Unit 2

What is Filter in AngularJS?

A filter formats the value of an expression to display to the user.

For example, if you want to have your strings in either in lowercase or all in uppercase, you can do this by using filters in Angular.

There are built-in filters such as 'lowercase', 'uppercase' which can retrieve the output in lowercase and uppercase accordingly. Similarly, for numbers, you can use other filters.

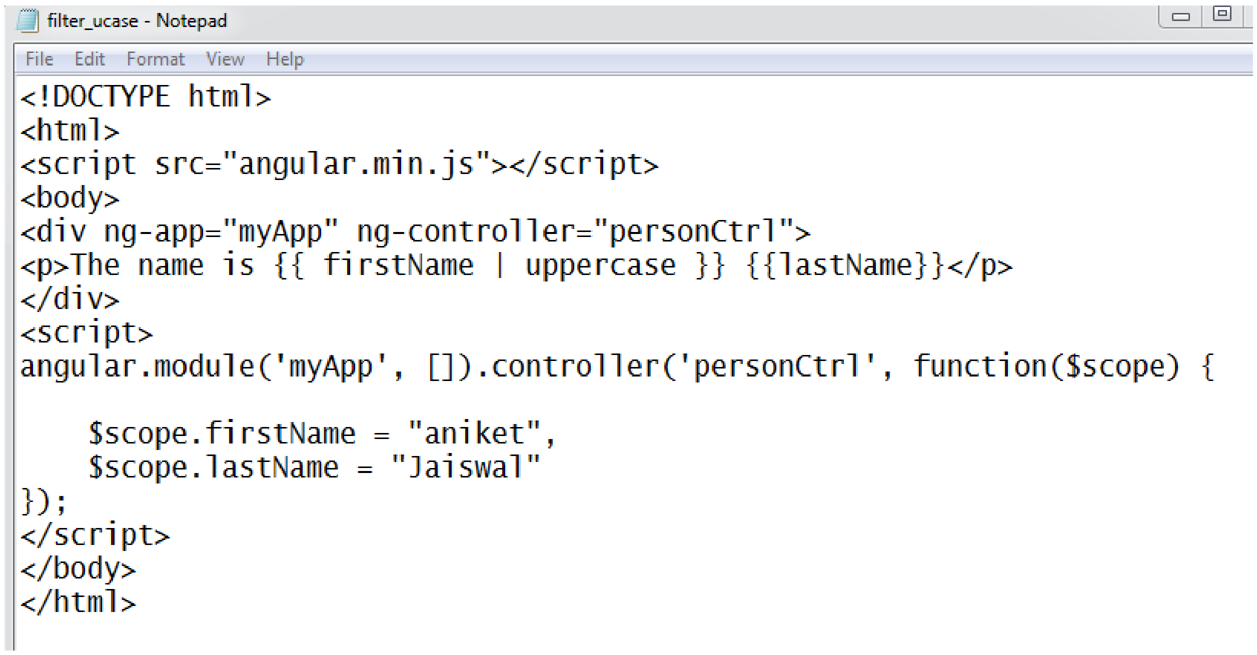
AngularJS provides filters to transform data:

- `currency` Format a number to a currency format.
- `date` Format a date to a specified format.
- `filter` Select a subset of items from an array.
- `json` Format an object to a JSON string.
- `limitTo` Limits an array/string, into a specified number of elements/characters.
- `lowercase` Format a string to lower case.
- `number` Format a number to a string.
- `orderBy` Orders an array by an expression.
- `uppercase` Format a string to upper case.

Adding Filters to Expressions

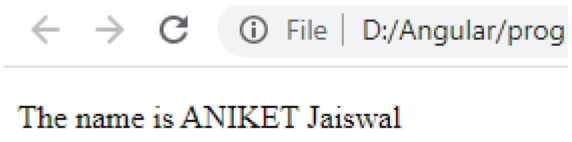
Filters can be added to expressions by using the pipe character `|`, followed by a filter.

`Uppercase and lowercase`



```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="personCtrl">
<p>The name is {{ firstName | uppercase }} {{lastName}}</p>
</div>
<script>
angular.module('myApp', []).controller('personCtrl', function($scope) {

    $scope.firstName = "aniket",
    $scope.lastName = "Jaiswal"
});
</script>
</body>
</html>
```

OUTPUT

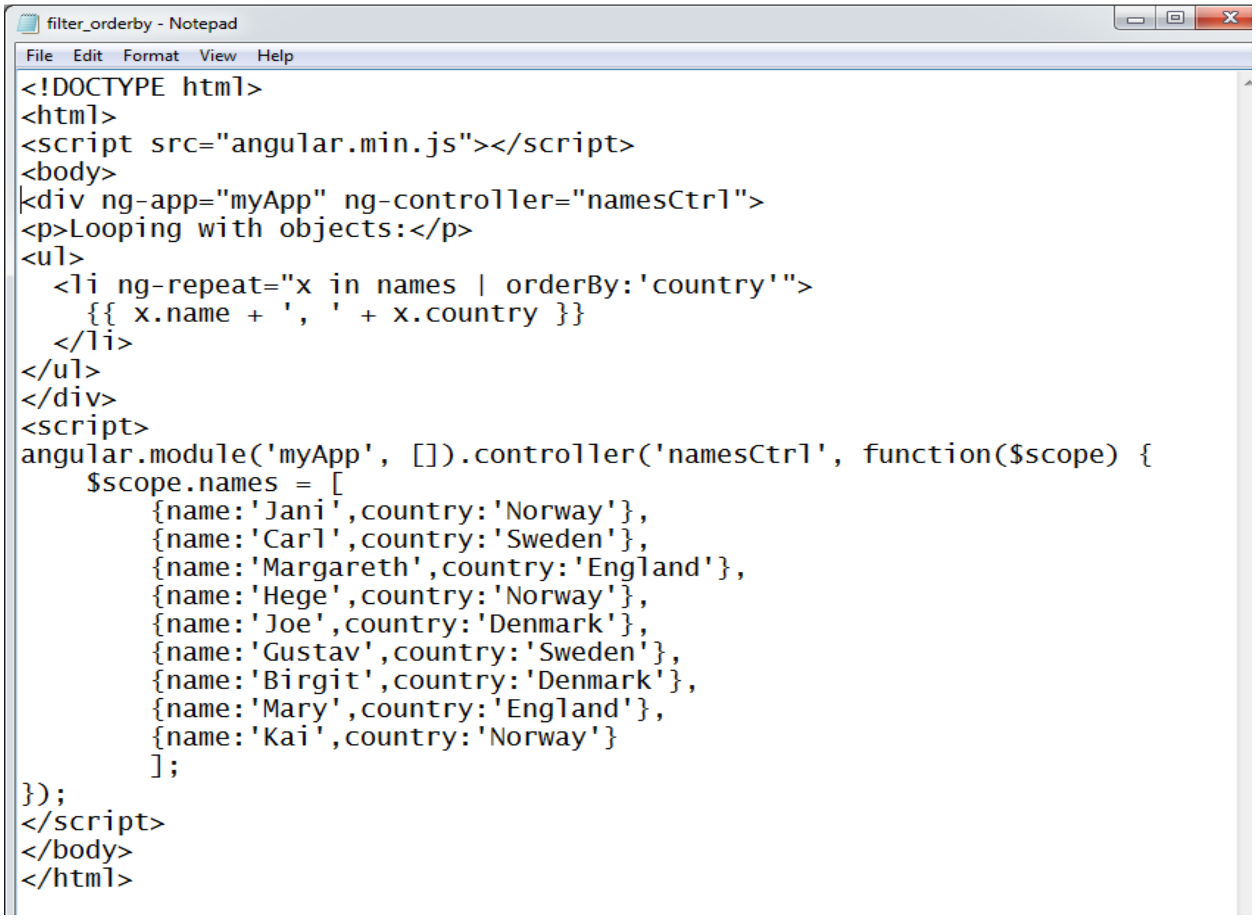


The name is ANIKET Jaiswal

Adding Filters to Directives

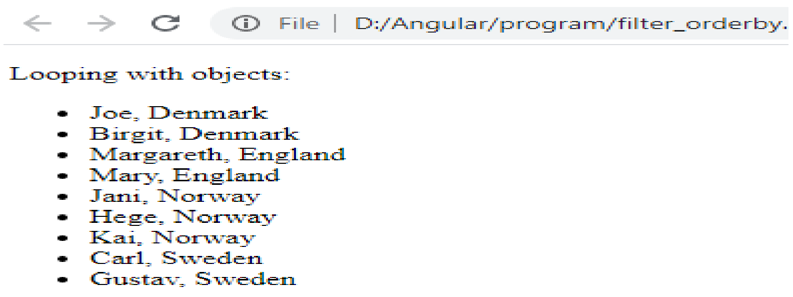
orderBy

Filters are added to directives, like `ng-repeat`, by using the pipe character `|`, followed by a filter:



```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="namesCtrl">
<p>Looping with objects:</p>
<ul>
  <li ng-repeat="x in names | orderBy:'country'">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>
</div>
<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Carl',country:'Sweden'},
        {name:'Margareth',country:'England'},
        {name:'Hege',country:'Norway'},
        {name:'Joe',country:'Denmark'},
        {name:'Gustav',country:'Sweden'},
        {name:'Birgit',country:'Denmark'},
        {name:'Mary',country:'England'},
        {name:'Kai',country:'Norway'}
        ];
});
</script>
</body>
</html>
```

