Lesson 1- What is AngularJS?

AngularJS is a javascript framework to build websites.

JS **frameworks** are collections of **JavaScript** code libraries that provide developers with pre-written JS code to **use for** routine programming features and tasks

Its **opensource** that means anyone can add new features in its code and mark the errors in existing one and it is available to everyone.

It builds:

- SPA (Single Page Applications) A ***single-page application*** (***SPA***) is a web ***application*** or ***website*** that interacts with the user by dynamically rewriting the current web ***page*** with new data from the web server, instead of the default method of the browser loading entire new ***pages***.
- Line of Business Applications **Line of business** (LOB) is a general term which refers to a product or a set of related products that serve a particular customer transaction or **business** need. In some industry sectors, like insurance, "**line of business**" also has a regulatory and accounting definition to meet a statutory set of insurance policies.

Benefits

- Dependency Injection
- Two way Data Binding (between model and view)
- Testing
- MVC
- Directives and Filters etc

*We can build an application using Model View Controllers and Angularjs will bind them automatically.

*Angularjs.org for downloading script of angularjs(angularjs.min.js and place it in Scripts folder) and cdn link.

Getting started with angular

- Add a reference to angular script in html page
- Add ng-app in body or div this denotes the start of angularjs application. Angular manages the page when it finds ng-app in body or div.
- For example, this page is handled by angularjs.

```
      <!DOCTYPE html>

<html>
    <head>
        <title>Sample Page</title>
        <script src="Scripts/angular.min.js"></script>
    </head>

    <body ng-app="">
        <div>
            10 + 20 = {{10+20}}
```

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

{{}} these double curly braces are binding expression anything inside it is considered as an angular expression and evaluated when page is loaded.

Anything inside ng-app is handled by angularjs nothing else on the page.

Valid Expressions in angularjs

- All mathematical expressions
- Boolean expressions
- JSON Object (the below code will get a page with “David” on it)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>
        <script src="Scripts/angular.min.js"></script>
    </head>

    <body ng-app="">
        <div>
            {{{name:"David", age:'30'}.name}}
        </div>
    </body>
</html>
```

- Iterators or arrays.

```
!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>
        <script src="Scripts/angular.min.js"></script>
    </head>

    <body ng-app="">
        <div>
            <!--This will return Sara from the array-->
            {{['David', 'Pam','Sara'][2]}}
        </div>
    </body>
</html>
```

Lesson 2- Angular modules and controllers

An **AngularJS module** defines an application. The module is a container for the different parts of an application. The module is a container for the application controllers. Controllers always belong to a module.

Module is the **main()** of angularjs. It depicts the flow and linking of items in an angular application.

The controller in AngularJS is a JavaScript function that maintains the application data and behaviour using $scope object. You can attach properties and methods to the $scope object inside a controller function, which in turn will add/update the data and attach behaviours to HTML elements. The $scope object is a glue between the controller and HTML.

In short, a controller manages the things between view and model. Model is just data and view are html page.

For example

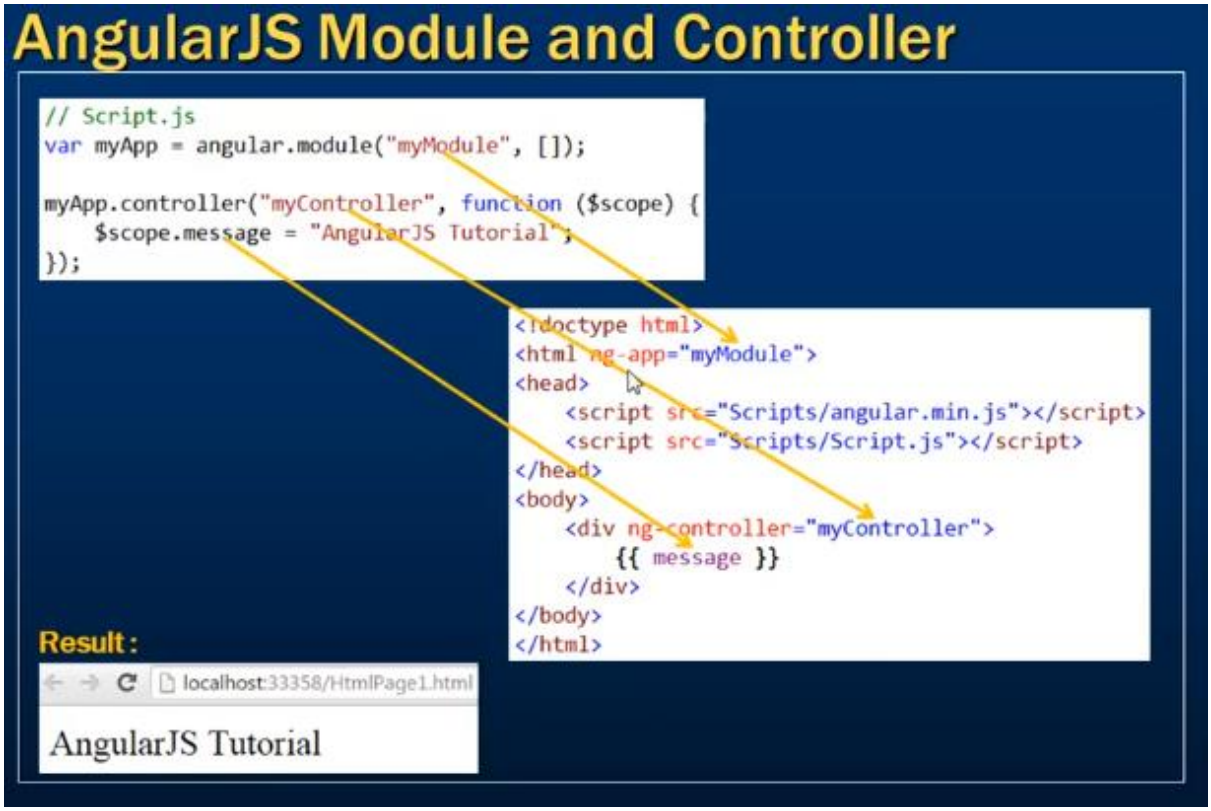
We will create a Script.js file in scripts and create a module and a controller in that module later we can use it in our view.

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

myApp.controller("myController",function($scope)
{
    $scope.message="AngularJS sample message";
});
```

