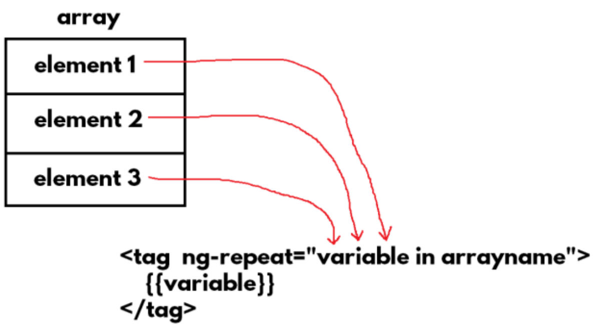
1.	uppercase	{{property | uppercase}}
2.	lowercase	{{property | lowercase}}
3.	number	{{property | number}}
4.	currency	{{property | currency}}
5.	orderBy	{{variable in array | orderBy: 'fieldname' }}
6.	filter	{{variable in array | filter: variable }}

Example on Filters

```
<!DOCTYPE html>
<html>
  <head>
    <title>AngularJS - Filters</title>
    <script src="angular.js"></script>
    <script>
      var app = angular.module("mymodule", [ ]);
      app.controller("mycontroller", fun1);
      function fun1($scope)
      {
        $scope.empid = 101;
        $scope.empname = "Scott";
        $scope.salary = 766784500;
      }
    </script>
  </head>
  <body>
    <div ng-app="mymodule" ng-controller="mycontroller">
      Emp ID: {{empid}}<br>
      Emp Name (uppercase): {{empname | uppercase}}<br>
      Emp Name (lowercase): {{empname | lowercase}}<br>
      Salary (number): {{salary | number}}<br>
      Salary (currency): {{salary | currency}}<br>
      Salary (currency): {{salary | currency:'&#8377;'}}<br>
    </div>
  </body>
</html>
```

ng-repeat

- ng-repeat is used to execute a tag repeatedly for each element of the array.



- **Adding new element into the array:**
  $scope.arrayname.push(newvalue);

- **Removing existing element from array:**
  $scope.arrayname.splice(index, no. of elements to remove);

Example on ng-repeat:

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <title>AngularJS - ng-repeat</title>
        <script src="angular.js"></script>

        <script>
            var app = angular.module("mymodule", [ ] );

            app.controller("mycontroller", fun1);
            function fun1($scope)
            {
                $scope.countries = [ "India", "Japan", "UK", "US" ];
            }
        </script>
    </head>
    <body>
        <h1>ng-repeat</h1>
        <div ng-app="mymodule" ng-controller="mycontroller">

            <ul>
                <li ng-repeat="country in countries">
                    {{country}}
                </li>
            </ul>

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

Example on ng-repeat with add, remove:

```
<html>
    <head>
        <title>AngularJS - Ng-repeat</title>
        <script src="angular.js"></script>

        <script>
            var app = angular.module("mymodule", []);

            app.controller("mycontroller", fun1);
            function fun1($scope)
            {
                $scope.countries = [ "India", "Canada", "China", "Japan", "UK" ];
                $scope.newcountry = "";
                $scope.f1 = function()
                {
                    $scope.countries.push($scope.newcountry);
                    $scope.newcountry = "";
                };
                $scope.f2 = function(n)
                {
                    var countryname = $scope.countries[n];
                    if (confirm("Are you sure to delete " + countryname))
                    {
                        $scope.countries.splice(n, 1);
                    }
                };
            }
        </script>
    </head>
    <body>
        <h1>AngularJS - Ng-repeat</h1>
        <div ng-app="mymodule">
            <div ng-controller="mycontroller">
                <ul>
                    <li ng-repeat="country in countries">
                        {{country}}
                        <input type="button" value="Remove" ng-click="f2($index)">
                    </li>
                </ul>
                <input type="text" placeholder="New Country" ng-model="newcountry">
                <input type="button" value="Add" ng-click="f1()">
            </div>
        </div>
    </body>
</html>
```

Example on ng-repeat with Object array add, remove:

```
<!DOCTYPE html>
<html>
    <head>
        <title>AngularJS - Object array - add, remove</title>
        <script src="angular.js"></script>

        <script>
            var app = angular.module("mymodule", [ ] );
            app.controller("mycontroller", fun1);
            function fun1($scope)
            {
                $scope.emps =
                [
                    { "empid": 101, "empname": "Scott", "salary": 3000 },
                    { "empid": 102, "empname": "Allen", "salary": 6500 },
                    { "empid": 103, "empname": "Jones", "salary": 7500 },
                    { "empid": 104, "empname": "Smith", "salary": 2400 },
                    { "empid": 105, "empname": "James", "salary": 9500 }
                ];

                $scope.newemp = { "empid": "", "empname": "", "salary": "" };

                $scope.save = function()
                {
                    $scope.emps.push({ "empid": $scope.newemp.empid, "empname": $scope.newemp.empname,
"salary": $scope.newemp.salary });

                    $scope.newemp.empid = "";
                    $scope.newemp.empname = "";
                    $scope.newemp.salary = "";
                }

                $scope.remove = function(currentindex)
                {
                    if (confirm("Are you sure to delete this employee") == true)
                    {
                        $scope.emps.splice(currentindex, 1);
                    }
                }
            }
        </script>
    </head>
    <body>
        <div ng-app="mymodule" ng-controller="mycontroller">
            <table border="1" cellpadding="5px">
                <tr>
                    <th>Emp ID</th>
                    <th>Emp Name</th>
                    <th>Salary</th>
                    <th></th>
                </tr>
                <tr ng-repeat="emp in emps">
                    <td>{{emp.empid}}</td>
                    <td>{{emp.empname}}</td>
                    <td>{{emp.salary}}</td>
                    <td>
                        <a href="#" ng-click="remove($index)">
                            Delete
                        </a>
                    </td>
                </tr>
                <tr>
                    <td>
                        <input type="text" id="newempid" ng-model="newemp.empid" autofocus="autofocus">
                    </td>
                    <td>
                        <input type="text" ng-model="newemp.empname">
                    </td>
                    <td>
                        <input type="text" ng-model="newemp.salary">
                    </td>
                    <td>
                        <input type="button" value="Save" ng-click="save()">
                    </td>
                </tr>
            </table>
        </div>
    </body>
</html>
```

Example on ng-repeat with Object Array with Filter and OrderBy:

```
<!DOCTYPE html>
<html>
    <head>
        <title>AngularJS - ng-repeat - json array - table</title>
        <script src="angular.js"></script>

        <script>
            var app = angular.module("mymodule", [ ]);
            app.controller("mycontroller", fun1);
            function fun1($scope)
            {
                $scope.emps =
                [
                    { "empid": 101, "empname": "Scott", "salary": 3000 },
                    { "empid": 102, "empname": "Allen", "salary": 6500 },
                    { "empid": 103, "empname": "Jones", "salary": 7500 },
                    { "empid": 104, "empname": "Smith", "salary": 2400 },
                    { "empid": 105, "empname": "James", "salary": 9500 }
                ];
        $scope.searchdata = "";
            }
        </script>
    </head>
    <body>
        <div ng-app="mymodule" ng-controller="mycontroller">
            Search: <input type="text" ng-model="searchdata.empname">
            <table border="1" cellpadding="5px">
                <tr>
                    <th>Emp ID</th>
                    <th>Emp Name</th>
                    <th>Salary</th>
                </tr>
            <tr ng-repeat="emp in emps | orderBy:'-salary' | filter: searchdata ">
                    <td>{{emp.empid}}</td>
                    <td>{{emp.empname}}</td>
                    <td>{{emp.salary}}</td>
                </tr>
            </table>
        </div>
    </body>
</html>
```

Animations

- We can perform animations while adding / removing items from an array, that is displayed by using "ng-repeat".
- Animation is a process of changing the css property value gradually, based on specified milliseconds.
- The "angular-animate.js" has "ngAnimate" module. To perform animations, we have to import the "ngAnimate" module.
- This module is used to invoke animations dynamically while adding / removing items from array in "ng-repeat".

CSS classes required for animations

- AngularJS automatically applies these css classes while adding / removing elements in the ng-repeat.

- **Adding element:**

.ng-enter	:	Specifies start value.
.ng-enter-active	:	Specifies end value.

- **Removing element:**

.ng-leave	:	Specifies start value.
.ng-leave-active	:	Specifies end value.

Example on Animations:

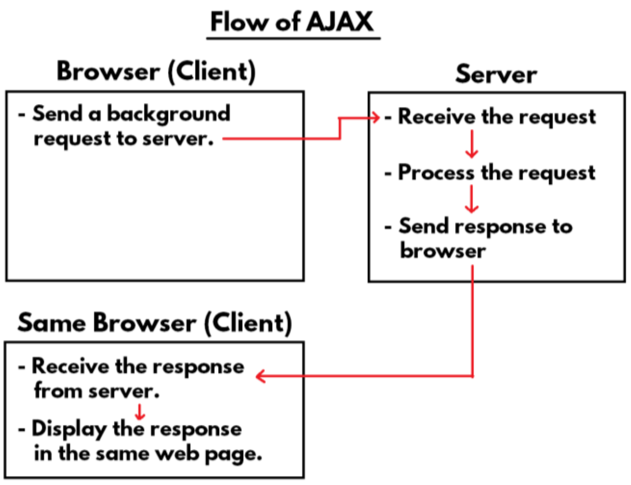
- Place the following files in "c:\angularjs" folder:
- c:\angularjs
    - animations.html
    - angular.js
    - angular-animate.js

Code for animations.html

```
<!DOCTYPE html>
<html>
   <head>
       <title>AngularJS - animations</title>
       <style type="text/css">
           .ng-enter
           {
               transition: 1s;
               opacity: 0;
               transform: translateY(30px);
           }
           .ng-enter-active
           {
               transform: translateY(0px);
               opacity: 1;
           }
           .ng-leave
           {
               transition: 1s;
               opacity: 1;
               transform: translateX(0);
           }
           .ng-leave-active
           {
               opacity: 0;
               transform: translateX(-100px);
           }
       </style>
       <script src="angular.js"></script>
       <script src="angular-animate.js"></script>
       <script>
           var app = angular.module("mymodule", [ "ngAnimate" ] );
           app.controller("mycontroller", fun1);
           function fun1($scope)
           {
               $scope.countries = [ "India", "Canada", "China", "Japan", "UK" ];
               $scope.search = "";
               $scope.newcountry = "";
               $scope.f1 = function()
               {
                   $scope.countries.push($scope.newcountry);
                   $scope.newcountry = "";
               };
               $scope.f2 = function(n)
               {
                    var countryname = $scope.countries[n];
                    if (confirm("Are you sure to delete " + countryname))
                    {
                        $scope.countries.splice(n, 1);
                    }
                };
            }
        </script>
    </head>
    <body>
        <div ng-app="mymodule" ng-controller="mycontroller">
            Seach: <input type="text" ng-model="search">
            <ul>
                <li ng-repeat="country in countries | filter:search">
                    {{country}}
                    <input type="button" value="Remove" ng-click="f2($index)">
                </li>
            </ul>
            <input type="text" placeholder="New Country" ng-model="newcountry">
            <input type="button" value="Add" ng-click="f1()">
        </div>
    </body>
</html>
```

AngularJS AJAX

- AJAX stands for "Asynchronous JavaScript And Xml".
- AJAX is a concept, which is used to "send a background request to server" and "get background response from server", without refreshing the web page.
- **Ex:** Google, Gmail, Facebook, IRCTC etc.


