

Angular js

Class notes

By


Mr . Narashima sir



**SRI RAGHAVENDRA
XEROX**

Software languages Material Available

Beside Bangalore Iyyager Bakery .Opp. CDAC, Balkampet Road,
Ameerpet,Hyd

 9951596199

19/10/15

Introduction

⇒ What is AngularJS?

AngularJS is a Opensource structural framework to develop dynamic web Applications.

AngularJS :-

- AngularJS is a completely Javascript base framework clientside programming.
- you need to integrate with webpage using Javascript.

Open Source :-

- It is a open source framework from Google.
- Anyone can download the AngularJS framework & integrate with their technology.

www.angularjs.org

Structural :-

- It is a structural framework. It implements the MVC design pattern to develop the application.

Note :- Generally MVC pattern is used by serverside technology. But AngularJS provides a facility to use this MVC pattern in clientside programming.

Framework :-

- AngularJS is framework not library. jquery is just library.
- AngularJS provides its own runtime environment, compile process at clientside.

Dynamic web Applications :-

- using AJS we can develop clientside dynamic webpages.
- update the html content at runtime without communicating with server.
- we can process the data at clientside as per the requirement.

```

<ul>
  <li ng-repeat = "item in names">
    directives
    {{ item }}
  </li>
</ul>

```

13/10/18

→ How MVC design pattern is implemented in AngularJS?

→ MVC is a design pattern.

→ MVC can be used in any technology.

Ex: Spring MVC in Java.

ASP.NET MVC in .Net.

→ AngularJS also uses MVC design pattern to implement client side programming.

→ In MVC based applications development process is split into three modules.

1. Model

2. View

3. Controller.

→ In AngularJS, the above modules are implemented as follows.

1. Model:-

→ Model is responsible for ^{data} storage.

→ It will be implemented by using JavaScript Variables, arrays, objects and array of objects.

→ These data will be shared to View to display.

2. View:-

→ View is responsible for data presentation.

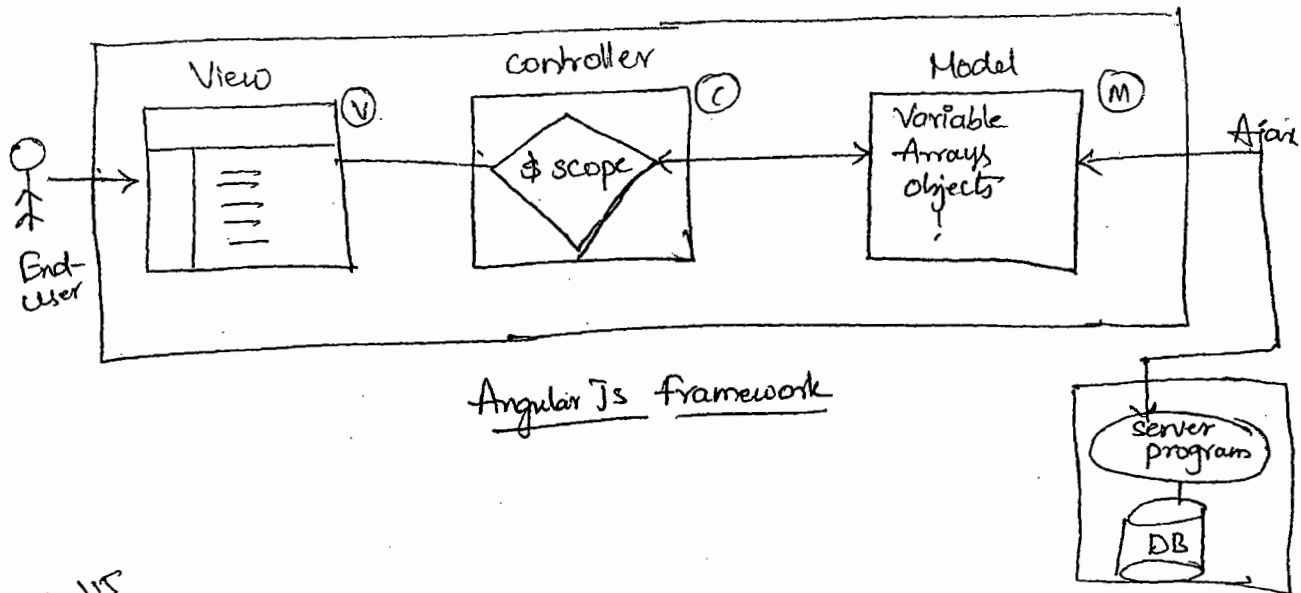
→ It can be implemented by using "HTML, CSS and AJS" directives.

Ex: ng-repeat.

→ Views are bind with Controller to get the data (ng-Controller)

3. Controller :-

- Controller is responsible for data processing.
- It act as mediator between View and Model.
- Controller will communicate with model to get the data.
- Controller will transfer the data to View.



14/10/15

⇒ Features of AngularJS :-

Q: What are the features of AngularJS?

Q: What are the Advantages of AngularJS?

Q: How AngularJS is differentiate from other JS libs/framework?

1) Data Binding :-

→ Data Binding is primary features of Angular JS.

→ Data Binding can be implemented in 2 different ways.

1) One-way Data Binding.

2) Two-way Data Binding.

→ In Data Binding, Model variable/objects are bind with Views.

→ First time in client side frameworks, AngularJS introduced this Data Binding concept.

→ These features will reduce lot of client side code. (JS code)

2) Structural Programming :-

→ Angular JS is structural framework.

→ It uses MVC structure (pattern) to implement the application development.

→ AngularJS is client side MVC.

→ Due to this structure, we can divide application development into multiple modules.

Model - View - Controller.

→ So, that we can reduce the complexity of the application development.

3) Highly Testable :-

→ AngularJS applications are more comfortable to perform unit testing.

→ Unit testing is a process of testing the required portion of the application.

→ Due to MVC structure, we can easily test required module in the application.

4) TDD support :- (Test Driven Development)

→ TDD stands for Test Driven Development.

→ It is the latest development process in which testers will provide the guidelines to developers.

→ According to their guidelines developers will modify/update the application development.

→ It is possible because of MVC structure. At any point of time MVC based applications are more comfortable to update.

→ AngularJS applications are MVC so that it will be more comfortable TDD Environment.

5) Parallel Development :-

→ In AngularJS applications, multiple programmers work parallel on controllers, Models & Views.

→ We are creating in separate files to implement each one.

Ex:- View - *.html

controller - *.js

→ This feature makes your application development more faster.

→ If you write View content and controller code in same file, only one programmer can work at a time.

is/ol/15

6) html as Template :-

→ AngularJS uses existing html tags as Template for Views preparation.

→ AngularJS provides couple of directives (similar to attributes in html) to implement these templates.

→ At the time of execution AngularJS framework executes these templates to generate actual content for View.

Ex:-

```
<ul>
  <li ng-repeat="item in names">
    {{ item }}
  </li>
</ul>
```

7) Routing support for SPA (Single page Applications)

→ SPA is used to develop the application with single page.

→ In this application, only first time page will be loaded from server.

→ Remaining all requests are processed at client side without loading the whole page.

→ AngularJS provides special features to support routing in SPA.

→ Routing concept of AngularJS provides organizing multiple routes for multiple requests with single page.

8) Client Side State Management :-

- One of the challenging task in SPA is State Management.
- AngularJS supports required options to manage the state between multiple requests of SPA.
- One of primary intention of AngularJS development is to support SPA development.

9) Dependency Injection :-

- AngularJS Applications are strongly supports Dependency Injection.
- It means developing the independent code & organize the dependent objects based on the requirement.
- Instead of Depending on fixed set of objects / values, we can make application modules are independent.
- Dependency Injection is concept is involved in several concepts of AJS
Ex:- Creating controllers, Creation Modules, etc.

10) Powered by Google :-

- Google Maintained and support the AngularJS framework.
- It organizes the AngularJS by group of Google team in order to address all technical issues.
- Google is one of the trustable & world famous IT organization.
- From 2012 onwards, officially AngularJS is supported by Google.

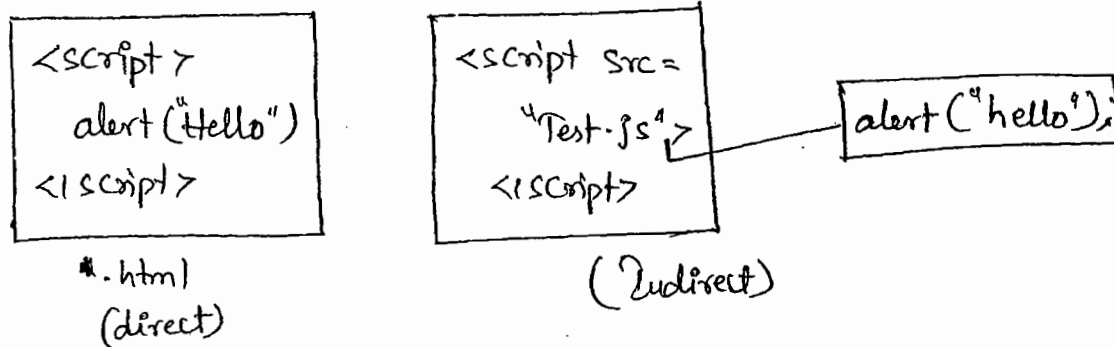
Note:- AngularJS was initially developed by Mr. Misko Hevery.

- He is also Google Employee & he is the author for several web related frameworks.

16/10/15

JavaScript

- JavaScript is a scripting language used for client side programming in webpages.
- JavaScript is light weight programming language.
- JavaScript is case sensitive.
- Control structures and operators are same as C/C#/Java languages.
- JavaScript is used to perform following operations.
 - * To perform client side validations.
 - * To perform dynamic operations.
 - * Client side calculations.
 - * To control the behaviour of html elements.
 - * To Implement AJAX Technology.



⇒ JavaScript Programming:-

- We should understand the following important rules/points to develop JavaScript Programs.

1) Include JavaScript in webpage:-

- We can include the JS in webpages either directly (or) indirectly.

Directly:- `<script type="text/JavaScript">`
 `// JavaScript code`
 `</script>`

Indirectly:- (external files)

`<script type="text/JavaScript" src="text.js">`
`</script>`

Note:- `<script>` tag can be add in any section of html document.
(either head (or) body)

2) Creating Variables:-

- JavaScript variables are declared with "var" keyword.
- These types of variables are called dynamic typing variables.
- It will change the behaviours, based on the value that we stored in it.

Examples:-
 `var x, y, z;`
 `var n=100;`
 `var str="HelloWorld";`
 `var ar=[10,20,30];`
 `var obj={};`

3) Numeric Conversions:-

- By default all html controls returns the values in string formatted data into numeric data.
- JavaScript contains two built-in functions to do this conversion.

- 1) `parseFloat()`
- 2) `parseInt()`

Ex: Var x, y, z;

x = ParseInt(obj1.value);

y = ParseInt(obj2.value);

4) Accessing references of html elements :-

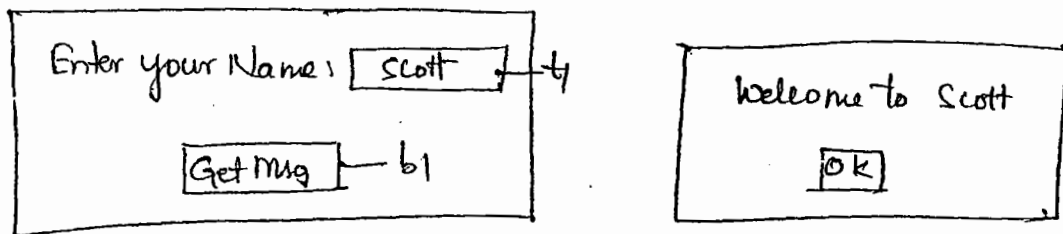
→ we can get the reference of any html element by using "document" object.