Our new html file with module and controller looks like this it will render the message assigned in controller function in script.js file.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">
           {{message}}
        </div>
    </body>
</html>
```

Anything assigned to $scope can be passed through controller to view or vice versa.



Lesson 3- Controllers in AngularJS

Controller in AngularJS

Controller manipulates the DOM directly by scope object.

How to add behavior to a scope object?

We will add a complex object in a controller and attach it with scope to use it later in the views like so:

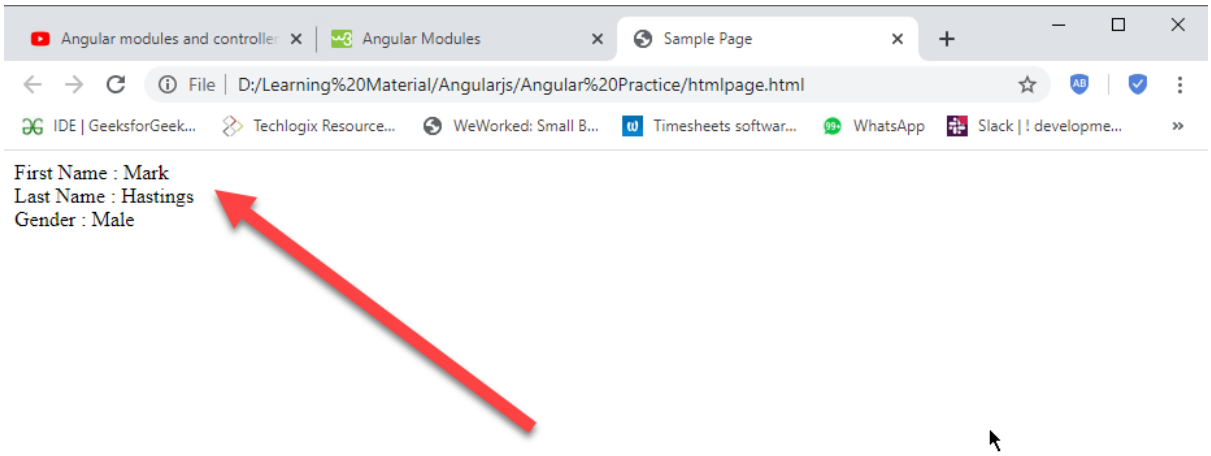
Script.js

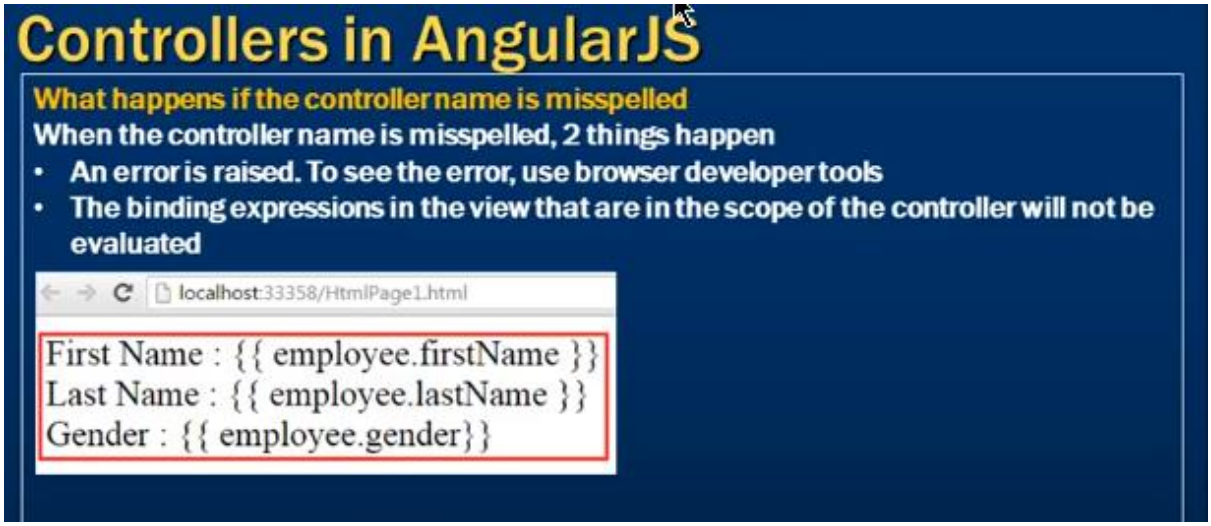
```
var myApp = angular.module("myModule",[]);

myApp.controller("myController",function($scope)
{
    var employee={
        firstName:"Mark",
        lastName: 'Hastings',
        gender:'Male'
    };
    $scope.employee= employee;
});
```

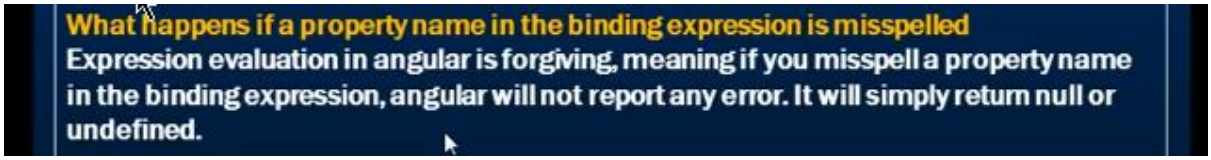
Htmlpage:

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">
           <div>First Name : {{employee.firstName}}</div>
           <div>
               Last Name : {{employee.lastName}}
           </div>
           <div>
               Gender : {{employee.gender}}
           </div>
        </div>
    </body>
</html>
```

Web page loaded:







Creating module and registering a controller with it in one line by method chaining mechanism.

```
var myApp = angular
.module("myModule",[])
.controller("myController",function($scope)
{
    var employee={
        firstName:"Mark",
        lastName: 'Hastings',
        gender:'Male'
    };
    $scope.employee= employee;
});
```

Lesson 4- AngularJS ng-src directive

ng-src directive in angular.

404 error???

Binding expression in img src produces this error because server loads the image at the start but the binding expression is not evaluated since then.

Htmlpage

```html
<!doctype html>
<html ng-app="myModule">
<head>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/Script.js"></script>
</head>
<body>
    <div ng-controller="myController">
        <div>
            Name : {{ country.name }}
        </div>
        <div>
            Capital : {{ country.capital }}
        </div>
        <div>
            <img src="{{ country.flag }}"
                 alt="{{ country.name + ' Flag' }}"
                 style="height:100px; width:200px" />
        </div>
    </div>
</body>
</html>
```

Scripts.js

Image is present in the given path in flag

```javascript
/// <reference path="angular.min.js" />

var myApp = angular
                .module("myModule", [])
                .controller("myController", function ($scope) {
                    var country = {
                        name: "USA",
                        capital: "Washington, D.C.",
                        flag: "/Images/usa-flag.png"
                    };

                    $scope.country = country;
                });
```

DOM is parsed a request is generated to server to load that image but the angular binding expression is not evaluated by then so it produces an error. The 2nd request is generated when angular binding expression is evaluated then the image is loaded.

To fix this error we use ng-src directive.

Instead of src in img tag.

```
<!doctype html>
<html ng-app="myModule">
<head>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/Script.js"></script>
</head>
<body>
    <div ng-controller="myController">
        <div>
            Name : {{ country.name }}
        </div>
        <div>
            Capital : {{ country.capital }}
        </div>
        <div>
            <img ng-src="{{ country.flag }}"
                 alt="{{ country.name + ' Flag' }}"
                 style="height:100px; width:200px" />
        </div>
    </div>
</body>
</html>
```

Lesson 5- Two-way Data Binding

Two way data binding keeps the model and view in sync all the time.

ng-model directive is used to change the data of model as changed in the view.

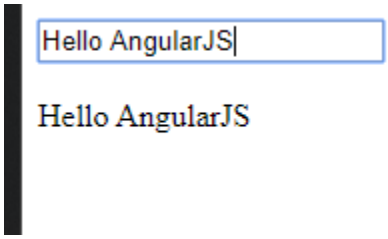
Htmlpage:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">
           <input type="text" ng-model="message"/>
           <br/><br/>
           <div>
                {{ message }}
           </div>
        </div>
    </body>
</html>
```

Script.js

```
var myApp = angular
.module("myModule",[])
.controller("myController",function($scope)
{

    $scope.message= "Hello Angular!";
});
```

Web page loaded:



It shows if we change the value of input the value of next div changes which is bind to a scope message that means ng-model changes the value of message by DOM manipulation.

*it also work with complex objects like employee in previous example.

Lesson 6- AngularJS ng-repeat Directive

Ng-repeat is used to iterate through an array an example is:

Script.js

```
var myApp = angular
.module("myModule",[])
.controller("myController",function($scope)
{
    var employees=[
        {firstName:"ABC", lastName:"dshv", gender:"Male", salary:55000},
        {firstName:"bhhj", lastName:"dshv", gender:"Female", salary:65000},
        {firstName:"nbvf", lastName:"dshv", gender:"Male", salary:54000},
        {firstName:"hjbc", lastName:"dshv", gender:"Female", salary:85000}
    ];
    $scope.employees=employees;
});
```

Htmlpage

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>
```

