

## What is AngularJS ?

Angular JS is an open source **JavaScript framework** that is used to build web applications.

It can be freely used, changed and shared by anyone.

**Misko Hevery and Adam** Abrons developed AngularJS initially in 2009. Later on, supported by Google 2010.

## Features of AngularJS

The various features of angularJS are as follows:

1. Data Binding
2. Architecture
3. Directives
4. Not Browser Specific
5. Codeless
6. Speed and Performance
7. Dependency Injection

### i. Data Binding

In an angular application, we don't need to write separate code to perform the **data binding** functionality. By adding some snippets of code we can easily bind data from HTML control to application data. Any extra code is not written to bind with HTML control.

### ii. Architecture

An angular application is built using MVC architecture that stands for Model View and Controller. It separates the application into three parts model part, view part and controller part as per the components of MVC architecture. Using this, architecture presentation part, logic part and application data part is split into the separate section which allows managing of application in a very fluent manner.

### iii. Directives

View of angularJS, mix data from a model into HyperText Markup Language templates. Angular JS directives are used for the same purpose. It tells how to combine data into the HTML template. With the use of directive, we can provide extra functionality to our angular application. Angular provides a way to create custom directives too.

#### iv. Not Browser Specific

Angular applications are not browser specific means there is no browser constraint on an angular application.

- It claims to support browsers such as Chrome, Firefox, Safari, IE8, Android.

#### v. Code Less

- A programmer can write less and perform more functionalities with the same code.
- Filters in angular provide the functionality of write less do more. The various filters in angular are uppercase, lowercase, currency etc. You can use the filter and easily format the data.

#### vi. Speed and Performance

Speed and performance of angular are faster because of code reusability

#### vii. Dependency Injection

- This built-in injection helps in developing the application easily as well as it is easy to understand.
- It helps an application easier to test.
- Whenever angular JS detect that you need a service then it immediately provides an instance for that.
- It allows you to ask for your dependencies rather than having to go look for them or making it by yourself.

#### viii. Deep Linking

- It allows to bookmarks the web page. The page gets saved by its URL without getting its state changed.

- Whenever the request is made by a user for that page it will get displayed in the same state as before.

## **Advantage of AngularJS**

There are a lot of JavaScript frameworks for building web applications. So, it is a genuine question, why to use Angular JS.

### **Following are the advantages of AngularJS over other JavaScript frameworks:**

- **Dependency Injection:** Dependency Injection specifies a design pattern in which components are given their dependencies instead of hard coding them within the component.
- **Two way data binding:** Data binding is the synchronization of data between business logic and view of the application. It serves as a bridge between two components of angular that is model part and view part. Data Binding is automatic and provides a way to wire the two important part of an application that is the UI and application data..
- **Testing:** Angular JS is designed in a way that we can test right from the start. So, it is very easy to test any of its components through unit testing and end-to-end testing.
- **Model View Controller:** In Angular JS, it is very easy to develop application in a clean MVC way. You just have to split your application code into MVC components i.e. Model, View and the Controller.
- **Google supported**  
AngularJs framework is supported by a large community, Google.

The various advantages of Google supported sites are:

- Regular updates are done.
- For distributed or remote users, the anywhere/anytime access capability to the corporate intranet by google supported sites.
- It provides a capacity to work across the operating system.
- **Easy to extend and customize**

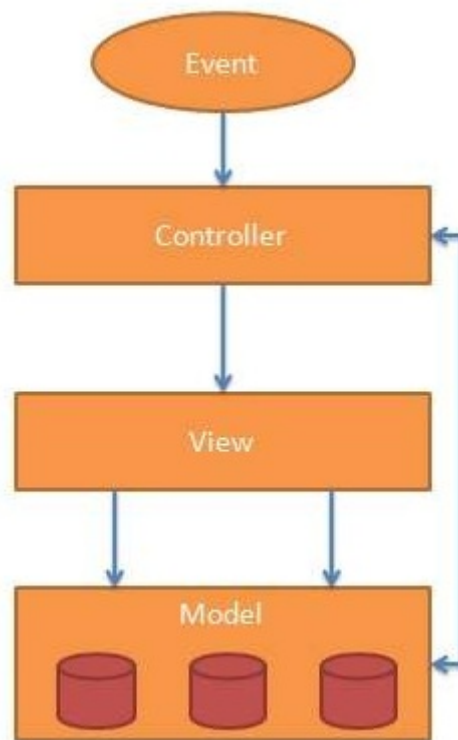
- Due to certain built-in attributes, it is easy to extend. These attributes make it possible to extend the functionality of HTML by attaching a specific behavior with it. One can create its own directives too in it therefore it is customized.
- Customized means adding or removing features or functionality, which is done to satisfy the specific needs. Also, customized software consist of user-friendly features rather than unnecessary elements as it is purely according to individual needs.

- **Single Page Application (SPA)**

Single page application means only a single HTML web page is loaded and further updation is done on that single page only. Since it is mostly used to create single page application and single page application works fast as well as it is user-friendly.

## AngularJS MVC Architecture

MVC stands for Model View Controller. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.



The MVC pattern is made up of the following three parts:

1. **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.

2. **View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
3. **Controller:** It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

### Downloading and Installing AngularJS

Downloading and installing AngularJS is easy, takes very little time, and doesn't require your credit card. It is

completely free of charge (under the MIT license).

