AngularJS

What is AngularJS
AngularJS is a JavaScript framework that helps build applications that run in a web browser.

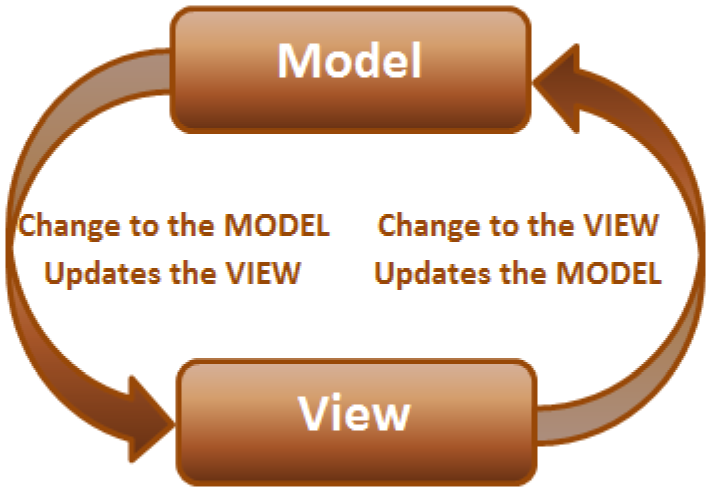
Who developed AngularJS
Google is the company that developed AngularJS. AngularJS is an open source project, which means it can be be freely used, changed, and shared by anyone.

AngularJS is an excellent framework for building both Single Page Applications (SPA) and Line of Business Applications. Many companies are using Angular today, and there are many public facing web sites that are built with angular.

There is a website, https://www.madewithangular.com, that has the list of web sites that are built using AngularJS. Within this list you can find many popular websites.

What are the benefits of using AngularJS
1. Dependency Injection : Dependency Injection is something AngularJS does quite well. If you are new to Dependency Injection, don't worry, we will discuss it in detail with examples in a later video.

2. Two Way Data-Binding : One of the most useful feature in AngularJS is the Two Way Data-Binding. The Two Way Data-Binding, keeps the model and the view in sync at all times, that is a change in the model updates the view and a change in the view updates the model.


Model
Change to the MODEL Updates the VIEW
Change to the VIEW Updates the MODEL
View

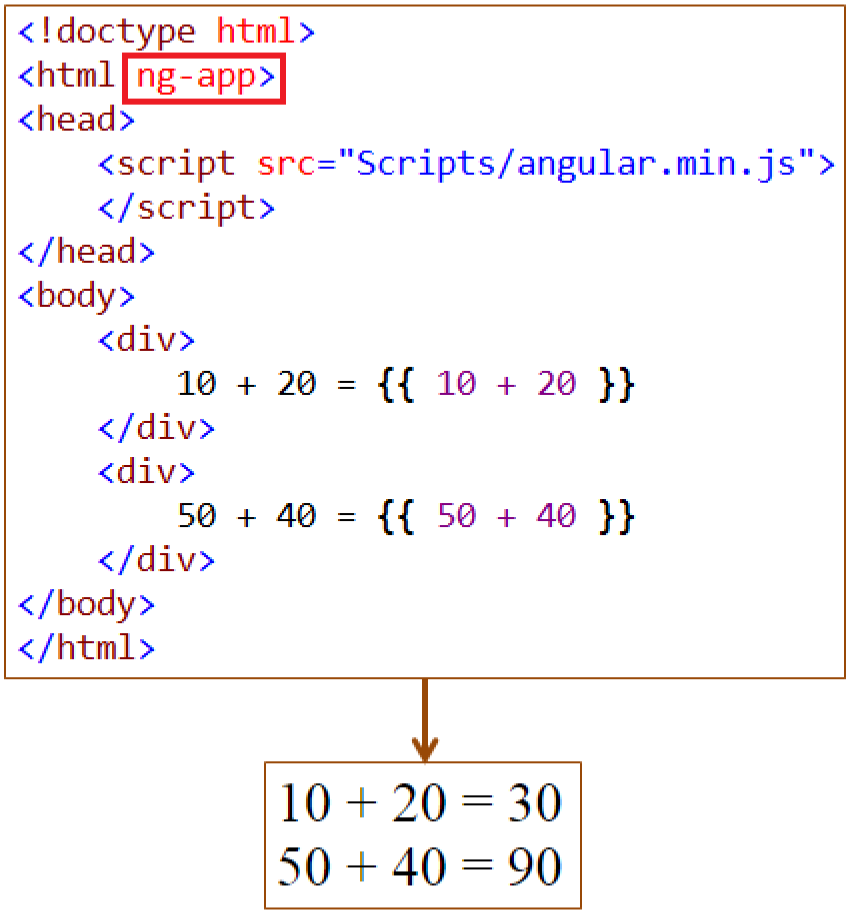
3. Testing : Testing is an area where Angular really shines. Angular is designed with testing in mind right from the start. Angular makes it very easy to test any of it's components through both unit testing and end to end testing. So there's really no excuse for not testing any of your angular application code.