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

    <body ng-app="myModule">
        <div ng-controller="myController">

           <div>
                <table>
                    <thead>
                        <th>
                            Firstname
                        </th>
                        <th>
                            Lastname
                        </th>
                        <th>
                            Gender
                        </th>
                        <th>
                            Salary
                        </th>
                    </thead>
                    <tr ng-repeat="employee in employees">
                        <td>
                            {{employee.firstName}}
                        </td>
                        <td>
                            {{employee.lastName}}
                        </td>
                        <td>
                            {{employee.gender}}
                        </td>
                        <td>
                            {{employee.salary}}
                        </td>
                    </tr>
                </table>
           </div>
        </div>
    </body>
</html>
```

Web page loaded:

Firstname	Lastname	Gender	Salary
ABC	dshv	Male	55000
bhhj	dshv	Female	65000
nbvf	dshv	Male	54000
hjbc	dshv	Female	85000

NESTED ng-repeat

We can nest ng-repeat if we want to show something like this:

- UK
  - London
  - Manchester
  - Birmingham
- USA
  - LA
  - Chicago
  - Houston
- India
  - Hyderabad
  - Chennai
  - Mumbai

The model for this will be like this:

```
var myApp = angular
.module("myModule",[])
.controller("myController",function($scope)
{
    var countries=[
        {
            name:"UK",
            cities:[
                {name:"London"},
                {name: "Manchester"},
                {name:"Birmingham"}
            ]
        },
        {
            name:"USA",
            cities:[
                {name:"LA"},
                {name: "Chicago"},
                {name:"Houston"}
```

```
            ]
        },
        {
            name:"India",
            cities:[
                {name:"Hyderabad"},
                {name: "Chennai"},
                {name:"Mumbai"}
            ]
        }
    ];
    $scope.countries= countries;
});
```

And homepage where nested ng-repeat will take place will look like this:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">

           <div>
               <ul>
                   <li ng-repeat="country in countries">
                       {{country.name}}
                       <ul>
                           <li ng-repeat="city in country.cities">
                               {{city.name}}
                           </li>
                       </ul>
                   </li>
               </ul>
                </div>
        </div>
    </body>
</html>
```

How to find the index of an item in the collection?

We can use $index to find the index.

Like this in htmlpage:

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">

           <div>
               <ul>
                   <li ng-repeat="country in countries">
                       {{country.name}}-Index = {{$index}}
                       <ul>
                           <li ng-repeat="city in country.cities">
                               {{city.name}} - Index = {{$index}}
                           </li>
                       </ul>
                   </li>
               </ul>
                </div>
        </div>
    </body>
</html>
```

Output web page will be like this:

- UK-Index = 0
  - London - Index = 0
  - Manchester - Index = 1
  - Birmingham - Index = 2
- USA-Index = 1
  - LA - Index = 0
  - Chicago - Index = 1
  - Houston - Index = 2
- India-Index = 2
  - Hyderabad - Index = 0
  - Chennai - Index = 1
  - Mumbai - Index = 2

But what if we need to find the index of parent object?

1. By $parent.$index
   Like this:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">

           <div>
               <ul>
                   <li ng-repeat="country in countries">
                       {{country.name}}-Index = {{$index}}
                       <ul>
                           <li ng-repeat="city in country.cities">
                               {{city.name}} - Index = {{$index}} Parent Index
 = {{$parent.$index}}
                           </li>
                       </ul>
                   </li>
               </ul>
                </div>
        </div>
    </body>
</html>
```

2. 2nd method is by initializing a variable as ng-init="parentindex=$index" we can use that variable inside anything of its scope. It will look like this:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">

           <div>
               <ul>
                   <li ng-repeat="country in countries" ng-
init="parentindex=$index">
                       {{country.name}}-Index = {{$index}}
                       <ul>
```

```
                            <li ng-repeat="city in country.cities">
                                {{city.name}} - Index = {{$index}} Parent In
dex = {{parentindex}}
                            </li>
                        </ul>
                    </li>
                </ul>
                 </div>
        </div>
    </body>
</html>
```

The output of both type of parent index method is same as:

- UK-Index = 0
  - London - Index = 0 Parent Index = 0
  - Manchester - Index = 1 Parent Index = 0
  - Birmingham - Index = 2 Parent Index = 0
- USA-Index = 1
  - LA - Index = 0 Parent Index = 1
  - Chicago - Index = 1 Parent Index = 1
  - Houston - Index = 2 Parent Index = 1
- India-Index = 2
  - Hyderabad - Index = 0 Parent Index = 2
  - Chennai - Index = 1 Parent Index = 2
  - Mumbai - Index = 2 Parent Index = 2

Lesson 7- Handling events in AngularJS

Handling Events in AngularJS

1. Button click event handling
   If we want a page like this to like and dislike a technology:

Name	Likes	Sislikes	Like/Dislike
C#	0	0	Like Dislike
ASP.NET	0	0	Like Dislike
SQL SERVER	0	0	Like Dislike
AngularJS	0	0	Like Dislike

The htmlpage will look like this:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
        <link href="Style.css" rel="stylesheet"/>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">

           <div>
               <table>
                   <thead>
                       <th>Name</th>
                       <th>Likes</th>
                       <th>Sislikes</th>
                       <th>Like/Dislike</th>
                   </thead>
                   <tbody>
                       <tr ng-repeat="technology in technologies">
                           <td>{{ technology.name }}</td>
                           <td>{{ technology.likes }}</td>
                           <td>{{ technology.dislikes }}</td>
                           <td>
                               <input type="button" value="Like" ng-
click="incrementLikes(technology)"/>
                               <input type="button" value="Dislike" ng-
click="incrementDislikes(technology)"/>
                           </td>
                       </tr>
```

```
                    </tbody>
                </table>
                 </div>
        </div>
    </body>
</html>
```

The script.js file will look like this

```
var myApp = angular
.module("myModule",[])
.controller("myController",function($scope)
{
    var technologies=[
        {name:"C#",likes:0,dislikes:0},
        {name:"ASP.NET",likes:0,dislikes:0},
        {name:"SQL SERVER",likes:0,dislikes:0},
        {name:"AngularJS",likes:0,dislikes:0},
    ];
    $scope.technologies=technologies;
    $scope.incrementLikes=function(technology){
        technology.likes++;
    }
    $scope.incrementDislikes=function(technology){
        technology.dislikes++;
    }
});
```

But the look and style of table is not quite same so we have to provide a style sheet Style.css and link it in htmlpage:

```
table
{
    border-collapse: collapse;
    font-family: Arial;
}
td{
    border: 1px solid black;
    padding: 5px;
}
th{
    border: 1px solid black;
    padding: 5px;
    text-align: left;
}
```

In this code the functions for click event are bind with scope object.

Lesson 8- AngularJS filters

Filters in angularJS can do three things for data:

1. Format
2. Sort
3. Filter

They can be used with a binding expression or a directive.

A pipe(|) character is used with binding expression {{ expression | filterName:parameter }}.

Filters for formatting data will change the format the data for example to lowercase, uppercase, number, currency, date to string based on the format provided.

limitTo filter limits the number of rows or characters in a string or to limit an ng-repeat rows to be displayed.