To download AngularJS, head on over to <http://angularjs.org> and follow these steps:

1. Create a folder on your computer called BeginningAngularJS. Inside this folder, create a subfolder called js to contain your JavaScript files.
2. On the AngularJS home page, click the Download button. You will see a dialog box like the one shown in Figure .
3. You want the 1.2.x-minified version, so make sure that you choose *1.2.X (legacy)* for the branch option and *Minified* for the build option.
4. Click the Download button to start the download process.
5. Once the download has completed, move the downloaded file, *angular.min.js*, into the js folder that you created earlier (assuming you did not save it there directly).
6. That's it! You just downloaded and installed AngularJS.

Throughout this book, I will assume that you have followed the preceding steps when I refer to file system locations and folder names. If you are comfortable with the Content Delivery Network (CDN), and prefer to use it, feel free to do so. Likewise, if your preference is to use the non-minified version of the AngularJS library, go right ahead. This won't affect the output of any of the code listings (assuming that you have things set up correctly otherwise).

## Download AngularJS

**Branch**   ?

**Build**    ?

**CDN**  ?

**Bower**  ?

**Extras** [Browse additional modules](#)

[Previous Versions](#)

 **Download**

### First Angular js Program

```
first - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html ng-app="">
<head>
<title>Listing 2-1</title>
<script src="angular.min.js"></script>
</head>
<body>
<p>Hello {{'wor' + 'ld'}}</p>
</body>
</html>
```

OUTPUT:

← → ↻ ⓘ File | D:/

Hello World

## AngularJS Expressions

AngularJS expressions can be written inside double braces: `{{ expression }}`.

AngularJS expressions can also be written inside a directive: `ng-bind="expression"`.

AngularJS will resolve the expression, and return the result exactly where the expression is written.

**AngularJS expressions** are much like **JavaScript expressions**: They can contain literals, operators, and variables.

Example `{{ 5 + 5 }}` or `{{ firstName + " " + lastName }}`

Example:

```
<!DOCTYPE html>
<html >
<head>
<title>Listing 2-2</title>
<script src="angular.min.js"></script>
</head>
<body>

<p ng-app| >Hello {{'wor' + 'ld'}}</p>
<p >Hello {{ 2 + 4 }}</p>
</body>
</html>
```

Two interesting things happen:

1. The first interesting thing is that the expression binding in the first paragraph worked just as it did before. Even though we relocated the *ngApp* directive, the expression binding is still nested within its boundaries and, therefore, still under AngularJS control.
2. The second interesting thing is that the second paragraph uses an expression too. However, this expression binding simply renders as is; it is not evaluated at all. AngularJS simply isn't interested in it, because it is not contained within the boundaries of an *ngApp* directive. In fact, AngularJS has no knowledge of this particular paragraph element or anything contained within it.



```

<!DOCTYPE html>
<html ng-app>
<head>
<title>Listing 2-2</title>
<script src="angular.min.js"></script>
</head>
<body>

<p>Hello {{'wor' + 'ld'}}</p>
<p>Hello {{ 2 + 4 }}</p>
</body>
</html>

```



Hello World

Hello 6

## AngularJS Numbers

Expressions can be used to work with numbers as well.

```

an_number - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
  <title>Expression</title>
</head>
<body>
  <script src="angular.min.js"></script>
  <h1> Angular Expression Numbers</h1>
  <div ng-app="" ng-init="a=5;b=10">
    Two strings are |{{a}} and {{b}}</div>
</body>
</html>

```

OUTPUT:

← → ↻ ⓘ File | D:/Angular/program/an\_number.html

# Angular Expression Numbers

Two strings are 5 and 10

## AngularJS Strings

AngularJS strings are like JavaScript strings:

```
an_string - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<head>
  <title>Expression</title>
</head>
<body>
  <script src="angular.min.js"></script>
  <h1> Angular Expression String</h1>
  <div ng-app="" ng-init="first_name='ramesh';l_name='xyz'">
    Two strings are{{first_name}} and <br> {{l_name}}</div>
</body>
</html>
```

OUTPUT:

# Angular Expression String

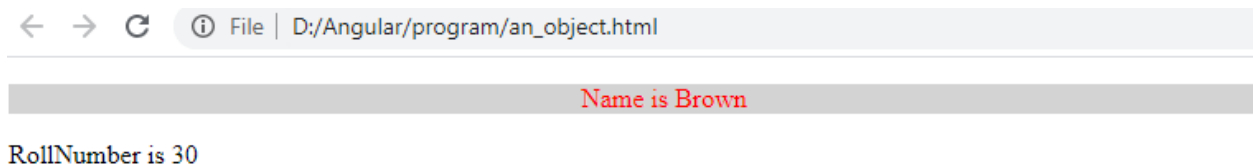
Two strings are ramesh and  
xyz

## AngularJS Object:

AngularJS Objects are like JavaScript Object So you can easily access via dot(.) operator

```
an_object - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html ng-app>
<head>
<title>Expression</title>
<script src="angular.min.js"></script>
</head>
<body>
<div ng-init="student={Name:'Brown',RollNo:30}">
<p style="color:red;text-align:center;background-
color:LightGrey;">Name is {{student.Name}}</p>
<p> RollNumber is {{student.RollNo}}</p>
</body>
</html>
```

OUTPUT:

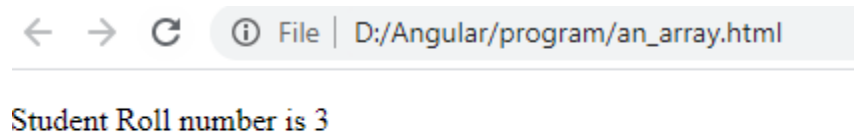


## AngularJS Arrays:

AngularJS arrays are like JavaScript array so you can easily access via index.

```
an_array - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html ng-app>
<head>
<title>Expression</title>
<script src="angular.min.js"></script>
</head>
<body>
<div ng-init="student=[501,502,3]">
<p> Student Roll number is {{student[2]}}</p>
</body>
</html>
```

OUTPUT:



## AngularJS Expression capabilities and Limitations

### AngularJS Expressions vs. JavaScript Expressions

Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.

Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.

AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.

AngularJS expressions support filters, while JavaScript expressions do not.

## Angular JS Expression limitations

1. There is currently no availability to use conditionals, loops, or exceptions in an Angular expression
2. You cannot declare functions in an Angular expression, even inside ng-init directive.
3. One cannot create regular expressions in an Angular expression. A regular expression is a combination of symbols and characters, which are used to find for strings such as `.*\txt$`. Such expressions cannot be used within Angular JS expressions.
4. Also, one cannot use `use`, or `void` in an Angular expression.

## What Is a Directive?

AngularJS uses directives to augment HTML with extra functionality. Essentially, directives are a convenient way to declaratively call JavaScript functions

### ✓ ngShow Directive

The ngShow directive will show, or hide, the element on which it is declared, based on the expression provided to it.

```
<!DOCTYPE html>
<html ng-app>
<head>
<title>ng show</title>
<script src="angular.min.js"></script>
</head>
<body>
<p ng-show="true">Paragraph 1, can you see me?</p>
<p ng-show="false">Paragraph 2, can you see me?</p>
<p ng-show="1 == 1">Paragraph 3, can you see me?</p>
<p ng-show="1 == 2">Paragraph 4, can you see me?</p>
</body>
</html>
```

OUTPUT:

---

Paragraph 1, can you see me?

Paragraph 3, can you see me?

### ✓ Ng-Init Directive

The **ng-init directive** is used to initialize application data or value or you can say put the value to the variable.

### ✓ ng-click Directive

The AngularJS ng-click directive facilitates you to specify custom behavior when an element is clicked. So, it is responsible for the result what you get after clicking. It is supported by all HTML elements.

```
<element ng-click="expression"></element>
```

```

<!doctype html>
<html ng-app>
<head>
<title>Listing 2-5</title>
<script src="angular.min.js"></script>
</head>
<body>
<button ng-click="count = count + 1" ng-init="count = 0">
Increment
</button>
count: {{count}}|
</body>
</html>

```

OUTPUT:

Increment count: 4

```

<!DOCTYPE html>
<html >
<head>
<script src="angular.min.js"></script>
</head>
<body >
<div ng-app ng-init="greet='Hello world!'; amount= 100;
myArr =
[100, 200]; person = { firstName:'Steve', lastName : 'Jobs'}">
    {{amount}}      <br />
    {{myArr[1]}}    <br />
    {{person.firstName}}
</div>
</body>
</html>

```

OUTPUT:

100  
200  
Steve

✓ **Ng-Model Directive**

The ng-model directive is used binds the value of HTML controls (like input) to application data.

This directive binds the value from the web page, mostly from the input field to the application variable. Basically, this directive allows sending data from input to AngularJS application which can be used somewhere else

#### Syntax

```
<div ng-app = "">
  <p>Type Name: <input type = "text" ng-model = "Name"></p>
</div>
```

## ng-bind Directive

The AngularJS ng-bind directive replaces the content of an HTML element with the value of a given variable, or expression. If you change the value of the given variable or expression, AngularJS changes the content of the specified HTML element as well as.

This directive binds the value from the AngularJS application to the web page. i.e. It allows forwarding data from the application to HTML tags.

It is an alternative to the interpolation directive.

#### **Syntax:**

```
<element ng-bind="expression"> </element>
```

## Using ngmodel and ngbind

```
<!DOCTYPE html>
<html ng-app>
<head>
<title>Listing 2-3</title>
<script src="angular.min.js"></script>
</head>
<body>
<label>City: </label><input ng-model="city1" type="text"/></label>
<p>You entered: {{city1}}</p>
</body>
</html>
```

**OUTPUT:**



---

City:

You entered: mumbai

```
<!DOCTYPE html>
<html ng-app>
<head>
<title>Listing 2-3</title>
<script src="angular.min.js"></script>
</head>
<body>
<label>City: </label><input ng-model="city1" type="text"/></label><br>
<p ng-bind="city1"></p>
</body>
</html>
```

**OUTPUT:**

---

City:

Mumbai

## The ng-repeat Directive

The ng-repeat directive is used to iterate the html elements for each item in a collection. In following example, we've iterated over array of cities.

### Syntax

```
<div ng-app="" ng-init="cities=['Delhi','Noida','Gurgaon']">
  <ul>
    <li ng-repeat="city in cities">
      {{ city }}
    </li>
  </ul>
</div>
```

```

<!DOCTYPE html>
<html >
<head>
  <script src="angular.min.js"></script>
</head>
<body >
  <div ng-app="" ng-init="names=[
{name:'Jani',country:'Norway'},
{name:'Hege',country:'Sweden'},
{name:'Kai',country:'Denmark'}]">

  <ul>
    <li ng-repeat="x in names">
      {{ x.name + ', ' + x.country }}
    </li>
  </ul>
</div>
</body>
</html>

```

- Jani, Norway
- Hege, Sweden
- Kai, Denmark

## Scope in AngularJS

The \$scope in an AngularJS is a built-in object, which contains application data and methods. You can create properties to a \$scope object inside a controller function and assign a value or function to it.

The \$scope is glue between a controller and view (HTML). It transfers data from the controller to view and vice-versa.



**What is an AngularJS Module?**

A module defines the application functionality that is applied to the entire HTML page using the ng-app directive. It defines functionality, such as services, directives, and filters, in a way that makes it easy to reuse it in different applications.