4. Model View Controller : With angular it is very easy to develop applications in a clean MVC way. All you have to do is split your application code into MVC components. The rest, that is managing those components and connecting them together is done by angular.

To get started with angular
1. Add a reference to the angular script
2. Include ng-app attribute

What is ng-app
In angular, ng-app is called a directive. There are many directives in angular. You can find the complete list of directives on https://angularjs.org. The ng prefix in the directive stands for angular. The ng-app directive is a starting point of AngularJS Application. Angular framework will first check for ng-app directive in an HTML page after the entire page is loaded. If ng-app directive is found, angular bootstraps itself and starts to manage the section of the page that has the ng-app directive.

Example : In the example below, the **ng-app** directive is placed at the <html> tag level. So the binding expressions in both the div elements are evaluated and displayed as expected.



```
<!doctype html>
<html ng-app>
<head>
    <script src="Scripts/angular.min.js">
    </script>
</head>
<body>
    <div>
        10 + 20 = {{ 10 + 20 }}
    </div>
    <div>
        50 + 40 = {{ 50 + 40 }}
    </div>
</body>
</html>
```

```
10 + 20 = 30
50 + 40 = 90
```

What is a module in AngularJS
A module is a container for different parts of your application i.e controllers, services, directives, filters, etc. In this video we will also discuss controllers.

Why is a module required
You can think of a module as a Main() method in other types of applications. For example, a Dot Net console application has a Main() method which is the entry point into the application and it wires together the different parts of the application.

Modules are the angular application's equivalent of the Main() method. Modules declaratively specify how the angular application should be bootstrapped.

There are several benefits of the modular approach.

How to create a module
Creating a module in angular is staright forward. Use the angular object's module() method to create a module. The angular object is provided by angular script. The following example, creates a module.

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

The first parameter specifies the name of the module.
The second parameter specifies the dependencies for this module

What is a controller in angular
In angular a controller is a JavaScript function. The job of the controller is to build a model for the view to display. The model is the data. In a real world application, the controller may call into a service to retrieve data from the database.

How to create a controller in angular
Simple, create a JavaScript function and assign it to a variable

```
var myController = function ($scope) {
   $scope.message = "AngularJS Tutorial";
}
```

What is $scope
$scope is a parameter that is passed to the controller function by angular framework. We attach the model to the $scope object, which will then be available in the view. With in the view, we can retrieve the data from the scope object and display.

How to register the controller with the module
Use module object's controller function to register the controller with the module

```
myApp.controller("myController", myController);
```

Here is the complete code

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

//Create the controller
var myController = function ($scope) {
    $scope.message = "AngularJS Tutorial";
}

// Register the controller with the module
myApp.controller("myController", myController);
```

The above code can also be written as shown below

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

// Creating the controller and registering with the module all done in one line.
myApp.controller("myController", function ($scope) {
    $scope.message = "AngularJS Tutorial";
});
```

How to use the module that we created to bootstrap the angular application
Associate the module name with ng-app directive

```
ng-app="myModule"
```

Similarly associate the controller using ng-controller directive

```
ng-controller="myController"
```

Here is the complete HTML

```
<!doctype html>
<html ng-app="myModule">
<head>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
</head>
<body>
    <div ng-controller="myController">
        {{ message }}
    </div>
</body>
</html>
```

Two way databinding in AngularJS

When the model changes the view is automatically updated. This is achieved using the data binding expression in the view.

Script.js : The code in the controller attaches message property to the scope which is the model.