For example if we want to display a page like this:

Rows to display: 3
Start Row Number: 0

Name	Date of Birth	Gender	Salary (number)	Salary (currency)
BEN	23/11/1980	male	55,000.79	$55,000.8
SARA	05/05/1970	female	68,000.00	$68,000.0
MARK	15/08/1974	male	57,000.00	$57,000.0

Htmlpage:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Page</title>

        <script src="Scripts/angular.min.js"></script>
        <script src="Scripts/Script.js"></script>
        <link href="Style.css" rel="stylesheet"/>
    </head>

    <body ng-app="myModule">
        <div ng-controller="myController">
            <div>
                Rows to display: <input type="number" step="1" min="0" max="5"
ng-model="rowLimit"/>
                <br/>
                Start Row Number: <input type="number" step="1" min="0" max="5"
 ng-model="rowStart"/>
                 <table>
                     <thead>
                         <th>Name</th>
```

```
                        <th>Date of Birth</th>
                        <th>Gender</th>
                        <th>Salary (number)</th>
                        <th>Salary (currency)</th>
                    </thead>
                    <tbody>
                        <tr ng-
repeat="employee in employees | limitTo: rowLimit: rowStart">
                            <td>{{ employee.name | uppercase }}</td>
                            <td>{{ employee.dateOfBirth | date:"dd/MM/yyyy"}}<
/td>
                            <td>{{ employee.gender | lowercase }}</td>
                            <td>{{ employee.salary | number:2}}</td>
                            <td>{{ employee.salary | currency: "$":1 }}</td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </body>
</html>
```

Script.js page:

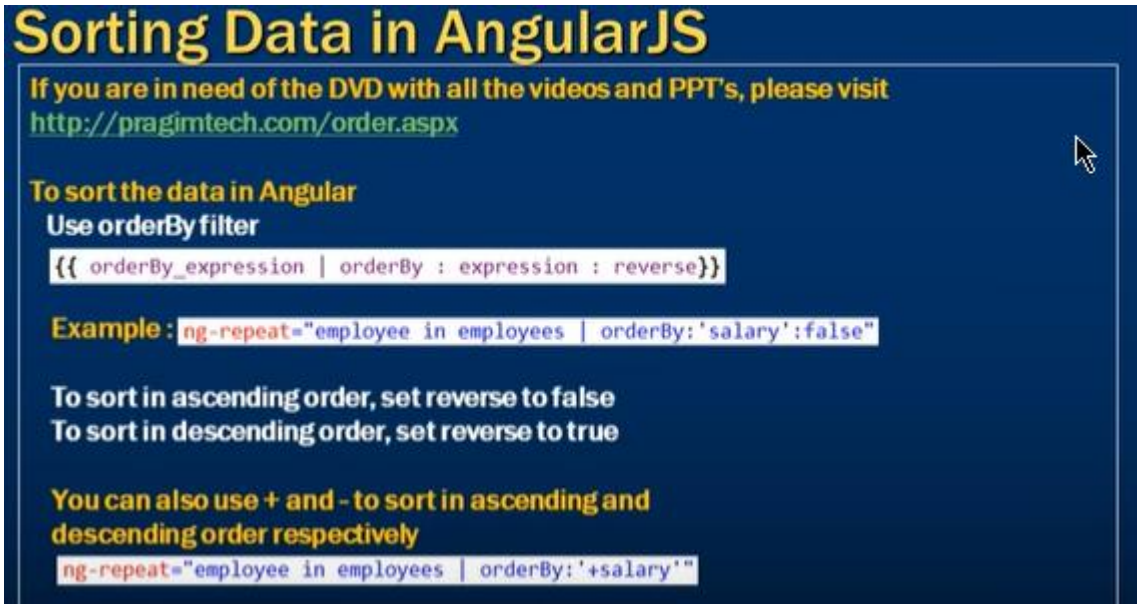
```
var myApp = angular
.module("myModule",[])
.controller("myController",function($scope)
{
    var employees =[
        {name:"Ben", dateOfBirth:new Date("November 23, 1980"), gender:"Male",
 salary: 55000.788},
        {name:"Sara", dateOfBirth:new Date("May 05, 1970"), gender:"Female", s
alary: 68000},
        {name:"Mark", dateOfBirth:new Date("August 15, 1974"), gender:"Male",
salary: 57000},
        {name:"Pam", dateOfBirth:new Date("October 27, 1979"), gender:"Female"
, salary: 53000},
        {name:"Todd", dateOfBirth:new Date("December 30, 1983"), gender:"Male"
, salary: 60000},
    ];

    $scope.employees = employees;
    $scope.rowLimit = 3;
    $scope.rowStart = 0;
});
```

All the filters are applied in htmlpage.

Lesson 9- Sorting data in AngularJS

To sort the data we use orderBy filter.

{{ orderBy_expression | orderBy : expression: reverse}}



We can hard code the filter to sort the data by a filter or we can bind its value to a model and change the parameter on which we are sorting the data. We use a select element and its value is passed to ng-model which is passed to orderBy filter.

Lesson 10 – AngularJS sort rows by table header

In this lesson we are learning how to sort a table by clicking on table header. Whichever row is clicked table is sorted by that.

```
$scope.employees = employees;
$scope.sortColumn = "name";
$scope.reverseSort = false;

$scope.sortData = function (column) {
    $scope.reverseSort = ($scope.sortColumn == column) ? !$scope.reverseSort : false;
    $scope.sortColumn = column;
}

$scope.getSortClass = function (column) {
    if ($scope.sortColumn == column) {
        return $scope.reverseSort ? 'arrow-down' : 'arrow-up'
    }

    return '';
}
```

sortData function will set all the properties for orderBy and getSortClass function will return a css class to apply an arrow up or arrow down to that row header which shows the order of data and null for all other row headers.

These are the classes in style.css

```
    text-align: left;
}

.arrow-up {
    width: 0;
    height: 0;
    border-left: 5px solid transparent;
    border-right: 5px solid transparent;
    border-bottom: 10px solid black;
}

.arrow-down {
    width: 0;
    height: 0;
    border-left: 5px solid transparent;
    border-right: 5px solid transparent;
    border-top: 10px solid black;
}
```

Htmlpage will be changed ng-click will be used to make the row header clickable and call the sortData function then for css class add a div and call getSortClass function to get the respective arrow.

```
<body ng-app="myModule">
    <div ng-controller="myController">
        <table>
            <thead>
                <tr>
                    <th ng-click="sortData('name')">
                        Name <div ng-class="getSortClass('name')"></div>
                    </th>
                    <th ng-click="sortData('dateOfBirth')">
                        Date of Birth <div ng-class="getSortClass('dateOfBirth')"></div>
                    </th>
                    <th ng-click="sortData('gender')">
                        Gender <div ng-class="getSortClass('gender')"></div>
                    </th>
                    <th ng-click="sortData('salary')">
                        Salary <div ng-class="getSortClass('salary')"></div>
                    </th>
                </tr>
            </thead>
            <tbody>
```

Then change the ng-repeat to orderBy and pass the parameters defined in Script.js

```
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees | orderBy:sortColumn:reverseSort">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.dateOfBirth | date:"dd/MM/yyyy" }} </td>
                    <td> {{ employee.gender }} </td>
                    <td> {{ employee.salary  }} </td>
                </tr>
            </tbody>
        </table>
```

The loaded web page will look like this:

Name	Date of Birth ▼	Gender	Salary
Todd	30/12/1983	Male	60000
Ben	23/11/1980	Male	55000
Pam	27/10/1979	Female	53000
Mark	13/08/1974	Male	57000
Sara	05/05/1970	Female	68000

Lesson 11- Search filter in AngularJS

Search filter will search throughout the table and fetch all the matching rows.

Htmlpage will look like this:

```
Search : <input type="text" ng-model="searchText" placeholder="Search employees" />
<br /><br />
<table>
    <thead>
        <tr>
            <th>Name</th>
            <th>Gender</th>
            <th>Salary</th>
            <th>City</th>
        </tr>
    </thead>
    <tbody>
        <tr ng-repeat="employee in employees | filter:searchText">
            <td> {{ employee.name }} </td>
            <td> {{ employee.gender }} </td>
            <td> {{ employee.salary  }} </td>
            <td> {{ employee.city  }} </td>
        </tr>
    </tbody>
</table>
```

Style.css:

```
body {
    font-family: Arial;
}

table {
    border-collapse: collapse;
}

td {
    border: 1px solid black;
    padding: 5px;
}

th {
    border: 1px solid black;
    padding: 5px;
    text-align: left;
}
```

Script.js is same as before.

Lesson 12- AngularJS filter by multiple properties

By multiple properties and exact match.

Htmlpage code for multiple properties and exact match