Flow of AJAX
Browser (Client)
- Send a background request to server.
Server
- Receive the request
- Process the request
- Send response to browser
Same Browser (Client)
- Receive the response from server.
- Display the response in the same web page.

AngularJS AJAX

- AngularJS provides "$http" server to perform ajax requests and responses.
- AngularJS AJAX supports the following four types of requests:

GET	:	Retrieving / searching
POST	:	Inserting
PUT	:	Updating

- DELETE : Deleting

1. **GET:**
   - Used to retrieve data from server.
   - Examples:
     - Google search
     - Retrieving students marks sheet based on hall ticket no.
     - Retrieving list of trains based on the source and destination stations in IRCTC website.
   - Syntax: $http.get("url").then(successfunction, errorfunction);

2. **POST:**
   - Used to insert data into database table.
   - Examples:
     - User registration.
   - Syntax: $http.post("url", { data }).then(successfunction, errorfunction);

3. **PUT:**
   - Used to update data in the database table.
   - Examples:
     - Updating user profile.
   - Syntax: $http.put("url", { data }).then(successfunction, errorfunction);

4. **DELETE:**
   - Used to delete data from the database table.
   - Examples:
     - Deleting user profile.
   - Syntax: $http.delete("url").then(successfunction, errorfunction);

AngularJS AJAX - NodeJS - Simple - Example

Creating the application

- Create "c:\angularjs" folder.
- Place "angular.js" file in "c:\angularjs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\ajax" folder.

c:\angularjs
- httpserver.js
- index.html
- angular.js

c:\angularjs\httpserver.js

```
var http = require("http");
var fs = require("fs");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
        if (request.url == "/" || request.url == "/index.html")
        {
            fs.readFile("index.html", "utf8", fun1);
            function fun1(error, data)
            {
                if (error)
                {
                    response.setHeader("content-type", "text/html");
                    response.writeHead(404);
                    response.write("<h1 style='color:red'>Page not found</h1>");
                    response.end();
                }
                else
                {
                    response.setHeader("content-type", "text/html");
                    response.writeHead(200);
                    response.write(data);
                    response.end();
                }
            }
        }
        else if (request.url == "/angular.js")
        {
            fs.readFile("angular.js", "utf8", fun1);
            function fun1(error, data)
            {
                if (error)
                {
                    response.setHeader("content-type", "text/html");
                    response.writeHead(404);
                    response.write("<h1 style='color:red'>Page not found</h1>");
                    response.end();
                }
                else
                {
                    response.setHeader("content-type", "text/javascript");
                    response.writeHead(200);
                    response.write(data);
                    response.end();
                }
            }
        }
        else if (request.url.startsWith("/getdata"))
        {
            console.log("request received at " + new Date().toLocaleTimeString());
            response.setHeader("content-type", "text/html");
            response.writeHead(200);
            response.write("Response from server at " + new Date().toLocaleTimeString());
            response.end();
        }
    }
```

c:\angularjs\index.html

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>AngularJS AJAX - NodeJS - Simple</title>
  <script src="angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.getdata = function ()
      {
        $http.get("/getdata").then(fun2, fun3);
      };
      function fun2(response)
      {
        $scope.message = response.data;
      }
      function fun3(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <h1>AngularJS AJAX - NodeJS - Simple</h1>
    <input type="button" ng-click="getdata()" value="Get data from server">
    <div>{{message}}</div>
  </div>
</body>
</html>
```

Execution:

- Download and install "nodejs" from "https://nodejs.org".
- Open "Command Prompt" and run the following commands:
  ```
  cd c:\angularjs
  node httpserver.js
  ```
- Open browser and enter the url: http://localhost:8080/index.html

AngularJS AJAX - NodeJS - Get - Example

Creating the application

- Create "c:\angularjs" folder.
- Place "angular.js" file in "c:\angularjs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\ajax" folder.

c:\angularjs

- httpserver.js
- index.html
- angular.js

c:\angularjs\httpserver.js

```
var http = require("http");
var fs = require("fs");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/angular.js")
    {
        fs.readFile("angular.js", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/javascript");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
```

```
        else if (request.url.startsWith("/getemployees"))
        {
            console.log("request received at " + new Date().toLocaleTimeString());
            response.setHeader("content-type", "application/json");
            response.writeHead(200);
            response.write(`[
                { "empid": 1, "empname": "Scott", "salary": 4000 },
                { "empid": 2, "empname": "Allen", "salary": 7500 },
                { "empid": 3, "empname": "Jones", "salary": 9200 },
                { "empid": 4, "empname": "James", "salary": 8400 },
                { "empid": 5, "empname": "Smith", "salary": 5600 }
            ]`);
            response.end();
        }
    }
```

c:\angularjs\index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - NodeJS - Get</title>
  <script src="angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.employees= [];
      $scope.f1 = function ()
      {
        $http.get("/getemployees").then(f2, f3);
      };
      function f2(response)
      {
        $scope.employees = response.data;
      }
      function f3(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - NodeJS - Get</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <input type="button" value="Load data" ng-click="f1()">
    <table border="1" id="table1">
      <tr>
        <th>Emp ID</th>
        <th>Emp Name</th>
        <th>Salary</th>
      </tr>
      <tr ng-repeat="emp in employees">
```

```
            <td>{{emp.empid}}</td>
            <td>{{emp.empname}}</td>
            <td>{{emp.salary}}</td>
          </tr>
        </table>
      </div>
    </body>
    </html>
```

Execution:

- Download and install "nodejs" from "https://nodejs.org".
- Open "Command Prompt" and run the following commands:

```
cd c:\angularjs
node httpserver.js
```

- Open browser and enter the url: http://localhost:8080/index.html

AngularJS AJAX - NodeJS - Search - Example

Creating the application

- Create "c:\angularjs" folder.
- Place "angular.js" file in "c:\angularjs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\ajax" folder.

c:\angularjs

- httpserver.js
- index.html
- angular.js

c:\angularjs\httpserver.js

```
var http = require("http");
var fs = require("fs");
var querystring = require("querystring");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
   console.log("Server started at port 8080");
}

function engine(request, response)
{
   if (request.url == "/" || request.url == "/index.html")
   {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
             if (error)
             {
                  response.setHeader("content-type", "text/html");
                  response.writeHead(404);
                  response.write("<h1 style='color:red'>Page not found</h1>");
                  response.end();
```

```
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/angular.js")
    {
        fs.readFile("angular.js", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/javascript");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url.startsWith("/searchemployees"))
    {
        console.log("request received at " + new Date().toLocaleTimeString());
        response.setHeader("content-type", "application/json");
        response.writeHead(200);
        var employees = [
            { "empid": 1, "empname": "Scott", "salary": 4000 },
            { "empid": 2, "empname": "Allen", "salary": 7500 },
            { "empid": 3, "empname": "Jones", "salary": 9200 },
            { "empid": 4, "empname": "James", "salary": 8400 },
            { "empid": 5, "empname": "Smith", "salary": 5600 }
        ];
        var q = querystring.parse((request.url.split('?')[1]));
        var employees2 = [];
        for (var i = 0; i < employees.length; i++)
        {
            if (employees[i].empname.indexOf(q.searchstr) >= 0)
            {
                employees2.push(employees[i]);
            }
        }
        response.write(JSON.stringify(employees2));
        response.end();
    }
```

```
}
```

c:\angularjs\index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - NodeJS - Search</title>
  <script src="angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);

    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
         $scope.str = "";
      $scope.employees = [];
      $scope.getdata = function ()
      {
        $http.get("/searchemployees?searchstr=" + $scope.str).then(f1, f2);
      };
      function f1(response)
      {
        $scope.employees = response.data;
      }
      function f2(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - NodeJS - Search</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <form>
      <input type="text" placeholder="Search" ng-model="str">
      <input type="submit" value="OK" ng-click="getdata()"><br>
                      <table id="table1" border="1">
                           <tr> <th>Emp ID</th> <th>Emp Name</th> <th>Salary</th> </tr>
                           <tr ng-repeat="emp in employees">
                                <td>{{emp.empid}}</td>
                                <td>{{emp.empname}}</td>
                                <td>{{emp.salary}}</td>
                           </tr>
                      </table>
    </form>
  </div>
</body>
</html>
```

Execution:

- Download and install "nodejs" from "https://nodejs.org".
- Open "Command Prompt" and run the following commands:

```
cd c:\angularjs
node httpserver.js
```

- Open browser and enter the url: http://localhost:8080/index.html

AngularJS AJAX - NodeJS - Post - Example

Creating the application

- Create "c:\angularjs" folder.
- Place "angular.js" file in "c:\angularjs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\ajax" folder.

c:\angularjs

- httpserver.js
- index.html
- angular.js

c:\angularjs\httpserver.js

```
var http = require("http");
var fs = require("fs");
var querystring = require("querystring");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/angular.js")
    {
        fs.readFile("angular.js", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/javascript");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url.startsWith("/insertemployee"))
    {
        console.log("request received at " + new Date().toLocaleTimeString());
        response.setHeader("content-type", "text/html");
        response.write("Successfully Inserted");
        response.end();
    }
}
```

c:\angularjs\index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - NodeJS - Post</title>
  <script src="angular.js"></script>
  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.emp = { "empid": null, "empname": null, "salary": null };
      $scope.message = "";

      $scope.savedata = function ()
      {
        $http.post("/insertemployee", $scope.emp).then(f1, f2);
      };
      function f1(response)
      {
        $scope.message = response.data;
      }
      function f2(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - NodeJS - Post</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <form>
      Emp ID: <input type="text" ng-model="emp.empid"><br>
      Emp Name: <input type="text" ng-model="emp.empname"><br>
      Salary: <input type="text" ng-model="emp.salary"><br>
      <input type="submit" value="Insert" ng-click="savedata()">
      <br>{{message}}
    </form>
  </div>
</body>
</html>
```

Execution:

- Download and install "nodejs" from "https://nodejs.org".
- Open "Command Prompt" and run the following commands:

```
cd c:\angularjs
node httpserver.js
```

- Open browser and enter the url: http://localhost:8080/index.html

AngularJS AJAX - NodeJS - Put - Example

Creating the application

- Create "c:\angularjs" folder.
- Place "angular.js" file in "c:\angularjs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\ajax" folder.

c:\angularjs

- httpserver.js
- index.html
- angular.js

c:\angularjs\httpserver.js

```
var http = require("http");
var fs = require("fs");
var querystring = require("querystring");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/angular.js")
    {
        fs.readFile("angular.js", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/javascript");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url.startsWith("/updateemployee"))
    {
        console.log("request received at " + new Date().toLocaleTimeString());
        response.setHeader("content-type", "text/html");
        response.write("Successfully Updated");
        response.end();
    }
}
```

c:\angularjs\index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - NodeJS - Put</title>
  <script src="angular.js"></script>
  <script>
    var app = angular.module("mymodule", []);
```

```
        app.controller("mycontroller", fun1);
        function fun1($scope, $http)
        {
          $scope.emp = { "empid": null, "empname": null, "salary": null };
          $scope.message = "";
          $scope.savedata = function ()
          {
            $http.put("/updateemployee", $scope.emp).then(f1, f2);
          };
          function f1(response)
          {
            $scope.message = response.data;
          }
          function f2(err)
          {
            alert(JSON.stringify(err));
          }
        }
      </script>
    </head>
    <body>
      <h1>AngularJS AJAX - NodeJS - Put</h1>
      <div ng-app="mymodule" ng-controller="mycontroller">
        <form>
          Existing Emp ID: <input type="text" ng-model="emp.empid"><br>
          Emp Name: <input type="text" ng-model="emp.empname"><br>
          Salary: <input type="text" ng-model="emp.salary"><br>
          <input type="submit" value="Update" ng-click="savedata()">
          <br>{{message}}
        </form>
      </div>
    </body>
    </html>
```

Execution:

- Download and install "nodejs" from "https://nodejs.org".
- Open "Command Prompt" and run the following commands:

```
cd c:\angularjs
node httpserver.js
```

- Open browser and enter the url: http://localhost:8080/index.html

AngularJS AJAX - NodeJS - Delete - Example

Creating the application

- Create "c:\angularjs" folder.
- Place "angular.js" file in "c:\angularjs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\ajax" folder.

c:\angularjs

- httpserver.js
- index.html
- angular.js

c:\angularjs\httpserver.js

```
var http = require("http");
var fs = require("fs");
var querystring = require("querystring");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
  console.log("Server started at port 8080");
}

function engine(request, response)
{
  if (request.url == "/" || request.url == "/index.html")
  {
    fs.readFile("index.html", "utf8", fun1);
    function fun1(error, data)
    {
      if (error)
      {
        response.setHeader("content-type", "text/html");
        response.writeHead(404);
        response.write("<h1 style='color:red'>Page not found</h1>");
        response.end();
      }
      else
      {
        response.setHeader("content-type", "text/html");
        response.writeHead(200);
        response.write(data);
        response.end();
      }
    }
  }
     else if (request.url == "/angular.js")
   {
       fs.readFile("angular.js", "utf8", fun1);
       function fun1(error, data)
       {
            if (error)
            {
                 response.setHeader("content-type", "text/html");
                 response.writeHead(404);
                 response.write("<h1 style='color:red'>Page not found</h1>");
                 response.end();
            }
            else
            {
                 response.setHeader("content-type", "text/javascript");
                 response.writeHead(200);
                 response.write(data);
                 response.end();
            }
       }
```

```
        }
        else if (request.url.startsWith("/deleteemployee"))
        {
                console.log("request received at " + new Date().toLocaleTimeString());
                response.setHeader("content-type", "text/html");
                response.write("Successfully Deleted");
                response.end();
        }
    }
```

c:\angularjs\index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - NodeJS - Delete</title>
  <script src="angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.empid = "";
      $scope.message = "";
      $scope.deletedata = function ()
      {
        $http.delete("/deleteemployee?empid=" + $scope.empid).then(f1, f2);
      };
      function f1(response)
      {
        $scope.message = response.data;
      }
      function f2(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - NodeJS - Delete</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <form>
      Emp ID to delete: <input type="text" ng-model="empid">Ex: 1<br>
      <input type="submit" value="Delete" ng-click="deletedata()"><br>
      {{message}}
    </form>
  </div>
</body>
</html>
```

Execution:

- Download and install "nodejs" from "https://nodejs.org".
- Open "Command Prompt" and run the following commands:

```
cd c:\angularjs
node httpserver.js
```

- Open browser and enter the url: http://localhost:8080/index.html

AngularJS AJAX - NodeJS - Grid - Example

Creating the application

- Create "c:\angularjs" folder.
- Place "angular.js" file in "c:\angularjs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\ajax" folder.

c:\angularjs

- httpserver.js
- index.html
- angular.js

c:\angularjs\httpserver.js

```
var http = require("http");
var fs = require("fs");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/angular.js")
    {
        fs.readFile("angular.js", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/javascript");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url.startsWith("/getemployees"))
    {
        console.log("request received at " + new Date().toLocaleTimeString());
        response.setHeader("content-type", "application/json");
        response.writeHead(200);
        response.write(`[
            { "empid": 1, "empname": "Scott", "salary": 4000 },
            { "empid": 2, "empname": "Allen", "salary": 7500 },
            { "empid": 3, "empname": "Jones", "salary": 9200 },
            { "empid": 4, "empname": "James", "salary": 8400 },
            { "empid": 5, "empname": "Smith", "salary": 5600 }
        ]`);
        response.end();
    }
    else if (request.url.startsWith("/insertemployee"))
    {
        console.log("request received at " + new Date().toLocaleTimeString());
        response.setHeader("content-type", "text/html");
        response.write("Successfully Inserted");
        response.end();
    }
    else if (request.url.startsWith("/updateemployee"))
    {
        console.log("request received at " + new Date().toLocaleTimeString());
        response.setHeader("content-type", "text/html");
        response.write("Successfully Updated");
        response.end();
    }
    else if (request.url.startsWith("/deleteemployee"))
    {
            console.log("request received at " + new Date().toLocaleTimeString());
            response.setHeader("content-type", "text/html");
            response.write("Successfully Deleted");
            response.end();
    }
}
```

c:\angularjs\index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS - AJAX - Grid</title>
  <script src="angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);

    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.showloading = true;
      $scope.employees = [];
      $scope.newemployee = { "empid": "", "empname": "", "salary": "" };
      $scope.editemployee = { "empid": "", "empname": "", "salary": "" };

      $http.get("/getemployees").then(fun2, fun3);

      function fun2(response)
      {
        for (var i = 0; i < response.data.length; i++)
        {
          $scope.employees.push({ "empid": response.data[i].empid, "empname": response.data[i].empname, "salary":
response.data[i].salary, "showedit": false });
        }
        $scope.showloading = false;
      }

      function fun3(err)
      {
        alert(JSON.stringify(err));
      }

      $scope.insert = function ()
      {
        if ($scope.newemployee.empid != "" && $scope.newemployee.empname != "" && $scope.newemployee.salary !=
"")
        {
          $http.post("/insertemployee", $scope.newemployee).then(fun4, fun5);
        }

        function fun4(response)
        {
          $scope.employees.push({ "empid": $scope.newemployee.empid, "empname":
$scope.newemployee.empname, "salary": $scope.newemployee.salary });
          $scope.newemployee.empid = "";
          $scope.newemployee.empname = "";
          $scope.newemployee.salary = "";
        }
        function fun5(err)
        {
          alert(JSON.stringify(err));
        }
      };

      $scope.delete = function (i)
      {
        if (confirm("Are you sure to delete?"))
        {
          $http.delete("/deleteemployee?empid=" + $scope.employees[i].empid).then(fun6, fun7);
        }
        function fun6(response)
        {
          $scope.employees.splice(i, 1);
        }
        function fun7(err)
        {
          alert(JSON.stringify(err));
        }
      }

      $scope.edit = function (i)
      {
        $scope.employees[i].showedit = true;
        $scope.editemployee.empid = $scope.employees[i].empid;
        $scope.editemployee.empname = $scope.employees[i].empname;
        $scope.editemployee.salary = $scope.employees[i].salary;
      };

      $scope.update = function (i)
      {
        if ($scope.editemployee.empid != "" && $scope.editemployee.empname != "" && $scope.editemployee.salary !=
"")
        {
          $http.put("/updateemployee", $scope.editemployee).then(fun8, fun9);
        }
        function fun8(response)
        {
          $scope.employees[i].empid = $scope.editemployee.empid;
          $scope.employees[i].empname = $scope.editemployee.empname;
          $scope.employees[i].salary = $scope.editemployee.salary;
          $scope.employees[i].showedit = false;
        }
        function fun9(err)
        {
          alert(JSON.stringify(err));
        }
      };
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - NodeJS - Grid</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <span ng-show="showloading">Loading</span>
    <table border="1" id="table1" cellpadding="5px">
      <tr>
        <th>Emp ID</th>
```

```
          <th>Emp Name</th>
          <th>Salary</th>
          <th>Options</th>
        </tr>
        <tr ng-repeat="emp in employees">
          <td>
            <span ng-show="!emp.showedit">{{emp.empid}}</span>
            <input type="text" ng-model="editemployee.empid" ng-show="emp.showedit" readonly="readonly">
          </td>
          <td>
            <span ng-show="!emp.showedit">{{emp.empname}}</span>
            <input type="text" ng-model="editemployee.empname" ng-show="emp.showedit">
          </td>
          <td>
            <span ng-show="!emp.showedit">{{emp.salary}}</span>
            <input type="text" ng-model="editemployee.salary" ng-show="emp.showedit">
          </td>
          <td>
            <input type="button" value="Edit" ng-click="edit($index)" ng-show="!emp.showedit">
            <input type="button" value="Update" ng-click="update($index)" ng-show="emp.showedit">
            <input type="button" value="Delete" ng-click="delete($index)" ng-show="!emp.showedit">
          </td>
        </tr>
        <tr class="newrow">
          <td><input type="text" ng-model="newemployee.empid" autofocus="autofocus"></td>
          <td><input type="text" ng-model="newemployee.empname"></td>
          <td><input type="text" ng-model="newemployee.salary"></td>
          <td><input type="button" value="Insert" ng-click="insert()"></td>
        </tr>
      </table>
    </div>
  </body>
</html>
```

Execution:

- Download and install "nodejs" from "https://nodejs.org".
- Open "Command Prompt" and run the following commands:

```
cd c:\angularjs
node httpserver.js
```

- Open browser and enter the url: http://localhost:8080/index.html

AngularJS AJAX - .NET - Simple - Example

Creating the application

- Open Visual Studio 2017.
- Click on "File" menu – "New" – "Project".
- Select ".NET Framework 4.6".
- Select "Visual C#".
- Select "ASP.NET Web Application (.NET Framework)".
- Name: AjaxSimple
- Location: c:\angularjs
- Solution name: AjaxSimple

- Click on OK.
- Click on "Empty".
- Check the check boxes "MVC" and "Web API".
- Click on OK.
- Open Solution Explorer.
- Place "angular.js" file in the project.
- Right click on the "Controllers" folder and click on "Add" – "Controller". Select "MVC 5 Controller – Empty". Click on "Add". Enter the controller name as "HomeController". Click on "Add".
- Right click on "Views\Home" folder and click on "Add" – "View". Enter the view name as "Index". Select the template as "Empty (without model)". Uncheck the checkbox "Use a layout page". Click on "Add".

AjaxSimple

- angular.js
    - Models
        - Employee.cs
    - Controllers
        - HomeController.cs
    - Views
        - Home
            - Index.cshtml

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace AjaxSimple.Controllers
{
  public class HomeController : Controller
  {
    public ActionResult Index()
    {
      return View();
    }
    public ActionResult GetData()
    {
      return Content("Hello from Server at " + DateTime.Now);
    }
  }
}
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - .NET - Simple</title>
  <script src="~/angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
```

```
      function fun1($scope, $http)
      {
        $scope.getdata = function ()
        {
          $http.get("/home/getdata").then(fun2, fun3);
        };
        function fun2(response)
        {
          $scope.message = response.data;
        }
        function fun3(err)
        {
          alert(JSON.stringify(err));
        }
      }
    </script>
  </head>
  <body>
    <div ng-app="mymodule" ng-controller="mycontroller">
      <h1>AngularJS AJAX - .NET - Simple</h1>
      <input type="button" ng-click="getdata()" value="Get data from server">
      <div>{{message}}</div>
    </div>
  </body>
  </html>
```

Execution:

- Go to "Debug" menu – "Start Debugging" in Visual Studio.

AngularJS AJAX - .NET - Get - Example

Creating the application

- Open Visual Studio 2017.
- Click on "File" menu – "New" – "Project".
- Select ".NET Framework 4.6".
- Select "Visual C#".
- Select "ASP.NET Web Application (.NET Framework)".
- Name: AjaxGet
- Location: c:\angularjs
- Solution name: AjaxGet
- Click on OK.
- Click on "Empty".
- Check the check boxes "MVC" and "Web API".
- Click on OK.
- Open Solution Explorer.
- Place "angular.js" file in the project.
- Right click on "Models" and click on "Add" – "New Item". Click on "Visual C#" – "Class". Enter the name as "Employee.cs". Click on "Add".
- Right click on the "Controllers" folder and click on "Add" – "Controller". Select "MVC 5 Controller – Empty". Click on "Add". Enter the controller name as "HomeController". Click on "Add".