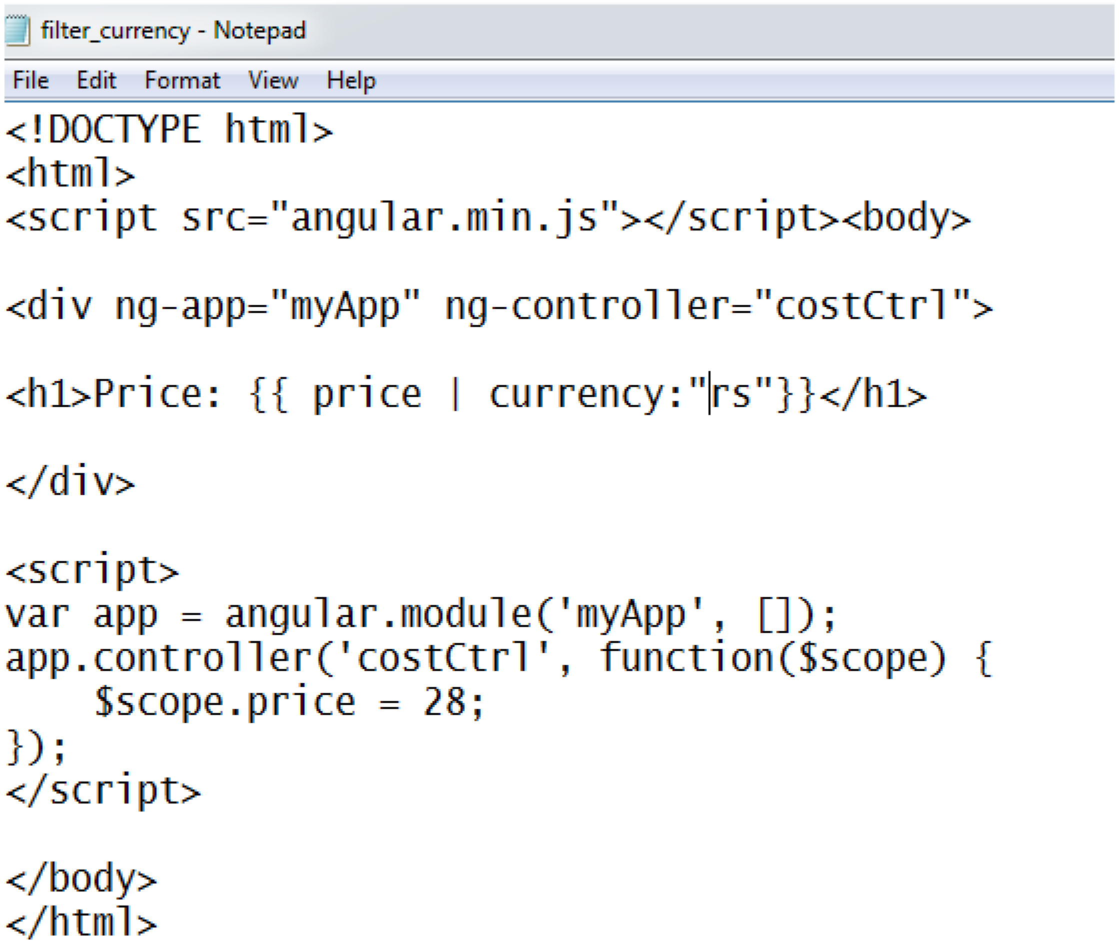
OUTPUT



```
Looping with objects:

  • Joe, Denmark
  • Birgit, Denmark
  • Margareth, England
  • Mary, England
  • Jani, Norway
  • Hege, Norway
  • Kai, Norway
  • Carl, Sweden
  • Gustav, Sweden
```

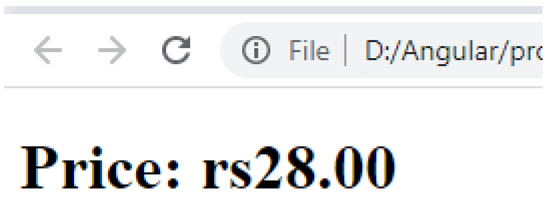
currency Filter

The `currency` filter formats a number as currency



```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script><body>

<div ng-app="myApp" ng-controller="costCtrl">

<h1>Price: {{ price | currency:"rs"}}</h1>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('costCtrl', function($scope) {
    $scope.price = 28;
});
</script>

</body>
</html>
```

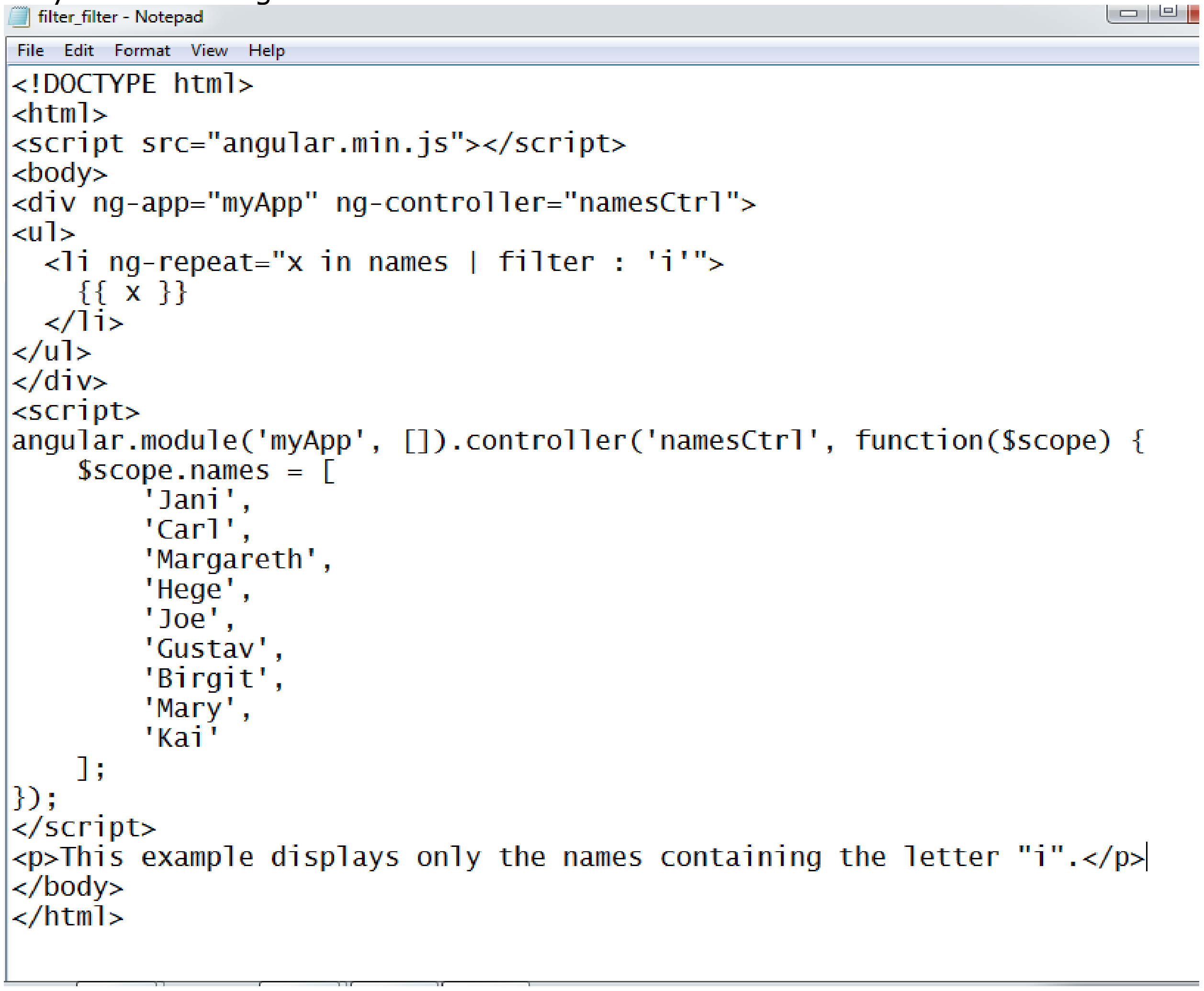
OUTPUT:



```
Price: rs28.00
```

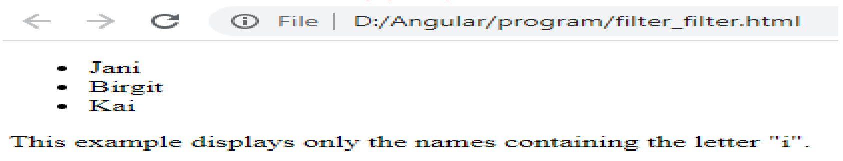
The filter Filter

The `filter` filter selects a subset of an array.
The `filter` filter can only be used on arrays, and it returns an array containing only the matching items.