```
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <input type="text" ng-model="searchText.name" placeholder="Search name" />
        <input type="text" ng-model="searchText.city" placeholder="Search city" />
        <input type="checkbox" ng-model="exactMatch"/> Exact Match
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>Salary</th>
                    <th>City</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees | filter:searchText:exactMatch">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.gender }} </td>
```

Script.js and style.css is same as before.

Loaded web page is shown below:

It searches for both and if exact match is checked it searches for Exact match of value in that respective row.

Now if we want to search by two columns but by one text box that can also be achieved by the following code.

HTMLpage:

```
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <input type="text" ng-model="searchText" placeholder="Search city & name" />
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>Salary</th>
                    <th>City</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees | filter:search">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.gender }} </td>
                    <td> {{ employee.salary  }} </td>
                    <td> {{ employee.city  }} </td>
                </tr>
            </tbody>
```

Script.js

```
Demo - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Google Chrome - Debug - Any CPU
Styles.css  Script.js  HtmlPage1.html
        { name: "Sara", gender: "Female", salary: 68000, city: "Chennai" },
        { name: "Mark", gender: "Male", salary: 57000, city: "London" },
        { name: "Pam", gender: "Female", salary: 53000, city: "Chennai" },
        { name: "Todd", gender: "Male", salary: 60000, city: "London" },
    ];

    $scope.employees = employees;

    $scope.search = function (item) {
        if ($scope.searchText == undefined) {
            return true;
        }
        else {
            if(item.name.toLowerCase().indexOf($scope.searchText.toLowerCase()) != -1 ||
                item.city.toLowerCase().indexOf($scope.searchText.toLowerCase()) != -1)
            {
                return true;
            }
        }

        return false;
    }
```

Search function works for all items and if that item matches the criteria of name and city columns true is returned and that item is displayed.

M

Name	Gender	Salary	City
Mark	Male	57000	London
Pam	Female	53000	Chennai

Lesson 13- Create a custom filter in AngularJS

Create a Custom Filter in AngularJS

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

Custom filter in AngularJS

- Is a function that returns a function
- Use the filter function to create a custom filter

Example : Create a custom filter to convert integer values 1, 2, 3 to Male, Female and Not disclosed respectively

Name	Gender	Salary
Ben	1	55000
Sara	2	68000
Mark	1	57000
Pam	2	53000
Todd	3	60000

Name	Gender	Salary
Ben	Male	55000
Sara	Female	68000
Mark	Male	57000
Pam	Female	53000
Todd	Not disclosed	60000

Customer filter will convert integer values to their string representation.

That's how a filter is created in script.js

```
var app = angular
        .module("myModule", [])
        .filter("gender", function () {
            return function (gender) {
                switch (gender) {
                    case 1:
                        return "Male";
                    case 2:
                        return "Female";
                    case 3:
                        return "Not disclosed";
                }
            }
        })
        .controller("myController", function ($scope) {

            var employees = [
                { name: "Ben", gender: 1, salary: 55000 },
                { name: "Sara", gender: 2, salary: 68000 },
                { name: "Mark", gender: 1, salary: 57000 },
                { name: "Pam", gender: 2, salary: 53000 },
                { name: "Todd", gender: 3, salary: 60000 }
```

And we apply that filter in htmlpage like this on binding expression gender property.

```
<tbody>
    <tr ng-repeat="employee in employees">
        <td> {{ employee.name }} </td>
        <td> {{ employee.gender | gender }} </td>
        <td> {{ employee.salary  }} </td>
    </tr>
</tbody>
```

Without filter web page loaded:

Name	Gender	Salary
Ben	1	55000
Sara	2	68000
Mark	1	57000
Pam	2	53000
Todd	3	60000

With filter web page loaded.

Name	Gender	Salary
Ben	Male	55000
Sara	Female	68000
Mark	Male	57000
Pam	Female	53000
Todd	Not disclosed	60000

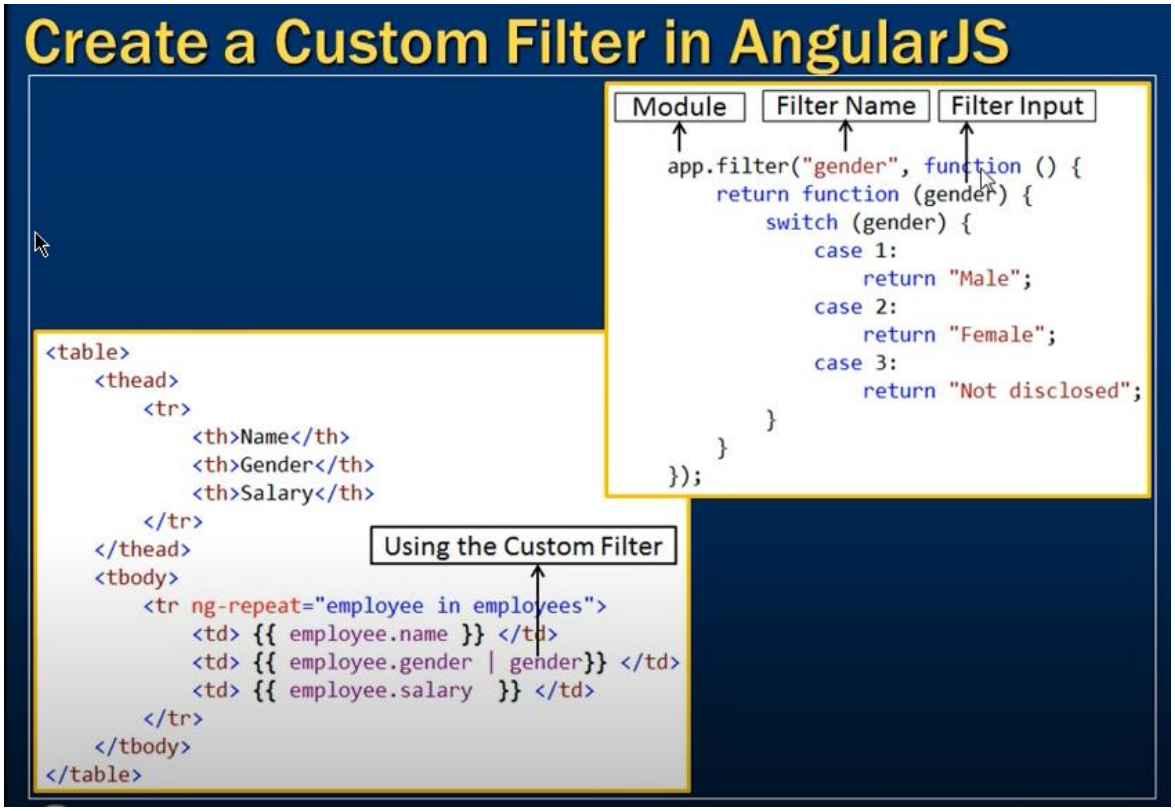
We can also create a separate filter file as filter.js and add this code in it for filter creation.

