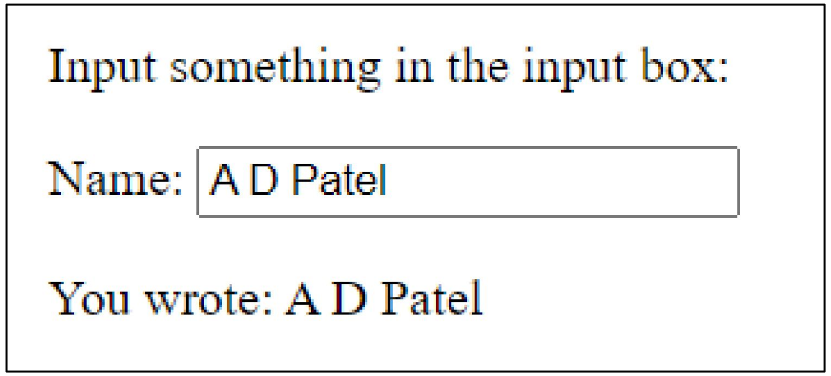
Angular JS

https://angularjs.org/

First Example: Angular JS

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
  <div ng-app=" " ng-init=" firstName='A D Patel'">
    <p>Input something in the input box:</p>
    <p>Name: <input type="text" ng-model="firstName"></p>
    <p>You wrote: {{ firstName }}</p>
  </div>
</body>
</html>
```



Input something in the input box:

Name: A D Patel

You wrote: A D Patel

Introduction

- **Angular** is a development platform, built on **TypeScript**.
- As a platform, Angular includes:
  - A **component-based framework** for building scalable web applications
  - A collection of well-integrated **libraries** that cover a wide variety of features, including routing, forms management, client-server communication, and more
  - A suite of developer tools to help you **develop**, **build**, **test**, and **update** your code
- With Angular, you're taking advantage of a platform that can **scale** from single-developer projects to enterprise-level applications.
- Angular is designed to make updating as **straightforward** as possible.
- The Angular ecosystem consists of a diverse group of over **1.7 million** developers, library authors, and content creators.

Cont’d

- 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.
- Angular JS is maintained by Google.
- 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:** AngularJS creates a two way data-binding between the select element and the orderProp model. orderProp is then used as the input for the orderBy filter.
  - **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.

It’s all to avoid postback…

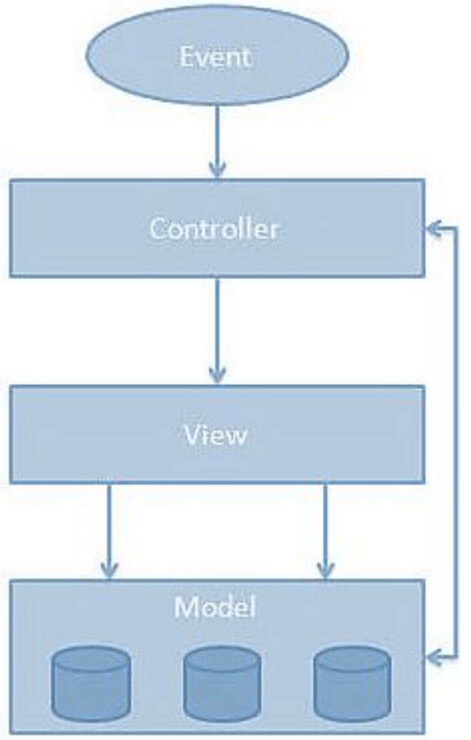
- Check the Angular JS website.

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:

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



```
<!DOCTYPE html>
<html lang="en">
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">  </script>
</head>

<body ng-app="myapp">

<div ng-controller="HelloController" >
<h2>Hello {{helloTo.title}} !</h2>
</div>
```

View Part