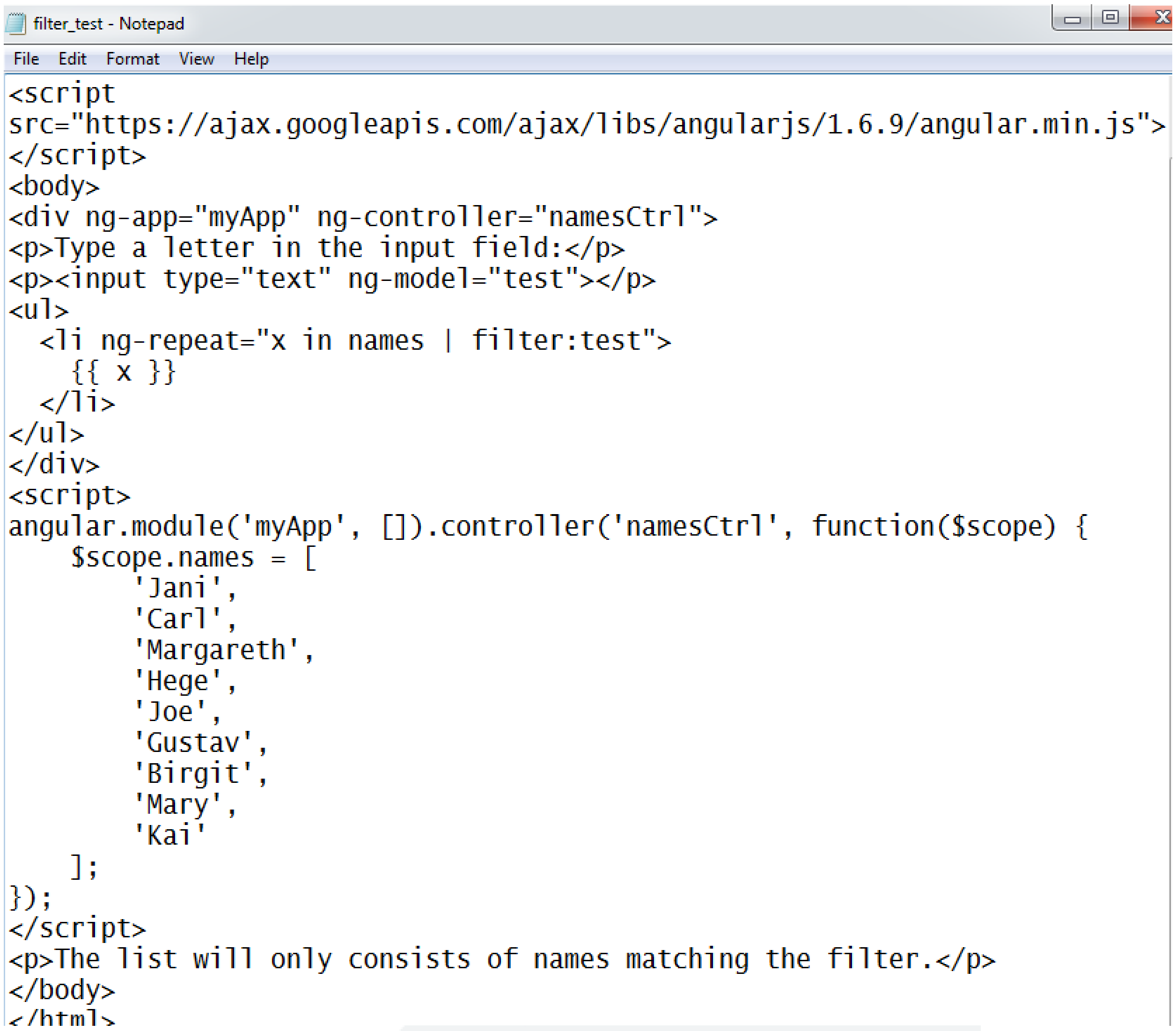


```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="namesCtrl">
<ul>
  <li ng-repeat="x in names | filter : 'i'">
    {{ x }}
  </li>
</ul>
</div>
<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        'Jani',
        'Carl',
        'Margareth',
        'Hege',
        'Joe',
        'Gustav',
        'Birgit',
        'Mary',
        'Kai'
    ];
});
</script>
<p>This example displays only the names containing the letter "i".</p>
</body>
</html>
```

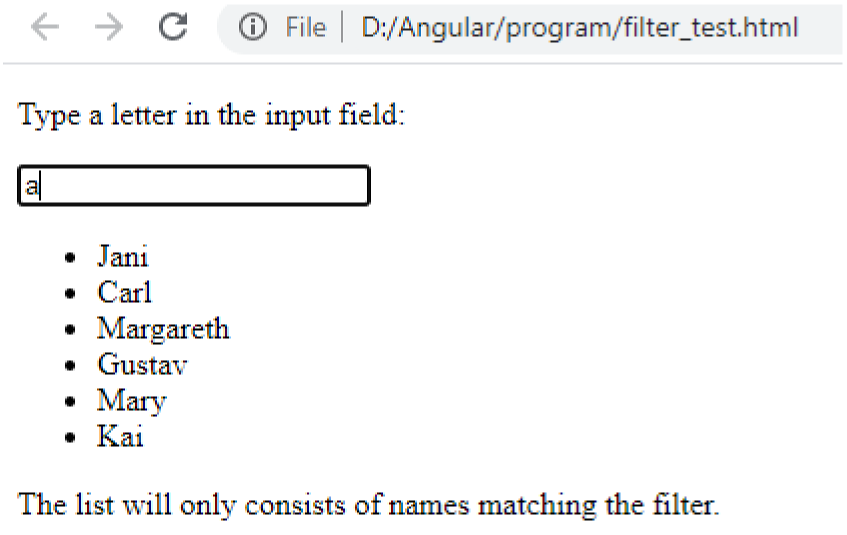
OUTPUT:



```
• Jani
• Birgit
• Kai

This example displays only the names containing the letter "i".
```



```
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-controller="namesCtrl">
<p>Type a letter in the input field:</p>
<p><input type="text" ng-model="test"></p>
<ul>
  <li ng-repeat="x in names | filter:test">
    {{ x }}
  </li>
</ul>
</div>
<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        'Jani',
        'Carl',
        'Margareth',
        'Hege',
        'Joe',
        'Gustav',
        'Birgit',
        'Mary',
        'Kai'
    ];
});
</script>
<p>The list will only consists of names matching the filter.</p>
</body>
</html>
```

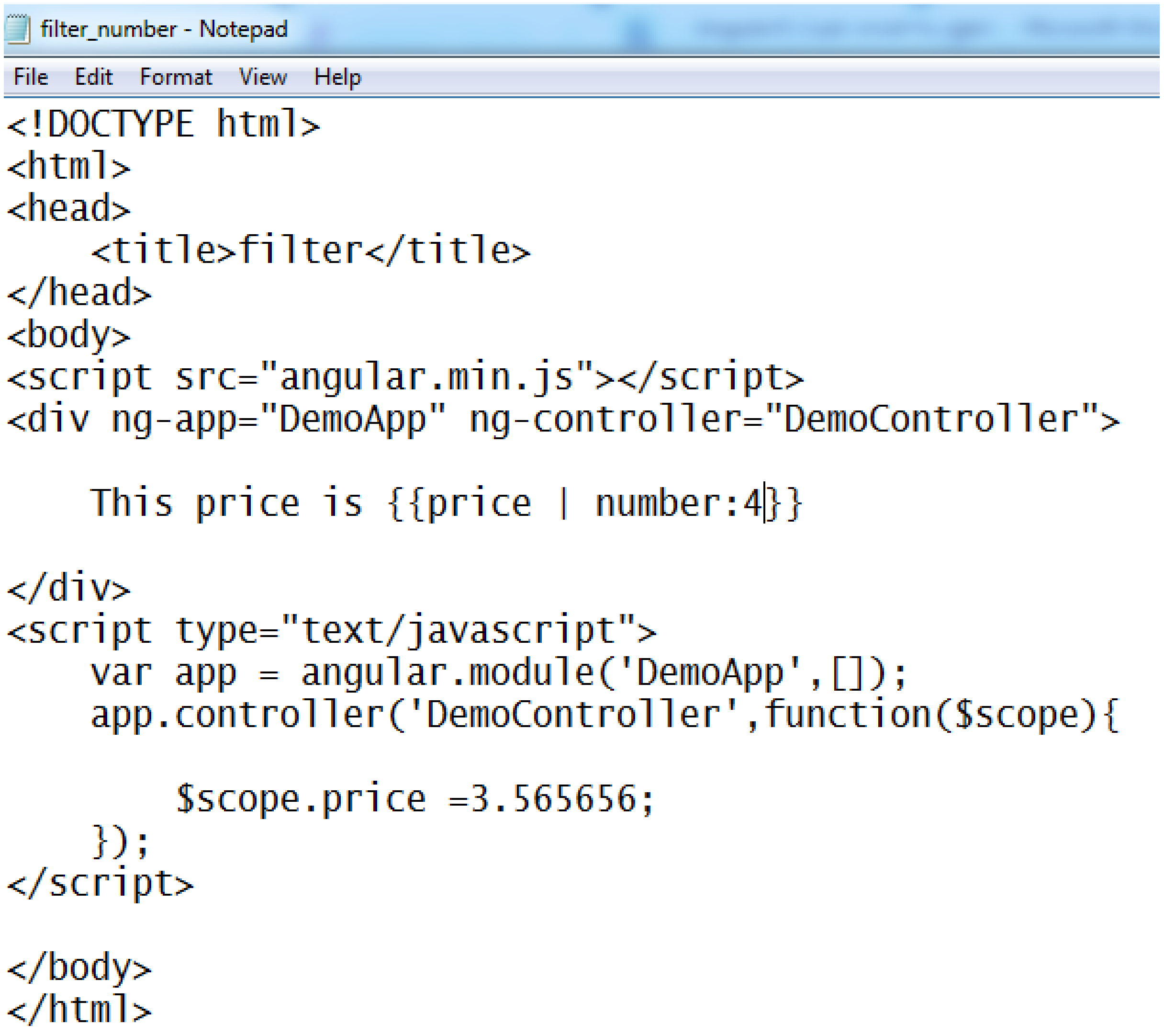


Type a letter in the input field:

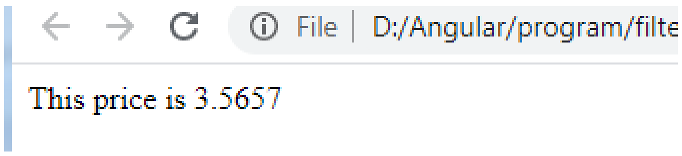
a

- Jani
- Carl
- Margareth
- Gustav
- Mary
- Kai

The list will only consists of names matching the filter.



```
<!DOCTYPE html>
<html>
<head>
    <title>filter</title>
</head>
<body>
<script src="angular.min.js"></script>
<div ng-app="DemoApp" ng-controller="DemoController">

    This price is {{price | number:4}}

</div>
<script type="text/javascript">
    var app = angular.module('DemoApp',[]);
    app.controller('DemoController',function($scope){

        $scope.price =3.565656;
    });
</script>

</body>
</html>
```



```
This price is 3.5657
```

Date Filter

The date filter is used to format a date to a specified format. The date can be a date object, milliseconds, or a datetime string like "2016-05-05T09:05:05.035Z" By default, the format is "MMM d, y" (May 5, 2016).