→ "getElementById" is a function of "document" object

usage: document.getElementById("id");

Note:- Once we get the reference, we can access the details of that elements like value, text, attributes etc.

⇒ Example:-



```
<html>
```

```
<head>
```

```
<script>
```

```
function f1()
```

```
{
```

```
var obj = document.getElementById("t1");
```

```
alert("Welcome to " + obj.value);
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
Enter your Name : <input type="text" id="t1">
```

```
<input type="button" id="b1" onClick="f1()" Value="Getmsg">
```

```
</body>
```

```
</html>
```

→ Create a webpage to handle mouse & key events?

```
<html>
```

```
<head>
```

```
<script>
```

```
function f1()
```

```
{
```

```
    alert("welcome");
```

```
}
```

```
function f2()
```

```
{
```

```
    var obj = document.getElementById("t1");
```

```
    var spobj = document.getElementById("spi");
```

```
    spobj.innerHTML = obj.value;
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<h1 onMouseOver="f1()">Programming JavaScript Demo </h1>
```

```
Enter your name : <input type="text" id="t1" onKeyUp="f2()"><br/>
```

```
welcome to <span id="spi"></span>
```

```
</body>
```

```
</html>
```

→ <!-- create html page background color <div> tag based on mouse events -->

```
<html>
<head>
  <style>
```

```
    div
```

```
    {
```

```
      border : 2px solid Red;
```

```
      width : 300px;
```

```
      height : 100px;
```

```
    }
```

```
</style>
```

```
<script>
```

```
  function f1c()
```

```
  {
```

```
    var obj = document.getElementById ("div1");
```

```
    obj.style.backgroundColor = "Pink";
```

```
    obj.style.Color = "Blue";
```

```
  }
```

```
  function f2c()
```

```
  {
```

```
    var obj = document.getElementById ("div1");
```

```
    obj.style.backgroundColor = "white";
```

```
    obj.style.Color = "black";
```

```
  }
```

```
</script>
```

```
</head>
```

```
<body>
```

```
  <h1> Java Script Demo </h1>
```

```
  <div id="div1" onmouseover="f1c()" onmouseout="f2c()"> <br>
```

```
    dynamic styles using JS </div>
```

```
</body>
```

```
</html>
```

→ Implementing uppercase and lowercase

```
<html>
  <head>
    <script>
      function f1()
      {
        var obj = document.getElementById ("t1");
        var spobj1 = document.getElementById ("sp1");
        var spobj2 = document.getElementById ("sp2");
        spobj1.innerHTML = obj.value.toUpperCase();
        spobj2.innerHTML = obj.value.toLowerCase();
      }
    </script>
  </head>
  <body>
    <h1> Program Javascript Demo </h1> <hr>
    Enter your Name: <input type="text" onkeyup="f1()" id="t1"/>
    <br><br>
    <span id="sp1"> </span> <br>
    <span id="sp2"> </span>
  </body>
</html>
```

17/10/15

Arrays

→ <!-- create a webpage to display array elements as unordered
list format -->

```

<html>
  <head>
    <script>
      function f1()
      {
        var ar = [10, 20, 30, 40];
        var obj = document.getElementById ('list1');
        obj.innerHTML = " ";
        for (i = 0; i < ar.length; i++)
        {
          obj.innerHTML = "<li>" + ar[i] + "</li>";
        }
      }
    </script>
  </head>
  <body>
    <h1> JavaScript Programming with Array </h1><hr>
    <input type="button" onclick="f1()" value="Display" /> <br>
    <ul id="list1"> </ul>
  </body>
</html>

```

→ <!-- Creating a webpage to perform dynamic Operations -->

```

<html>
  <head>
    <script>
      var course = ["ASP", "JSP", "PHP"];
      function f1()
      {
        var obj = document.getElementById ('list1');
        obj.innerHTML = " ";
      }
    </script>
  </head>
  <body>
    <h1>

```

```

for(i=0; i < course.length; i++)
{
    obj.innerHTML = "<options> + course[i] + "</option>";
}
}
function f2()
{
    var item = document.getElementById("t1").value;
    course.push(item); f1(); document.getElementById("t1").value="";
}

```

```

function f3()
{
    var obj = document.getElementById("list1");
    course.splice(obj.selectedIndex); f1();
}
function f4()
{
    course.splice(); f1();
}

```

</script>

</head>

<body>

Enter Course Name

<input type="text" id="t1"/>

<input type="button" onclick="f1()" value="Display"/>

<input type="button" onclick="f2()" value="Add"/>

<input type="button" onclick="f3()" value="Remove"/>

<input type="button" onclick="f4()" value="Remove all"/>

<select size="10" id="test" style="width: 100px">

</select> </body>

</html>

ArrayName.splice (index, count); ^{No. of items to remove.}

ArrayName.Push (item); ^{Index of item to remove}

↓
item name that you want to pass. we can pass more than one item also by using (.)

Sort()
remove()
join()
splice()
pop()

→ we can perform these operations by using array methods

→ <!-- create a webpage to display student details using JS obj -->

```
<html>
<head>
  <script>
    function f2()
    {
      var obj = { sno: 1025, sname: 'scott', course: 'AngularJS' };
      var str = "Student Id: " + obj.sno + "<br>";
      str = str + "Student Name: " + obj.sname + "<br>";
      str = str + "Student Course: " + obj.course + "<br>";
      document.getElementById("sp1").innerHTML = str;
    }
  </script>
</head>
<body>
  <input type="button" onclick="f1()" value="show details" />
  <br><br>
  <span id="sp1"></span>
</body>
</html>
```


19/10/15

Javascript

⇒ Object Creation in Javascript :-

- Creating objects is one of the frequently required task in Javascript programming.
- In fact every html tag internally converted as object at runtime.
- In order to organize multiple values with single variable objects are the best options.
- So, that we can store / transfer the data in the form of objects.
- we can create objects in Javascript in 3 ways.

- 1) Direct Object creation with { }.
- 2) Object creation with "new" keyword & "Object".
- 3) By using construction function.

1) Direct object creation

- If you want to create an object with properties & values at a time, you can use this option.

Syntax :-

```
Var obj = { prop : Value, prop : Value ... };
```

Ex:-

```
Var s1 = { Sno : 1025, sname : "scott", course : "AngularJS" };
```

2) Object Creation with "new" keyword & "Object" :-

- This operation is suitable attaching properties after creating objects.

Syntax :-

```
Var obj = new Object();
```

```
obj.prop = value;
```

```
obj.prop = value;
```

!

Ex 1:-
Var s1 = new Object();
s1.sno = 1025;
s1.sname = "scott";
s1.course = "Angular JS";

3) Object Creation with Construction function.

- This option is suitable to create multiple objects by initializing the required values.
- In this option we can reuse the same function to create multiple objects.
- To implement this option, we have to create a function with arguments.
- Argument values need to be assigned to properties with "this" keyword.
- "this" refers in javascript as current object (instance).

Syntax:-

```
function functionName(arg1, arg2)
{
    this.prop = arg1;
    this.prop = arg2;
    ---
}
```

Ex 2:-

```
function student(n, name, cname)
{
    this.sno = n;
    this.sname = name;
    this.course = cname;
}
```

```
Var s1 = new Student(1026, "scott", "AngularJS");
```

⇒ Example: for Object Creation :-

<html>

<head>

<script>

// 1. Object creation with { }

Var s1 = { sno: 1025, sname: "Scott", course: "Angular Js" };

// 2. using "new" & "Object"

Var s2 = new Object();

s2.sno = 1026;

s2.sname = "smith";

s2.course = "jQuery";

// 3. using construction function

~~Var s3 = new Student~~

function Student(x, y, z)

{

this.sno = x;

this.sname = y;

this.course = z;

}

Var s3 = new Student("1027", "Sathy", "Bootstrap");

</script>

</head>

<body>

<h1> Object Creation in Java Script </h1>

<hr></hr>

<h2> 1. Object creation with { } </h2>

<h2> 2. Object Creation using "new" & "Object" </h2>

<h2> 3. By using construction function </h2>

<script>

Var spobj1 = document.getElementById("sp1");

Var str = "Student Id: " + s1.sno + ",";

str = str + "Student Name: " + s1.sname + ",";

str = str + "Student course: " + s1.course;

Var spobj2 = document.getElementById("sp2");

Var str = "Student id: " + s2.sno + ",";

str = str + "Student Name: " + s2.sname + ",";

str = str + "Course: " + s2.course;

Var spobj3 = document.getElementById("sp3");

Var str = "Student id: " + s3.sno + ",";

str = str + "Student Name: " + s3.sname + ",";

str = str + "Course: " + s3.course;

</script>

</body>

</html>

⇒ Create, update, show ^{the objects} ~~Emp for object creation~~ in JavaScript

Create	Show	update
Emp No	<input type="text"/>	→ t1
Ename	<input type="text"/>	→ t2
Job	<input type="text"/>	→ t3

```
<html>
```

```
<head>
```

```
<script>
```

```
1) Var eno = document.getElementById ("t1");
```

```
1) Var ename = document.getElementById ("t2");
```

```
1) Var job = document.getElementById ("t3");
```

```
Var obj;
```

```
function Student (x,y,z)
```

```
function f1()
```

```
{
```

```
function Student (x,y,z)
```

```
{
```

```
this.eno = x;
```

```
this.ename = y;
```

```
this.job = z;
```

```
}
```

```
Var obj = new Student (eno.value, ename.value, job.value);
```

```
}
```

```
(11, "rama", "Engineer");
```

```
(eno.value, ename.value, job.value);
```

```
function f2()
```

```
{
```

```
Var eno = document.getElementById ("t1");
```

```
Var ename = document.getElementById ("t2");
```

```
Var job = document.getElementById ("t3");
```

```
eno.value = obj.eno;
```

```
ename.value = obj.ename;
```

```
job.value = obj.job;
```

```
}
```

```

function f3()
{
    Var eno = document.getElementById ("t1");
    Var ename = document.getElementById ("t2");
    Var job = document.getElementById ("t3");
    obj.Enno = eno.Value;
    obj.Enname = ename.Value;
    obj.Job = job.Value;
}

```

<body>

~~<div>~~

<Button onclick="f1()"> Create </Button>

<Button onclick="f2()"> Show </Button>

<Button onclick="f3()"> update </Button>

EmpNo <input type="text" id="t1" />

ENName <input type="text" id="t2" />

Job <input type="text" id="t3" />

</body>

</html>

⇒

```

<script>
Var obj;
function f1()
{
    obj = { prop: Value -- }
}
for f2()
{
    t1.Value = obj.enno;
}

```

```

fun f3()
{
    obj.enno = t1.Value;
}

```


20/10/15

→ Example to create objects & store the values into Table

```
<html>
  <head>
    <script>
      function f1()
      {
        var Students = [ { sno: 1025, sname: "scott", course: "AJS" },
                          { sno: 1026, sname: "sott1", course: "jQuery" },
                          { sno: 1027, sname: "scott2", course: "HTML" },
                          { sno: 1028, sname: "scott3", course: "css" } ];

        var str = "";
        for (str i=0; i < students.length; i++)
        {
          str = str + "<tr>";
          str = str + "<td>" + students[i].sno + "</td>";
          str = str + "<td>" + students[i].sname + "</td>";
          str = str + "<td>" + students[i].course + "</td>";
          str = str + "</tr>";
        }
        document.getElementById("table1").innerHTML = str;
      }
    </script>
  </head>
  <body>
    <h1> Array of objects - JS </h1>
    <hr>
    <button id="b1" onclick="f1()"> show Student Details </button>
    <hr></hr>
    <table id="table1" border="1" width="400px">
    </table>
  </body>
</html>
```

22/10/15

⇒ Directives in AngularJS:-

⇒ Developing webpages using AJS:-

→ AJS is a Javascript framework. It is used to perform all client-side activities easily.