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

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

HtmlPage1.html : Whenever the message property value changes, the data binding expression in the view updates the view automatically.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></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>
```

How about the other way round. How to keep the model up to date when the view changes. That's exactly is the purpose of ng-model directive.

In the html below, notice the input element is decorated with **ng-model** directive. This ensures that whenever the value in the textbox is changed, angular will automatically update the message property of the $scope object. This means the ng-model directive automatically takes the form values and updates the model. The binding expression does the opposite, i.e whenever the model changes the view is automatically updated.
Because of the two way data binding provided by angular, as you type in the textbox, the value is immediately displayed on the view just below the textbox. This two way binding feature provided by angular, eliminates the need to write any custom code to move data from the model to the view or from the view to the model.

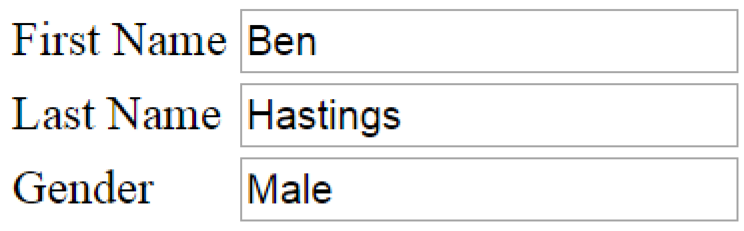
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <input type="text" placeholder="Type your message here" ng-model="message" />
        <br /><br />
        {{ message }}
    </div>
</body>
</html>
```

ng-model directive can be used with the following 3 html elements

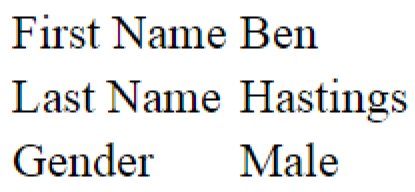
- input

- select
- textarea

Two way binding example with complex object :



```
First Name  Ben
Last Name   Hastings
Gender      Male
```



```
First Name Ben
Last Name  Hastings
Gender     Male
```

Script.js code : In the following example, the model is employee which is a complex object with properties like firstName, lastName and gender.

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

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

HtmlPage1.html : When the view loads, the model data is display in both, the textbox and td elements on the page. As you start to type in any of the textboxes, the respective employee model object property is updated, and the change is immediately reflected in the respective td element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <table>
            <tr>
                <td>
            First Name
        </td>
        <td>
            <input type="text" placeholder="Firstname"
                   ng-model="employee.firstName" />
        </td>
    </tr>
    <tr>
        <td>
            Last Name
        </td>
        <td>
            <input type="text" placeholder="Lastname"
                   ng-model="employee.lastName" />
        </td>
    </tr>
    <tr>
        <td>
            Gender
        </td>
        <td>
            <input type="text" placeholder="Gender"
                   ng-model="employee.gender" />
        </td>
    </tr>
</table>

<br />

<table>
    <tr>
        <td>
            First Name
        </td>
        <td>
            {{ employee.firstName }}
        </td>
    </tr>
    <tr>
        <td>
            Last Name
        </td>
        <td>
            {{ employee.lastName }}
        </td>
    </tr>
    <tr>
        <td>
            Gender
        </td>
        <td>
            {{ employee.gender }}
```

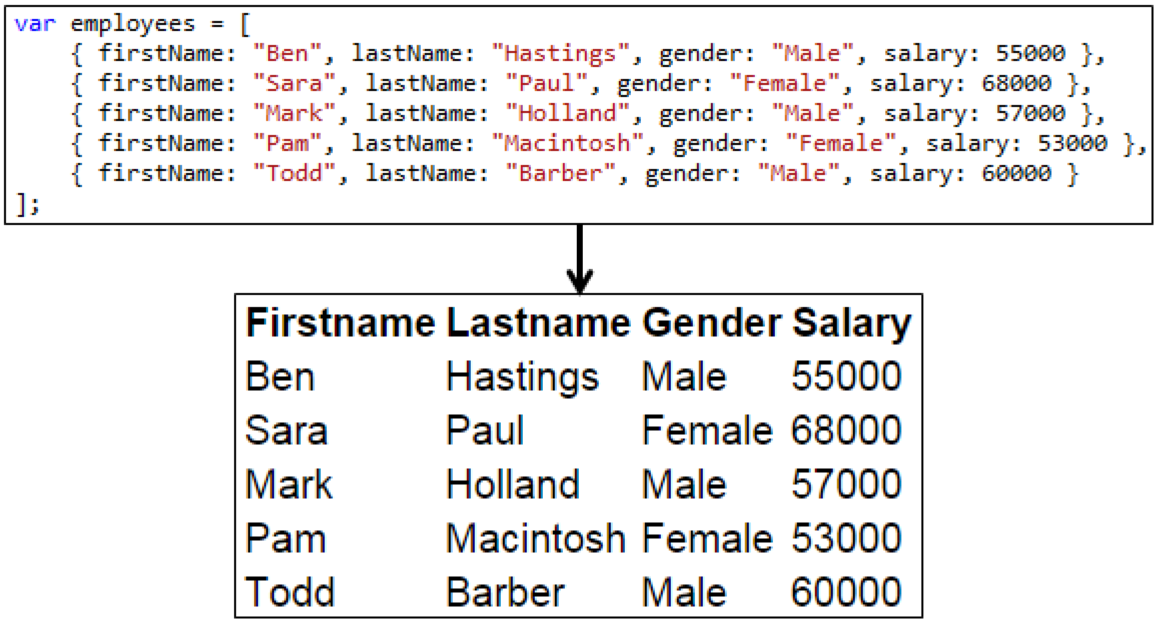
```
                </td>
            </tr>
        </table>
    </div>
</body>
</html>
```

AngularJS ng-repeat directive

ng-repeat is similar to foreach loop in C#.

Let us understand this with an example. Here is what we want to do.
1. For each employee we have in the employees array we want a table row. With in each cell of the table row we to display employee

- Firstname
- Lastname
- Gender
- Salary