Syntax:

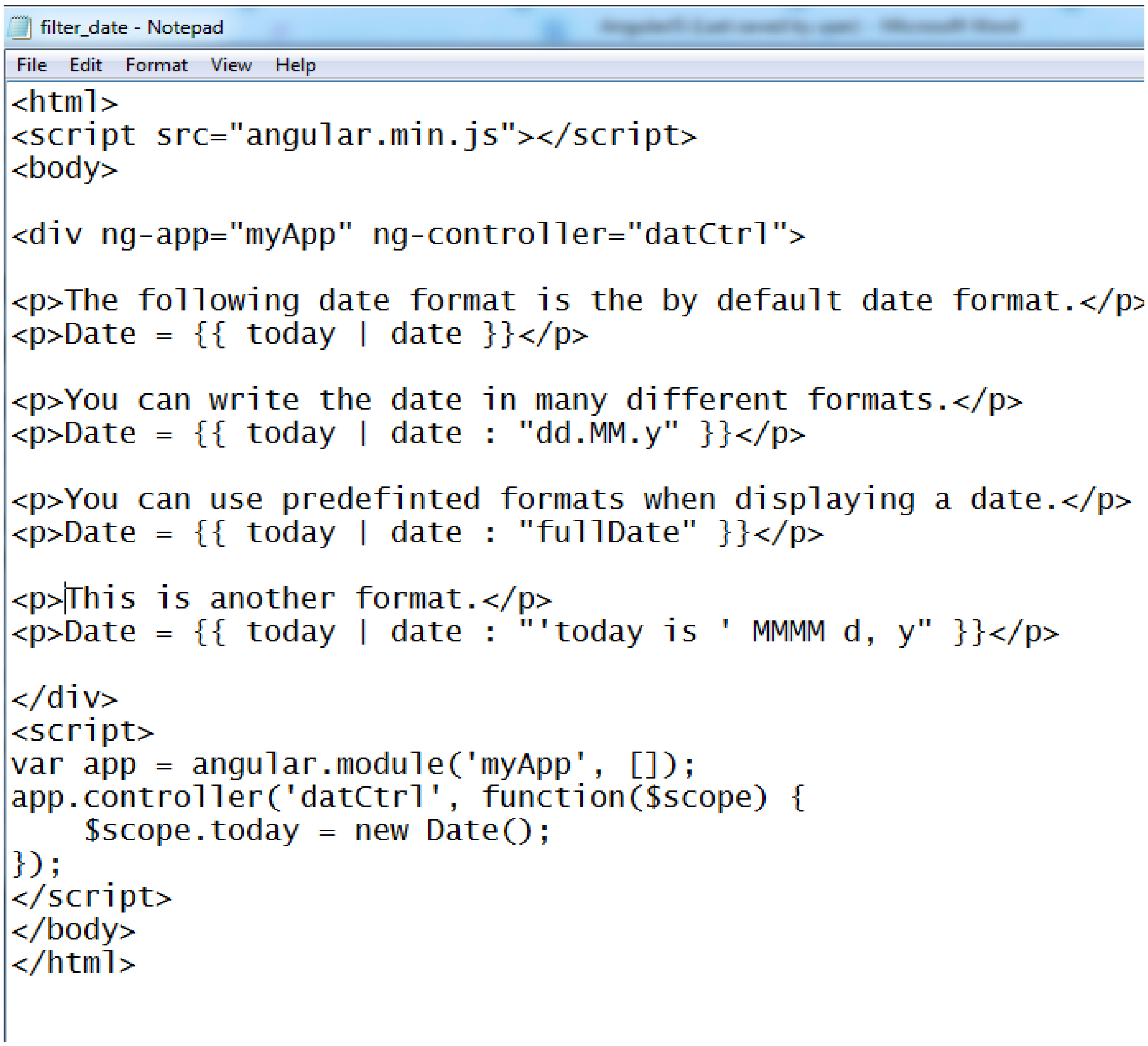
1. {{ date_expression | date : format : timezone}}
2. .
3. .
4. $filter('date')(date, format, timezone)

date: It specifies a date format either as Date object, milliseconds (string or number) or various ISO 8601 datetime string formats (e.g. yyyy-MM-ddTHH:mm:ss.sssZ and its shorter versions like yyyy-MM-ddTHH:mmZ, yyyy-MM-dd or yyyyMMddTHHmmssZ).

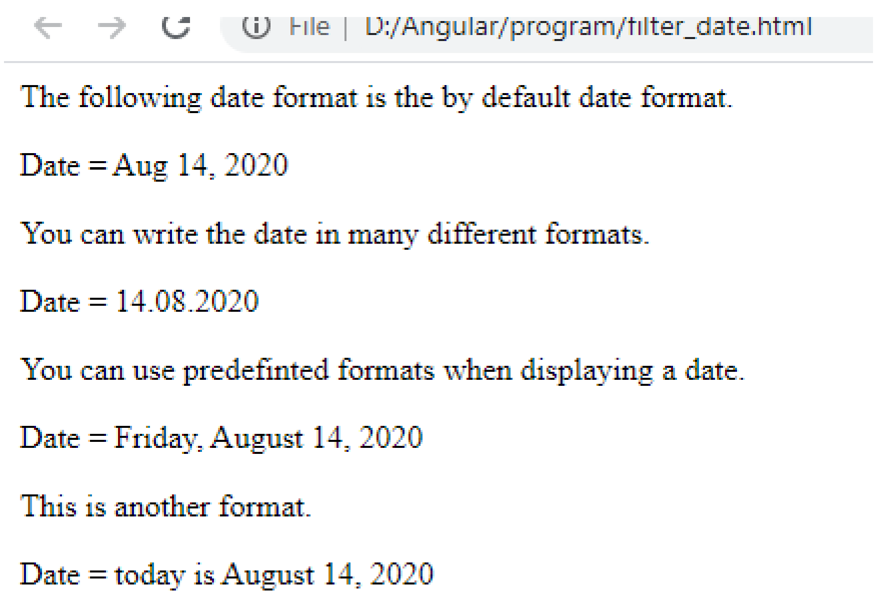
format: It is optional. It is used to display the date in, which can be composed of the following elements.

- "yyyy" year (2016)
- "yy" year (16)
- "y" year (2016)
- "MMMM" month (January)
- "MMM" month (Jan)
- "MM" month (01)
- "M" month (1)
- "dd" day (06)
- "d" day (6)
- "EEEE" day (Tuesday)
- "EEE" day (Tue)
- "HH" hour, 00-23 (09)
- "H" hour 0-23 (9)
- "hh" hour in AM/PM, 00-12 (09)
- "h" hour in AM/PM, 0-12 (9)
- "mm" minute (05)
- "m" minute (5)
- "ss" second (05)
- "s" second (5)
- "sss" millisecond (035)

- "a" (AM/PM)
- "Z" timezone (from -1200 to +1200)
- "ww" week (00-53)
- "w" week (0-53)



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

<div ng-app="myApp" ng-controller="datCtrl">

<p>The following date format is the by default date format.</p>
<p>Date = {{ today | date }}</p>

<p>You can write the date in many different formats.</p>
<p>Date = {{ today | date : "dd.MM.y" }}</p>

<p>You can use predefinted formats when displaying a date.</p>
<p>Date = {{ today | date : "fullDate" }}</p>

<p>This is another format.</p>
<p>Date = {{ today | date : "'today is ' MMMM d, y" }}</p>

</div>
<script>
var app = angular.module('myApp', []);
app.controller('datCtrl', function($scope) {
    $scope.today = new Date();
});
</script>
</body>
</html>
```


File | D:/Angular/program/filter_date.html

```
The following date format is the by default date format.

Date = Aug 14, 2020

You can write the date in many different formats.

Date = 14.08.2020

You can use predefinted formats when displaying a date.

Date = Friday, August 14, 2020

This is another format.