```
app.filter("gender", function () {
    return function (gender) {
        switch (gender) {
            case 1:
                return "Male";
            case 2:
                return "Female";
            case 3:
                return "Not disclosed";
        }
    }
})
```

Then we will reference filter.js file I htmlpage file

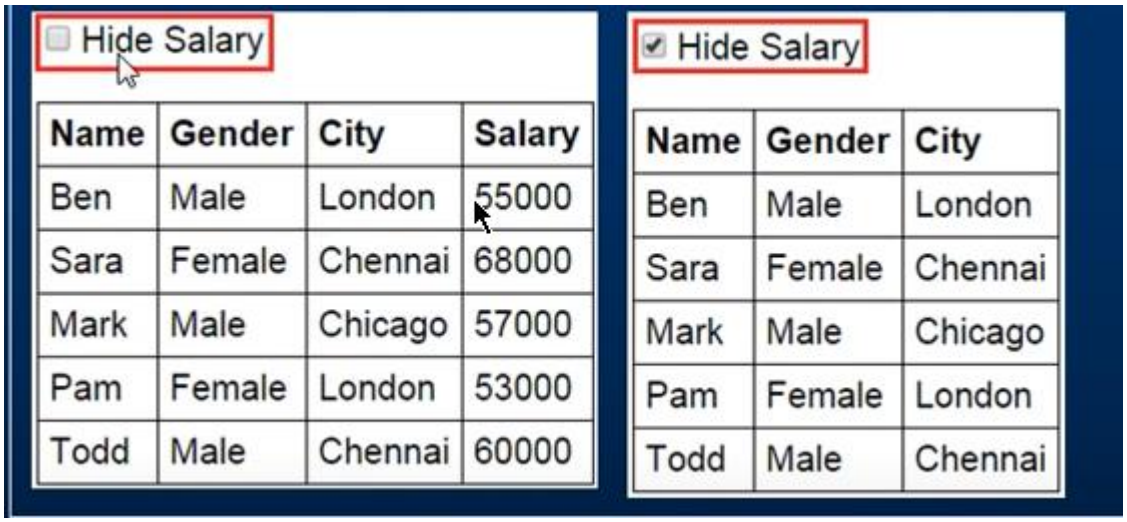
```
<head>
    <title></title>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
    <script src="Scripts/Filters.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
```

Overview of custom filter



Lesson 14- ng-hide and ng-show in AngularJS

Both directives are used to control the visibility of html elements.



Htmlpage to hide the salary based on a checkbox and ng-hide directive.

```
<body ng-app="myModule">
    <div ng-controller="myController">
        <input type="checkbox" ng-model="hideSalary" /> Hide Salary
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>City</th>
                    <th ng-hide="hideSalary">Salary</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.gender}} </td>
                    <td> {{ employee.city}} </td>
                    <td ng-hide="hideSalary"> {{ employee.salary  }} </td>
                </tr>
            </tbody>
        </table>
    </div>
```

Now we will achieve the same using ng-show directive.

```
<body ng-app="myModule">
    <div ng-controller="myController">
        <input type="checkbox" ng-model="hideSalary" /> Hide Salary
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>City</th>
                    <th ng-show="!hideSalary">Salary</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.gender}} </td>
                    <td> {{ employee.city}} </td>
                    <td ng-show="!hideSalary"> {{ employee.salary  }} </td>
                </tr>
            </tbody>
        </table>
    </div>
```

Controller code:

Script.js - Controller

```
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {

            var employees = [
                { name: "Ben", gender: "Male", city: "London", salary: 55000 },
                { name: "Sara", gender: "Female", city: "Chennai", salary: 68000 },
                { name: "Mark", gender: "Male", city: "Chicago", salary: 57000 },
                { name: "Pam", gender: "Female", city: "London", salary: 53000 },
                { name: "Todd", gender: "Male", city: "Chennai", salary: 60000 }
            ];

            $scope.employees = employees;
        });
```

Now instead of hiding the entire salary column we will mask and unmask the values like this:

Mask and unmask Salary column values

☐ Hide Salary

Name	Gender	City	Salary
Ben	Male	London	55000
Sara	Female	Chennai	68000
Mark	Male	Chicago	57000
Pam	Female	London	53000
Todd	Male	Chennai	60000

☑ Hide Salary

Name	Gender	City	Salary
Ben	Male	London	#####
Sara	Female	Chennai	#####
Mark	Male	Chicago	#####
Pam	Female	London	#####
Todd	Male	Chennai	#####

Html page will change like this:

```
<body ng-app="myModule">
    <div ng-controller="myController">
        <input type="checkbox" ng-model="hideSalary" /> Hide Salary
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>City</th>
                    <th ng-hide="hideSalary">Salary</th>
                    <th ng-show="hideSalary">Salary</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.gender}} </td>
                    <td> {{ employee.city}} </td>
                    <td ng-hide="hideSalary"> {{ employee.salary  }} </td>
                    <td ng-show="hideSalary"> #### </td>
                </tr>
            </tbody>
```

Lesson 15- AngularJS ng-init directive

Ng-init evaluates an expression. In given ng-init we are creating an array of employees which will be accessible by all inner elements of this div.

```
<div ng-init="employees = [
            { name: 'Ben', gender: 'Male', city: 'London' },
            { name: 'Sara', gender: 'Female', city: 'Chennai' },
            { name: 'Mark', gender: 'Male', city: 'Chicago' },
            { name: 'Pam', gender: 'Female', city: 'London' },
            { name: 'Todd', gender: 'Male', city: 'Chennai' }
        ]">
```

Then we can show these employee objects In a table

```
<body ng-app>
    <div ng-init="employees = [
            { name: 'Ben', gender: 'Male', city: 'London' },
            { name: 'Sara', gender: 'Female', city: 'Chennai' },
            { name: 'Mark', gender: 'Male', city: 'Chicago' },
            { name: 'Pam', gender: 'Female', city: 'London' },
            { name: 'Todd', gender: 'Male', city: 'Chennai' }
        ]">

        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>City</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees">
                    <td>{{employee.name}}</td>
                    <td>{{employee.gender}}</td>
                    <td>{{employee.city}}</td>
                </tr>
```

*In real world we should always use a controller to initialize a value instead of using ng-init.

*ng-init should only be used for aliasing special properties of ng-repeat directive.

In the given example we have countries array which further have cities array in it so to get the parentIndex(index of country) in city loop we can use ng-init to initialize it in county loop div and the later can use it with binding expression as shown in 2nd image.

```
body ng-app="myModule">
    <div ng-controller="myController">
        <ul>
            <li ng-repeat="country in countries" ng-init="parentIndex = $index">
                {{country.name}} - Index = {{$index}}
                <ul>
                    <li ng-repeat="city in country.cities">
                        {{city.name}} - Index = {{$index}}
                    </li>
                </ul>
            </li>
        </ul>

    </div>
/body>
```

We can use that variable like this:

```
ul>
    <li ng-repeat="city in country.cities">
        {{city.name}} - Parent Index = {{ parentIndex }}, Index = {{$index}}
    </li>
/ul>
```

Lesson 16- ng-include directive in AngularJS

It is used to embed an html page into another html page. It increases the reusability of html pages for same view.

To show the list of employees in a table by another employee.html page.

Employee.html page:

```
table>
    <thead>
        <tr>
            <th>Name</th>
            <th>Gender</th>
            <th>Salary</th>
        </tr>
    </thead>
    <tbody>
        <tr ng-repeat="employee in employees">
            <td>{{employee.name}}</td>
            <td>{{employee.gender}}</td>
            <td>{{employee.salary}}</td>
        </tr>
    </tbody>
/table>
```

We don't need to add head and other sections because they are already added in htmlpage.html we just have to include it in our htmlpage like this:

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/Script.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <div ng-include="'EmployeeTable.html'"></div>
    </div>
</body>
</html>
```

Now the value can also be a $scope property and we can define it in script.js like this.