A module is a collection of controllers, directives, filters, services, and other configuration information. The main player in all this is angular.module, as it is the gateway into the Module API, the mechanism used to configure angular modules. It is used to register, create, and retrieve previously created AngularJS modules.

This probably all sounds rather abstract, so let's look at a practical example by walking through the process of setting up a default module for our application. The default module is the module that AngularJS will use as the entry point into your application. (It may even be the only module you use.) Don't worry if all this doesn't make a lot of sense at the moment, as we will look at a complete listing and talk more about what is happening when we build our custom filter.

Add the following code to a new JavaScript file, which you can name myAppModule.js.

```
// Create a new module
```

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

You just created a module. Wasn't that easy? The module method was used to create a module named myAppModule.

We also captured the returned object (a reference to the module just created) in a variable, also named myAppModule.

You will notice that we also passed an empty array to the module method. This can be used to pass a list of dependencies; that is, other modules that this module depends upon. We don't have any dependencies, so we simply pass an empty array instead.

## What is Controller in AngularJs?

A Controllers in AngularJs takes the data from the View, processes the data, and then sends that data across to the view which is displayed to the end user. The Controller will have your core business logic.

The controller will use the data model, carry out the required processing and then pass the output to the view which in turn is displayed to the end user.

Following is a simple definition of working of Angular JS Controller.

The controller's primary responsibility is to control the data which gets passed to the view. The scope and the view have two-way communication.

The following example demonstrates creating a **myController**, which attaches a **message** property containing the string 'Hello World' to the \$scope.

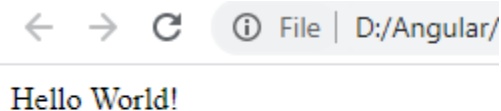
We create an AngularJS Module, **myNgApp**, for our application. Then we add the controller's constructor function to the module using the .controller() method. This keeps the controller's constructor function out of the global scope.

We attach our controller to the DOM using the ng-controller directive. The **message** property can now be data-bound to the template:

```
ng_control1 - Notepad
File Edit Format View Help
<html >
<head>
  <title>AngularJS Controller</title>
  <script src="angular.min.js"></script>
</head>
<body ng-app="myNgApp">
  <div ng-controller="myController">
    {{message}}
  </div>
  <script>
    var ngApp = angular.module('myNgApp', []);

    ngApp.controller('myController', function ($scope) {
      $scope.message = "Hello World!";
    });
  </script>
</body>
</html>
```

OUTPUT:



Note: The \$ sign is used as prefix in all the built-in objects in AngularJS, so that we can differentiate AngularJS built-in objects and other objects

## Creating separate file for controller

You can create separate .js file link it in html using script tag

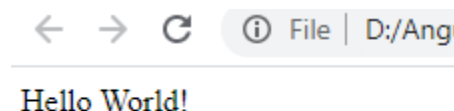
```
ng_control1_1 - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html >
<head>
  <title>AngularJS Controller</title>
  <script src="angular.min.js"></script>
  <script src="first.js"></script>
</head>
<body ng-app="myNgApp">
  <div ng-controller="myController">
    {{message}}
  </div>
</body>
</html>
```

#### First.js

```
File Edit Format View Help
var ngApp = angular.module('myNgApp', []);

    ngApp.controller('myController', function ($scope) {
        $scope.message = "Hello World!";
    });
```

#### OUTPUT:



```
ng_control2_1 - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html >
<head>
  <title>AngularJS Controller</title>
  <script src="angular.min.js"></script>
  <script src="second.js"></script>
</head>
<body ng-app="myNgApp1">
  <div ng-controller="myController1">
    <div>
      first Name :{{employee.firstName}}
    </div>
    <div>
      last Name :{{employee.lastName}}
    </div>
  </div>
</body>
```

```
second - Notepad
File Edit Format View Help
var ngApp = angular.module('myNgApp1', []);

    ngApp.controller('myController1', function ($scope) {
        $scope.employee = {
            firstName:"rohan",
            lastName:"xyz"
        };
    });
```