Date = today is August 14, 2020
```

Limit to filter

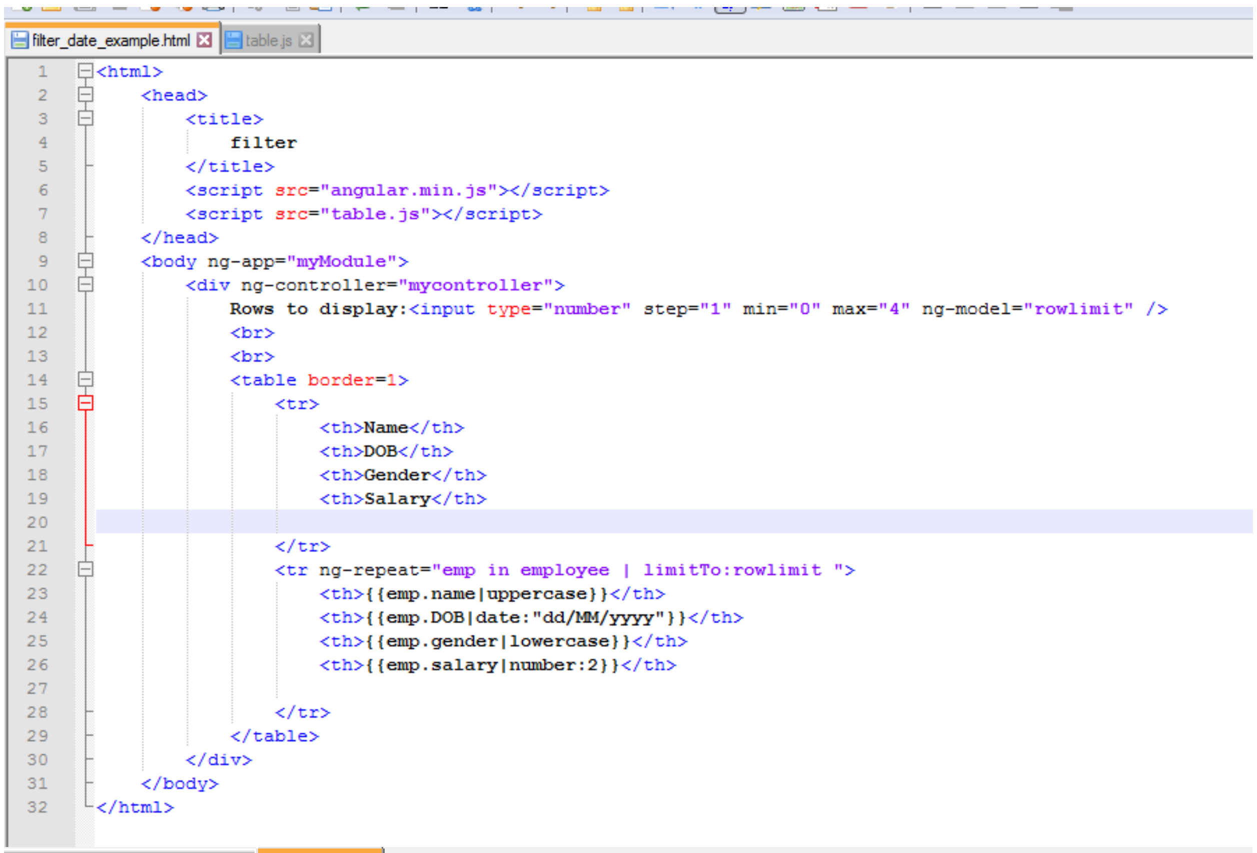
The `limitTo` filter returns an array or a string containing only a specified number of elements.

When the `limitTo` filter is used for arrays, it returns an array containing only the specified number of items.

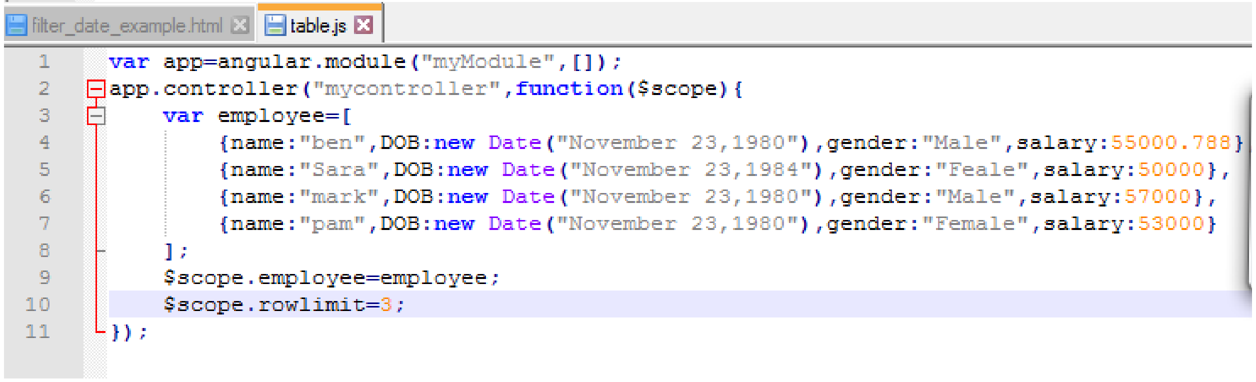
When the `limitTo` filter is used for strings, it returns a string containing, only the specified number of characters.

When the `limitTo` filter is used for numbers, it returns a string containing only the specified number of digits.

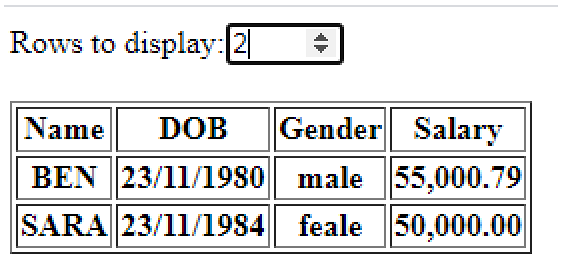
Use negative numbers to return elements starting from the end of the element, instead of the beginning


filter_date_example.html | table.js

```
<html>
    <head>
        <title>
            filter
        </title>
        <script src="angular.min.js"></script>
        <script src="table.js"></script>
    </head>
    <body ng-app="myModule">
        <div ng-controller="mycontroller">
            Rows to display:<input type="number" step="1" min="0" max="4" ng-model="rowlimit" />
            <br>
            <br>
            <table border=1>
                <tr>
                    <th>Name</th>
                    <th>DOB</th>
                    <th>Gender</th>
                    <th>Salary</th>

                </tr>
                <tr ng-repeat="emp in employee | limitTo:rowlimit ">
                    <th>{{emp.name|uppercase}}</th>
                    <th>{{emp.DOB|date:"dd/MM/yyyy"}}</th>
                    <th>{{emp.gender|lowercase}}</th>
                    <th>{{emp.salary|number:2}}</th>

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


filter_date_example.html | table.js

```
var app=angular.module("myModule",[]);
app.controller("mycontroller",function($scope){
    var employee=[
        {name:"ben",DOB:new Date("November 23,1980"),gender:"Male",salary:55000.788}
        {name:"Sara",DOB:new Date("November 23,1984"),gender:"Feale",salary:50000},
        {name:"mark",DOB:new Date("November 23,1980"),gender:"Male",salary:57000},
        {name:"pam",DOB:new Date("November 23,1980"),gender:"Female",salary:53000}
    ];
    $scope.employee=employee;
    $scope.rowlimit=3;
});
```

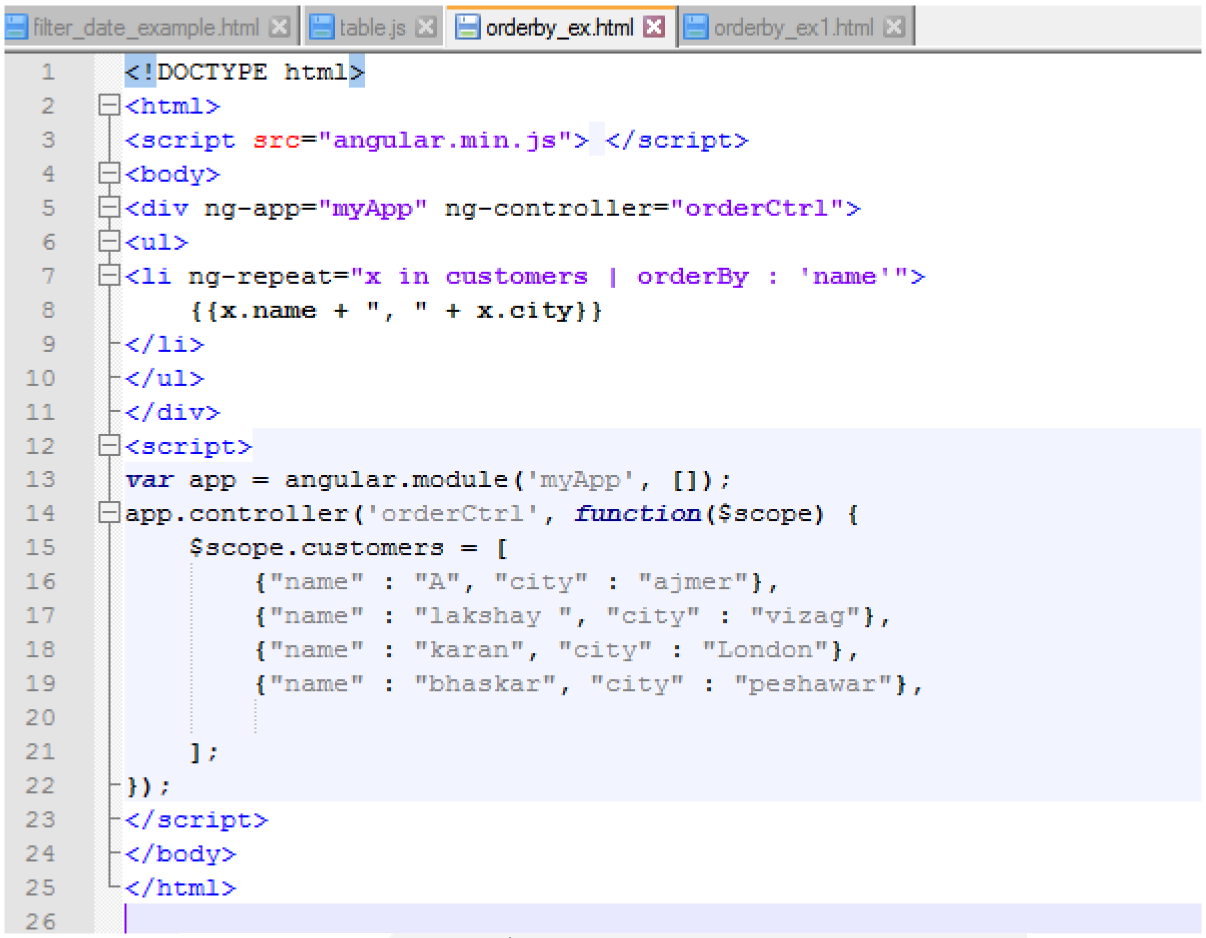

Rows to display: 2

Name	DOB	Gender	Salary
BEN	23/11/1980	male	55,000.79
SARA	23/11/1984	feale	50,000.00

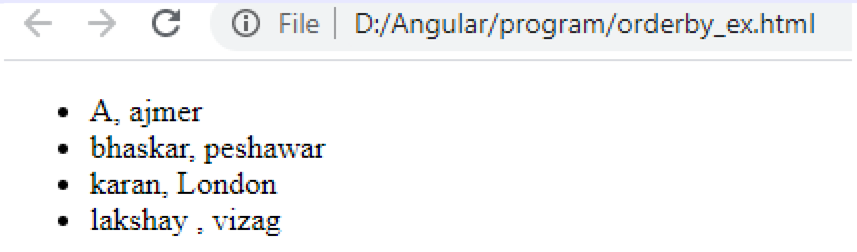
orderBy filter:

The **orderBy filter** is a handy tool used in angular js.
The orderBy channel encourages you to sort an exhibit.
Of course, it sorts strings in sequential order requests and numbers in the numerical request.
Syntax:

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

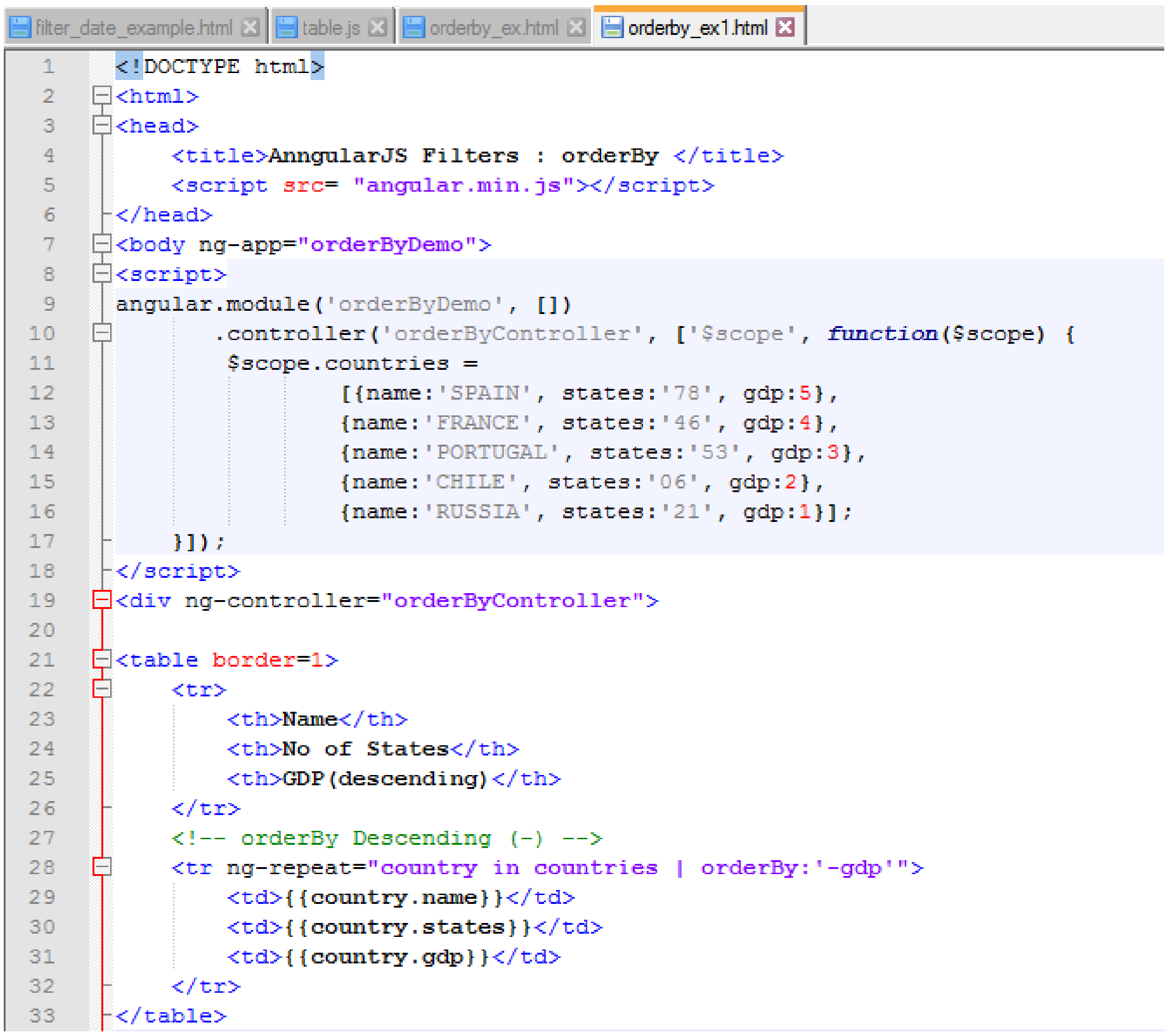


```
<!DOCTYPE html>
<html>
<script src="angular.min.js"> </script>
<body>
<div ng-app="myApp" ng-controller="orderCtrl">
<ul>
<li ng-repeat="x in customers | orderBy : 'name'">
    {{x.name + ", " + x.city}}
</li>
</ul>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('orderCtrl', function($scope) {
    $scope.customers = [
        {"name" : "A", "city" : "ajmer"},
        {"name" : "lakshay ", "city" : "vizag"},
        {"name" : "karan", "city" : "London"},
        {"name" : "bhaskar", "city" : "peshawar"},

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

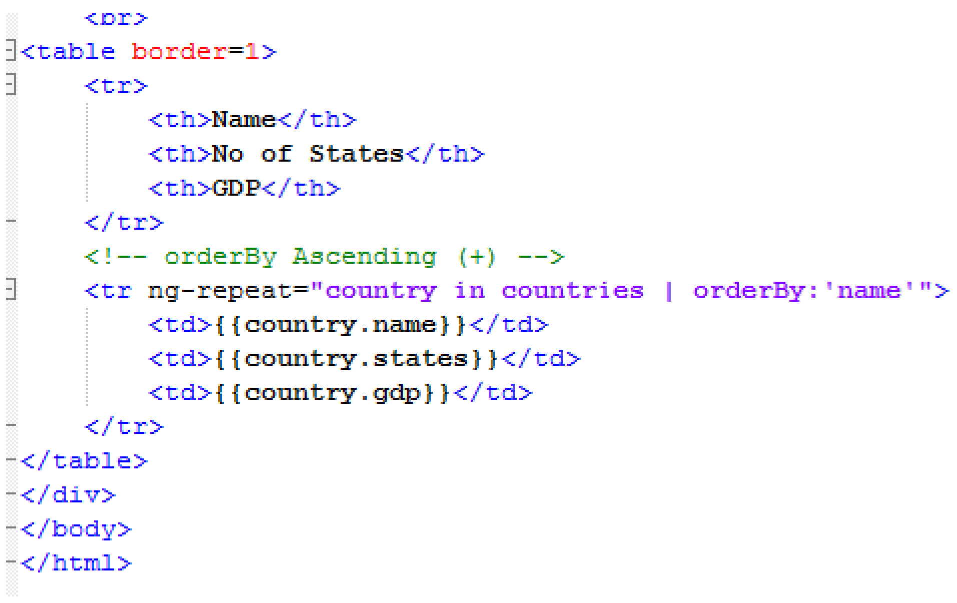


- A, ajmer
- bhaskar, peshawar
- karan, London
- lakshay , vizag

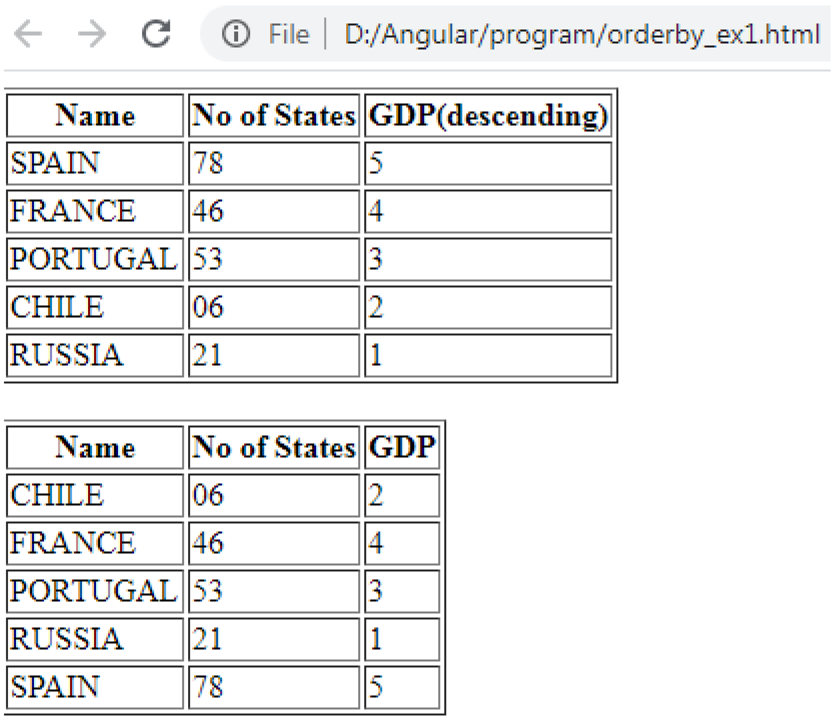
Orderby(ascending)



```
<!DOCTYPE html>
<html>
<head>
    <title>AnngularJS Filters : orderBy </title>
    <script src= "angular.min.js"></script>
</head>
<body ng-app="orderByDemo">
<script>
angular.module('orderByDemo', [])
        .controller('orderByController', ['$scope', function($scope) {
         $scope.countries =
                 [{name:'SPAIN', states:'78', gdp:5},
                 {name:'FRANCE', states:'46', gdp:4},
                 {name:'PORTUGAL', states:'53', gdp:3},
                 {name:'CHILE', states:'06', gdp:2},
                 {name:'RUSSIA', states:'21', gdp:1}];
    }]);
</script>
<div ng-controller="orderByController">

<table border=1>
    <tr>
        <th>Name</th>
        <th>No of States</th>
        <th>GDP(descending)</th>
    </tr>
    <!-- orderBy Descending (-) -->
    <tr ng-repeat="country in countries | orderBy:'-gdp'">
        <td>{{country.name}}</td>
        <td>{{country.states}}</td>
        <td>{{country.gdp}}</td>
    </tr>
</table>
```