```
<script>
angular.module("myapp", [])
   .controller("HelloController", function($scope) {
      $scope.helloTo = {};
      $scope.helloTo.title = "World, AngularJS";
   } );
</script>
```

Controller Part

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

AngularJS Directives

- AngularJS *directives* are used to extend HTML. Directives are markers on HTML DOM element that tell AngularJS to attach a *specified behavior* to that HTML element. Directives in AngularJS are special attributes starting with *ng- prefix* where *ng* stands for Angular. AngularJS includes various *built-in* directives, you may also create your *own directive* in AngularJS. Some built-in directives are listed here.

Directive	Description
ng-app	This directive starts an AngularJS Application
ng-init	This Initializes AngularJS variables i.e. application data.
ng-repeat	This directive repeats HTML elements for each item in a collection.
ng-model	This directive binds the value of HTML controls (input, select, textarea) to application data.
ng-bind	This directive Replaces the value of HTML control with the value of specified AngularJS expression.
Other built-in directive-	
ng-controller	Attaches the controller of MVC to the view.
ng-show	Display HTML element based on the value of the specified expression.
ng-readonly	Makes HTML element read-only based on the value of the specified expression.
ng-disabled	Sets the disable attribute on the HTML element if specified expression evaluates to true.
ng-if	Removes or recreates HTML element based on an expression.
ng-click	Specifies custom behavior when an element is clicked.

ng-app directive

- The ng-app directive defines the root element of an AngularJS application and starts an AngularJS Application.
- The ng-app directive will automatically initialize the application when a web page is loaded.
- It is also used to load various AngularJS modules in AngularJS Application.

Example

```
<div ng-app = "">
...
</div>
```

In the above example code, we have defined a default AngularJS application using **ng-app** attribute of a **<div> element**.

ng-init directive

- ng-init directive initializes an AngularJS Application data. It is used to declare and assign values to the variables for an AngularJS application.

In following example, we initialize variable firstName and lastName and assign values to it.

```
<div ng-app = "" ng-init="firstName='A D';lastName=‘Patel' ">
...
</div>
```

ng-model directive

- The ng-model directive is used for two-way data binding in AngularJS. It is used to get value of input controls like textbox, label, etc and use these values in web pages.

In the following example, we define a model named my*name*.

```
<div ng-app = " ">
...
<input type = "text" ng-model = "myname">
<p>My name is(enter it): {{myname}} </p>

</div>
```

ng-repeat directive

- The ng-repeat directive repeats HTML elements for each item in a collection. Or simply say that it is used to loop through items in collection element and it will act as for loop. In the following example, we iterate over the name of the students:

```
<div ng-app="" ng-init="students=[‘Ramesh',‘Mahesh',‘Suresh']">
  <ul>
    <li ng-repeat="name in students">
      {{ ‘Student Name: ’ +name }}
    </li>
  </ul>
</div>
```

ng-bind directive

- The ng-bind directive binds the model property declared via ng-model directive or the result of an expression to the HTML element. It also updates an element if the value of an expression changes.

```
<html >
  <head>
    <title>
       ng-bind Example
    </title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
</head>
<h1> Showing ng-bind Directive</h2>
<body ng-app="">
<div>
  Sum of 10 + 5 = <span ng-bind="10 + 5"></span> <br />
  Please Enter your Mobile no: <input type="text" ng-model="mobileno" /><br />
  Hi, my Mobile no is <span ng-bind="mobileno"></span>
</div>
</body>
</html>
```

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 }}*

```
<!DOCTYPE html>
<html>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">    </script>
<body>
 <div ng-app>
  <p> Expression: {{ 5 + 5 }} </p>
 </div>
</body>
</html>
```

If you remove the ng-app directive, HTML will display the expression as it is, without solving it

Change color of input box

- You can write expressions wherever you like, AngularJS will simply resolve the expression and return the result.
- Example: Let AngularJS change the value of CSS properties.
- Change the color of the input box below, by changing its value

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
 <p>Change the value of the input field:</p>
  <div ng-app="" ng-init="myCol='lightblue'">
    <input style="background-color:{{myCol}}" ng-model="myCol">
  </div>
 <p>AngularJS resolves the expression and returns the result.</p>
 <p>The background color of the input box that you write in the input field.</p>
</body>
</html>
```

AngularJS Numbers

- AngularJS numbers are like JavaScript numbers

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
  <div ng-app="" ng-init="quantity=3;cost=5">
  <p>Total in dollar: {{ quantity * cost }}</p>
  </div>
</body>
</html>
```

- The same can be achieved using *ng-bind*

```
<div ng-app="" ng-init="quantity=1;cost=5">
<p>Total in dollar: <span ng-bind="quantity * cost"></span></p>
</div>
```

AngularJS Strings

- String example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="" ng-init="firstName='NMIMS';lastName='Navi Mumbai'">
<p> Example: {{ firstName + " " + lastName }} </p>
</div>
</body>
</html>
```