← → ↺ ⓘ File | D:/Angular/prograr

first Name :rohan  
last Name :xyz

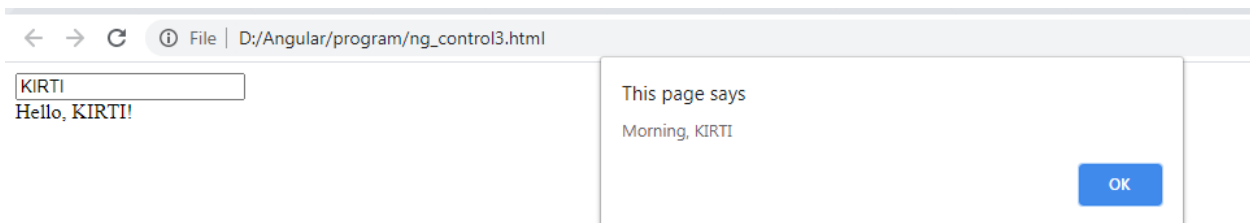
## Using function

The properties of the view can call "functions" on the scope. Moreover events on the view can call "methods" on the scope

```
ng_control3 - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html ng-app="helloApp">

<head>
  <title>Hello AngularJS</title>
  <script src="angular.min.js"></script>
</head>

<body ng-controller="HelloController">
  <input type="text" ng-model="name" />
  <div ng-click="sayMorning(name)">
    Hello, {{name}}!
  </div>
<script>
var ngApp=angular.module("helloApp", [])
  ngApp.controller("HelloController", function ($scope) {
    $scope.sayMorning = function (ans) {
      alert("Morning, " + ans)
    }
  });
</script>
</body>
</html>
```





```
ng_con_sample2 - Notepad
File Edit Format View Help
<body>
<div ng-controller="DoubleController">
  Two times <input ng-model="num"> equals {{ double(num) }}
</div>
<script>
var myApp = angular.module('myApp', []);

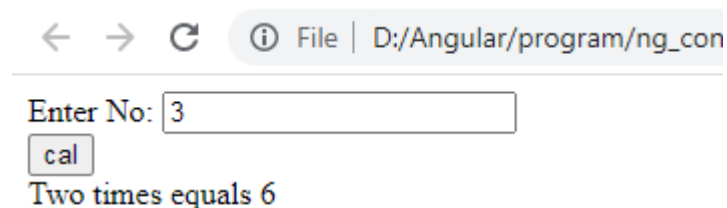
myApp.controller('DoubleController', function($scope) {
  $scope.value=1;
  $scope.double = function(value)
    { return value * 2; };
});
</script>
</body>
</html>
```

← → ↻ ⓘ File | D:/Angular/program/ng\_con\_sample2

Two times  equals 6

Using button click

```
ng_control4 - Notepad
File Edit Format View Help
<html ng-app="myApp">
<head>
    <script src="angular.min.js"></script>
</head>
<body>
<div ng-controller="DoubleController">
    Enter No: <input ng-model="num"><br>
    <button ng-click="double(num)">cal</button> <br>
    Two times equals {{value}}
</div>
<script>
var myApp = angular.module('myApp',[]);
myApp.controller('DoubleController',function($scope) {
    $scope.value=1;
    $scope.double = function(num)
    { $scope.value= num* 2; };
});
</script>
</body>
</html>
```



## 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

```
filter_ucose - Notepad
File Edit Format View Help
<!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

← → ↻ ⓘ File | D:/Angular/prog

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:

```
filter_orderby - Notepad
File Edit Format View Help
<!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

```
< > ↺ ⓘ File | D:/Angular/program/filter_orderby.
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

```
filter_currency - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script><body>

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

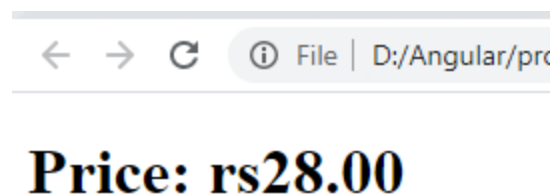
<h1>Price: {{ price | currency:"rs"}}</h1>

</div>

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

</body>
</html>
```

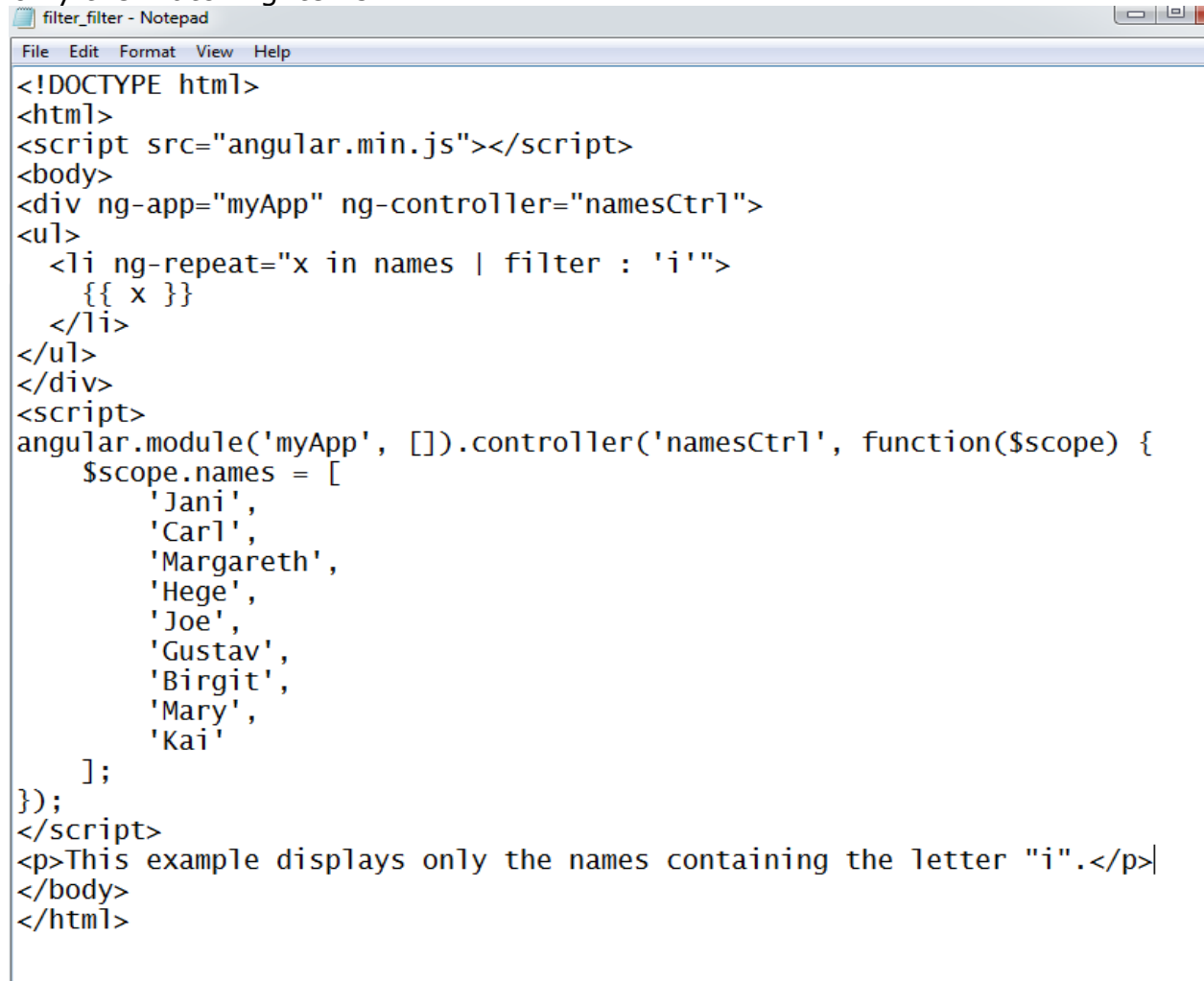
OUTPUT:



The filter Filter

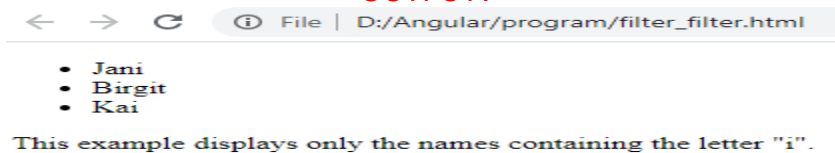
The **filter** filter selects a subset of an array.

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:





```
filter_test - Notepad
File Edit Format View Help
<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>
```

← → ↻ ⓘ File | D:/Angular/program/filter\_test.html

Type a letter in the input field:

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

The list will only consists of names matching the filter.

```
filter_number - Notepad
File Edit Format View Help
<!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>
```

← → ↻ ⓘ File | D:/Angular/program/filte

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)

```
filter_date - Notepad
File Edit Format View Help

<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 predefined 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>
```

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 predefined formats when displaying a date.

Date = Friday, August 14, 2020

This is another format.

Date = today is August 14, 2020

## 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

```
filter_ucose - Notepad
File Edit Format View Help
<!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

← → ↻ ⓘ File | D:/Angular/prog

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:



```
filter_orderby - Notepad
File Edit Format View Help
<!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

```
< > ↺ ⓘ File | D:/Angular/program/filter_orderby.
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

```
filter_currency - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script><body>

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

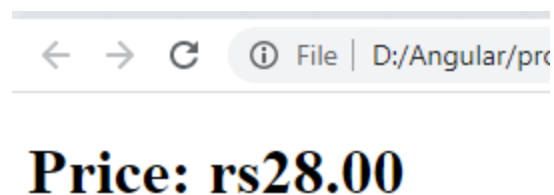
<h1>Price: {{ price | currency:"rs"}}</h1>

</div>

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

</body>
</html>
```

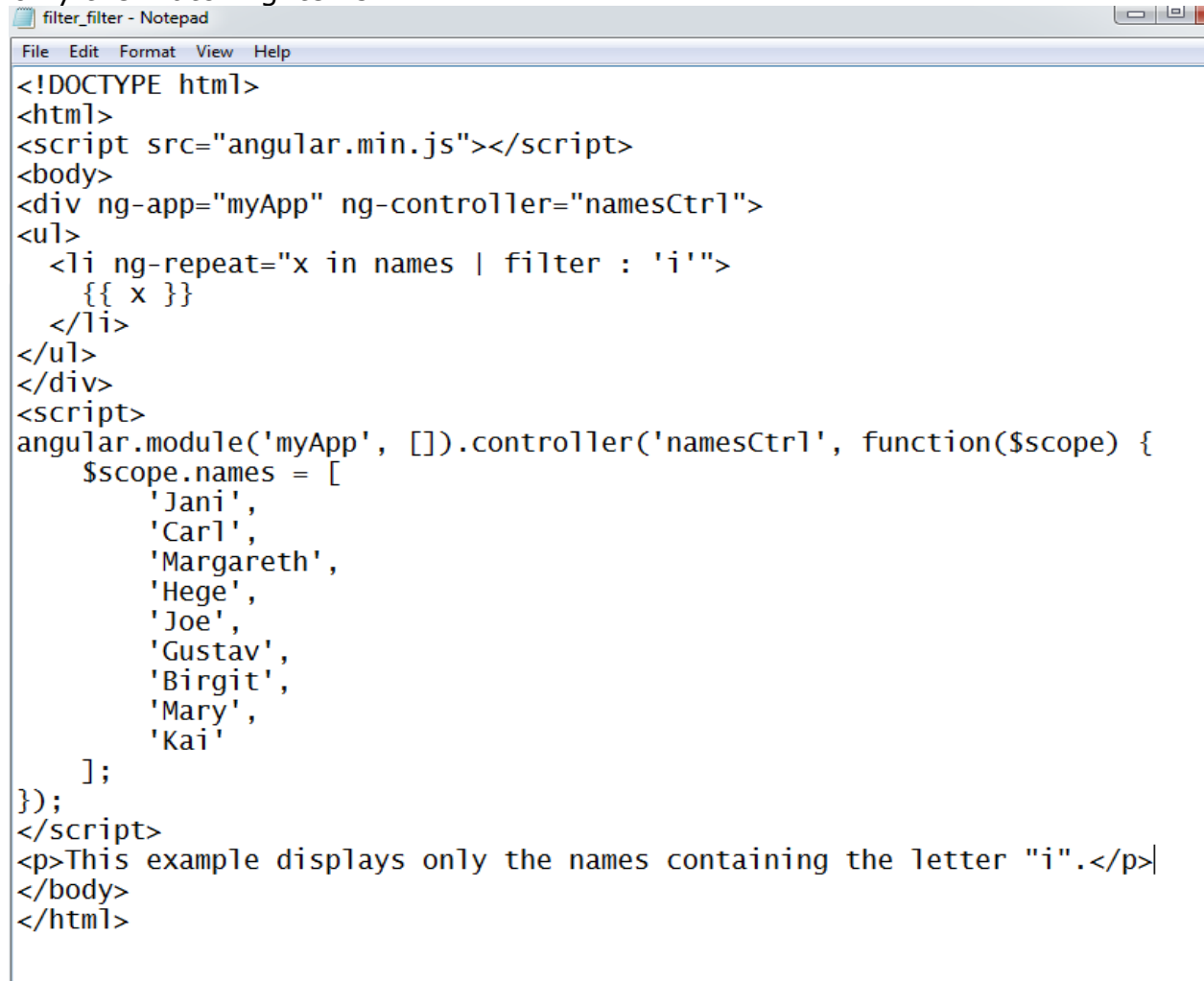
OUTPUT:



The filter Filter

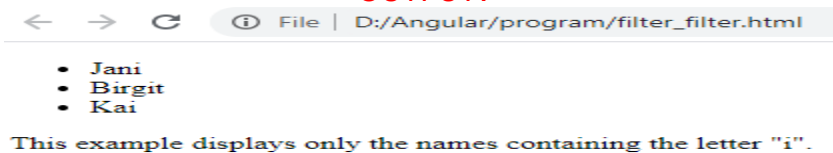
The **filter** filter selects a subset of an array.

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:



```
filter_test - Notepad
File Edit Format View Help

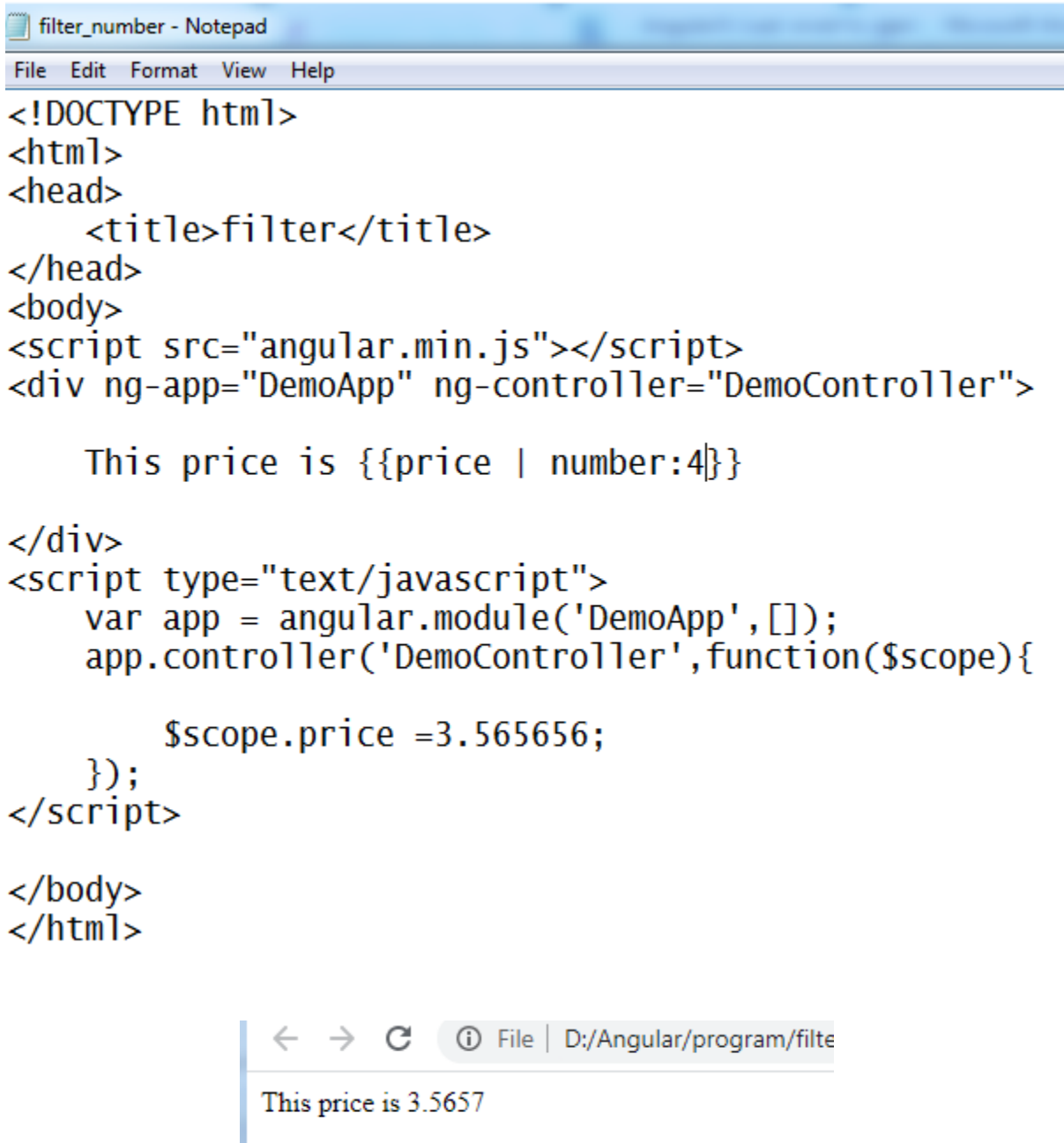
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-controller="namesCtrl">
<p>Type a letter in the input field:</p>
<p><input type="text" ng-model="test"></p>
<ul>
  <li ng-repeat="x in names | filter:test">
    {{ x }}
  </li>
</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>
```

← → ↻ ⓘ File | D:/Angular/program/filter\_test.html

Type a letter in the input field:

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

The list will only consists of names matching the filter.