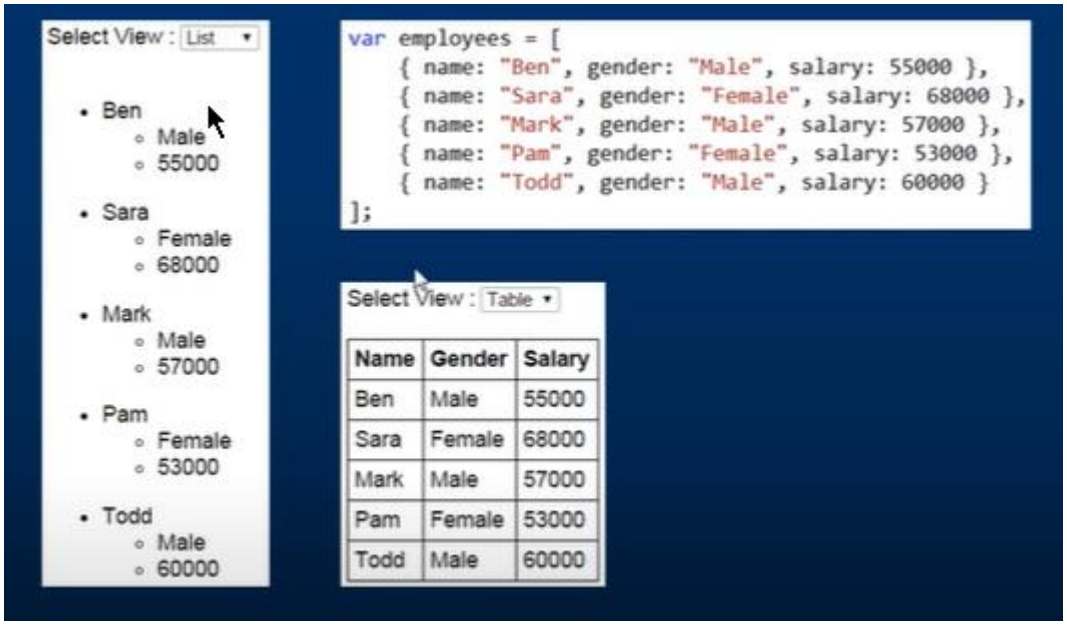
```javascript
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {

            var employees = [
                { name: "Ben", gender: "Male", salary: 55000 },
                { name: "Sara", gender: "Female", salary: 68000 },
                { name: "Mark", gender: "Male", salary: 57000 },
                { name: "Pam", gender: "Female", salary: 53000 },
                { name: "Todd", gender: "Male", salary: 60000 }
            ];

            $scope.employees = employees;
            $scope.employeeView = "EmployeeTable.html";
        });
```

And in htmlpage this will be used like this:

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/Script.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <div ng-include="employeeView"></div>
    </div>
</body>
</html>
```

Both will do the same thing.

By ng-include directive we can specify the view format of a table using a dropdown like this:



For this we will add two pages employeeList.html and employeeTable.html to show the data in desired format. EmployeeTable will look like the previous example and employeeList will look like this:

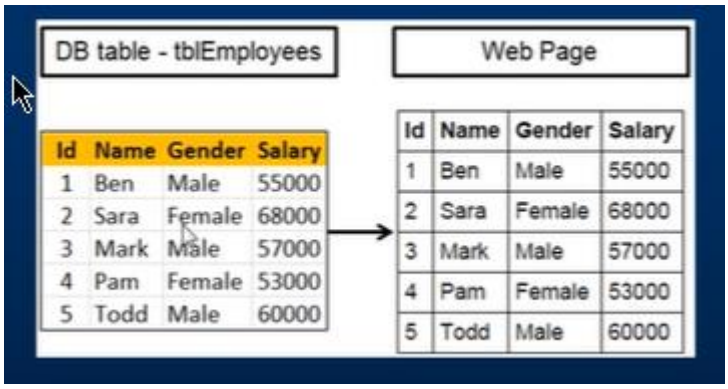
```
<ul ng-repeat="employee in employees">
    <li>
        {{employee.name}}
        <ul>
            <li>
                {{employee.gender}}
            </li>
            <li>
                {{employee.salary}}
            </li>
        </ul>
    </li>
</ul>
```

Htmlpage.html will look like this:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/Script.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        Select View :
        <select ng-model="employeeView">
            <option value="EmployeeTable.html">Table</option>
            <option value="EmployeeList.html">List</option>
        </select>
        <br /><br />
        <div ng-include="employeeView"></div>
    </div>
</body>
</html>
```

Lesson 17- Cosuming ASP NET Web Services in AngularJS Using $http

We want to retrieve data from a database table and display it on a web page like so:



In this lesson we created a database table in sql server management studio and updated the connection string in web.config file of an asp.net project for that database. We added a service to access that database and table to serialize it and add it in our custom employee list to show on page.

The service file will look like this:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

using System.Configuration;

using System.Data.SqlClient;

using System.Web.Script.Serialization;


namespace sampleWebApp

{

    /// <summary>

    /// Summary description for EmployeeService

    /// </summary>

    [WebService(Namespace = "http://tempuri.org/")]

    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]

    [System.ComponentModel.ToolboxItem(false)]

    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

    [System.Web.Script.Services.ScriptService]
```

```
  public class EmployeeService : System.Web.Services.WebService
  {

    [WebMethod]
    public void GetAllEmployees()
    {
      List<Employee> listEmployees = new List<Employee>();
      string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
      using(SqlConnection con = new SqlConnection(cs))
      {
        SqlCommand cmd = new SqlCommand("Select * from tblEmployees", con);
        con.Open();
        SqlDataReader rdr = cmd.ExecuteReader();
        while(rdr.Read())
        {
          Employee employee = new Employee();
          employee.id = Convert.ToInt32(rdr["Id"]);

          employee.name = rdr["Name"].ToString();
          employee.gender = rdr["Gender"].ToString();
          employee.salary = Convert.ToInt32(rdr["Salary"]);
          listEmployees.Add(employee);
        }

      }
      JavaScriptSerializer js = new JavaScriptSerializer();

      Context.Response.Write(js.Serialize(listEmployees));
    }
  }
}
```

Web.config file will be updated like this:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
        <connectionStrings>
                <add name="DBCS"
                        connectionString="server=.; database=sampleDB;integrated security=SSPI"/>
        </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2"/>
    <httpRuntime targetFramework="4.7.2"/>
          <webServices>
                <protocols>
                        <add name="HttpGet"/>
                </protocols>
          </webServices>
  </system.web>
  <system.codedom>
    <compilers>
      <compiler language="c#;cs;csharp" extension=".cs"
        type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
        warningLevel="4" compilerOptions="/langversion:default /nowarn:1659;1699;1701"/>
      <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
        type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
        warningLevel="4" compilerOptions="/langversion:default /nowarn:41008
/define:_MYTYPE=\&quot;Web\&quot; /optionInfer+"/>
    </compilers>
  </system.codedom>
```

```
</configuration>
```

HttpGet method is also added in web.config file to use it later with $http object.

To call the service we will use a controller so the script.js file will look like this:

```
var myApp = angular
.module("myModule",[])
.controller("myController",function($scope, $http)
{
   $http.get('EmployeeService.asmx/GetAllEmployees')
      .then(function (response) {
         $scope.employees = response.data;
      });
});
```

We will also add a employee class in project to populate the data of database table in it.

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

namespace sampleWebApp
{
   public class Employee
   {
      public int id { get; set; }
      public string name { get; set; }
      public string gender { get; set; }
      public int salary { get; set; }