```
var employees = [
    { firstName: "Ben", lastName: "Hastings", gender: "Male", salary: 55000 },
    { firstName: "Sara", lastName: "Paul", gender: "Female", salary: 68000 },
    { firstName: "Mark", lastName: "Holland", gender: "Male", salary: 57000 },
    { firstName: "Pam", lastName: "Macintosh", gender: "Female", salary: 53000 },
    { firstName: "Todd", lastName: "Barber", gender: "Male", salary: 60000 }
];
```

Firstname	Lastname	Gender	Salary
Ben	Hastings	Male	55000
Sara	Paul	Female	68000
Mark	Holland	Male	57000
Pam	Macintosh	Female	53000
Todd	Barber	Male	60000

This can be achieved very easily using **ng-repeat directive**

Script.js : The controll function builds the model for the view. The model employees has the list of all employees.

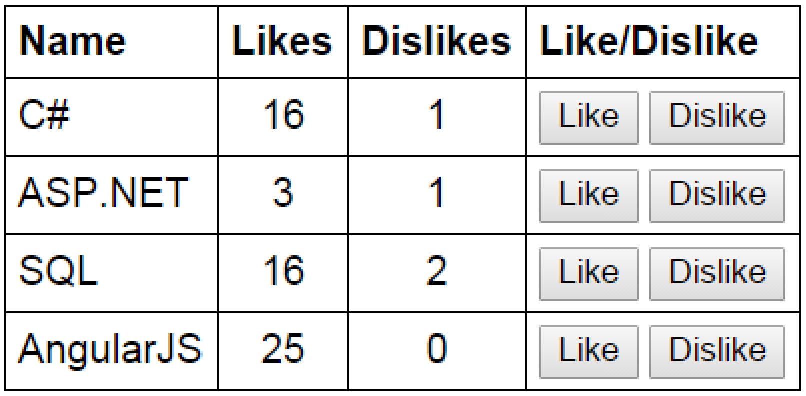
```
var app = angular
            .module("myModule", [])
            .controller("myController", function ($scope) {

                var employees = [
                    { firstName: "Ben", lastName: "Hastings", gender: "Male", salary: 55000 },
                    { firstName: "Sara", lastName: "Paul", gender: "Female", salary: 68000 },
                    { firstName: "Mark", lastName: "Holland", gender: "Male", salary: 57000 },
                    { firstName: "Pam", lastName: "Macintosh", gender: "Female", salary: 53000 },
                    { firstName: "Todd", lastName: "Barber", gender: "Male", salary: 60000 }
                ];

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

HtmlPage1.html : In the view, we are using ng-repeat directive which loops thru each employee in employees array. For each employee, we a get a table row, and in each table cell of the table row, the respective employee details (Firstname, Lastname, Gender, Salary) are retrieved using the angular binding expression.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <table>
            <thead>
                <tr>
                    <th>Firstname</th>
                    <th>Lastname</th>
                    <th>Gender</th>
                    <th>Salary</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees">
                    <td> {{ employee.firstName }} </td>
                    <td> {{ employee.lastName }} </td>
                    <td> {{ employee.gender }} </td>
                    <td> {{ employee.salary }} </td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```

Handling events in AngularJS

Let us understand with an example. Here is what we want to do.