→ In order to use AJS, we need to download AngularJS framework (*.js file) from the following website.

www.angularjs.org

filename: angular.min.js.

→ Import the above *.js file in your webpage so, that we can develop webpages with AngularJS.

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

(∵ script tag is having another attribute called type that is optional.)

Example:-

<!-- create a webpage to display welcome message using AJS -->

```
<html>
```

```
<head>
```

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

```
</head>
```

```
<body>
```

```
<h1> AngularJS - Demo </h1> </h1>
```

```
<div ng-app="" ng-init="x='Sathya'">
```

```
  username: <input type="text" ng-model="x" />
```

```
</div></div>
```

```
<span> welcome to {{x}} </span>
```

```
</div>
```

```
</body>
```

```
</html>
```

Execution process of AJS:-

- 1) Browser will load the html document.
- 2) AJS framework will load. At the time of loading AJS framework, it creates a global object called "angular".
- 3) Angular object will compile and execute the AJS related elements.
- 4) AJS framework generates dynamic content based on the directives that we involved.

Note:- AJS directives are special attributes which can be understood by only AJS framework.

* In the above example, we involved following directives

- 1) ng-app :- to specify AJS application so, that it will specify the region on which we can apply AJS.
- 2) ng-model :- to bind i/p that value with AJS variable value. It is 2-way binding.
- 3) {{x}} :- It is a AJS "expression". It is considered as 1-way binding.

Example ②:-

<!-- create a webpage to change the bgcolor of table dynamically using AJS -->

```
<html>
  <head>
    <script src="angular.min.js"> </script>
  </head>
  <body>
    <h1> Angular JS - Demo2 </h1>
    <hr> </hr>
```

```

<div ng-app="" ng-init="x='lightgreen'">
  <select colorName: <select ng-model="x">
    <option> pink </option>
    <option> light Red </option>
    <option> Red </option>
  </select>
  <br><br>
  <table bgcolor="{{x}}" width="80%" border="2">
    <tr>
      <td> 1025 </td>
      <td> Scott </td>
      <td> AJS </td>
    </tr>
    <tr>
      <td> 1026 </td>
      <td> Tiger </td>
      <td> jQuery </td>
    </tr>
  </table>
</div>
</body>
</html>

```

27/10/15

⇒ Directives in AngularJS:-

- In AngularJS application development will split into Model, View and controller.
- Views are developed by using HTML tags & AngularJS directives.
- AngularJS uses HTML as template & generate the dynamic content at runtime.
- Directives plays very important role to generate dynamic content.

What is directives in AngularJS?

- Directives are used as HTML attributes.
- Directives can extend the behaviour of HTML tags.
- Every View (UI Logic) in AngularJS is depend on the directives.

Important Directives:-

- 1) ng-app
- 2) ng-model.
- 3) ng-init
- 4) ng-controller.
- 5) ng-view
- 6) ng-repeat
- 7) ng-show
- 8) ng-hide.
- 9) ng-if
- 10) ng-switch.

Note:- Directives are prefix with "ng-".

→ According to HTML 5, these user defined attributes are prefix with "data-".

data-ng-model
data-ng-app

Example 1-

<!-- Create a webpage to change the image dynamically using AngularJS -->

<html>

<head>

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

</head>

<body> ng-app="" ng-init="fname='Image1.jpg'">

<h1> AngularJS directives - Demo1 </h1>

<hr></hr>

<select ng-model="fname">

<option> Image1.jpg </option>

<option> Image2.jpg </option>

<option> Image3.jpg </option>

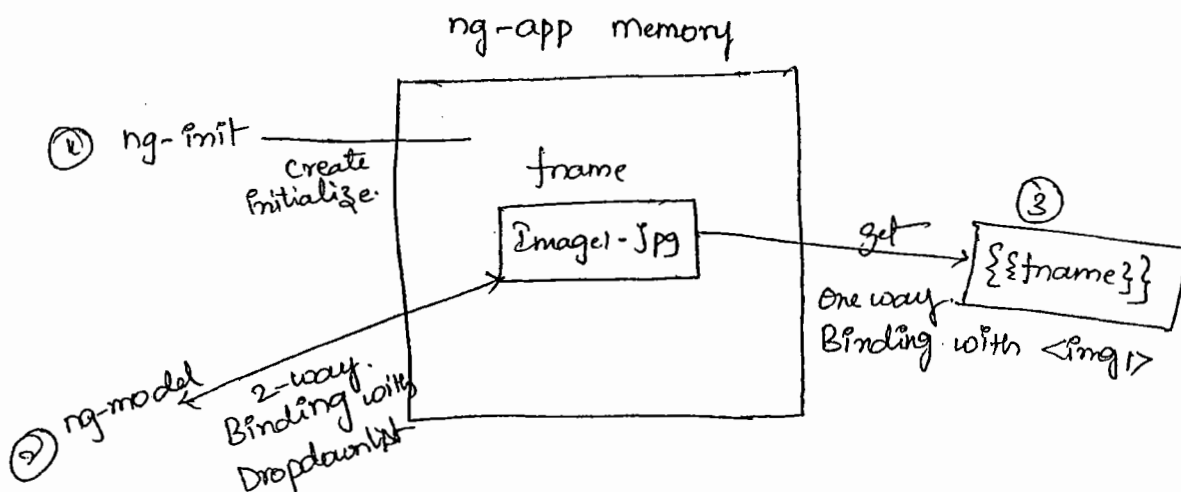
<option> Image4.jpg </option>

</select>
</br>

 Selected Image : {{fname}}
</br>

</body>

</html>



⇒ Example

<!-- create a webpage to change the size of the image with range controls using AngularJS -->

<html>

<head>

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

</head>

<body ng-init="w=300, h=190" ng-app="">

<h1> AngularJS Directive- Demo </h1>

<hr></hr>

Height <input type="range" ng-model="h" min="100" max="390"/>

</br>

width <input type="range" ng-model="w" min="100" max="500"/>

</br>

 {{h}} , {{w}}
</br>

</body>

</html>

⇒ Data Binding in AngularJS :-

- one of the important feature of AngularJS is "Data Binding".
- Data Binding concept is used to bind UI related items (html tags) with programming variables/objects.
- UI will update automatically whenever these programming items are updated.
- AngularJS supports 2 types of Bindings:
 - 1) one-way Binding.
 - 2) Two-way Binding

1) one-way Binding:-

- In one-way Binding, UI will get the updated values whenever you modify the variables.
- In this binding only reading the value from variables & updates in UI.
- UI cannot update the value of variables.
- It can be implemented by using AngularJS expressions (or) ng-bind.

Ex:- ` {{ x }} `

2) Two-way Binding:-

- In two-way binding, html controls are bind with variables.
- If you update the variable, it will reflect the controls.
- If you update the control value, it will reflect the variable.
- It can be implemented by using "ng-model" directive.

Ex:- `<input type="text" ng-model="x" />`

⇒ ng-repeat

- ng-repeat directive is used to process the collections like arrays.
- This directive generates the given content for every item in the collection.
- At the time of execution, angularJS will process each & every item from the given collection.

Syntax:- `ng-repeat="variable in collection"`

Ex:- `<div ng-repeat="item in ar">
 {{ item }}
</div>`

→ The Above example will generate content of div tag (span) for each item in the array.

→ "item" is a temporary variable. which holds each value from array.

28/10/15

Example:-

<!-- create a webpage to process collection with ng-repeat directive of AngularJS -->

<html>

<head>

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

</head>

<body ng-app="" ng-init="ar=['HTML','css','jQuery','AngularJS','JSP','ASP','PHP']">

<h1> AngularJS Directive-Demo </h1>

<hr/>

<table border="2">

<tr>

<th> S.No. </th>

<th> CourseName </th>

</tr>

<tr ng-repeat="item in ar">

<td> {{ \$index + 1 }} </td>

<td> {{ item }} </td>

</tr>

</table>

</body>

</html>

→ Example

<!-- Create a webpage to process student details with JS object in Angular JS -->

```
<html>
<head>
  <script src="angular.min.js"> </script>
</head>
<body ng-app="" ng-init="s1 = {sno: 1025, sname = 'scott',
                           course = 'AngularJS' }">

  <h1> Angular JS Directives </h1>
  <hr />
  <span>
    studentId : {{ s1.sno }}
    studentName : {{ s1.sname }}
    Course Name : {{ s1.course }}
  </span>
</body>
</html>
```

Controllers in AngularJS

- Angular JS uses MVC pattern to develop client side web applications.
- In this pattern the C-stands for controller.
- Controller contains application logic to organizing data flow with in the application.
- In AngularJS controllers are developed by using constructor function.
- At the time of Execution AngularJS framework create objects for Controller. by using constructor function.
- The Views are binding with controller by using a directive called ng-controller.

Ex: <div ng-controller="StudentController">
=</div>

Note:- The Above Binding statement generate the new object for controller. so, that View can access controller members (Variables & methods)

- The Variables (or) methods that we prepared inside the controller are not accessible to View.
- AngularJS will provides a special object called "\$scope" to sharing data b/w controller and View.

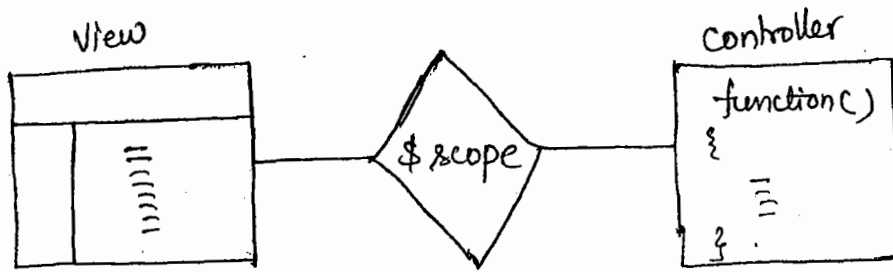
"\$scope" object :

- "\$scope" is a builtin object in AngularJS.
- AngularJS integrate a new scope object for every instance of ~~object~~ controller.
- The Variables & methods that you want to share with view should be attach with "\$scope" object.

Ex: `$scope.x = 123;`
`$scope.arr = [10, 20, 30];`
`$scope.getTotal = function()`
`{`
`=`
`}`
`$scope.s1 = { --- };`

Note:- If we are not using any controller AngularJS uses "Global Controller & Global \$scope" to organizing your data.

- \$scope object should be mentioned with in the controller to attach ^{with} ~~the~~ Variables & methods. \$scope object name no need to mention in the View. (we can directly access variables & methods)
- \$scope is act as a ~~mediator~~ mediator b/w controller & View.



29/10/15

→ Developing controllers in AngularJS :-

→ AngularJS uses controllers to make data flow b/w model & View.

→ These controllers are created by using module object.

→ In AJS, modules are called AngularJS application. Module is combination of AJS programming elements like : controllers, Directives, filters....

→ Module can be created by using built-in object called "angular".

Steps to create controller :-

Step 1 :- Create a module.

```
var obj = angular.module("ModuleName", [])
```

Step 2 :- Create controller.

```
obj.controller("controllerName", constructor = function);
```

Note :-

→ constructor function that we prepared to create controller object should take "\$scope" as argument.