   }
}
```

Now everything is linked htmlpage will call the controller that will call the service which will fetch the data from database.

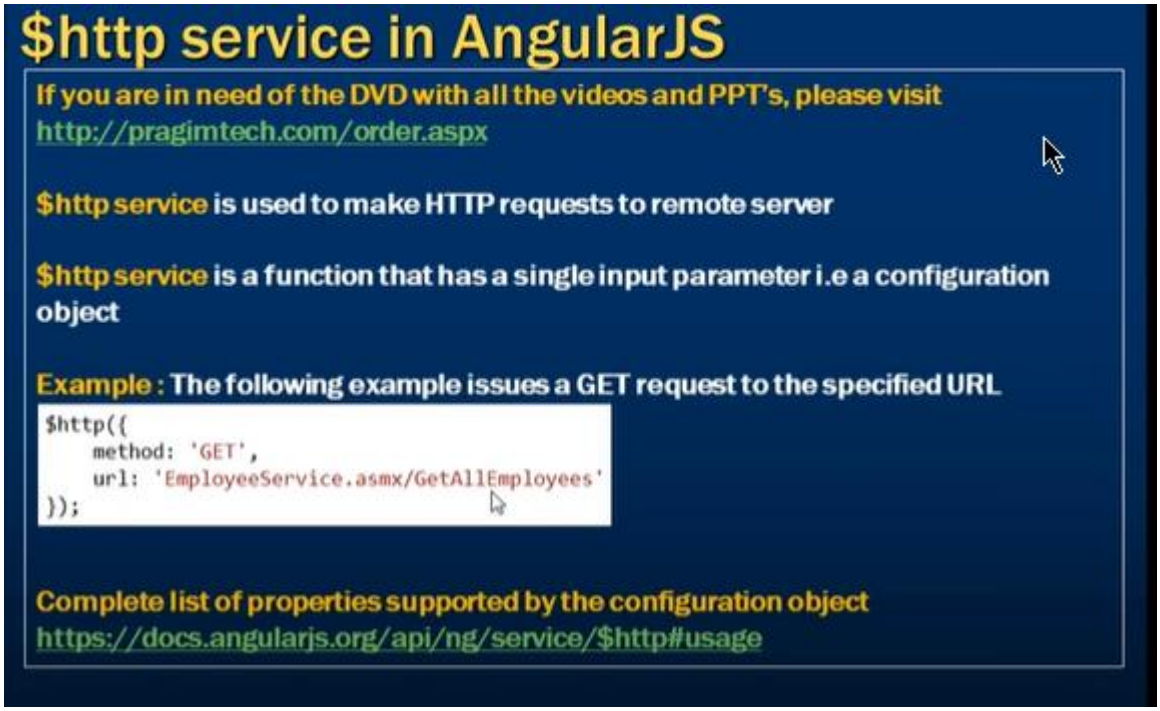
```
<!DOCTYPE html>
<html>
```

```
<head>
   <title>Sample Page</title>

   <script src="Scripts/angular.min.js"></script>
   <script src="Scripts/Script.js"></script>
   <link href="Style.css" rel="stylesheet" />
</head>

<body ng-app="myModule">
   <div ng-controller="myController">
      <div>

         <table>
            <thead>
               <th>Id</th>
               <th>Name</th>
               <th>Gender</th>
               <th>Salary (number)</th>
               <th>Salary (currency)</th>
            </thead>
            <tbody>
               <tr ng-repeat="employee in employees">
                  <td>{{ employee.id }}</td>
                  <td>{{ employee.name | uppercase }}</td>
                  <td>{{ employee.gender | lowercase }}</td>
                  <td>{{ employee.salary | number:2}}</td>
                  <td>{{ employee.salary | currency: "$":1 }}</td>
               </tr>
            </tbody>
         </table>
      </div>
   </div>
</body>
```

</html>

Lesson 18- $http service in AngularJS

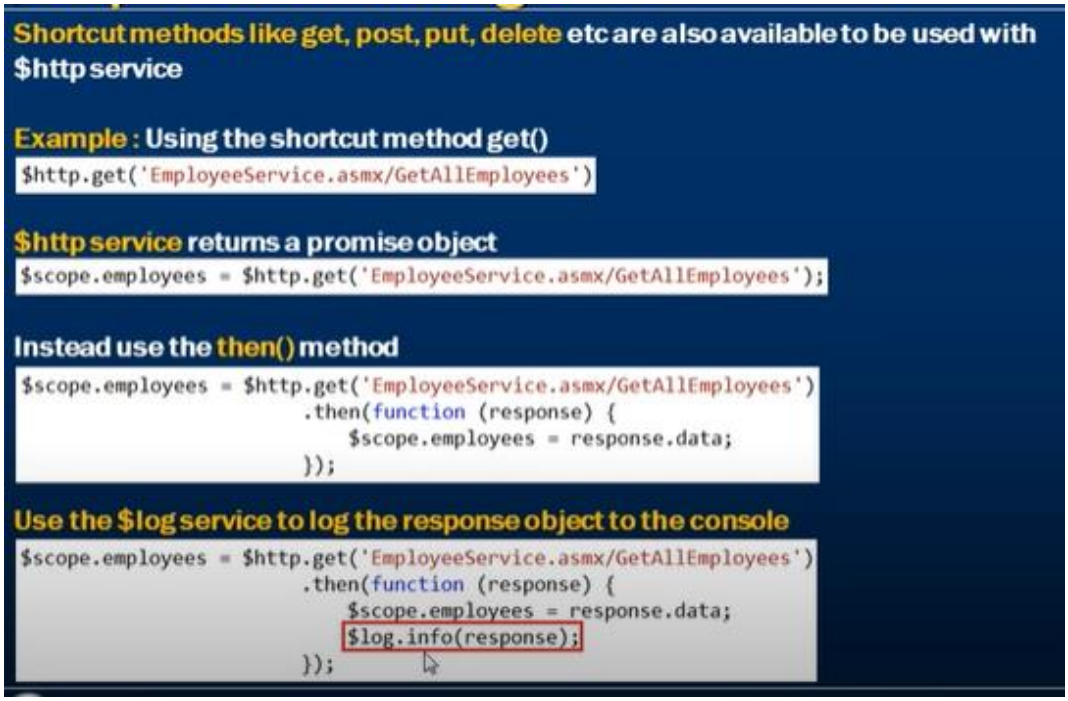
It is used to make http requests to remote server. http has only one parameter i.e. configuration object.



The **Promise object** represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

The first function in then() is success callback function it is called after the success of request.

A response object is passed in it which has several properties.



The 2nd function in then is failure callback function which takes an argument of “reason” for failure or problem.

If there is an error processing the request, the errorCallback function is called

```
$scope.employees = $http.get('EmployeeService.asmx/GetAllEmployee')
                        .then(function (response) {
                            $scope.employees = response.data;
                        }, function (reason) {
                            $scope.error = reason.data;
                        });
```

You can also create separate functions and associate them as successCallback and errorCallback functions

```
var successCallBack = function (response) {
    $scope.employees = response.data;
};

var errorCallBack = function (reason) {
    $scope.error = reason.data;
}

$scope.employees = $http.get('EmployeeService.asmx/GetAllEmployees')
                        .then(successCallBack, errorCallBack);
```

Default Transformations provided by Angular's http service

- If the data property of the request configuration object contains a JavaScript object, it is automatically converted into JSON object
- If JSON response is detected, it is automatically converted into a JavaScript object

Lesson 19- AngularJS Services

What is a service in AngularJS

A service in Angular is simply an object that provide some sort of service that can be reused with in an angular application

Why do we need services

Services encapsulate reusable logic that does not belong anywhere else (i.e Directives, Filters, Views, Models, & Controllers)

What are the benefits of using services

- Reusability
- Dependency Injection
- Testability

Lesson 20- Creating Custom Service

Service code in controller:

Create custom service in AngularJS

```
// All logic in the controller. No custom service used
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {
            $scope.transformString = function (input) {
                if (!input)
                    return input;

                var output = "";

                for (var i = 0; i < input.length; i++) {
                    if (i > 0 && input[i] == input[i].toUpperCase()) {
                        output = output + " ";
                    }

                    output = output + input[i];
                }

                $scope.output = output;
            };
        });
```

Create another service.js file and inject its name in controller so that we can use its methods like this:

```
// factory method is used to create and register the service with Angular
app.factory('stringService', function () {
    return {
        processString: function (input) {
            if (!input)
                return input;

            var output = "";

            for (var i = 0; i < input.length; i++) {
                if (i > 0 && input[i] == input[i].toUpperCase()) {
                    output = output + " ";
                }

                output = output + input[i];
            }

            return output;
        }
    };
});
```

```
// The controller is now simple and clean. Notice the stringService injection
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope, stringService) {
            $scope.transformString = function (input) {
                $scope.output = stringService.processString(input);
            };
        });
```