- Right click on "Views\Home" folder and click on "Add" – "View". Enter the view name as "Index". Select the template as "Empty (without model)". Uncheck the checkbox "Use a layout page". Click on "Add".

AjaxGet

- angular.js
    - Models
        - Employee.cs
    - Controllers
        - HomeController.cs
    - Views
        - Home
            - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxGet.Models
{
  public class Employee
  {
    public int empid { get; set; }
    public string empname { get; set; }
    public double salary { get; set; }
  }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxGet.Models;

namespace AjaxGet.Controllers
{
  public class HomeController : Controller
  {
    public ActionResult Index()
    {
      return View();
    }
    public ActionResult GetEmployees()
    {
      List<Employee> emps = new List<Employee>()
      {
        new Employee() { empid = 1, empname = "Scott", salary = 4000 },
        new Employee() { empid = 2, empname = "Allen", salary = 7500 },
        new Employee() { empid = 3, empname = "Jones", salary = 9200 },
        new Employee() { empid = 4, empname = "James", salary = 8400 },
        new Employee() { empid = 5, empname = "Smith", salary = 5600 }
      };
      return Json(emps, JsonRequestBehavior.AllowGet);
    }
  }
}
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - .NET - Get</title>
  <script src="~/angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.employees= [];
      $scope.f1 = function ()
      {
        $http.get("/home/getemployees").then(f2, f3);
      };
      function f2(response)
      {
        $scope.employees = response.data;
      }
      function f3(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - .NET - Get</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <input type="button" value="Load data" ng-click="f1()">
    <table border="1" id="table1">
      <tr>
        <th>Emp ID</th>
        <th>Emp Name</th>
        <th>Salary</th>
      </tr>
      <tr ng-repeat="emp in employees">
        <td>{{emp.empid}}</td>
        <td>{{emp.empname}}</td>
        <td>{{emp.salary}}</td>
      </tr>
    </table>
  </div>
</body>
</html>
```

Execution:

- Go to "Debug" menu – "Start Debugging" in Visual Studio.

AngularJS AJAX - .NET - Search - Example

Creating the application

- Open Visual Studio 2017.
- Click on "File" menu – "New" – "Project".
- Select ".NET Framework 4.6".
- Select "Visual C#".
- Select "ASP.NET Web Application (.NET Framework)".
- Name: AjaxSearch
- Location: c:\angularjs
- Solution name: AjaxSearch
- Click on OK.
- Click on "Empty".
- Check the check boxes "MVC" and "Web API".
- Click on OK.
- Open Solution Explorer.
- Place "angular.js" file in the project.
- Right click on "Models" and click on "Add" – "New Item". Click on "Visual C#" – "Class". Enter the name as "Employee.cs". Click on "Add".
- Right click on the "Controllers" folder and click on "Add" – "Controller". Select "MVC 5 Controller – Empty". Click on "Add". Enter the controller name as "HomeController". Click on "Add".
- Right click on "Views\Home" folder and click on "Add" – "View". Enter the view name as "Index". Select the template as "Empty (without model)". Uncheck the checkbox "Use a layout page". Click on "Add".

AjaxSearch

- angular.js
    - Models
        - Employee.cs
    - Controllers
        - HomeController.cs
    - Views
        - Home
            - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxSearch.Models
{
  public class Employee
  {
    public int empid { get; set; }
    public string empname { get; set; }
    public double salary { get; set; }
  }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxSearch.Models;

namespace AjaxSearch.Controllers
{
  public class HomeController : Controller
  {
    // GET: Home/Index
    public ActionResult Index()
    {
      return View();
    }

    // GET: Home/searchemployees
    public ActionResult searchemployees(string searchstr)
    {
      List<Employee> emps = new List<Employee>()
      {
        new Employee() { empid = 1, empname = "Scott", salary = 4000 },
        new Employee() { empid = 2, empname = "Allen", salary = 7500 },
        new Employee() { empid = 3, empname = "Jones", salary = 9200 },
        new Employee() { empid = 4, empname = "James", salary = 8400 },
        new Employee() { empid = 5, empname = "Smith", salary = 5600 }
      };

      List<Employee> emps2 = new List<Employee>();
      for (int i = 0; i < emps.Count; i++)
      {
        if (emps[i].empname.Contains(searchstr))
        {
          emps2.Add(emps[i]);
        }
      }
      return Json(emps2, JsonRequestBehavior.AllowGet);
    }
  }
}
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - .NET - Search</title>
  <script src="~/angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.str = "";
      $scope.employees = [];
      $scope.getdata = function ()
      {
        $http.get("/home/searchemployees?searchstr=" + $scope.str).then(f1, f2);
      };
      function f1(response)
      {
        $scope.employees = response.data;
      }
      function f2(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - .NET - Search</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <form>
      <input type="text" placeholder="Search" ng-model="str">
      <input type="submit" value="OK" ng-click="getdata()"><br>
      <table id="table1" border="1">
        <tr> <th>Emp ID</th> <th>Emp Name</th> <th>Salary</th> </tr>
        <tr ng-repeat="emp in employees">
          <td>{{emp.empid}}</td>
          <td>{{emp.empname}}</td>
          <td>{{emp.salary}}</td>
        </tr>
      </table>
    </form>
  </div>
</body>
</html>
```

Execution:

- Go to "Debug" menu – "Start Debugging" in Visual Studio.

AngularJS AJAX - .NET - Post - Example

Creating the application

- Open Visual Studio 2017.
- Click on "File" menu – "New" – "Project".
- Select ".NET Framework 4.6".
- Select "Visual C#".
- Select "ASP.NET Web Application (.NET Framework)".
- Name: AjaxPost
- Location: c:\angularjs
- Solution name: AjaxPost
- Click on OK.
- Click on "Empty".
- Check the check boxes "MVC" and "Web API".
- Click on OK.
- Open Solution Explorer.

- Place "angular.js" file in the project.
- Right click on "Models" and click on "Add" – "New Item". Click on "Visual C#" – "Class". Enter the name as "Employee.cs". Click on "Add".
- Right click on the "Controllers" folder and click on "Add" – "Controller". Select "MVC 5 Controller – Empty". Click on "Add". Enter the controller name as "HomeController". Click on "Add".
- Right click on "Views\Home" folder and click on "Add" – "View". Enter the view name as "Index". Select the template as "Empty (without model)". Uncheck the checkbox "Use a layout page". Click on "Add".

AjaxPost

- angular.js
    - Models
        - Employee.cs
    - Controllers
        - HomeController.cs
    - Views
        - Home
            - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxPost.Models
{
  public class Employee
  {
    public int empid { get; set; }
    public string empname { get; set; }
    public double salary { get; set; }
  }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxPost.Models;

namespace AjaxPost.Controllers
{
  public class HomeController : Controller
  {
    // GET: Home/Index
    public ActionResult Index()
    {
      return View();
    }

    // GET: Home/InsertEmployee
    public ActionResult InsertEmployee(Employee emp)
    {
      return Content("Successfuly Inserted");
    }
    }
  }
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - .NET - Post</title>
  <script src="~/angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);

    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.emp = { "empid": null, "empname": null, "salary": null };
      $scope.message = "";

      $scope.savedata = function ()
      {
        $http.post("/home/insertemployee", $scope.emp).then(f1, f2);
      };

      function f1(response)
      {
        $scope.message = response.data;
      }

      function f2(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - .NET - Post</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <form>
      Emp ID: <input type="text" ng-model="emp.empid"><br>
      Emp Name: <input type="text" ng-model="emp.empname"><br>
      Salary: <input type="text" ng-model="emp.salary"><br>
      <input type="submit" value="Insert" ng-click="savedata()">
      <br>{{message}}
    </form>
  </div>
</body>
</html>
```

Execution:

- Go to "Debug" menu – "Start Debugging" in Visual Studio.

AngularJS AJAX - .NET - Put - Example

Creating the application

- Open Visual Studio 2017.
- Click on "File" menu – "New" – "Project".
- Select ".NET Framework 4.6".
- Select "Visual C#".
- Select "ASP.NET Web Application (.NET Framework)".
- Name: AjaxPut
- Location: c:\angularjs
- Solution name: AjaxPut
- Click on OK.
- Click on "Empty".
- Check the check boxes "MVC" and "Web API".
- Click on OK.
- Open Solution Explorer.
- Place "angular.js" file in the project.
- Right click on "Models" and click on "Add" – "New Item". Click on "Visual C#" – "Class". Enter the name as "Employee.cs". Click on "Add".
- Right click on the "Controllers" folder and click on "Add" – "Controller". Select "MVC 5 Controller – Empty". Click on "Add". Enter the controller name as "HomeController". Click on "Add".
- Right click on "Views\Home" folder and click on "Add" – "View". Enter the view name as "Index". Select the template as "Empty (without model)". Uncheck the checkbox "Use a layout page". Click on "Add".

AjaxPut

- angular.js
    - Models
        - Employee.cs
    - Controllers
        - HomeController.cs
    - Views
        - Home
            - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxPut.Models
{
  public class Employee
  {
    public int empid { get; set; }
    public string empname { get; set; }
    public double salary { get; set; }
  }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxPut.Models;

namespace AjaxPut.Controllers
{
  public class HomeController : Controller
  {
    // GET: Home/Index
    public ActionResult Index()
    {
      return View();
    }

    // GET: Home/UpdateEmployee
    public ActionResult UpdateEmployee(Employee emp)
    {
      return Content("Successfuly Updated");
    }
  }
}
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - .NET - Put</title>
  <script src="~/angular.js"></script>
  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.emp = { "empid": null, "empname": null, "salary": null };
      $scope.message = "";
      $scope.savedata = function ()
      {
        $http.put("/home/updateemployee", $scope.emp).then(f1, f2);
      };
      function f1(response)
      {
        $scope.message = response.data;
      }
      function f2(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - .NET - Put</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <form>
      Existing Emp ID: <input type="text" ng-model="emp.empid"><br>
```

```
        Emp Name: <input type="text" ng-model="emp.empname"><br>
        Salary: <input type="text" ng-model="emp.salary"><br>
        <input type="submit" value="Update" ng-click="savedata()">
        <br>{{message}}
      </form>
    </div>
  </body>
  </html>
```

Execution:

- Go to "Debug" menu – "Start Debugging" in Visual Studio.

AngularJS AJAX - .NET - Delete - Example

Creating the application

- Open Visual Studio 2017.
- Click on "File" menu – "New" – "Project".
- Select ".NET Framework 4.6".
- Select "Visual C#".
- Select "ASP.NET Web Application (.NET Framework)".
- Name: AjaxDelete
- Location: c:\angularjs
- Solution name: AjaxDelete
- Click on OK.
- Click on "Empty".
- Check the check boxes "MVC" and "Web API".
- Click on OK.
- Open Solution Explorer.
- Place "angular.js" file in the project.
- Right click on the "Controllers" folder and click on "Add" – "Controller". Select "MVC 5 Controller – Empty". Click on "Add". Enter the controller name as "HomeController". Click on "Add".
- Right click on "Views\Home" folder and click on "Add" – "View". Enter the view name as "Index". Select the template as "Empty (without model)". Uncheck the checkbox "Use a layout page". Click on "Add".