→ At runtime AJS framework will create object for constructor with \$scope as parameter.

Step 3 :- Bind Module.

```
<body ng-app="ModuleName" > </body>
```

Step 4 :- Bind controller

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

<!-- Access All variables & methods of controller -->
</div>

Example 1

<!-- Create a webpage to process employee details using AngularJS controller -->

<html>

<head>

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

<script>

// Step 1

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

// Step 2

obj.controller("EmpController", function(\$scope)

{

\$scope.empno = 1025;

\$scope.ename = "scott";

\$scope.job = "Manager";

});

</head>

<body> ng-app = "myApp" > // Step 3

<h1> AngularJS Directives - Demo </h1>
Controllers

<hr></hr>

<div ng-controller="EmpController" > // Step 4

 EmployeeId : {{ empno }}
</br>

EmployeeName : {{ ename }}
</br>

Designation : {{ job }}

</div>

</body>

</html>

Example 1-

<!-- Create a webpage to create multiple objects of Empcontroller -->

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

step 1: Var obj = angular.module('myApp', []);

step 2: obj.controller('EmpController', function(\$scope)

```
{
  $scope.empno = 1025;
  $scope.ename = 'scott';
  $scope.job = 'Manager';
});
```

```
</script>
```

```
</head>
```

step 3: <body> ng-app='myApp'

```
<h1> Angular JS controllers </h1>
<hr>
```

step 4: <div style="border: 2px solid blue; background-color: pink;"

```
ng-controller="Empcontroller">
```

```
<input type="text" ng-model="empno"/>
```

```
<input type="text" ng-model="ename"/>
```

```
<input type="text" ng-model="job"/> <br><br>
```

```
<span> Employee Id: {{ empno }} <br><br>
```

```
Employee Name: {{ ename }} <br><br>
```

```
Designation: {{ job }} <br><br>
```

```
</span>
```

```
</div>
```

```
</body>
```

```
</html>
```

Example:-

<1 -- create a webpage to display Department details in following format using AngularJS controller -->

DeptNo	10
DName	Accounts
Location	Hyderabad

<html>

<head>

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

<script>

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

obj.controller('DeptController', function(\$scope)

{

\$scope.Dno = 10;

\$scope.Dname = 'Accounts';

\$scope.Loc = 'Hyderabad';

});

</script>

</head>

<body ng-app="myApp">

<h1> Angular JS Controller </h1>

</hr>

//<div ng-controller="DeptController"> <table border="1" ng-controller="DeptController">

<table border="1">
<tr>
<td> Dept No </td>
<td> DName </td>
<td> Location </td>
</tr>
<tr>
<td> {{ Dno }} </td>
<td> {{ Dname }} </td>
<td> {{ Loc }} </td>
</tr>
</table>

<tr>
<td> DeptNo </td>
<td> {{ Dno }} </td>
</tr>
<tr>
<td> DName </td>
<td> {{ Dname }} </td>
</tr>
<tr>
<td> Location </td>
<td> {{ Loc }} </td>
</tr>
</table>

</div>

20/10/15

→ Example:-

<!-- create a webpage to process student marks using AngularJS Controller -->

```
<html>
<head>
  <script src="angular.min.js"> </script>
  <script>
    var obj = angular.module('myApp', []);
    obj.controller('studentcontroller', function($scope)
    {
      $scope.marks = [95, 85, 75, 65, 55];
    });
  </script>
</head>
<body> ng-app = "myApp" >
  <h1> student details </h1>
  <hr>
  <div ng-controller = "studentcontroller">
    <ul>
      <li ng-repeat = "item in marks" >
        subject - { {{ index + 1 }} } = {{ item }}
      </li>
    </ul>
  </div>
</body>
</html>
```

→ Example:-

<!-- create a webpage to process student details using AngularJS Controller organize the data using javascript object -->

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



```
<script>
```

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

```
obj.controller('studentController', function($scope)
```

```
{
```

```
  $scope.studentobj = { sno: 1025,
```

```
    sname: "Scott",
```

```
    course: "Angular JS",
```

```
    Marks: [95, 85, 75, 65, 55]
```

```
  };
```

```
});
```

```
</script>
```

```
</head>
```

```
<body ng-app="myApp">
```

```
  <h1> Student details </h1>
```

```
  <hr>
```

```
  <table ng-controller="studentController" border="2" width="250px">
```

```
    <tr>
```

```
      <td> Student Id </td>
```

```
      <td> {{studentobj.sno}} </td>
```

```
    </tr>
```

```
    <tr>
```

```
      <td> Student Name </td>
```

```
      <td> {{studentobj.sname}} </td>
```

```
    </tr>
```

```
    <tr>
```

```
      <td> Course </td>
```

```
      <td> {{studentobj.course}} </td>
```

```
    </tr>
```

```
    <tr>
```

```
      <td> Marks </td>
```

```
      <td> <ul> <li ng-repeat="item in studentobj.Marks">
```

```
        subject - {{ $indent + 1 }} : {{ item }} </li> </ul>
```

```
      </td>
```

```
    </tr>
```

```
  </table>
```

```
</body>
```

```
</html>
```

→ Example

<!-- create a webpage to process product details using Angular JS
controller organize products data using Array of objects -->

```
<html>
```

```
<head>
```

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

```
<script>
```

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

```
obj.controller('Product controller', function($scope)
```

```
{
```

```
$scope.products = [
```

```
{ pid: 14589, pname: "Nikon camera", uprice: 5200,
```

```
image: "Image1.jpg" },
```

```
{ pid: 14589, pname: "Canon camera", uprice: 5200,
```

```
image: "Image2.jpg" },
```

```
{ pid: 14589, pname: "Samsung camera", uprice: 5200,
```

```
image: "Image3.jpg" },
```

```
{ pid: 14589, pname: "Sony camera", uprice: 5200,
```

```
image: "Image4.jpg" } ];
```

```
});
```

```
</script>
```

```
</head>
```

```
<body ng-app="myApp">
```

```
<h1> Product Details </h1>
```

```
<h2>
```

```
<div ng-controller="productcontroller">
```

```
<div style="text-align: center; float: left; padding: 3px;
```

```
margin: 5px; border: 2px solid blue; height: 150px;
```

```
width: 200px;" ng-repeat="item in products">
```

 <u> {{ item.pname }} </u>

Product Id : {{ item.pid }}

Price : INR {{ item.uprice }}.00

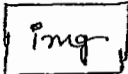
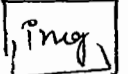
</div>

</div>

</body>

</html>

→ update the previous Example to display the product details in the following format.

productName	Brand	Unit price	Picture
Nikon Coolpix S3700	Nikon	\$ 6,275.00	
Canon PowerShot SX400	Canon	\$ 9,850.00	

</html>

<head>

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

~~var~~ obj =

</script>

<script>

// same as above

</script>

</head>

<body> ng-app="myApp"

<h1> Product Details </h1>


```
<table ng-controller="productController" border="2">
```

```
<tr>
```

```
<th> <u> item.Product Name </u> </th>
```

```
<th> <u> Product Id </u> </th>
```

```
<th> <u> Unit Price </u> </th>
```

```
<th> <u> Picture </u> </th>
```

```
</tr>
```

```
<tr> ng-repeat="item in products">
```

```
<td> item.
```

```
<td> ng-repeat="item in products"
```

```
<td> {{ item.pname }} </td>
```

```
<td> {{ item.pid }} </td>
```

```
<td> {{ item.uprice }} </td>
```

```
<td>  </td>
```

```
</tr>
```

```
</table>
```

```
</body>
```

```
</html>
```

31/10/15

Filters in AngularJS

- AngularJS filters are used to format (filter/sort) the data that we display to the user.
- Filters can be applied at client side. No need to communicate with Server.
- By using filters we can perform following operations:
 - 1) Convert the data into lowercase/uppercase
 - 2) Attach currency symbols.

3) filters the items based on search strings.

4) Sort the items based on requirement.

filter options:

→ AngularJS provides us several built-in ~~functions~~ filters to perform above operations.

→ use the following options to implement filters:

- 1) lower case.
- 2) uppercase.
- 3) currency.
- 4) order by.
- 5) filter.

Note:-

→ order by & filter option applicable on ng-repeat directive.

⇒ Example:-

<!-- create a webpage to demonstrate usage of filters in AngularJS -->

```
<html>
```

```
<head>
```

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

```
<script>
```

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

```
obj.controller("EmpController", function($scope)
```

```
{
```

```
  $scope.ename = "scott";
```

```
  $scope.job = "Manager";
```

```
  $scope.sal = 2000;
```

```
});
```

```
</script>
```

```
</head>
```

```
<body ng-app="myApp">
```

```
<h1> Employee details - filters Demo </h1>
```

```
<hr/>
```

```

<div ng-controller="EmpController">
  <span>
    Employee Name : {{ ename | uppercase }} <br>
    Designation : {{ job | lowercase }} <br>
    Salary : {{ sal | currency }}
  </span>
</div>
</body>
</html>

```

⇒ Example 1

<!-- Create a webpage to demonstrate usage of orderBy filter. in AngularJS -->

```

<html>
<head>
  <script src="angular.min.js"> </script>
  var obj = angular.module("myApp", []);
  obj.controller("DemoController", function($scope)
  {
    $scope.names = ["Scott", "Steve", "Sathya", "Abhrona", "Mark",
      "James", "Ritesh", "Adam"];
  });
</script>
</head>
<body ng-app="myApp">
  <h1> orderBy filters - Demo </h1> </hr>
  <table border="1" width="150px" ng-controller="DemoController">
    <tr ng-repeat="item in names | orderBy: item:true">
      <td> {{ item }} </td>
    </tr>
  </table>
</body>
</html>

```

⇒ Example 2

<!-- Create a webpage to sort the student details using order by filter -->

```
<html>
<head>
  <script src="angular.min.js"></script>
  <script>
    var obj = angular.module("myApp", []);
    obj.controller("Democontroller", function($scope)
    {
      $scope.students = [
        {sname: "scott", course: "AngularJS"},
        {sname: "Abram", course: "HTML"},
        {sname: "smith", course: "jQuery"},
        {sname: "James", course: "css"},
        {sname: "Adam", course: "Java"}
      ];
      $scope.x = false;
    });
  </script>
</head>
<body ng-app="myApp">
  <h1>order by filter - Demo</h1> <h2>
  <div ng-controller="DemoController">
    Descending order: <input type="checkbox" ng-model="x"/> <br><br>
    <table border="1" width="500px">
      <tr bgcolor="skyBlue">
        <th>Student Name</th>
        <th>Course Name</th>
      </tr>
      <tr bgcolor="pink" ng-repeat="item in students | orderBy:'sname':x">
        <td>{{ item.sname }}</td>
        <td>{{ item.course }}</td>
      </tr>
    </table>
  </div>
</body>
</html>
```

→ update the previous program to sort the data based on the Selected ^{ask} property from dropdown list.