```
    <br>
<table border=1>
    <tr>
        <th>Name</th>
        <th>No of States</th>
        <th>GDP</th>
    </tr>
    <!-- orderBy Ascending (+) -->
    <tr ng-repeat="country in countries | orderBy:'name'">
        <td>{{country.name}}</td>
        <td>{{country.states}}</td>
        <td>{{country.gdp}}</td>
    </tr>
</table>
</div>
</body>
</html>
```


File | D:/Angular/program/orderby_ex1.html

Name	No of States	GDP(descending)
SPAIN	78	5
FRANCE	46	4
PORTUGAL	53	3
CHILE	06	2
RUSSIA	21	1

Name	No of States	GDP
CHILE	06	2
FRANCE	46	4
PORTUGAL	53	3
RUSSIA	21	1
SPAIN	78	5

Custom filter

```
<!doctype html>
<html lang="en">
<head>
  <title>Example - example-filter-reverse-production</title>
  <script src="angular.min.js"> </script>
 </head>
<body ng-app="ReverseFilter">
            <div ng-controller="ReverseController">
                        <input ng-model="msg" type="text"><br>
                        Content with No filter: {{msg}}<br>
                        Content after Reverse filter: {{msg|reverse}}<br>
            </div>
</body>
<script>



var app =angular.module('ReverseFilter', [])
```

```
            app.filter('reverse', function() {
                                                            return function(input) {
                                                    input = input || '';
                                                    var out = '';
                                                    for (var x = 0; x < input.length; x++) {
                                                                    //window.alert(out);
                                                                    out = input.charAt(x) + out;

                        }
                                return out;

        };
                                });
                app.controller('ReverseController', function($scope) {
                        $scope.msg = 'angular';
                        });


</script>
</html>
```

What is Single Page Applications?

Single page applications or (SPAs) are web applications that load a single HTML page and dynamically update the page based on the user interaction with the web application.

What is Routing in AngularJS?

In AngularJS, routing is what allows you to create Single Page Applications.

- AngularJS routes enable you to create different URLs for different content in your application.
- AngularJS routes allow one to show multiple contents depending on which route is chosen.
- A route is specified in the URL after the # sign.

ngRoute

AngularJS ngRoute module provides routing, deep linking services and directives for angular applications. We have to download `angular-route.js` script that contains the ngRoute module from AngularJS official website to use the routing feature.

If you are bundling this file into your application, then you can add it to your page with below code.

```
<script src="angular-route.js">
```

Then load the ngRoute module in your AngularJS application by adding it as a dependent module as shown below.

```
angular.module('appName', ['ngRoute']);
```

ngView

ngView directive is used to display the HTML templates or views in the specified routes. Every time the current route changes, the included view changes with it according to the configuration of the $route service.

$routeProvider

$routeProvider is used to configure the routes. We use the ngRoute config() to configure the $routeProvider. The config() takes a function which takes the $routeProvider as parameter and the routing configuration goes inside the function.

$routeProvider has a simple API, accepting either the when() or otherwise() method.

AngularJS Routing Syntax

The following syntax is used to configure the routes in AngularJS.

```
var app = angular.module("appName", ['ngRoute']);

app.config(function($routeProvider) {
      $routeProvider
             .when('/view1', {
                    templateUrl: 'view1.html',
                    controller: 'FirstController'
             })
             .when('/view2', {
                    templateUrl: 'view2.html',
                    controller: 'SecondController'
             })
             .otherwise({
                    redirectTo: '/view1'
             });
});
```

when() method takes a **path**and a **route** as parameters.

path is a part of the URL after the # symbol.

route contains two properties – templateUrl and controller.

templateUrl property defines which HTML template AngularJS should load and display inside the div with the ngView directive.

controller property defines which controllers should be used with the HTML template.

When the application is loaded, **path** is matched against the part of the URL after the # symbol. If no route paths matches the given URL the browser will be redirected to the path specified in the otherwise() function.

Example:

Index.html

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"> </script>
<script src="angular-route.min.js"></script>

<body ng-app="myApp">
<p><a href="#/!">
<img src="logo.png" alt="logo" style="width: 50 height:50;"></a></p>

<a href="#!/courses">Courses</a>
<br>
<a href="#!/internships">Internships</a>
<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
        $routeProvider
        .when("/", {
                template : '<h1>Welcome to Institute</h1> <p> Click on the links to
change this content </p>'
        })
        .when("/courses", {
                template : `<h1>Courses Offered</h1> <p> <ul> <li>Machine Learning
Foundation</li>
                                        <li>Web Classes</li> <li>System
Design</li>
                                        </ul></p>`
        })
        .when("/internships", {
                template : `<h1>Hire With Us</h1>
                                <p>
                                        <ul>
```

```
                                                        <li>Software Developer</li>
                                                        <li>Technical Content Writer</li>
                                                        <li>Technical Content Engineer</li>
                                                        </ul>
                                                </p>`
        });
});
</script>

</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html >
<head>
  <title>Routing</title>

  <script src="angular.min.js"></script>
  <script src="angular-route.min.js"></script>
<script>
        angular.module("DemoApp", ['ngRoute'])
           .controller("DemoController", function($scope) {
              $scope.title = "Simple Router Example";
           })
           .config(['$routeProvider', function($routeProvider) {
              $routeProvider.
                 when('/abc', {
                    template: '<h2>ABC</h2> from the template',
                 }).
                 when('/def', {
                    templateUrl: 'def.html',
                 }).
                 otherwise({
                    redirectTo: '/'
                 });
           }]);
</script>
</head>

<body ng-app="DemoApp" ng-controller="DemoController">
```

```
<h1>{{title}}</h1>
        <a href="#!">home</a>
        <a href="#!/abc">abc</a>
        <a href="#!/def">def</a>
<div ng-view></div>
</body>
</html>
```

Route Parameters

You can embed parameters into the route path. Here is an AngularJS route path parameter example:

```
#/books/12345
```

This is a URL with a route path in. In fact it pretty much consists of just the route path. The parameter part is the `12345` which is the specific id of the book the URL points to.

AngularJS can extract values from the route path if we define parameters in the route paths when we configure the `$routeProvider`. Here is the example `$routeProvider` from earlier, but with parameters inserted into the route paths:

The value 12345 will be extracted as parameter.

Your controller functions can get access to route parameters via the AngularJS $routeParams service like this:

```
module.controller("RouteController", function($scope, $routeParams) {
   $scope.param = $routeParams.param;
})
```

Example:

```
<!DOCTYPE html>
<html>
        <head>
        <title>AngularJS Routing with Prameters Example</title>
        <script src="angular.min.js"></script>
        <script src="angular-route.min.js"></script>
```

```
          <script>
                    var app = angular.module("routesApp", ['ngRoute']);
                    app.config(['$routeProvider',function ($routeProvider) {
                    $routeProvider.when('/routeURL1/:userId', {
                              templateUrl: 'sample1.html',
                              controller: 'sample1Controller'
                                             }).
                                        when('/routeURL2', {
                              templateUrl: 'sample2.html',
                              controller: 'sample2Controller'
                                             }).
                                        otherwise({
                              redirectTo: '/login'
                                             });

                         }
                                   ]);

                    app.controller('sample1Controller',function($scope,$routeParams){
                              $scope.uid = $routeParams.userId;
                                        })
                    app.controller('sample2Controller',function($scope){
                              $scope.message='Test Sample Page 2 URL';
                                   })

</script>
</head>
<body>
<h2>AngularJS Routing with Parameters Example</h2>
<div ng-app="routesApp">
          <ul>
                    <li>
                              <a href="#!/routeURL1/34124">Route1 with Parameters</a>
                    </li>
                    <li>
                              <a href="#!/routeURL2">Sample Route2</a>
                    </li>
          </ul>
          <div ng-view></div>
</div></body></html>
```

When a route change events occurs the following events are triggered,

- $locationChangeStart
- $routeChangeStart
- $locationChangeSuccess
- $routeChangeSuccess

$routeChangeStart

When routeChangeStart is triggered a function is called and that function will have parameters like event which is called, the second parameter is the next parameter in which the next route is called and the last parameter is current parameter to stick to that route only.

```
.controller("studentsController", function($http, $route, $scope) {
            $scope.$on(“$routeChangeStart”, , function(event, next, current) {})
```

Now the next thing which we have to do is to display a confirmation which we will do using JavaScript confirm function,

```
.controller("studentsController", function($http, $route, $scope) {
            $scope.$on(“$routeChangeStart”, , function(event, next, current) {
                if (confirm("Are you sure you want to Navigate away from this page ")) {}
            })
```

Now we want the confirmation if the user clicks ok it will return true and route to the other URL and if users click cancel it will return false and it will stay on the current route only. So when user clicks on cancel it should return false so we are using NOT operator,

```
if (!confirm("Are you sure you want to Navigate away from this page ")) {

}
```

To cancel route Change we are going to use event object -- we use prevent default function as,

```
.controller("studentsController", function($http, $route, $scope) {
            $scope.$on(“$routeChangeStart”, function(event, next, current) {
                if (!confirm("Are you sure you want to Navigate away from this page ")) {
                    event.preventDefault();
                }
            })
```