AjaxDelete

- angular.js
    - Controllers
        - HomeController.cs
    - Views
        - Home
            - Index.cshtml

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxDelete.Models;

namespace AjaxDelete.Controllers
```

```
{
  public class HomeController : Controller
  {
    // GET: Home/Index
    public ActionResult Index()
    {
      return View();
    }

    // GET: Home/DeleteEmployee
    public ActionResult DeleteEmployee(int empid)
    {
      return Content("Successfuly Deleted");
    }
  }
}
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS AJAX - .NET - Delete</title>
  <script src="~/angular.js"></script>

  <script>
    var app = angular.module("mymodule", []);
    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.empid = "";
      $scope.message = "";
      $scope.deletedata = function ()
      {
        $http.delete("/home/deleteemployee?empid=" + $scope.empid).then(f1, f2);
      };
      function f1(response)
      {
        $scope.message = response.data;
      }
      function f2(err)
      {
        alert(JSON.stringify(err));
      }
    }
  </script>
</head>
<body>
  <h1>AngularJS AJAX - .NET - Delete</h1>
  <div ng-app="mymodule" ng-controller="mycontroller">
    <form>
      Emp ID to delete: <input type="text" ng-model="empid">Ex: 1<br>
      <input type="submit" value="Delete" ng-click="deletedata()"><br>
      {{message}}
    </form>
  </div>
</body>
</html>
```

Execution:

- Go to "Debug" menu – "Start Debugging" in Visual Studio.

AngularJS AJAX - .NET - Grid - Example

Creating the application

- Open Visual Studio 2017.
- Click on "File" menu – "New" – "Project".
- Select ".NET Framework 4.6".
- Select "Visual C#".
- Select "ASP.NET Web Application (.NET Framework)".
- Name: AjaxGrid
- Location: c:\angularjs
- Solution name: AjaxGrid
- Click on OK.
- Click on "Empty".
- Check the check boxes "MVC" and "Web API".
- Click on OK.
- Open Solution Explorer.
- Place "angular.js" file in the project.
- Right click on "Models" and click on "Add" – "New Item". Click on "Visual C#" – "Class". Enter the name as "Employee.cs". Click on "Add".
- Right click on the "Controllers" folder and click on "Add" – "Controller". Select "MVC 5 Controller – Empty". Click on "Add". Enter the controller name as "HomeController". Click on "Add".
- Right click on "Views\Home" folder and click on "Add" – "View". Enter the view name as "Index". Select the template as "Empty (without model)". Uncheck the checkbox "Use a layout page". Click on "Add".

AjaxGrid

- angular.js
  - Models
    - Employee.cs
  - Controllers
    - HomeController.cs
  - Views
    - Home
      - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxGrid.Models
{
  public class Employee
  {
    public int empid { get; set; }
    public string empname { get; set; }
```

```
        public double salary { get; set; }
    }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxGrid.Models;

namespace AjaxGrid.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult GetEmployees()
        {
            List<Employee> emps = new List<Employee>()
            {
                new Employee() { empid = 1, empname = "Scott", salary = 4000 },
                new Employee() { empid = 2, empname = "Allen", salary = 7500 },
                new Employee() { empid = 3, empname = "Jones", salary = 9200 },
                new Employee() { empid = 4, empname = "James", salary = 8400 },
                new Employee() { empid = 5, empname = "Smith", salary = 5600 }
            };
            return Json(emps, JsonRequestBehavior.AllowGet);
        }

        public ActionResult InsertEmployee(Employee emp)
        {
            return Content("Successfully Inserted");
        }

        public ActionResult UpdateEmployee(Employee emp)
        {
            return Content("Successfully Updated");
        }

        public ActionResult DeleteEmployee(int empid)
        {
            return Content("Successfully Deleted");
        }
    }
}
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <title>AngularJS AJAX - .NET - Grid</title>
    <script src="~/angular.js"></script>
```

```
  <script>
    var app = angular.module("mymodule", []);

    app.controller("mycontroller", fun1);
    function fun1($scope, $http)
    {
      $scope.showloading = true;
      $scope.employees = [];
      $scope.newemployee = { "empid": "", "empname": "", "salary": "" };
      $scope.editemployee = { "empid": "", "empname": "", "salary": "" };

      $http.get("/home/getemployees").then(fun2, fun3);

      function fun2(response)
      {
        for (var i = 0; i < response.data.length; i++)
        {
          $scope.employees.push({ "empid": response.data[i].empid, "empname": response.data[i].empname, "salary":
response.data[i].salary, "showedit": false });
        }
        $scope.showloading = false;
      }

      function fun3(err)
      {
        alert(JSON.stringify(err));
      }

      $scope.insert = function ()
      {
        if ($scope.newemployee.empid != "" && $scope.newemployee.empname != "" && $scope.newemployee.salary !=
"")
        {
          $http.post("/home/insertemployee", $scope.newemployee).then(fun4, fun5);
        }

        function fun4(response)
        {
          $scope.employees.push({ "empid": $scope.newemployee.empid, "empname":
$scope.newemployee.empname, "salary": $scope.newemployee.salary });
          $scope.newemployee.empid = "";
          $scope.newemployee.empname = "";
          $scope.newemployee.salary = "";
        }
        function fun5(err)
        {
          alert(JSON.stringify(err));
        }
      };

      $scope.delete = function (i)
      {
        if (confirm("Are you sure to delete?"))
        {
```

```
      $http.delete("/home/deleteemployee?empid=" + $scope.employees[i].empid).then(fun6, fun7);
    }
    function fun6(response)
    {
      $scope.employees.splice(i, 1);
    }
    function fun7(err)
    {
      alert(JSON.stringify(err));
    }
  }

  $scope.edit = function (i)
  {
    $scope.employees[i].showedit = true;
    $scope.editemployee.empid = $scope.employees[i].empid;
    $scope.editemployee.empname = $scope.employees[i].empname;
    $scope.editemployee.salary = $scope.employees[i].salary;
  };

  $scope.update = function (i)
  {
    if ($scope.editemployee.empid != "" && $scope.editemployee.empname != "" && $scope.editemployee.salary !=
"")
    {
      $http.put("/home/updateemployee", $scope.editemployee).then(fun8, fun9);
    }
    function fun8(response)
    {
      $scope.employees[i].empid = $scope.editemployee.empid;
      $scope.employees[i].empname = $scope.editemployee.empname;
      $scope.employees[i].salary = $scope.editemployee.salary;
      $scope.employees[i].showedit = false;
    }
    function fun9(err)
    {
      alert(JSON.stringify(err));
    }
  };
 }
 </script>
</head>
<body>
 <h1>AngularJS AJAX - .NET - Grid</h1>
 <div ng-app="mymodule" ng-controller="mycontroller">
   <span ng-show="showloading">Loading</span>
   <table border="1" id="table1" cellpadding="5px">
     <tr>
       <th>Emp ID</th>
       <th>Emp Name</th>
       <th>Salary</th>
       <th>Options</th>
     </tr>
     <tr ng-repeat="emp in employees">
       <td>
```

```
          <span ng-show="!emp.showedit">{{emp.empid}}</span>
          <input type="text" ng-model="editemployee.empid" ng-show="emp.showedit" readonly="readonly">
        </td>
        <td>
          <span ng-show="!emp.showedit">{{emp.empname}}</span>
          <input type="text" ng-model="editemployee.empname" ng-show="emp.showedit">
        </td>
        <td>
          <span ng-show="!emp.showedit">{{emp.salary}}</span>
          <input type="text" ng-model="editemployee.salary" ng-show="emp.showedit">
        </td>
        <td>
          <input type="button" value="Edit" ng-click="edit($index)" ng-show="!emp.showedit">
          <input type="button" value="Update" ng-click="update($index)" ng-show="emp.showedit">
          <input type="button" value="Delete" ng-click="delete($index)" ng-show="!emp.showedit">
        </td>
      </tr>
      <tr class="newrow">
        <td><input type="text" ng-model="newemployee.empid" autofocus="autofocus"></td>
        <td><input type="text" ng-model="newemployee.empname"></td>
        <td><input type="text" ng-model="newemployee.salary"></td>
        <td><input type="button" value="Insert" ng-click="insert()"></td>
      </tr>
    </table>
  </div>
</body>
</html>
```

Execution:

- Go to "Debug" menu – "Start Debugging" in Visual Studio.

Validations

- Validation is a process of checking whether the input values are correct or not.
- In angularjs, we create validation rules using HTML 5; and we use AngularJS – specific variables to show / hide error messages.

HTML 5 – Validation Rules:

1. required="required"
2. min="minimum"
3. max="maximum"
4. pattern="regular expression"
5. type="number"
6. type="email"

etc.

List of AngularJS Variables to Show / Hide Error Messages

1. $untouched
2. $touched
3. $prestine
4. $dirty

5. $invalid
6. $valid
7. $error

1. $untouched

- Returns a Boolean value that indicates whether the user has touched (cursor entered in) the element or not.
- **True:** The user didn't touched the element.
- **False:** The user has touched the element.

2. $touched

- Returns a Boolean value that indicates whether the user has touched (cursor entered in) the element or not.
- **True:** The user has touched the element.
- **False:** The user didn't touched the element.

3. $prestine

- Returns a Boolean value that indicates whether the user has modified the value of the element or not.
- **True:** The user has not modified the value.
- **False:** The user has modified the value.

4. $dirty

- Returns a Boolean value that indicates whether the user has modified the value of the element or not.
- **True:** The user has modified the value.
- **False:** The user has not modified the value.

5. $invalid

- Returns a Boolean value that the value is invalid or not.
- **True:** The value is invalid.
- **False:** The value is valid.

6. $valid

- Returns a Boolean value that the value is valid or not.
- **True:** The value is valid.
- **False:** The value is invalid.

7. $error

- Returns a Boolean value that the value is invalid because of specific type of validation rule or not.
- **$error.required:** true / false
- **$error.min:** true / false
- **$error.max:** true / false
- **$error.pattern:** true / false
- **$error.number:** true / false
- **$error.email:** true / false

etc.