2/11/15

Example:-

<!-- Create a webpage to sort the student details based on selected column using orderBy filter -->

<html>

<head>

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

<script>

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

obj.controller("DemoController", function(\$scope)

{

\$scope.students =

[{ sid: 1025, sname: "Scott", course: "AngularJS" },

{ sid: 1026, sname: "Smith", course: "jQuery" },

{ sid: 1027, sname: "Abrons", course: "HTML" },

{ sid: 1028, sname: "James", course: "css" },

{ sid: 1029, sname: "Adam", course: "JavaScript" }];

\$scope.x = false;

\$scope.y = "sid";

});

</script>

</head>

<body> ng-app="myApp" >

<h1> Order By filters - Demo 3 </h1>

<hr />

<div ng-controller="DemoController">


```

<table border="2" width="350px">
  <tr bgcolor="skyBlue">
    <th><a href="#" ng-click="x=1x; y='sid'">StudentId</a></th>
    <th><a href="#" ng-click="x=1x; y='sname'">StudentName</a></th>
    <th><a href="#" ng-click="x=1x; y='course'">CourseName</a></th>
  </tr>
  <tr bgcolor="pink" ng-repeat="item in students | orderBy:y:x">
    <td>{{ item.sid }}</td>
    <td>{{ item.sname }}</td>
    <td>{{ item.course }}</td>
  </tr>
</table> </div>
</body>
</html>

```

Example:-

<!-- create a webpage to apply searching on student details by using filter option of AngularJS -->

```

<html>
<head>
  <script src="angular.min.js"></script>
  <script>
    var obj = angular.module("myApp", []);
    obj.controller("DemoController", function($scope)
    {
      $scope.students = [
        {sid:1025, sname:'scott', course:'AngularJS'},
        {sid:1026, sname:'Abram', course:'HTML'},
        {sid:1027, sname:'Smith', course:'jQuery'},
        {sid:1028, sname:'James', course:'AngularJS'},
        {sid:1029, sname:'Adam', course:'Javascript'}
      ];
    });
  </script>

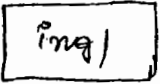
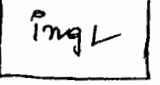
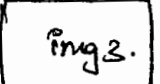
```

```

    $scope.s1 = "";
    $scope.s2 = "";
  });
</script>
</head>
<body ng-app="myApp">
  <h1> filter option - Demo4 </h1>
  <br>
  <div ng-controller="Democontroller">
    Enter Student Name to Search:
    <input type="text" ng-model="s1" /> <br><br>
    select course to search:
    <select ng-model="s2">
      <option> AngularJs </option>
      <option> JQuery </option>
      <option> HTML </option>
      <option> Javascript </option>
      <option value=""> All courses </option>
    </select> <br><br>
    <table border="1" border width="350px">
      <tr bgcolor="SkyBlue">
        <th> studentId </th>
        <th> StudentName </th>
        <th> CourseName </th>
      </tr>
      <tr bgcolor="Pink">
        <td colspan="3">
          ng-repeat="item in Students | filter:
            { 'sname': s1, 'course': s2 }
          <td> {{ sid }} </td>
          <td> {{ sname }} </td>
          <td> {{ course }} </td>
        </td>
      </tr>
    </table>
  </div>
</body> </html>

```

Example 1 - Product Details

Product Name	Brand	Unit Price	Picture
<input type="text"/>	<input type="text"/>	<input type="text"/>	
Dell Laptop	Dell	\$45,000.	
Sony laptop.	Sony.	\$40,000	
Lenovo Laptop	lenovo	\$35,000	

Create a webpage to organize products Data so, that it allows the user to search & sorting based on the given scenarios.

```
<html>
```

```
<head>
```

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

```
<script>
```

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

```
obj.controller("Demo Controller", function($scope)
```

```
{
```

```
$scope.products =
```

```
[{ ProductName: "Dell Laptop", Brand: "Dell", unitprice: "45000",  
  Picture: "Images/img1.jpg" },
```

```
{ ProductName: "Sony laptop", Brand: "sony", unitprice: "40,000",  
  Picture: "Images/img2.jpg" },
```

```
{ ProductName: "lenovolaptop", Brand: "lenovo", unitprice: 35000,  
  Picture: "Images/img3.jpg" } ];
```

```
$scope.x = false;
```

```
$scope.isReverse = false;
```

```
$scope.y = "productName";
```

```
$scope.sortColumn = "price";
```

```
$scope.s1 = " ";
```

```
$scope.s2 = " "; $scope.s3 = " ";
```

```
});
```

```

</script>
</head>
<body ng-app='myApp'>
  <h1> filters in AngularJS </h1>
  <h2>
    <div ng-controller='Democontroller'>
      <table border='2' width='500px'>
        <tr bgcolor='SkyBlue'>
          <th><a href='' ng-click='x=1;x'; y='prameproductName'>
            ProductName </a> </th>
          <th><a href='' ng-click='x=1;x'; y='Brand'> Brand </a> </th>
          <th><a href='' ng-click='x=1;x'; y='upriceunitPrice'> UnitPrice </a> </th>
          <th><a href=''> Picture </th>
        </tr>
        <tr>
          <td colspan='4'>
            <tr> bgcolor='pink' ng-repeat='item in products | orderBy:y:x
              filter: { 'ProductName': s1,
                placeholder='Product Name'
            </tr>
            <td> <input type='text' x ng-model='s1' /> </td>
            <td> <input type='text' x ng-model='s1' /> </td>
            <td> <input type='text' x ng-model='s2' />
            <td> <input type='text' x ng-model='s2' />
              <select ng-model='s2'>
                <option> Dell </option>
                <option> Sony </option> <option> All Brands </option>
                <option> lenovo </option> </select> </td>
            <td> <input type='text' x ng-model='s3' /> </td>
          </td>
        </tr>
        <tr>
          <td colspan='4'>
            <tr> bgcolor='pink' ng-repeat='item in products | orderBy:y:x
              filter: { 'productName': s1, 'Brand': s2, 'unitPrice': s3 }>
            <td> { { item.ProductName } } </td>
            <td> { { item.Brand } } </td> "INR"
            <td> { { item.unitPrice | currency:'$' } } </td>
            <td> <img alt='item image' /> </td>
          </td>
        </tr>
      </table>
    </div>
  </body>

```

```

</table>      src = "Images/ {{ item.image }} "/> </td>
</div>
</body>
</html>

```

3/11/15
Above Example

```

<html>
<head>
<script src = "angular.min.js"> </script>
<script>
    var obj = angular.module ("myApp", []);
    obj.controller ("ProductController", function ($scope)
    {
        $scope.Products =
        [ { pname: "Apple mP841 HN ultra book", Brand = "Apple", price = 119500,
          image = img1.jpg },
          { pname: "Asus EeeBook X205TA", brand = "Asus", price = 15490,
            image = img2.jpg } ];
        $scope.Reverse = false;
        $scope.sortColumn = "pname";
        $scope.findpname = "";
        $scope.findBrand = "";
        $scope.findPrice = "";
    });
</script>
</head>
<body ng-app = "myApp">
    <h1> All filter options </h1>
    <hr>
    <div ng-controller = "ProductController">

```


⇒ Events in AngularJS :-

→ similar to Javascript, AngularJS also supports all client side events.

→ In Javascript all events are prefix with "on" word.

✶ Eg:- onclick
onchange
onfocus
etc..

→ AngularJS, all event names are prefix with "ng-"

Eg:- ng-click
ng-change
ng-focus
etc..

→ Events are handled in AngularJS by attaching required functions in controller with \$scope.

→ we can store function reference in a variable that will be referred by \$scope.

Example:-

```
$scope.test = function()  
{  
    ==  
    ==  
    ==  
}
```

→ we can bind the above function with event in view as follows.

```
<input type="button" ng-click="test()" -- />
```

Example:-

<!-- create a webpage to implement login function in AngularJS --> ^{Events}

```
<html>
```

```
<head>
```

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

```
<script>
```

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

```
obj.controller('DemoController', function($scope)
```

```
{
```

```
  $scope.uid = '';
```

```
  $scope.pwd = '';
```

```
  $scope.message = ''; $scope.str = "Black"
```

```
  $scope.Login = function()
```

```
{
```

```
  if($scope.uid == "admin" && $scope.pwd == "admin123")
```

```
{
```

```
  $scope.message = "Welcome to Admin";
```

```
  $scope.str = "Green"
```

```
else
```

```
{
```

```
  $scope.message = "Invalid userid or Password";
```

```
  $scope.str = "Red"
```

```
  }  
  }  
}
```

```
</script>
```

```
<head>  
</body>
```

```
<body ng-app="myApp">
```

```
  <h1> Angular JS Event-Demo </h1>
```

```
  <hr>  
  <div ng-controller="DemoController">
```

```
    Username : <input type="text" ng-model="uid" /> <br> </br>
```

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

```
    <input type="button" ng-click="login()" value="Login" />
```

```
  <br> </br>
```

```
  <span> {{ message }} </span>
```

```
</body> <div style="color: {{ str }}">
```

```
</html>
```


4/11/15

Example 1-

<!-- create a webpage to perform Math operations using AngularJS Events -->

```
<html>
<head>
  <script src="angular.min.js"> </script>
  <script>
    var obj = angular.module('myApp', []);
    obj.controller('DemoController', function($scope)
    {
      $scope.x = 0;
      $scope.y = 0;
      $scope.result = '';
      $scope.Mystyle = {color: 'Red'};
      $scope.fz = function(arg)
      {
        var z = 0;
        switch(arg)
        {
          case '+':
            z = parseInt($scope.x) + parseInt($scope.y);
            $scope.Mystyle = {color: 'Red'}; break;
          case '-':
            z = parseInt($scope.x) - parseInt($scope.y);
            $scope.Mystyle = {color: 'Green'};
            break;
          case '*':
            z = parseInt($scope.x) * parseInt($scope.y);
            $scope.Mystyle = {color: 'Blue'};
            break;
        }
        $scope.result = "Result : " + z;
      }
    });
  </script>
</head>
```

```

</script>
</head>
<body> ng-app = "myApp"
  <h1> AngularJS Events </h1> <br/>
  <div ng-controller="Democontroller">
    Enter first Num: <input type="text" ng-model="x"> <br/> </div>
    Enter second Num: <input type="text" ng-model="y"> <br/> </div>
    <input type="button" ng-click="f2('+')" value="sum"/>
    <input type="button" ng-click="f2('-')" value="sub"/>
    <input type="button" ng-click="f2('*')" value="mul"/> <br/> </div>
    <span ng-style="MyStyle"> {{ Result }} </span>
  </div>
</body>
</html>

```

Example :-

<!-- Create a webpage to display the images based on mouse over event using AngularJS Events -->

```

<html>
  <head>
    <style>
      .c1
      {
        border : 1px solid Blue;
        height : 80px;
        width : 100px;
        margin : 3px;
        padding : 3px;
      }
      #img1
      {
        border : 2px solid Red;
        height : 230px;
        width : 300px;
        margin : 3px; padding : 3px;
      }
    </style>
  </head>
  <body>
    <div>
      <img alt="A 100x80px box with a blue border." data-bbox="100 600 200 680"/>
      <img alt="A 300x230px box with a red border." data-bbox="100 700 300 900"/>
    </div>
  </body>
</html>

```

```

</head>
<body ng-app='myApp'>
  <h1> Angular JS Events </h1> <hr>
  <div ng-controller='DemoController'>
    <img class='ci' ng-repeat='item in images'
      ng-mouseover='updateImage(item)' src='images/{{item}}' />
    <br></br>
    <img id='img1' src='images/{{str}}' />
  </div>
  <script src='angular.min.js'></script>
  <script>
    var obj = angular.module('myApp', []);
    obj.controller('DemoController', function($scope)
    {
      $scope.images = ['Image1.jpg', 'Image2.jpg', 'Image3.jpg'];
      $scope.str = 'Image1.jpg';
      $scope.updateImage = function(arg)
      {
        $scope.str = arg;
      };
    });
  </script>
</body>
</html>

```

⇒ Apply Styles with AngularJS :

→ we can apply styles to views in AngularJS in the following ways:

1. styles with expressions.
2. using "ng-style"
3. using "ng-class"

1) style with expressions :-

→ It is a regular approach. In this we store required style properties value in variables.

→ Apply these values using expressions.