Name	Likes	Dislikes	Like/Dislike
C#	16	1	Like Dislike
ASP.NET	3	1	Like Dislike
SQL	16	2	Like Dislike
AngularJS	25	0	Like Dislike

1. Display the list of technologies in a table
2. Provide the ability to like and dislike a technology
3. Increment the likes and dislikes when the respective buttons are clicked

Script.js : In the controller function we have 2 methods to increment likes and dislikes. Both the functions have the technology object that we want to like or dislike as a parameter.

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

            var technologies = [
                { name: "C#", likes: 0, dislikes: 0 },
                { name: "ASP.NET", likes: 0, dislikes: 0 },
                { name: "SQL", 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++;
            };
        });
```

HtmlPage1.html : Notice in the html below, we are associating **incrementLikes()** and **incrementDislikes()** functions with the respective button. When any of these buttons are clicked, the corresponsing technology object is automatically passed to the function, and the likes or dislikes property is incremented depending on which button is clicked.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Likes</th>
                    <th>Dislikes</th>
                    <th>Like/Dislike</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="technology in technologies">
                    <td> {{ technology.name }} </td>
                    <td style="text-align:center"> {{ technology.likes }} </td>
                    <td style="text-align:center"> {{ technology.dislikes }} </td>
                    <td>
                        <input type="button" ng-
click="incrementLikes(technology)" value="Like" />
                        <input type="button" ng-
click="incrementDislikes(technology)" value="Dislike" />
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```

AngularJS filters

Filters in angular can do 3 different things

1. Format data
2. Sort data
3. Filter data

Filters can be used with a binding expression or a directive

To apply a filter use pipe (|) character

Syntax : {{ expression | filterName:parameter **}}**

Angular filters for formatting data

Filter	**Description**
Lowercase	Formats all characters to lowercase
Uppercase	Formats all characters to uppercase
Number	Formats a number as text. Includes comma as thousands separator and the number of decimal places can be specified
Currency	Formats a number as a currency. $ is default. Custom currency and decimal places can be specified
Date	Formats date to a string based on the requested format

Angular Date formats

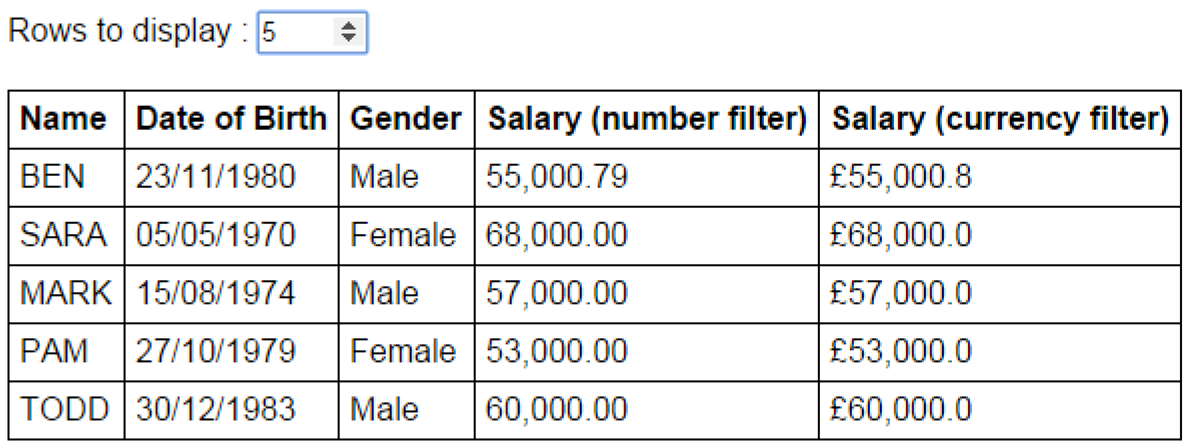
Format	**Result**
Yyyy	4 digit year. Exampe 1998
Yy	2 digit year. Example 1998 => 98
MMMM	January – December
MMM	Jan – Dec
MM	01 – 12
M	1 - 12 (No leading ZEROS)
Dd	01 – 31
D	1 - 31 (No leading ZEROS)

Angular date format documentation
https://docs.angularjs.org/api/ng/filter/date

limitTo filter : Can be used to limit the number of rows or characters in a string.

Syntax : {{ expression | limitTo : limit : begin**}}**

The following example uses all the above filters



Rows to display : 5

Name	Date of Birth	Gender	Salary (number filter)	Salary (currency filter)
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
PAM	27/10/1979	Female	53,000.00	£53,000.0
TODD	30/12/1983	Male	60,000.00	£60,000.0

Script.js