The image shows a Notepad window titled 'filter\_number - Notepad' with the following HTML code:

```
<!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>
```

Below the Notepad window, a web browser window is shown with the address bar displaying 'D:/Angular/program/filte'. The browser content area displays the text '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)

```

filter_date - Notepad
File Edit Format View Help

<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 predefined 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>

```

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 predefined 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 x table.js x
1 <html>
2 <head>
3   <title>
4     filter
5   </title>
6   <script src="angular.min.js"></script>
7   <script src="table.js"></script>
8 </head>
9 <body ng-app="myModule">
10   <div ng-controller="mycontroller">
11     Rows to display:<input type="number" step="1" min="0" max="4" ng-model="rowlimit" />
12     <br>
13     <br>
14     <table border=1>
15       <tr>
16         <th>Name</th>
17         <th>DOB</th>
18         <th>Gender</th>
19         <th>Salary</th>
20       </tr>
21       <tr ng-repeat="emp in employee | limitTo:rowlimit">
22         <th>{{emp.name|uppercase}}</th>
23         <th>{{emp.DOB|date:"dd/MM/yyyy"}}</th>
24         <th>{{emp.gender|lowercase}}</th>
25         <th>{{emp.salary|number:2}}</th>
26       </tr>
27     </table>
28   </div>
29 </body>
30 </html>
31
32
filter_date_example.html x table.js x
1 var app=angular.module("myModule",[]);
2 app.controller("mycontroller",function($scope){
3   var employee=[
4     {name:"ben",DOB:new Date("November 23,1980"),gender:"Male",salary:55000.788},
5     {name:"Sara",DOB:new Date("November 23,1984"),gender:"Feale",salary:50000},
6     {name:"mark",DOB:new Date("November 23,1980"),gender:"Male",salary:57000},
7     {name:"pam",DOB:new Date("November 23,1980"),gender:"Female",salary:53000}
8   ];
9   $scope.employee=employee;
10   $scope.rowlimit=3;
11 });
```

Rows to display:

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 }}
```



```
1 <!DOCTYPE html>
2 <html>
3 <script src="angular.min.js"> </script>
4 <body>
5 <div ng-app="myApp" ng-controller="orderCtrl">
6 <ul>
7 <li ng-repeat="x in customers | orderBy : 'name'">
8 {{x.name + ", " + x.city}}
9 </li>
10 </ul>
11 </div>
12 <script>
13 var app = angular.module('myApp', []);
14 app.controller('orderCtrl', function($scope) {
15     $scope.customers = [
16         {"name" : "A", "city" : "ajmer"},
17         {"name" : "lakshay ", "city" : "vizag"},
18         {"name" : "karan", "city" : "London"},
19         {"name" : "bhaskar", "city" : "peshawar"},
20     ];
21 });
22 </script>
23 </body>
24 </html>
```

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

OrderBy(ascending )

```
filter_date_example.html x table.js x orderby_ex.html x orderby_ex1.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>AngularJS Filters : orderBy </title>
5 <script src= "angular.min.js"></script>
6 </head>
7 <body ng-app="orderByDemo">
8 <script>
9 angular.module('orderByDemo', [])
10 .controller('orderByController', ['$scope', function($scope) {
11     $scope.countries =
12         [{name:'SPAIN', states:'78', gdp:5},
13          {name:'FRANCE', states:'46', gdp:4},
14          {name:'PORTUGAL', states:'53', gdp:3},
15          {name:'CHILE', states:'06', gdp:2},
16          {name:'RUSSIA', states:'21', gdp:1}];
17     });
18 </script>
19 <div ng-controller="orderByController">
20
21 <table border=1>
22 <tr>
23 <th>Name</th>
24 <th>No of States</th>
25 <th>GDP(descending)</th>
26 </tr>
27 <!-- orderBy Descending (-) -->
28 <tr ng-repeat="country in countries | orderBy:'-gdp'">
29 <td>{{country.name}}</td>
30 <td>{{country.states}}</td>
31 <td>{{country.gdp}}</td>
32 </tr>
33 </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>
```

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="#/!">
</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.

```
1. .controller("studentsController", function($http, $route, $scope) {  
2.     $scope.$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,

```
1. .controller("studentsController", function($http, $route, $scope) {  
2.     $scope.$on("$routeChangeStart", , function(event, next, current) {  
3.         if (confirm("Are you sure you want to Navigate away from this page ")) {}  
4.     })
```

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,

```
1. if (!confirm("Are you sure you want to Navigate away from this page ")) {  
2.  
3. }
```

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

```
1. .controller("studentsController", function($http, $route, $scope) {  
2.     $scope.$on("$routeChangeStart", function(event, next, current) {  
3.         if (!confirm("Are you sure you want to Navigate away from this page ")) {  
4.             event.preventDefault();  
5.         }  
6.     })
```

### 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="il.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="il.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}}
    </li>
</ul>
</div>
</body>
</html>

```

### empTable.html

```

<html>
<head>

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

### Example:

```

<!DOCTYPE html>
<html>

```



```

<head>
  <title>Welcome</title>
  <script src="angular.min.js"></script>
</head>
<body ng-app="Example">
  <div ng-controller="ExampleController">
    
  </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><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></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>

```