Syntax:- ` --- `

Ex:- ` HelloWorld `

2) Using ng-style :-

→ ng-style is AJS directive.

→ It will take style properties as object format.

→ This object we can pass as argument also.

Syntax:- ` --- `

~~Ex~~ ` --- `

Ex:- ` {{Result}} `

3) Using ng-class :-

→ ng-class is also AJS directive.

→ It is designed to apply CSS classes.

Syntax:- ` --- `

Ex:- ` --- `

5/11/15

Example:-

<!-- Create a webpage to allow the user to select the products-

Apply the styles to selected item & display total amount -->

<html>

<head>

<style>

img

{

height: 40px;

width: 60px;

border: 1px solid Black;

padding: 2px;

}

ul

{

List-style: none;

padding: 3px;

border: 2px solid Red;

width: 400px;

}

ul li

{

background-color: pink;

padding: 3px;

border: 2px solid Red;

width: 380px;

height: 50px;

margin: 5px;

}

.cl

{

background-color: Blue;

color: white;

}

</style> </head>

```

<body ng-app="myApp">
  <h1> Product Selection Page </h1> <hr/>
  <div ng-controller="DemoController">
    <ul>
      <li ng-repeat="item in Products"
          ng-class="{c1: item.isSelected}"
          ng-click="item.isSelected = !item.isSelected">
        
        {{item.pname}}
        <span style="float:right; padding: 15px">
          {{item.price | currency: "INR"}} </span>
      </li>
      <li style="background-color: LightGreen;">
        Total Amount :
        <span style="float:right; padding: 15px;">
          {{updateTotal() | currency: "INR"}} </span>
      </li>
    </ul>
  </div>
</body>
<script src="angular.min.js"> </script>
<script>
  var obj = angular.module("myApp", []);
  obj.controller("DemoController", function($scope)
  {
    $scope.products = [
      {pname: "Nikon Coolpix S3700", isSelected: false, price: 6275,
        picture: "Image1.jpg"},
      {pname: "Canon Powershot SX400", isSelected: false, price: 9890,
        picture: "Image2.jpg"}
    ];
  });

```

```
$scope.UpdateTotal = function()
```

```
{
```

```
  var total = 0;
```

```
  var n = $scope.products.length;
```

```
  for(var i=0; i<n; i++)
```

```
  {
```

```
    if ($scope.products[i].isSelected == true)
```

```
    {
```

```
      total = total + $scope.products[i].price;
```

```
    }
```

```
  }
```

```
  return total;
```

```
};
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```

Example :-

<1 Create a Webpage to display the product details in different display formats based on selected mode using ng-switch ->

```
<html>
```

```
  <head>
```

```
    <style>
```

```
      img
```

```
    {
```

```
      padding: 2px;
```

```
      margin: 5px;
```

```
    }
```

```
  </style>
```

```
</head>
```

```

<body ng-app='myApp'>
  <h1> ng-switch Demo </h1> <hr>
  <div ng-controller='DemoController'>
    <select ng-model='DisplayMode'>
      <option>None </option>
      <option>List </option>
      <option>Details </option>
      <option>SmallIcons </option>
    </select>
    <div ng-switch='DisplayMode'>
      <div ng-switch-when='List'>
        <ol>
          <li ng-repeat='item in products'>
            {{item.pname}} </li>
          </ol>
        </div>
        <div ng-switch-when='Details'>
          <table border='2'>
            <tr ng-repeat='item in products'>
              <td> {{item.pname}} </td>
              <td> {{item.price}} </td>
              <td> {{item.picture}} </td>
            </tr>
          </table>
        </div>
        <div ng-switch-when='SmallIcons'>
          <img ng-repeat='item in products' height='40px'
            width='40px' border='1' src='Images/{{item.picture}}' />
        </div>
        <div ng-switch-default>
          <h1> It is default mode </h1>
        </div>
      </div> </div>

```



```

<script> src="angular.min.js" </script>
<script>
var obj = angular.module("myApp", [])
obj.controller("DemoController", function($scope)
{
    $scope.DisplayMode = "None";
    $scope.products =
    [{ pname: "Nikon coolpix s3700", price: 6275, picture: "Image1.jpg"},
    { pname: "Canon Powershot sx400", price: 9890, picture: "Image1.jpg"}];

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

```

6/11/15

Example:-

<!-- create a webpage to demonstrate usage of ng-show, ng-hide and ng-disabled directives -->

```

<html>
<head>
<style>
#div1
{
border: 2px solid Blue;
width: 200px;
height: 250px;
padding: 5px;
}
</style>
</head>
<body ng-app="myApp">
<h1> Special Directives </h1> <hr/>

```

```
<div ng-controller="DemoController">
```

```
<h3> usage of ng-disabled </h3>
```

```
<input type="checkbox" ng-model="x" />
```

Are you Agree Terms & Conditions?

```
<br><br>
```

```
<input type="button" ng-disabled="!x" value="Register" />
```

```
<hr></hr>
```

```
<h3> usage of ng-show </h3>
```

```
<a href="#" ng-click="y=!y"> show/hide user details </a>
```

```
<br></br>
```

```
<div ng-show="y"> This div contain user details </div>
```

```
<hr></hr>
```

```
<h3> usage of ng-hide </h3>
```

```
<a href="#" ng-click="z=!z"> hide/show Product Details </a>
```

```
<br></br>
```

```
<div id="div1" ng-hide="z">
```

```
<h3> ProductName </h3></br>
```

```
<span> Product details are present in this section </span> </br>
```

```

```

```
</div>
```

```
<script>
```

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

```
obj.controller("DemoController", function($scope)
```

```
{
  $scope.x = false;
```

```
  $scope.y = false;
```

```
  $scope.z = false;
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```

Validations in AngularJS Application using AngularJS forms

- According to webpage development, browser point of view "form is a collection of input fields".
- forms are used to collect the data from the user.
- At the time of getting data from user and submitting, we need to perform validations.
- AngularJS provides its own options to implement validations.
- AngularJS provides built-in directives, objects to perform validations at client side.

Rules that should follow to implement validations in AJS:

- 1) `<form>` tag should be placed with in the tag that is bind with `ng-controller`.

```
<div ng-controller=" -- ">  
  <form>  
  =  
  </form>  
</div>
```

- 2) `<form>` tag should have "name" and "novalidate" attributes.

```
<form name="f1" novalidate>  
  =  
  </form>
```

- 3) Input controls should have name attribute.

```
<input type="Text" name="t1" --- />
```

- 4) Prepare custom Error Message using `` tag with `ng-show` directive.

```
<span ng-show=" " > Error Message </span>
```

5) Identify the failures of validations by using pre-defined AJS options:

Ex) \$error.

usage: ft.tl.\$error.required.

6) Apply validations to input controls by using AJS Validation directives.

ng-required

ng-maxlength

ng-minlength

ng-pattern -- etc.

Example 4

< Create a webpage to implement AngularJS Validations ->

```
<html>
  <head>
    <style>
      .C1
      {
        color: Red;
      }
    </style>
    <script src="angular.min.js"> </script>
    <script>
      var obj = angular.module("myApp", []);
      obj.controller("Demo Controller", function($scope)
      {
        $scope.fname = "Ravi";
        $scope.lname = "Kumar";
      });
    </script>
  </head>
  <body ng-app="myApp">
    <h1>AngularJS Validations </h1>
    <hr>
```

```

<div ng-controller="DemoController">
  <form name="f1" novalidate>
    firstName: <input type="text" name="t1" ng-model="fname"
      ng-required="true"/>
    <span class="c1" ng-show="f1.t1.$error.required">
      firstName is required... </span>
    <br><br>
    LastName: <input type="text" name="t2" ng-model="lname"
      ng-required="true"/>
    <span class="c1" ng-show="f1.t2.$error.required">
      LastName is required... </span> <br><br>
    <input type="button" value="submit" name="b1"/> <br><br>
  </form>
</div>
</body>
</html>

```

~~7/11/18~~ Note:-

\$dirty: This property gives whether user is used the i/p field (or) not.

7/11/18

\$invalid:-

- This property can be applied at form level to know all validations (or) passed (or) field.
- If any one validation in field it gives true, all validations are valid & returns false.
- we can use this option to disable/enable submit button.

Ex:

```
<input type="submit" ng-disabled="f1.$invalid"/>
```

⇒ AngularJS Validations :

```
<html>
<head>
<style>
    .cl
    {
        color: Red;
    }
</style>
<script src="angular.min.js"> </script>
<script>
    var obj = angular.module('myApp', []);
    obj.controller('DemoController', function($scope)
    {
        $scope.fname = " ";
        $scope.lname = " ";
        $scope.pincod = " ";
        $scope.emailid = " ";
        $scope.city = " ";
    });

```

```
</script>
<head>
<body ng-app="myApp">
  <h1>AngularJS Validations </h1> </hr>
  <div ng-controller="DemoController">
    <form name="f1" novalidate>
      firstName <input type="text" name="t1" ng-model="fname"
        ng-controller="required" />
      <span class="c1" ng-show="f1.t1.$error.required && f1.t1.$dirty">
        firstName is Required </span> <br/>
      LastName <input type="text" ng-model="lname" name="t2"
        ng-required="true" />
```


LastName is Required

city: <input type="text" ng-model="city" name="t3" ng-maxlength="10"
ng-minlength="3" />

cityname should not be more than 10 character

city name should be 3character

pincode: <input type="text" ng-model="pincode" name="t4"
ng-pattern="/^[1-9]{6}\$/"/>

plz enter valid pincode

Email: <input type="email" ng-model="emailid" name="t5" />

plz enter valid Email id

<input type="submit" value="submit" name="b1" ng-disabled=
"f1.\$invalid"/>

</form>

</div>

</body>

</html>

→ Modules in AJS!

What is module in AJS?

→ Module is consider as a container for the different programming items of application like controllers, filters, services, directives etc.,

→ Module specify how an application should be bootstrapped.
(initializing your application items).

→ Module can implement separation of code features of MVC.

→ Module is an important part of the AJS dependency injection system.

Note:- "ng" is a predefined global module that will be prepared by AngularJS framework.

Q- what we can define in a AJS Module?

Q- What are the content of the AJS Module?

→ we can define different items of the AngularJS application.

→ AngularJS module contains any of the following items-

- 1) controllers
- 2) Services
- 3) factories.
- 4) Directives
- 5) filters.
- 6) Configurations.
- 7) Routes.

Q- How to create Modules in AJS?

Var obj = angular.module ("moduleName", []);

Arguments:-

- 1) Module Name - string
- 2) Array of dependent modules - string array.

Creating module with dependant Modules

```
var obj1 = angular.module("myApp", []);  
var obj2 = angular.module("myApp", ["myapp"]);
```

9/11/15

<!-- create a webpage to implement a module that will dependent on another module to organize employee data using AngularJS -->

<html>

<head>

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

<script>

var obj1 = angular.module("myApp1", ["PersonController"]);

obj1.controller("PersonController", function(\$scope)

{

\$scope.name = "Scott";

\$scope.age = 30;

});

var obj2 = angular.module("myApp2", ["myApp1"]);

obj2.controller("EmpController", function(\$scope)

{

\$scope.empno = 1234;

\$scope.job = "Manager";

\$scope.sal = 25,000;

});

</script>

</head>

<body ng-app="myApp2">

<h1> AngularJS Modules </h1>

<div ng-controller="PersonController">

Employee Name : {{name}}

Employee Age : {{age}}

</div>

```

<div Ag-controller = "Emp Controller">
  Employee Number : {{empno}} <br>
  Employee Job : {{job}} <br>
  Employee Salary : {{sal}} <br>
</div>
</div>
<body>
</html>

```

Angular JS - Services

→ AngularJS provides built-in services to perform required operations across multiple controllers.