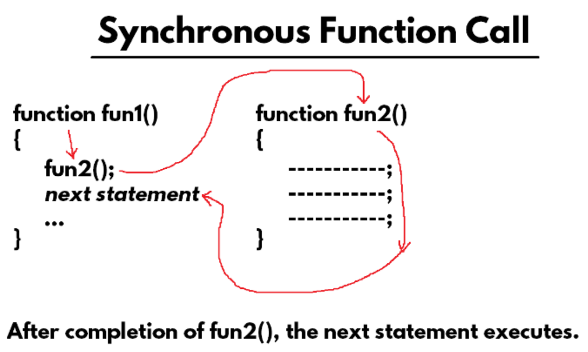
Example on Validations

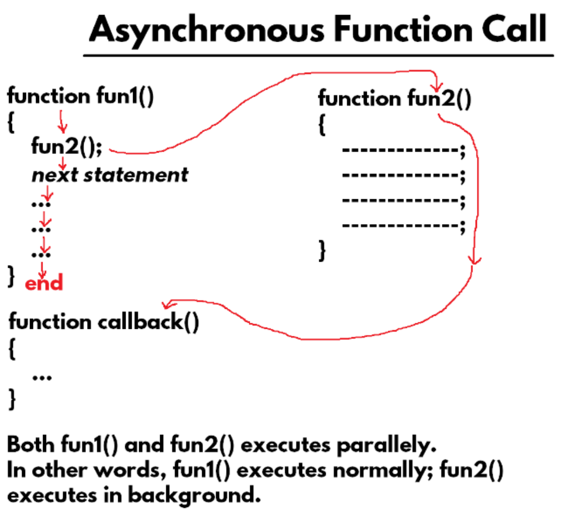
```
<html>
 <head>
  <title>AngularJS - Validations</title>
  <script src="angular.js"></script>
  <script>
   var app = angular.module("mymodule", []);
   app.controller("mycontroller", fun1);
   function fun1($scope)
   {
    $scope.user = { "username": "", "email": "", "amount": "" };
   }
  </script>
 </head>
 <body>
  <div ng-app="mymodule">
   <div ng-controller="mycontroller">
    <form action="http://localhost/someaddress" novalidate="novalidate" name="f1">

     Name *:
     <input type="text" ng-model="user.username" name="username" required="required" pattern="^[a-zA-Z ]*$">
     <span style="color:red" ng-show="f1.username.$invalid && f1.username.$dirty">
      <span ng-show="f1.username.$error.required">Name can't be blank</span>
      <span ng-show="f1.username.$error.pattern">Name must have alphabets only</span>
     </span>
     <br>

     Email *: <input type="email" ng-model="user.email" name="email" required="required">
     <span style="color:red" ng-show="f1.email.$invalid && f1.email.$dirty">
      <span ng-show="f1.email.$error.required">Email can't be blank</span>
      <span ng-show="f1.email.$error.email">Invalid email</span>
     </span>
     <br>

     Amount *: <input type="number" ng-model="user.amount" name="amount" required="required" min="1000"
max="10000">
     <span style="color:red" ng-show="f1.amount.$invalid && f1.amount.$dirty">
      <span ng-show="f1.amount.$error.required">Amount can't be blank</span>
      <span ng-show="f1.amount.$error.number">Invalid number</span>
      <span ng-show="f1.amount.$error.min">Min: 1000</span>
      <span ng-show="f1.amount.$error.max">Max: 10000</span>
     </span>
     <br>

     <input type="submit" value="Submit" ng-disabled="f1.username.$invalid || f1.email.$invalid ||
f1.amount.$invalid">

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

$q

- $q is one of the pre-defined services in angularjs, which is used to execute the code asynchronously (in background).

  Ex: large amount of code, long loops, loading many records, attaching files etc.





Steps for development of $q

- **Get $q in controller & call async function:**

```
app.controller("controller name", controllerfunctionname);
function controllerfunctionname($q)
{
   var promise = asyncfunction($q);
}
```

- **Create the asynchronous function:**

```
function asyncfunction($q)
{
   var deferred = $q.defer( ); //Specifies that, this is async function.
   deferred.notify( ); //Passes a notification to the calling portion. It invokes "notification callback" function.
   deferred.resolve( ); //Finishes the async function successfully. It invokes "success callback" function.
   return deferred.promise; //Returns a promise to the calling portion.
}
```

- **Register the callback function:**
  promise.then(successcallback, errorcallback, notifycallback);

  Success callback: Executes when the async function resolved.
  Error callback: Executes when there is some exception while executing the async function.
  Notify callback: Executes when the async function passes a notification to the calling portion.

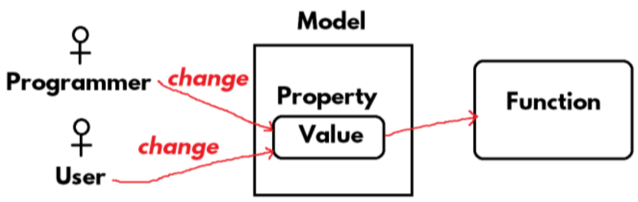
Example on $q

```
<html>
   <head>
      <title>AngularJS - $q</title>
      <style>
         body,input
         {
            font-family: Tahoma;
            font-size: 30px;
         }
         #span1
         {
            color: green;
         }
      </style>
      <script src="angular.js"></script>
      <script>
         function myfunction($q, x, y)
         {
            var deferred = $q.defer();
            setTimeout(f1, 100);
            setTimeout(f2, 3000);
            setTimeout(f3, 6000);
            function f1()
            {
               deferred.notify("Work started.... please wait...!!");
            }
            function f2()
            {
               deferred.notify("Thanx for your patience... Please wait more time....!!");
            }
            function f3()
            {
               var z = parseInt(x) + parseInt(y);
               deferred.resolve(z);
            }
            return deferred.promise;
         }

         var app = angular.module("mymodule", [ ]);
         app.controller("mycontroller", fun1);
         function fun1($scope, $q)
         {
            $scope.a = "";
            $scope.b = "";
            $scope.c = "";
```

```
                $scope.add = function()
                {
                    var promise = myfunction($q, $scope.a, $scope.b);
                    promise.then(x, y, z);
                    function x(data)
                    {
                        $scope.c = data;
                        $scope.message = "";
                    }
                    function y(error)
                    {
                        alert(error);
                    }
                    function z(data)
                    {
                        $scope.message = data;
                    }
                };
            }
        </script>
    </head>
    <body>
        <div ng-app="mymodule", ng-controller="mycontroller">
            Enter first numer: <input type="text" ng-model="a"><br>
            Enter second numer: <input type="text" ng-model="b"><br>
            <input type="button" value="Add" ng-click="add()">
            <span id="span1">{{message}}</span>
            <br>
            Result: <input type="text" ng-model="c" readonly="readonly">
        </div>
    </body>
</html>
```

$watch

- "$watch" is used to observe a model property & call a function automatically, every time the property's value gets changed by the programmer / user.



Syntax:

```
$scope.$watch(f1, f2);
function f1(scope)
{
    return scope.property;
}
function f2(newvalue, oldvalue)
{
    //do something with newvalue, oldvalue
    }
```

Example on $watch:

```
<html>
 <head>
  <title>AngularJS - $watch</title>
  <script src="angular.js"></script>

  <script>
   var app = angular.module("mymodule", []);
   app.controller("mycontroller", fun1);
   function fun1($scope)
   {
    $scope.x = "hello";
    $scope.$watch(f1, f2);
    function f1(scope)
    {
     return scope.x;
    }
    function f2(newvalue, oldvalue)
    {
     console.log(newvalue + ", " + oldvalue);
    }
   }
  </script>
 </head>
 <body>
  <h1>$watch</h1>
  <div ng-app="mymodule">
   <div ng-controller="mycontroller">
     <input type="text" ng-model="x"><br>
     <input type="text" ng-model="x"><br>
     <input type="text" ng-model="x"><br>
     {{x}}
   </div>
  </div>
 </body>
</html>
```

Dependency Injection

- It is used to specify the list of parameters that are required to run a controller / component.
- Advantage: Even though the javascript is minified, the parameters work correctly.

Syntax:

```
app.controller("controllername", [ "parameter1", "parameter2", "parameter3", …, functionname ] );

function functionname(parameter1, parameter2, …)
{
   code here
}
```

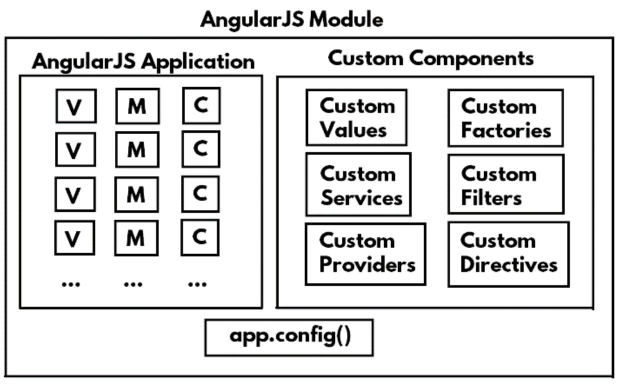
Example on Dependency Injection

```
<html>
   <head>
       <title>AngularJS - Dependency Injection</title>
       <style>
           body
           {
               font-family: Tahoma;
               font-size: 30px;
           }
           .class1
           {
               margin: 10px;
               padding: 10px;
               border: 5px solid red;
           }
       </style>

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

       <script>
           var app = angular.module("mymodule", [ ] );
           app.controller("controller1", [ "$scope", "$rootScope", fun1 ]);
           function fun1(s, r)
           {
               s.submessage = "message from sub model"; //s = $scope
               r.rootmessage = "message from root model"; //r = $rootScope
           }
       </script>
   </head>
   <body>
       <div ng-app="mymodule" class="class1">
           <p>{{rootmessage}}</p>
           <div ng-controller="controller1" class="class1">
               <p>{{submessage}}</p>
           </div>
       </div>
   </body>
</html>
```

Custom Recipes

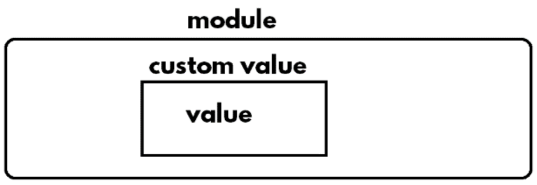
- An angularjs module is a collection of an angularjs application with a collection of custom recipes and app.config( ).
- Angularjs application contains collection of models, views and controllers.
- Custom recipes are:
  1. Custom values
  2. Custom factories
  3. Custom services
  4. Custom filters
  5. Custom providers
  6. Custom directives

Full Architecture of AngularJS Module



Custom Values

- "Custom value" component represents a value that can be stored in the module.
- The "value component" can store any type of data (number / string / object etc.).



Steps for working with value component

- Create a value component in the module

  Syntax: app.value("value name", actual value);

  Ex: app.value("x", 100);

- Retrieve the value component in the controller

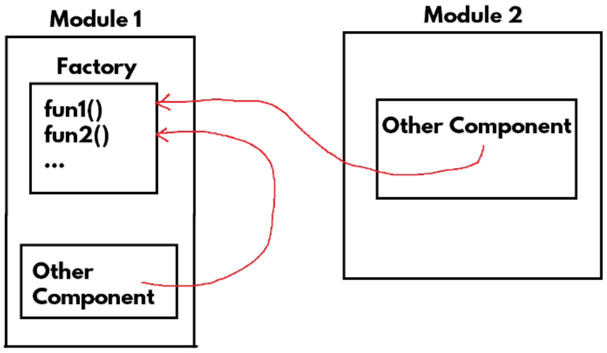
```
app.controller("controller name", functionname);
function functionname(value name)
{
    //do something with the value
}
```

Example on custom values:

```
<html>
    <head>
        <title>AngularJS - Custom Values</title>
        <style>
            body,input
            {
                font-family: Tahoma;
                font-size: 30px;
            }
            #div1
            {
                background-color: #3399ff;
                padding: 10px;
                margin: 10px;
            }
            #div2
            {
                background-color: #ffcc99;
                padding: 10px;
                margin: 10px;
            }
        </style>

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

        <script>
            var app = angular.module("mymodule", [ ]);
            app.value("x", 100);

            app.controller("mycontroller1", fun1);
            app.controller("mycontroller2", fun2);
            function fun1($scope, x)
            {
                $scope.message1 = x;
            }
            function fun2($scope, x)
            {
                $scope.message2 = x;
            }
        </script>
    </head>
    <body ng-app="mymodule">
        <h2>Angular JS Custom Values</h2>

        <div ng-controller="mycontroller1" id="div1">
            {{message1}}
        </div>

        <div ng-controller="mycontroller2" id="div2">
            {{message2}}
        </div>
    </body>
</html>
```

Custom Factories

- "Factory" is a collection of re-usable functions, which can be called from any other component within the same module and also in other modules.



Steps for development of Custom Factories

- **Create factory in the module:**

```
app.factory("factory name", functionname);
function functionname( )
{
   var temp = { };
   temp.function1 = function( )
{
   code here
}
temp.function2 = function( )
{
   code here
}
...
return temp;
}
```

- **Call the factory in the controller:**

```
app.controller("controller name", functionname);
function functionname(factoryname)
{
   factoryname.function1( );
   factoryname.function2( );
   ...
}
```

Example on Custom Factory:

```
<html>
    <head>
        <title>Angular JS - Custom Factories</title>
        <style>
            body, input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
        </style>
        <script src="angular.js"></script>
        <script>
            var app = angular.module("mymodule", [ ]);

            app.factory("myfactory", fun1);
            function fun1()
            {
                var temp = { };
                temp.add = function(a, b)
                {
                    var c = parseInt(a) + parseInt(b);
                    return c;
                }
```

```
                    temp.subtract = function(a, b)
                    {
                        var c = parseInt(a) - parseInt(b);
                        return c;
                    }
                    return temp;
                }

                app.controller("mycontroller", fun2);
                function fun2($scope, myfactory)
                {
                    $scope.firstnumber = "";
                    $scope.secondnumber = "";
                    $scope.result = "";
                    $scope.add = function()
                    {
                        $scope.result = myfactory.add($scope.firstnumber, $scope.secondnumber);
                    }
                }
            </script>
        </head>
        <body>
            <h2>Angular JS - Custom Factories</h2>
            <div ng-app="mymodule" ng-controller="mycontroller">
                First number: <input type="text" ng-model="firstnumber"><br>
                Second number: <input type="text" ng-model="secondnumber"><br>
                <input type="button" value="Add" ng-click="add()"><br>
                Result: <input type="text" ng-model="result" readonly="readonly"><br>
            </div>
        </body>
    </html>
```

Example on Custom Factory – Module to Module:

```
    <html>
        <head>
            <title>Angular JS - Custom Factories</title>
            <style>
                body, input
                {
                    font-family: 'Tahoma';
                    font-size: 30px;
                }
            </style>

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

            <script>
                var app1 = angular.module("mymodule1", [ ] );
                var app2 = angular.module("mymodule2", [ "mymodule1" ] );

                app1.factory("myfactory", fun1);
                function fun1()
                {
                    var temp = { };
                    temp.add = function(a, b)
                    {
                        var c = parseInt(a) + parseInt(b);
                        return c;
                    }
                    temp.subtract = function(a, b)
                    {
                        var c = parseInt(a) - parseInt(b);
                        return c;
                    }
                    return temp;
                }

                app2.controller("mycontroller", fun2);
                function fun2($scope, myfactory)
                {
                    $scope.firstnumber = "";
                    $scope.secondnumber = "";
                    $scope.result = "";
                    $scope.add = function()
                    {
                        $scope.result = myfactory.add($scope.firstnumber, $scope.secondnumber);
                    }
                }
            </script>
        </head>
        <body>
            <h2>Angular JS Custom Factories - Module to Module</h2>
            <div ng-app="mymodule2" ng-controller="mycontroller">
                First number: <input type="text" ng-model="firstnumber"><br>
                Second number: <input type="text" ng-model="secondnumber"><br>
                <input type="button" value="Add" ng-click="add()"><br>
                Result: <input type="text" ng-model="result" readonly="readonly"><br>
            </div>
        </body>
    </html>
```

Custom Services

- AngularJS services provide another way to create angularjs factories.
- AngularJS services are also set of re-usable functions (just like factories).

Sl. No	Factory	Service
1	Factory provides traditional JavaScript syntax to create a set of re-usable functions.	Service provides "JavaScript class" syntax to create set of re-usable functions.
2	In factory, we have to create a temporary javascript object, add functions to it and return the same.	In service, a temporary javascript object will be created and returned automatically, and the same is represented as "this".
3	Factories are "singleton". That means when we call the factory for the first time, an object for the factory will be created and the same object will be delivered when we call the factory next time.	

Syntaxes

Factory	Service

app.factory("factory name", functionname);
function functionname( )
{
    var temp = { };
    temp.function1 = function( )
    {
        code here
    }
    ….
    return temp;
}
	app.service("service name", functionname);
	function functionname( )
	{
	    this.function1 = function( )
	    {
	        code here
	    }
	    ….
	}

Example on Custom Services

```
<html>
   <head>
      <title>Angular JS - Custom Services</title>
      <style>
         body, input
         {
            font-family: 'Tahoma';
            font-size: 30px;
         }
      </style>

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

      <script>
         var app = angular.module("mymodule", [ ]);

         app.service("myservice", fun1);
         function fun1()
         {
            this.add = function(a, b)
            {
               var c = parseInt(a) + parseInt(b);
               return c;
            }
            this.subtract = function(a, b)
            {
               var c = parseInt(a) - parseInt(b);
               return c;
            }
         }

         app.controller("mycontroller", fun2);
         function fun2($scope, myservice)
         {
            $scope.firstnumber = "";
            $scope.secondnumber = "";
```

```
                    $scope.result = "";

                    $scope.add = function()
                    {
                         $scope.result = myservice.add($scope.firstnumber, $scope.secondnumber);
                    };
               }
          </script>
     </head>
     <body>
          <h2>Angular JS - Custom Services</h2>

          <div ng-app="mymodule" ng-controller="mycontroller">
               First number: <input type="text" ng-model="firstnumber"><br>
               Second number: <input type="text" ng-model="secondnumber"><br>
               <input type="button" value="Add" ng-click="add()"><br>
               Result: <input type="text" ng-model="result"><br>
          </div>

     </body>
</html>
```

Custom Filters

- Filter receives a value, convert it into expected format and displays the same in the output.

Steps for developing custom filters

- **Create a filter:**

```
app.filter("filtername", functionname);
function  functionname( )
{
  return function(argument)
  {
    //do some manipulation with argument.
    //return the result value.
  }
}
```

- **Call the filter in the expression:**

```
{{ property | filter }}
```

Example on custom filters:

```
<html>
     <head>
          <title>Angular JS - Custom Filters</title>
          <style>
               body, input
               {
                    font-family: 'Tahoma';
                    font-size: 30px;
               }
          </style>

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

```
        <script>
            var app = angular.module("mymodule", [ ]);

            app.filter("titlecase", fun1);
            function fun1()
            {
                function fun2(value)
                {
                    var temp = value[0].toString().toUpperCase() + value.slice(1);
                    return temp;
                }
                return fun2;
            }

            app.controller("mycontroller", fun3);
            function fun3($scope)
            {
                $scope.empname = "scott";
            }
        </script>
    </head>
    <body>
        <h2>Angular JS Custom Filters</h2>
        <div ng-app="mymodule" ng-controller="mycontroller">
            {{empname | titlecase}}
        </div>
    </body>
</html>
```

Custom Providers

- In angularjs, app.config( ) function is used to configure module level settings.
- A module can have only one app.config( ) function.
- The app.config( ) function will be called automatically, before each controller of the module.
- The app.config( ) function contains the common configuration settings that are applicable to all the controllers of the entire module.
- The app.config( ) function can access only "providers". A provider can represent an individual setting of app.config.

Steps for development of Custom Providers

- **Create a provider:**

```
app.provider("provider name", functionname1);
function functionname1( )
{
  var variable = value;
  this.setfunction = function(argument)
  {
    variable = argument;
  };
  this.$get = function( )
  {
    return { "functionname": function( ) { … } } );
  };
```

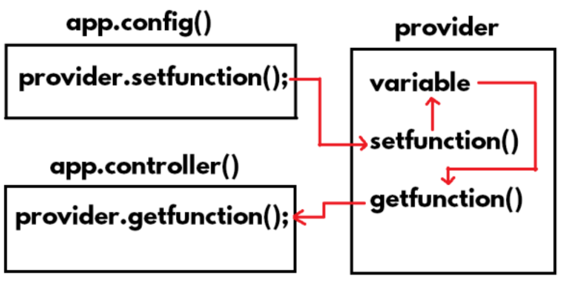
```
}
```

- **Access the provider in app.config( ):**

```
app.config(functionname2);
function  functionname2(providernameProvider)
{
   providernameProvider.setfunction(argument value);
}
```

- **Access the provider in controller:**

```
app.controller("controller name", functionname3);
function  functionname3(providername)
{
   providername.getfunction( );
}
```



Example on Custom Providers

```
<html>
   <head>
      <title>Angular JS - Custom Providers</title>
      <style>
         body, input
         {
            font-family: 'Tahoma';
            font-size: 30px;
         }
      </style>

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

      <script>
         var app = angular.module("mymodule", [ ]);
         //provider
         app.provider("sample", fun1);
         function fun1()
         {
            var message = "Hello";
            this.setmessage = function(msg)
            {
               message = msg;
            };
            this.$get = function()
            {
               function fun2()
```

```
                    {
                        return message;
                    }
                    return { getmessage: fun2 };
                };
            }

            //config
            app.config(fun3);
            function fun3(sampleProvider)
            {
                sampleProvider.setmessage("Hai");
                //sampleProvider.setmessage("Good Morning");
            }

            //controller
            app.controller("mycontroller", fun4);
            function fun4($scope, sample)
            {
                $scope.username = "";
                $scope.outputmessage = sample.getmessage();
            }
        </script>
    </head>
    <body>
        <h2>Angular JS Custom Providers</h2>
        <div ng-app="mymodule" ng-controller="mycontroller">
            Username:
            <input type="text" ng-model="username"><br>
            <span>{{outputmessage}} to {{username}}</span>
        </div>
    </body>
</html>
```

Custom Directives

- A "custom directive" represents a re-usable set of html tags, which can be called in the view.
- We can pass model data and other parameters to the directive.
- Custom directives are user-defined html tags.

Steps for working with Custom Directives

- **Create a directive in the module:**

```
app.directive("directive name", fun1);
function fun1( )
{
  var temp = { };
  temp.restrict = "E | A | EA";
  temp.scope = { "property": "value" };
  temp.link = fun2;

  function fun2($scope, element, attributes)
  {
    $scope.property
```

```
        element.html("content");
        element.css("property", "value");
        attributes.property
    }
    return temp;
}
```

- **Call the directive in the view:**

  As Element: <directive></directive>

  As Attribute: <tag directive></tag>

Example on Custom Directives

```
<html>
    <head>
        <title>Angular JS - Custom Directives</title>
        <style>
            body
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
        </style>

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

        <script>
            var app = angular.module("mymodule", [ ]);

            //directive
            app.directive("student", function()
            {
                var temp = { };
                temp.restrict = "E"; //E | A | EA
                temp.scope = { currentstudent : "=data" };

                temp.link = function($scope, element, attributes)
                {
                    element.html("Student name: <b>" + $scope.currentstudent.studentname + "</b>, Age: <b>" +
$scope.currentstudent.age + "</b>");

                    element.css("border", "5px solid " + attributes.bordercolor);
                    element.css("background-color", attributes.backgroundcolor);
                }
                return temp;
            });

            //controller
            app.controller("mycontroller", function($scope)
            {
                $scope.students = [
                    { "studentid": 1, "studentname": "Scott", "age": 25 },
                    { "studentid": 2, "studentname": "Allen", "age": 28 }
                ];
            });
        </script>
    </head>
    <body>
        <h2>Angular JS Custom Directives</h2>
        <div ng-app="mymodule" ng-controller="mycontroller">
            <p><student data="students[0]" bordercolor="red" backgroundcolor="yellow"></student></p>
            <p><student data="students[1]" bordercolor="blue" backgroundcolor="pink"></student></p>
        </div>
    </body>
</html>
```