```
var app = 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", salary: 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.rowCount = 3;
        });
```

HtmlPage1.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        Rows to display : <input type="number" step="1"
                                 ng-model="rowCount" max="5" min="0" />
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Date of Birth</th>
                    <th>Gender</th>
                    <th>Salary (number filter)</th>
                    <th>Salary (currency filter)</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees | limitTo:rowCount">
                    <td> {{ employee.name | uppercase }} </td>
                    <td> {{ employee.dateOfBirth | date:"dd/MM/yyyy" }} </td>
                    <td> {{ employee.gender }} </td>
                    <td> {{ employee.salary | number:2 }} </td>
                    <td> {{ employee.salary | currency : "£" : 1 }} </td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```

To sort the data in Angular

1. Use orderBy filter

{{ orderBy_expression | orderBy : expression : reverse**}}**

Example : ng-repeat="employee in employees | orderBy:'salary':false"

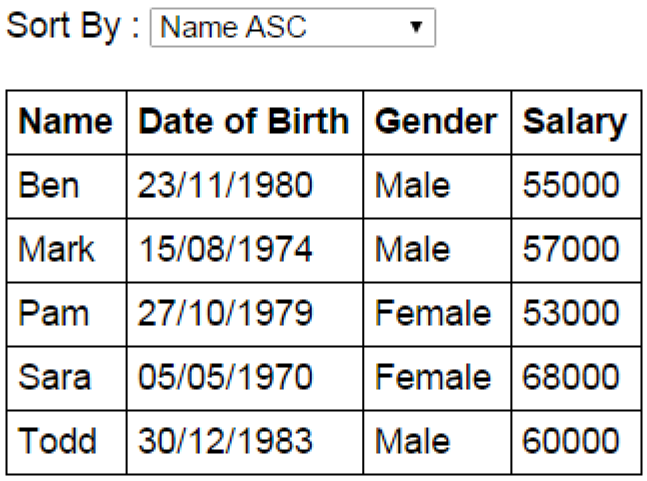
2. To sort in ascending order, set reverse to false

3. To sort in descending order, set reverse to true

4. You can also use + and - to sort in ascending and descending order respectively

Example : ng-repeat="employee in employees | orderBy:'+salary'"

Let us understand sorting data with an example.


Sort By : Name ASC

Name	Date of Birth	Gender	Salary
Ben	23/11/1980	Male	55000
Mark	15/08/1974	Male	57000
Pam	27/10/1979	Female	53000
Sara	05/05/1970	Female	68000
Todd	30/12/1983	Male	60000

The dropdownlist shows the columns and the direction we want to sort
When a dropdownlist item is selected, the table data should be sorted by the selected column
Script.js : The controller function builds the model. Also sortColumn property is added to the $scope object. Notice sortColumn property is initialized to name. This ensures that the data is sorted by name column in ascending order, when the form first loads.

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

            var employees = [
                {
                    name: "Ben", dateOfBirth: new Date("November 23, 1980"),
                    gender: "Male", salary: 55000
                },
                {
                    name: "Sara", dateOfBirth: new Date("May 05, 1970"),
                    gender: "Female", salary: 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.sortColumn = "name";

        });
```

HtmlPage1.html : The select element, has the list of columns by which the data should be sorted. + and - symbols control the sort direction. When the form initially loads notice that the data is sorted by name column in ascending order, and name option is automatically selected in the select element. Notice the orderBy filter is using the sortColumn property that is attached to the $scope object. When the selection in the select element changes, the sortColumn property of the $scope object will be updated automatically with the selected value, and in turn the updated value is used by the orderBy filter to sort the data.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        Sort By :
        <select ng-model="sortColumn">
            <option value="name">Name ASC</option>
            <option value="+dateOfBirth">Date of Birth ASC</option>
            <option value="+gender">Gender ASC</option>
            <option value="-salary">Salary DESC</option>
        </select>
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Date of Birth</th>
                    <th>Gender</th>
                    <th>Salary</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees | orderBy:sortColumn">
                    <td>
                        {{ employee.name }}
                    </td>
                    <td>
                        {{ employee.dateOfBirth | date:"dd/MM/yyyy" }}
                    </td>
                    <td>
                        {{ employee.gender }}
                    </td>
                    <td>
                        {{ employee.salary  }}
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```