→ What is Service in AngularJS?

→ Service is reusable component across multiple controllers.

→ A service can have variables & functions.

→ Service is defined the members as instance members by using "this" keyword.

```

this.Variable = " ";
this.fn-name = function()
{
    //
};

```

What are the examples of pre-defined service in AJS?

- 1) \$http - It is used to communicate with server asynchronously (AJAX).
- 2) \$route - It is used to implement routing concept for single page applications.

How to create services in AngularJS?

→ we can create services in AJS with "module" object reference (myApp).

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

→ Module object provides two functions to create user defined Service:

1) service()

2) factory()

Syntax for creating service()

1) using service()

```
myApp.service('ServiceName', function()
```

```
{
```

```
    // declare variables (or) functions with this keyword.
```

```
});
```

How to use service in controller?

→ we need to specify service details at the time of creating controller.

```
Ex: myApp.controller('ControllerName', function($scope, ServiceName)
```

```
{
```

```
    // Access the members of service by using ServiceName.
```

```
});
```

Example 1

<1> create web pages to share the programming logic by using AngularJS service →

Required files

1) angular.min.js

2) app.js

3) ServiceDemo1.js

4) ServiceDemo2.js

1) app.js

// 1. Creating module

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

// 2. creating service

```
myApp.service('TestService', function()
```

```
{
```

```
  this.sum = function(x, y)
```

```
  {
```

```
    var z = parseInt(x) + parseInt(y);
```

```
    return z;
```

```
  }
```

```
  this.multiply = function(x, y)
```

```
  {
```

```
    var z = parseInt(x) * parseInt(y);
```

```
    return z;
```

```
  }
```

```
});
```

// 3. Creating MathController

```
myApp.controller('MathController', function($scope, TestService)
```

```
{
```

```
  $scope.x = 10;
```

```
  $scope.y = 20;
```

```
  $scope.result = 0;
```

```
  $scope.getSum = function()
```

```
  {
```

```
    $scope.result = TestService.sum($scope.x, $scope.y);
```

```
  };
```

```
  $scope.getMultiply = function()
```

```
  {
```

```
    $scope.result = TestService.multiply($scope.x, $scope.y);
```

```
  };
```

```
});
```

// 4. Creating Product Controller

```
myApp.controller("Product Controller", function($scope, TestService)
{
    $scope.pname = "Printer";
    $scope.uprice = 2500;
    $scope.qty = 1;
    $scope.result = 0;

    $scope.getTotal = function()
    {
        $scope.result = TestService.multiply($scope.uprice, $scope.qty);
    }
});
```

2) Service Demo1. Html

```
<html>
<head>
    <script src="angular.min.js"></script>
    <script src="app.js"></script>
</head>
<body ng-app='myApp'>
    <h1>AngularJS service - MathController </h1>
    <hr></hr>
    <div ng-controller="MathController">
        First Value: <input type="text" ng-model="x"/> <br></br>
        Second Value: <input type="text" ng-model="y"/> <br></br>
        <button ng-click="getSum()"> Sum </button>
        <button ng-click="getMultiply()"> Multiply </button> <br></br>
        <span> Result: {{result}} </span>
    </div>
</body>
</html>
```

3) Service Demo 2. Html

```
<html>
<head>
  <script src="angular.min.js"></script>
  <script src="app.js"></script>
</head>
<body ng-app="myApp">
  <h1> Angular JS Service - ProductController </h1>
  <hr></hr>
  <div ng-controller="ProductController">
    Product Name : <input type="text" ng-model="pname"/> <br></br>
    Unit Price : <input type="text" ng-model="uprice"/> <br></br>
    Quantity : <input type="text" ng-model="qty"/> <br></br>
    <button ng-click="getTotal()"> Get Total Amount </button> <br></br>
    <span> Total Amount : {{ result }} </span>
  </div>
</body>
</html>
```

Creating Service with factory Method

Syntax: myApp.factory("Testfactory", function()

```
{
  var obj = { };
  obj.sum = function(x, y)
  {
    =
  }
  obj.multiple = function(x, y)
  {
    =
  }
  return obj;
}
```

10/11/15 creating service using factory()

myApp.factory("TestService", function()

```
{  
  var obj = {};  
  obj.sum = function(x, y)  
  {  
    var z = parseInt(x) + parseInt(y);  
    return z;  
  }  
  obj.multiply = function(x, y)  
  {  
    var z = parseInt(x) * parseInt(y);  
    return z;  
  }  
  return obj;  
});
```

Custom filters in AngularJS

- AJS Framework provides pre-defined filters to filter application data as per our requirement.
- If you want to create new filters as per your requirement, then you have to develop custom filters.
- ⇒ What is custom filters?
 - filters that we develop as per our customized requirement are called "custom filters".

Ex:-

- Display names starts with "A".
- filter only Even numbers.
- filter only five multiples.
- etc.,

How to create custom filters?

→ we can create custom filters with module object reference.

→ Angular object module object provides a method called "filter()" to create custom filters.

```
Ex:- myApp.filter("filterName", function()  
    {  
        // required code  
    });
```

Example :-

<!-- Create a webpage to implement custom filters for following requirements -->

1. Reversing the given string
2. Filtering items from array which are starts with specific character.

&

i) App.js :-

// creating module

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

// creating custom filter to reverse the string

```
myApp.filter("Reverse", function()
```

```
{
```

```
    var fn-test = function(input)
```

```
    {
```

```
        var ar = input.split('');
```

```
        ar.reverse();
```

```
        var output = ar.join('');
```

```
        return output;
```

```
    };
```

```
    return fn-test;
```

```
});
```


// custom filter to filter ^{items from} array ~~items~~ which are starts with specific character.

```
myApp.filter("StartsWith", function()
```

```
{
  var fn_test = function(input, option1)
  {
    var output = [];
    for (var i=0; i < input.length; i++)
    {
      if (input[i].charAt(0) == option1)
      {
        output.push(input[i]);
      }
    }
    return output;
  };
  return fn_test;
});
```

```
myApp.controller("DemoController", function($scope)
```

```
{
  $scope.username = "Hello World";
  $scope.courses = ["AngularJS", "ASP", "ASP.NET", "JSP", "PHP", "HTML",
    "SQL", "SQL server", "Javascript"];
});
```

2) custom-filter-Demo1-HTML

```
<html>
<head>
  <script src="angular.min.js"></script>
  <script src="app.js"></script>
</head>
```

```

<body ng-app='my-App'>
  <h1> Angular JS Custom filters </h1> </hr>
  <div ng-controller='DemoController'>
    username : {{ username | Reverse }}
    <br><br>
    <span> filtered Courses </span>
    <ol>
      <li ng-repeat='item in courses | StartsWith : 'A' '>
        {{ item }}
      </li>
    </ol>
  </div>
</body>
</html>

```

```

ex)
myApp.filter('TempEmps', function()
{
  var fn-Test = function(input)
  {
    var output = [];
    for (i=0; i < input.length; i++)
    {
      if (input[i].type == "PartTime")
      {
        output.push(input[i].ename);
      }
    }
    return output;
  };
  return fn-Test;
});

```

```
$scope.emps = [ { empno: 1015, ename: 'Scott', type: 'fulltime' },  
                 { empno: 1026, ename: 'James', type: 'parttime' } ];
```

12/11/15

Single Page Applications with Angular JS Router

- SPA means loading the content from different pages dynamically into single page.
- In SPA development, maintaining routes details is main challenging task
- If you use other frameworks, they do not provide much features to organize routes & history, state, etc.
- AJS will provide all the features to implement SPA with required options.

Steps to Implement

- 1) Include Angular JS routing library.
- 2) Module creation with depends on "ngRoute" module.
- 3) Provide config details to mapping the routes using "\$routeProvider".
- 4) Define required controller for the Views.
- 5) In HTML design, prepare required hyperlinks & div tag to load the dynamic content.
- 6) Provide "ng-view" directive to div tag.
`<div ng-view> </div>`

<1 -- create web Application to implement SPA using AJS Routing -->

Required files

(ASP-Net)

- 1) /scripts/angular.min.js
- 2) /scripts/angular-route.min.js
- 3) /scripts/app.js
- 4) index.html
- 5) student.html
- 6) courses.html

1) app.js :-

// creating Module object

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

// create configuration with module object it provides route details.

```
myApp.config(function($routeProvider)
```

```
{
```

```
  $routeProvider.when("/Route1", { templateUrl: "Student.html",  
                                controller: "StudentController"  
  });
```

```
  .when("/Route2", { templateUrl: "Courses.html",  
                    controller: "CoursesController"  
  });
```

```
});
```

// controller for student.html.

```
myApp.controller("StudentController", function($scope)
```

```
{
```

```
  $scope.sname = "scott";
```

```
  $scope.course = "Angular JS";
```

```
});
```

// Controller for courses.html.

myApp.controller("CoursesController", function(\$scope)

{
 \$scope.courses = ["AngularJS", "PHP", "jQuery", "JSP", "ASP"];
});

q) Index.html

<html>

<head>

<style>

 #div1

 {

 width: 50px;

 border: 2px Solid Green;

 background-color: Light Yellow;

 }

</style>

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

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

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

</head>

<body ng-app="myApp">

<h1> SPA with AJS Routing Demo </h1> <hr/>

 Student Details

 Course Details

<hr/>

<div id="div1" ng-view>

</div> <hr/>

</body>

</html>

3) student.html

<h3> students Details </h3> <hr/>

 StudentName : {{sname}}

 CourseName : {{course}} .

4) Courses.html

<h3> course Details </h3>

<h4>

<li ng-repeat="item in courses">
{{item}}

13/11/15

AJAX programming in AngularJS

What is AJAX?

→ Asynchronous Javascript And Xml.

→ AJAX is a client side programming technique used to communicate with server Asynchronously.

→ Note This Technique is first identified by the Google Team & used

Advantages of AJAX:-

- 1) Quick Response.
- 2) Reduced the burden on the server.
- 3) Reduce the network Traffic.

- 4) Comfortable for user. No need to wait more time for response.
- 5) Partial page updates. updates required portion of the page.

Implementing AJAX programming

- AJAX is a clientside programming.
- It is implemented with the help of JavaScript.
- If we implement with basic JavaScript, it will be difficult and time consuming programming.
- In order to make ajax programming is easy, there are several JavaScript libraries available.