Example 2 on Custom Directives

```
<html>
    <head>
        <title>Angular JS - Custom Directives 2</title>
        <script src="angular.js"></script>

        <script>
            var app = angular.module("mymodule", [ ]);

            //directive
            app.directive("students", fun1);
            function fun1()
            {
                var temp = { };
                temp.restrict = "E";
                temp.scope = { currentarray : "=data" };
                temp.link = fun2;

                function fun2($scope, element, attributes)
                {
                    var s = "<table border='1' width='300px'>";
                    s = s + "<caption>" + attributes.caption + "</caption>";
                    for (i = 0; i < $scope.currentarray.length; i++)
                    {
                        s = s + "<tr> <td>" + $scope.currentarray[i].studentid + "</td> <td>" +
$scope.currentarray[i].studentname + "</td> <tr>";
                    }
                    s = s + "</table>";
                    element.html(s);
                    element.find("td").css("background-color", "skyblue");
                    element.find("td").css("padding", "10px");
                    element.find("td").css("border", "2px solid " + attributes.bordercolor);
                }
                return temp;
            }

            //controller
            app.controller("mycontroller", fun3);
            function fun3($scope)
            {
                $scope.students2015 =
                [
                    { "studentid": 1, "studentname": "Scott", "age": 25 },
                    { "studentid": 2, "studentname": "Allen", "age": 28 }
                ];
                $scope.students2016 =
                [
                    { "studentid": 3, "studentname": "Smith", "age": 24 },
                    { "studentid": 4, "studentname": "Jones", "age": 26 }
                ];
            }
        </script>
    </head>
    <body>
        <h2>Angular JS Custom Directives</h2>
        <div ng-app="mymodule" ng-controller="mycontroller">
            <p><students data="students2015" bordercolor="red" caption="Students 2015"></students></p>
            <p><students data="students2016" bordercolor="blue" caption="Students 2016"></students></p>
        </div>
    </body>
</html>
```

Routing

- "Routing" concept is used to create "page navigation".
- That means we will create links from one page to another page.
- We will create "one main page" + "many subpages". Inside the "main page", "subpages" will appear.
- The main page contains html syntax tags like: <html>, <head>, <body> etc.
- The subpages will not contain html syntax tags; contains other normal tags.

Syntax of address:

#/home
#/about
#/contact
etc.

Example on Routing

- Place "angular.js" and "angular-route.js" file in the current folder (c:\angularjs).

index.html

```
<html ng-app="mymodule">
    <head>
        <title>Single Page Application</title>
        <style>
            body
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            #div1
            {
                background-color: #00ccff;
                width: 100%;
                height: 300px;
            }
        </style>
        <script src="angular.js"></script>
        <script src="angular-route.js"></script>

        <script>
            var app = angular.module("mymodule", ["ngRoute"]);
            app.controller("homecontroller", fun1);
            function fun1($scope)
            {
            }
            app.controller("aboutcontroller", fun2);
            function fun2($scope)
            {
            }
            app.controller("contactcontroller", fun3);
            function fun3($scope)
            {
            }
            app.config(fun4);
            function fun4($routeProvider)
            {
                $routeProvider.when("/home", { controller: "homecontroller", templateUrl: "home.html" });
                $routeProvider.when("/about", { controller: "aboutcontroller" , templateUrl: "about.html" });
                $routeProvider.when("/contact", { controller: "contactcontroller", templateUrl: "contact.html" });
                $routeProvider.otherwise({ redirectTo: "/home" });
            }
        </script>
    </head>
    <body>
        <h1>Web site name here</h1>
        <div id="navigation">
            <a href="#/home">Home</a>
            <a href="#/about">About</a>
            <a href="#/contact">Contact</a>
        </div>
        <div id="div1" ng-view>

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

<u>home.html</u>

```
<p>Home page content here</p>
```

<u>about.html</u>

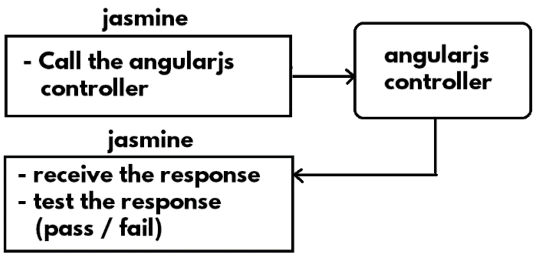
```
<p>About page content here</p>
```

<u>contact.html</u>

```
<p>Contact page content here</p>
```

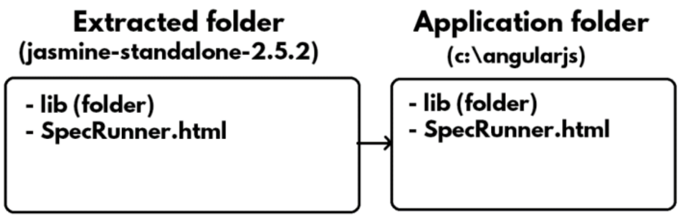
Jasmine

- "Jasmine" is a "Unit Testing" framework, which is used to unit test the angularjs controllers.


jasmine
- Call the angularjs controller
angularjs controller
jasmine
- receive the response
- test the response (pass / fail)

Downloading Jasmine

- Create "c:\angularjs" folder.
- Go to "https://github.com/jasmine/jasmine/releases".
- Click on "jasmine-standalone-2.5.2.zip".
- Extract the zip file.
- Copy the following files and folders from "extracted folder" (jasmine-standalone-2.5.2) into the application folder "c:\angularjs".
  - lib (folder)
  - SpecRunner.html



- Copy the following files from "extracted folder" (angular 1.6) into the application folder "c:\angularjs".
  - angular.js
  - angular-mocks.js

Importing files in SpecRunner.html

```
<!-- include angularjs files -->
<script src="angular.js"></script>
<script src="angular-mocks.js"></script>

<!-- include source files -->
<script src="module.js"></script>

<!-- include spec files -->
<script src="spec.js"></script>
```

Example 1 on Jasmine:

- Place "lib" folder, "ScriptRunner.html" file in "c:\angularjs" folder.
- Create "script.js", "spec.js" files in "c:\angularjs" folder.

c:\angularjs\script.js

```
var x = 100;
```

c:\angularjs\spec.js

```
describe("Jasmine - First Example", function() {
   it("x should be 100", function() {
        expect(x).toBe(100);
   });
});
```

c:\angularjs\SpecRunner.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>Jasmine Spec Runner v2.5.2</title>

 <link rel="shortcut icon" type="image/png" href="lib/jasmine-2.5.2/jasmine_favicon.png">
 <link rel="stylesheet" href="lib/jasmine-2.5.2/jasmine.css">

 <script src="lib/jasmine-2.5.2/jasmine.js"></script>
 <script src="lib/jasmine-2.5.2/jasmine-html.js"></script>
 <script src="lib/jasmine-2.5.2/boot.js"></script>

 <script src="script.js"></script>
 <script src="spec.js"></script>
</head>
<body>
</body>
</html>
```

- Double click on "SpecRunner.html".

Example 2 on Jasmine:

- Place "angular.js", "angular-mocks.js", "lib" folder, "ScriptRunner.html" file in "c:\angularjs" folder.
- Create "module.js", "spec.js", "index.html" files in "c:\angularjs" folder.

c:\angularjs\module.js

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

app.controller("mycontroller", fun1);

function fun1($scope)
{
  $scope.message = "Hello";
}
```

c:\angularjs\index.html

```
<html>
  <head>
       <title>AngularJS - Index</title>
       <style>
            body,input
            {
                 font-family: Tahoma;
```

```
                font-size: 30px;
            }
            #div1
            {
                background-color: #3399ff;
                padding: 10px;
                margin: 10px;
            }
        </style>
        <script src="angular.js"></script>
        <script src="module.js"></script>
    </head>
    <body ng-app="mymodule">
        <h2>AngularJS</h2>
        <div ng-controller="mycontroller" id="div1">
            {{message}}
        </div>
    </body>
</html>
```

c:\angularjs\spec.js

```
describe("Jasmine - Controller", function() {

    beforeEach(module("mymodule"));
    beforeEach(inject(function($controller)
        {
            controller = $controller("mycontroller", { $scope: scope });
        })
    );

    var scope = {};
    var controller;

    it("message should be hello", function() {
        expect(scope.message).toBe("Hello");
    });
});
```

c:\angularjs\SpecRunner.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>Jasmine Spec Runner v2.5.2</title>
 <link rel="shortcut icon" type="image/png" href="lib/jasmine-2.5.2/jasmine_favicon.png">
 <link rel="stylesheet" href="lib/jasmine-2.5.2/jasmine.css">

 <script src="lib/jasmine-2.5.2/jasmine.js"></script>
 <script src="lib/jasmine-2.5.2/jasmine-html.js"></script>
 <script src="lib/jasmine-2.5.2/boot.js"></script>
 <script src="angular.js"></script>
 <script src="angular-mocks.js"></script>

 <script src="module.js"></script>
 <script src="spec.js"></script>
</head>
<body>
</body>
</html>
```

- Double click on "SpecRunner.html".

Example 3 on Jasmine:

- Place "angular.js", "angular-mocks.js", "lib" folder, "ScriptRunner.html" file in "c:\angularjs" folder.
- Create "module.js", "spec.js", "index.html" files in "c:\angularjs" folder.

c:\angularjs\module.js

```
var app = angular.module("mymodule", [ ]);
app.controller("mycontroller", fun1);

function fun1($scope)
{
  $scope.username = null;
  $scope.password = null;
  $scope.msg = null;

  $scope.login = function()
  {
      if ($scope.username == "admin" && $scope.password == "manager")
      {
          $scope.msg = "Successful login";
      }
      else
      {
          $scope.msg = "Invalid login";
      }
  };
}
```

c:\angularjs\index.html

```
<!DOCTYPE html>
<html>
  <head>
      <title>AngularJS - Math</title>
      <style>
          body, input
          {
              font-family: 'Tahoma';
              font-size: 30px;
          }
      </style>

      <script src="angular.js"></script>
      <script src="module.js"></script>
  </head>
  <body>
      <div ng-app="mymodule" ng-controller="mycontroller">
```

```
            First number: <input type="text" ng-model="a"><br>
            Second number: <input type="text" ng-model="b"><br>
            <input type="button" value="Add" ng-click="add()">
            <input type="button" value="Subtract" ng-click="subtract()">
            <input type="button" value="Multiply" ng-click="multiply()">
            <input type="button" value="Divide" ng-click="divide()">
            <br>
            Result: <span ng-bind="c"></span>
        </div>
    </body>
</html>
```

c:\angularjs\spec.js

```
describe("Login test cases", function() {

    beforeEach(module("mymodule"));
    beforeEach(inject(function($controller)
        {
            controller = $controller("mycontroller", { $scope: scope });
        })
    );

    var scope = {};
    var controller;
    it("Login - success - test case", function() {
        scope.username = "admin";
        scope.password = "manager";
        scope.login();
        expect(scope.msg).toBe("Successful login");
    });

    it("Login - fail - test case", function() {
        scope.username = "fgdfg";
        scope.password = "dfgdgeth";
        scope.login();
        expect(scope.msg).toBe("Invalid login");
    });
});
```

c:\angularjs\SpecRunner.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>Jasmine Spec Runner v2.5.2</title>
 <link rel="shortcut icon" type="image/png" href="lib/jasmine-2.5.2/jasmine_favicon.png">
 <link rel="stylesheet" href="lib/jasmine-2.5.2/jasmine.css">
 <script src="lib/jasmine-2.5.2/jasmine.js"></script>
 <script src="lib/jasmine-2.5.2/jasmine-html.js"></script>
 <script src="lib/jasmine-2.5.2/boot.js"></script>
 <script src="angular.js"></script>
 <script src="angular-mocks.js"></script>

 <script src="module.js"></script>
 <script src="spec.js"></script>
</head>
<body>
</body>
</html>
```

- Double click on "SpecRunner.html".