EXAMPLE:

```
<!DOCTYPE html>
<html>
<head>
        <title>Angular JS Route Change</title>
        <script src = "angular.min.js"></script>
        <script src = "angular-route.min.js">
        </script>
</head>

<body style = "text-align:center;">


        <div>
                <p><a href = "#!/viewLink1">Link 1</a></p>
                <p><a href = "#!/viewLink2">Link 2</a></p>
                <div ng-app = "mainApp" ng-controller = "GFGController">
                <div ng-view></div>
        </div>

        <script>
                var mainApp = angular.module("mainApp", ['ngRoute']);
                mainApp.config(['$routeProvider', function($routeProvider) {
                        $routeProvider

                        .when('/viewLink1', {
                                template: "<p> This is Link 1 </p>"
                        })

                        .when('/viewLink2', {
                                template: "<p> This is Link 2 </p>"
                        })

                        .otherwise({
                                redirectTo: '/'
                        });
                }]);

                mainApp.controller( 'GFGController', function($scope, $location, $rootScope) {
```

```
          $rootScope.$on('$locationChangeStart',function(event,next,current)
                              {
                                        if(!confirm("Are you sure want to navigate away from this
page  "+next))
                                                  event.preventDefault();
                              });
                    $rootScope.$on('$routeChangeSuccess', function () {
                    console.log("route changed");

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

---

ng-include in AngularJS

AngularJS has a built-in directive to include the functionality from other AngularJS files by using the ng-include directive.The primary purpose of the **“ng-include directive”** is used to fetch, compile and include an external HTML file in the main AngularJS application.These are added as child nodes in the main application. The ng-include attribute’s value can also be an expression, that returns a filename. All HTML element supports this.

Syntax:

```
<element ng-include="filename" onload="expression" autoscroll="expression" >

Content...</element>
```

Note: Here the onload and autoscroll parameter are optional , onload define an expression to evaluate when the included file is loaded and autoscroll define whether or not the included section should be able to scroll into a specific view.

Example:1

Child.html

```
<html>
<body>
Demonstration of ng-include directive
</body>
</html>
```

Index.html

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

```
<head>
    <title>Expression</title>
</head>
<body ng-app="">
    <script src="angular.min.js"></script>
    <h1> Angular </h1>
         <div>
                  <div ng-include="'child.html'">
         </div>
</body>
</html>
```

Example:2

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Expression</title>
</head>
<body>
    <script src="angular.min.js"></script>
          <script src="i1.js"></script>
    <h1> Angular Include example</h1>
         <div ng-app="myapp" ng-controller="mycontroller">
                  <select ng-model="E">
                           <option value="empTable.html">Table</option>
                           <option value="emp_lst.html">List</option>
                  </select>
                  <div ng-include="E"></div>
         </div>
</body>
</html>
```

emp_lst.html

```
<html>
<head>
</head>
<body>
    <script src="angular.min.js"></script>
          <script src="i1.js"></script>
    <h1> Employee List </h1>
         <div ng-app="myapp" ng-controller="mycontroller">
                  <ul ng-repeat="emp in employee">
                           <li>{{emp.name}}
                                    <ul>
                                             <li>{{emp.gender}}
```

```
                                        <li>{{emp.salary}}
                                </ul>
                <ul>
        </div>
</body>
</html>
```

empTable.html

```
<html>
<head>

</head>
<body>
    <script src="angular.min.js"></script>
         <script src="i1.js"></script>
    <h1> Dsiplay in Table</h1>
        <div ng-app="myapp" ng-controller="mycontroller">

                        <table border=1>
                                <tr ng-repeat="emp in employee">
                                <td>{{emp.name}}</td>
                                <td>{{emp.gender}}</td>
                                <td>{{emp.salary}}</td>
                                </tr>
                        </table>

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

ng-src Directive

The **ng-src Directive** in AngularJS is used to specify the src attribute of an <img> element. It ensures that the wrong image is not produced until AngularJS has been evaluated. It is supported by <img> element.

Syntax:

```
<img ng-src="url"> </img>
```

Example:

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

```
<head>
   <title>Welcome</title>
   <script src="angular.min.js"></script>
 </head>
<body ng-app="Example">
   <div  ng-controller="ExampleController">
      <img ng-src="{{pic}}" height=300 width=300/>
   </div>
   <script>
      var app = angular.module("Example", []);
      app.controller('ExampleController', function ($scope) {
         $scope.pic = "tom.jpg";
      });
   </script>
</body>
```

Example:(display image in list format)

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

</head>
<body ng-app="myApp" ng-controller="myController">
   <div>
      <h3 >Images in list using ng-src</h3>
      <ul ng-repeat="pic in pics">
         <li><img ng-src="{{pic.url}}" style="height:300px;width:300px" /><span style="font-
size:14">{{pic.Name}}</span></li>
      </ul>
   </div>
   <script>
      var app = angular.module("myApp", []);
      app.controller('myController', function ($scope) {
         $scope.pics = [{ url: "mickey.jpg", Name: "Mickey" }, { url: "mickey2.jpg", Name:
"Mickey" }];
      });
   </script>
</body>
</html>
```

Example:(display image in table format)

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

```
<head>
   <title>Welcome</title>
   <script src="angular.min.js"></script
</head>
<body ng-app="myApp" ng-controller="myController">
   <div>
      <h3>Images in table using ng-src</h3>
      <table border=1>
            <tr>
               <th>Name</th>
               <th>Capital</th>
               <th>Currency</th>
               <th>Flag</th>
            </tr>
            <tr ng-repeat="country in countries">
               <td>{{country.Name}}</td>
               <td>{{country.Capital}}</td>
               <td>{{country.Currency}}</td>
               <td><img ng-src="{{country.Flag}}" width="50" height="30" /></td>
            </tr>
      </table>
   </div>
   <script>
      var app = angular.module("myApp", []);
      app.controller('myController', ['$scope', function ($scope) {
         $scope.countries =
         [{ Name: "India", Capital: "New Delhi", Currency: "Indian rupee", Flag: "india.png" },
        { Name: "Afghanistan", Capital: "Kabul", Currency: "Afghani", Flag: "afghanistan.png" },
        { Name: "Nepal", Capital: "Kathmandu", Currency: "Nepalese rupee", Flag: "nepal.jpg" }]
      }]);
   </script>
</body>
</html>
```

AngularJS Service

AngularJS services are JavaScript functions for specific tasks, which can be reused throughout the application.
AngularJS includes services for different purposes. For example, $http service can be used to send an AJAX request to the remote server. AngularJS also allows you to create custom service for your application
AngularJS provides many inbuilt services. For example, $http, $route, $window, $location, etc. Each service is responsible for a specific task such as the $http is used to make ajax call to get the server data, the $route is used to define the routing information, and so on. The inbuilt services are always prefixed with $ symbol.

There are two ways to create a service –

- Factory
- Service

AngularJS $http Service

In angularjs, $http is the most common service which is used in angularjs applications. By using $http service we can communicate with remote **HTTP** servers over the browser with the help of XMLHttpRequest object.

In **$http** service, we have a different methods available those are $http.get, $http.post, $http.put, $http.delete, etc. We will learn all these methods in-detail in next chapters.

Here we will see general usage of **$http** service in angularjs applications.

Syntax of using AngularJS $http Service

Generally the syntax of using **$http** service in angularjs applications will be like as shown following

```
var app = angular.module('serviceApp', []);
app.controller('serviceCtrl', function ($scope, $http) {
// Simple GET request example:
$http({
method: 'GET',
url: '/sampleUrl'
}).then(function success(response) {
// this function will be called when the request is success
}, function error(response) {
// this function will be called when the request returned error status
});
});
```

$log Service

AngularJs includes logging service $log, which logs the messages to the browser's console.

The $log service includes different methods to log the error, information, warning or debug information. It can be useful in debugging and auditing.

$interval Service

AngularJS includes $interval service which performs the same task as setInterval() method in JavaScript. The $interval is a wrapper for setInterval() method, so that it will be easy to override, remove or mocked for testing.

The $interval service executes the specified function on every specified milliseconds duration.

Signature: $interval(function, delay, [count]);

$window Service

AngularJs includes $window service which refers to the browser window object.

In the JavaScript, window is a global object which includes many built-in methods like alert(), prompt() etc.

The $window service is a wrapper around window object, so that it will be easy to override, remove or mocked for testing. It is recommended to use $window service in AngularJS instead of global window object directly.

Using Factory Method

In this method, we first define a factory and then assign method to it.

```
var mainApp = angular.module("mainApp", []);
mainApp.factory('MathService', function() {
  var factory = {};

  factory.multiply = function(a, b) {
    return a * b
  }
  return factory;
});
```

Using Service Method

In this method, we define a service and then assign method to it. We also inject an already available service to it.

```
mainApp.service('CalcService', function(MathService) {
  this.square = function(a) {
    return MathService.multiply(a,a);
  }
});
```

Example1:

Cal.html

```
<html>
    <head>
       <script src="angular.min.js"></script>
                   <script src="calc_ser.js"></script>
    </head>
    <body ng-app="MyApp">
         <div ng-controller="myctrl">
            First Value : <input type="number" ng-model="num1"/><br>
            Second Value : <input type="number" ng-model="num2"/><br>
            Result : {{ result }}<br>
            <button ng-click="sum()" > SUM </button>
            <button ng-click="mult()" > Multiplication </button>
         </div>
    </body>
</html>
```

calc_ser.js

```
var app=angular.module("MyApp",[]);
app.service("calcSer",function(){
   this.add = function (num1,num2) {
      return parseInt(num1)+parseInt(num2);
   }
         this.mul = function (num1,num2) {
      return parseInt(num1)*parseInt(num2);
   }
});
app.controller('myctrl',function($scope,calcSer){
         $scope.num1=10;
         $scope.num2=10;
         $scope.result=0;
         $scope.sum=function(){
         $scope.result=calcSer.add($scope.num1,$scope.num2)
         }
         $scope.mult=function(){
         $scope.result=calcSer.mul($scope.num1,$scope.num2)
         }
         }
);
```

Example2:

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
<p>Use a filter when displaying the array [255, 251, 200]:</p>

<ul>
  <li ng-repeat="x in counts">{{x | myFormat}}</li>
</ul>

<p>This filter uses a service that converts numbers into hexadecimal values.</p>
</div>

<script>
var app = angular.module('myApp', []);
app.service('hexafy', function() {
    this.myFunc = function (x) {
        return x.toString(16);
    }
});
app.filter('myFormat', function(hexafy) {
    return function(x) {
        return hexafy.myFunc(x);
    };
});
app.controller('myCtrl', function($scope) {
    $scope.counts = [255, 251, 200];
});
</script>

</body>
</html>
```