Ex: 1) AngularJS

- 2) JQuery
- 3) Dojo
- 4) MooTools etc.,

Implementing AJAX programming in AngularJS

- AngularJS framework provides special service called "\$http" to implement ajax programming.
- By using "\$http" service we can communicate with server asynchronously.
- In ajax programming, your JavaScript code is responsible to send the request, receive the response and display in the web page.

usage of \$http service

```
$http({ url: "serverPageUrl",
        method: 'RequestType',
        dataType: 'Json',
        data: '',
        headers: { "Content-Type": "application/json" }
    }).success(function(response) {
        // code
    });
```

<!-- Create a web Application to display the Dept details from server
using AJAX communication with AngularJS --> (ASP.net)

1) AJAX Demo1.html

<html>

<head>

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

<script>

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

obj.controller('Ajax Controller', function(\$scope, \$http)

{

\$scope.depths = [];

\$scope.x = "Deptno";

\$scope.y = false;

\$scope.getServerData = function()

{

\$http({ url: "WebService1.aspx/GetDepths",

method: 'POST';

dataType: 'JSON';

data: '';

headers: { "contentType": "application/json" };

});

.success(function(response)

{

\$scope.depths = response.d;

});

});

});

</script>

</head>


```

<body ng-app="myApp">
  <h1>Ajax Demo</h1> </hr>
  <div ng-controller="AjaxController">
    <button ng-click="getServerData()"> Get Dept Data </Button>
    <br></br>
    <table border="2" width="400" align="center">
      <tr>
        <th><a href="#" ng-click="x='Deptno'; y=!y"> Deptno</a></th>
        <th><a href="#" ng-click="x='Dname'; y=!y"> Dname</a></th>
        <th><a href="#" ng-click="x='Loc'; y=!y"> Loc</a></th>
      </tr>
      <tr>
        <td> {{ item.Deptno }} </td>
        <td> {{ item.Dname }} </td>
        <td> {{ item.Loc }} </td>
      </tr>
    </table>
  </div>
</body>
</html>

```

2) WebService.asmx.cs

using System.Data;

using System.Data.SqlClient;

namespace webApplication1

{

[WebService]

[ScriptService]



```
public class web.Services : WebService.
```

```
{
```

```
[WebMethod]
```

```
public List<Dept> GetDepts()
```

```
{
```

```
string connStr = "Server=.; database = Sathya; Trusted_Connection=true";
```

```
string cmdText = "Select * from Dept";
```

```
SqlDataAdapter da = new SqlDataAdapter (cmdText, connStr);
```

```
DataTable dt = new DataTable();
```

```
da.Fill(dt);
```

```
List<Dept> deptList = new List<Dept>();
```

```
foreach (DataRow item in dt.Rows)
```

```
{
```

```
Dept obj = new Dept();
```

```
obj.Deptno = (int) item[0];
```

```
obj.Dname = (intstring) item[1];
```

```
obj.Loc = (intstring) item[2];
```

```
deptList.Add(obj);
```

```
}
```

```
return deptList;
```

```
}
```

```
}
```

```
}
```

```
3) Dept.cs
```

```
public class Dept
```

```
{
```

```
public int Deptno {get; set;}
```

```
public string Dname {get; set;}
```

```
public string Loc {get; set;}
```

```
}
```

14/11/15

Custom Directives in AngularJS

- AngularJS directives are controls like the rendering of the HTML markup inside an AngularJS application (ng-app).
- Custom directives are used in AngularJS to extend the functionality of HTML.
- Custom directives are created by using "directive()" function.
- AIS module object provides "directive()" function.
- A custom directive simply replaces the element for which it is applied.
- AngularJS application during initialization time (loading time) executes the custom directive related functions.

Types of Attr directives

- The type of a directive specified how the directive is used.

1) Element directive

- An element directive is activated when AngularJS finds a matching HTML element in the HTML template (view).

Ex: <directive> data </directive>

2) Attribute directives

- An attribute directive is activated when AngularJS finds a matching HTML attribute in view.

Ex: <tag directive="data"> </tag>

Syntax for creating custom Directive:-

```
var myApp = angular.module("myApp", []);  
myApp.directive('directiveName', function()  
{  
    var obj = { };  
    ≡  
    return obj;  
});
```

→ Option that we assign to directive object.

- 1) restrict: "EA" / "E" / "A"
- 2) template: " "
- 3) replace: true / false

Ex:- Generating a webpage to develop custom directive by using AngularJS

1) custom-Directive-Demo1.html

```
<html>  
<head>  
    <script src="angular.min.js"></script>  
    <script src="app.js"></script>  
</head>  
<body ng-app="myApp">  
    <h1> Angular JS custom Directives </h1>  
    <hr></hr>  
    <div ng-controller="TestController">  
        <div welcome-Directive>  
            <div> <hr>
```

```
<welcome - Directive>
</welcome - Directive>
</div>
</body>
</html>
```

2) app.js

// Creating module

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

// Creating directive

```
myApp.directive('welcomeDirective', function()
```

```
{
```

```
  var obj = {} ,
```

```
  obj.restrict = 'EA';
```

```
  obj.template = '<h1> Welcome to {{sname}} </h1>';
```

```
  return obj;
```

```
});
```

// Creating required controller

```
myApp.controller('TestController', function($scope)
```

```
{
```

```
  $scope.sname = 'scott';
```

```
});
```

<1- Create a webpage to develop custom directive to process collections by using template url option in AngularJS →
(Asp. Net)

1) Index.html

```
<html>
  <head>
    <script src="scripts/angular.min.js"></script>
    <script src="scripts/app.js"></script>
  </head>
  <body ng-app="myApp">
    <h1> AngularJS custom Directives filters </h1> <hr></hr>
    <div ng-controller="StudentController">
      <custom-list> </custom-list>
    </div>
  </body>
</html>
```

2) app.js

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

```
// custom directive
```

```
myApp.directive("customList", function()
```

```
{
  var obj = {};
  obj.restrict = "EA";
  obj.templateUrl = "Templ.html";
  return obj;
}
```

```
});
```

// Controller

```
myApp.controller('Student Controller', function($scope)
{
    $scope.students = [ { sname: 'scott1', Course: 'AngularJS1' },
                        { sname: 'scott2', Course: 'AngularJS2' },
                        { sname: 'scott3', Course: 'AngularJS3' } ];
});
```

3) Templ. html

```
<table border="2" bgcolor="Pink" width="300px" align="center">
    <tr>
        <th> studentName </th>
        <th> Course Name </th>
    </tr>
    <tr ng-repeat="item in students">
        <td> {{ item.sname }} </td>
        <td> {{ item.Course }} </td>
    </tr>
</table>
```

Note: To apply alternate background color of table rows.

Templ.html <tr ng-class="{ c1: \$index % 2 == 0, c2: \$index % 2 == 1 }">

index.html <style>
 .c1
 {
 background-color: Yellow;
 }
 .c2
 {
 background-color: Green;
 }
</style>

* Integration of Bootstrap to develop Angular JS v2

<http://getbootstrap.com/> download (Views)

Bootstrap

What is Bootstrap?

- Bootstrap is a client-side programming framework for web Applications.
- It provides predefined styles in order to provide more richness to your webpages.
- Bootstrap files need to be download from the following website.

<http://getbootstrap.com/>

Required files

- bootstrap.min.css.

Including Bootstrap in webpage

```
<link href="bootstrap.min.css" rel="stylesheet">
```

Ex:- Applying bootstrap CSS classes to button.

```
<input type="button" class="btn btn-default" value="Register"/>
```

→ Class Names that you can apply for Buttons

- 1) btn btn-default
- 2) btn btn-primary.
- 3) btn btn-success
- 4) btn btn-info
- 5) btn btn-danger
- 6) btn btn-warning.

<!-- create a webpage to apply bootstrap styles to buttons -->

```
<html>
  <head>
    <link href="bootstrap.min.css" rel="stylesheet" />
  </head>
  <body>
    <h1> Bootstrap Demo </h1>
    <hr></hr>
    <button class="btn"> Get Total </button>
    <button class="btn btn-primary"> Register </button>
    <button class="btn btn-success"> Login </button>
  </body>
</html>
```

Note:-

→ Applying bootstrap styles to tables using AngularJS "ng-class"

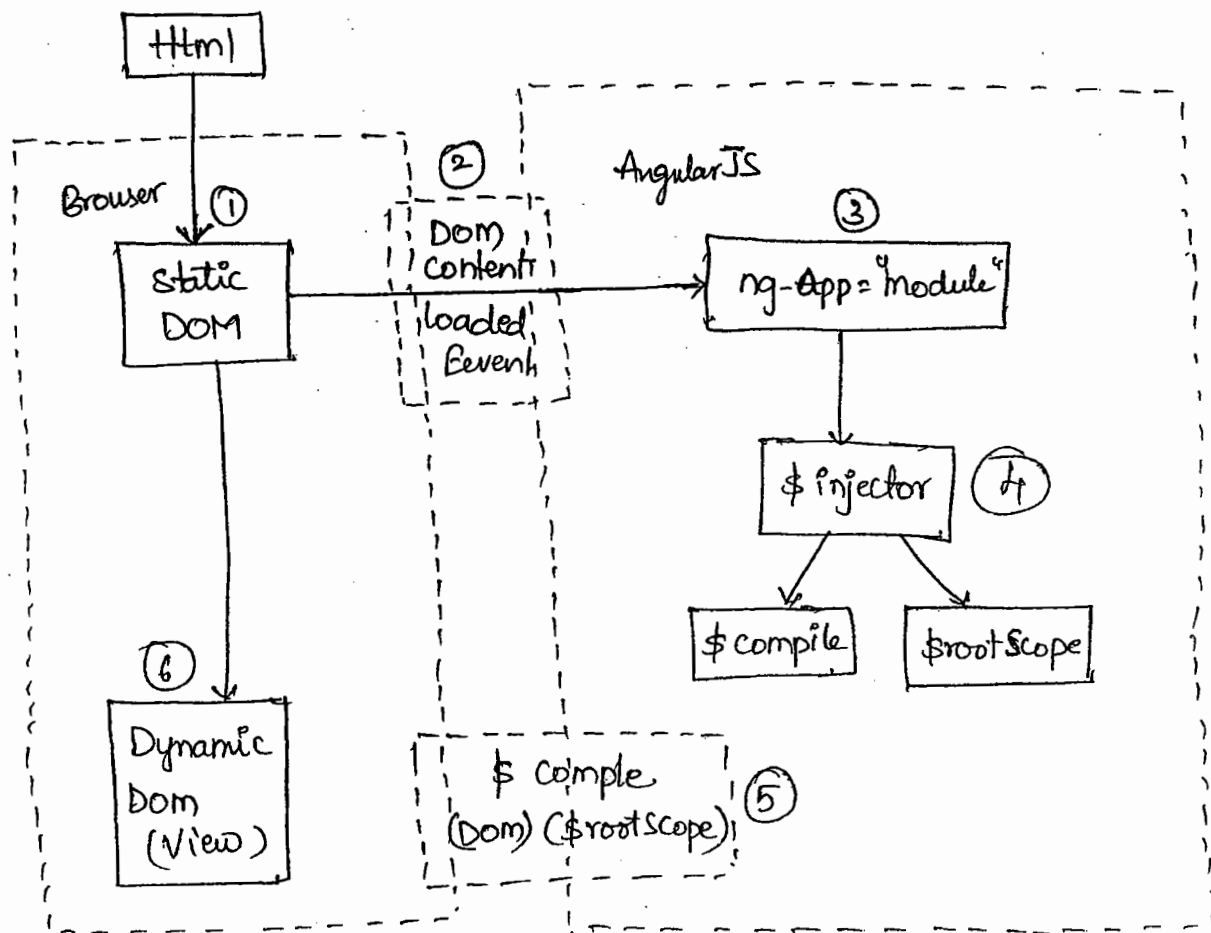
```
<table class="table">
  <tr ng-class="{ success: true }">
    ---
  </tr>
</table>
```

Interview Questions

Q:- what is AngularJS Bootstrap processing?

Q:- Explain the Execution process AngularJS Application?

Q:- Explain the auto-bootstrap initialization process of AngularJS Application?



Q:- How databinding works internally in AngularJS?

→ Angular objects uses the following predefined methods to perform this binding.

- 1) \$watch()
- 2) \$apply()

→ `$watch()` will verify continuously the model variable whether the values are updated or not.

→ `$apply()` will perform required task to update the values to view.

Q:- what is \$rootScope? what is the difference b/w \$rootScope & \$scope?

→ `$rootScope` is global for all controllers for entire angular JS application.

→ It is common for all controllers.

→ `$scope` will be created for every controller object.

→ `$scope` will be applied to specific controller.

Email : sir.narasimha@yahoo.com

Fb : <http://www.facebook.com/groups/dotnetnarasimha/>