- Same using *ng-bind*

```
<div ng-app="" ng-init="firstName='NMIMS';lastName='Navi Mumbai'">
<p>The name is <span ng-bind="firstName + ' ' + lastName"></span></p>
</div>
```

AngularJS Objects

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="" ng-init="person={firstName:‘NMIMS',lastName:‘Mumbai'}">
<p>My name is {{ person.firstName }}</p>
</div>
</body>
</html>
```

Same with ng-bind

```
<div ng-app="" ng-init="person={firstName:‘NMIMS',lastName:‘Navi Mumbai'}">
<p>The name is <span ng-bind="person.firstName"></span></p>
</div>
```

AngularJS Arrays

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The third element is {{ points[2] }}</p>
</div>
</body>
</html>
```

Same using ng-bind:

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The first result is <span ng-bind="points[0]"></span></p>
</div>
```

Cost Calculator using Angular JS

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div data-ng-app="" data-ng-init="quantity=1;price=20">
<h2>Cost Calculator</h2>
Quantity: <input type="number" ng-model="quantity">
Price: <input type="number" ng-model="price">
<p><b>Total in rupees:</b> {{quantity * price}}</p>
</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.
min.js">
</script>
<body>
<div ng-app>
<p>A simple expression example: {{ 5 + 5 }}</p>
</div>
</body>
</html>
```

Changing the color dynamically

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<p>Change the value of the input field:</p>
<div ng-app="" ng-init="myCol='pink'">
<input style="background-color:{{myCol}}" ng-model="myCol" value="{{myCol}}">
</div>
<p>AngularJS resolves the expression and returns the result.</p>
<p>The background color of the input box will be whatever you write in the input field.</p>
</body>
</html>
```

ng-list (directive)

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="">
<p>Write some names in the input field, use a comma to separate them:</p>
<input ng-model="customers" ng-list/>
<p>This example will convert your input into an array, one item for each name:</p>
<pre>{{customers}}</pre>
</div>
</body>
</html>
```

ng-keydown (directive)

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="">
<p>Press any key within the input box.</p>
<input ng-keydown="count = count + 1" ng-init="count=0" />
<h1>{{count}}</h1>
<p>Every time you press a key in the input field, this will increase the value of the variable "count".</p>
</body>
</html>
```

ng-style (directive)

```
<!DOCTYPE html>
<html>
<script src="   "></script>
<body ng-app="myApp" ng-controller="myCtrl">
<h1 ng-style="myObj">Welcome to NMIMS Navi Mumbai!</h1>
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
  $scope.myObj = {
    "color" : "white",
    "background-color" : "brown",
    "font-size" : "40px",
    "padding" : "40px"
  }
});
```

ng-required (directive)

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="">
<form name="myForm">
Click here to make the input field required:
<input type="checkbox" ng-model="myVar"><br><br>
<input name="myInput" ng-model="myInput" ng-required="myVar">
<h1 ng-if="!myForm.myInput.$valid">The input field cannot be empty</h1>
</form>
<p><strong>Note:</strong> This example shows an error if the input field is empty <strong>and</strong>
 the checkbox is checked for "required".</p>
</body>
</html>
```

ng-submit (directive)

```
<!DOCTYPE html>
<html>
<script src="   "></script>
<body ng-app="myApp" ng-controller="myCtrl">
<form ng-submit="myFunc()">
  <input type="text">
  <input type="submit">
</form>
<p>{{myTxt}}</p>
<p>ng-submit example.</p>



<script>
var app = angular.module("myApp", []);
```

AngularJS Modules & Controllers

Introduction

- An AngularJS module defines an application.
- The angular object's module() method is used to create a module.
- It is also called AngularJS function angular.module

```
<div ng-app="myApp">...</div>

<script>

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

</script>
```

- The "myApp" parameter refers to an HTML element in which the application will run.
- Now we can add controllers, directives, filters, and more, to AngularJS application.

How to add controller to a module

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
    $scope.firstName = "NMIMS";
    $scope.lastName = "Navi Mumbai";
});
</script>
</body>
</html>
```

How to add directive to a module

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>

<div ng-app="myApp" w3-test-directive></div>

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
    return {
        template : "This is a directive constructor. "
    };
});
</script>
</body>
</html>
```

Modules and controllers in file

Example.html

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

myApp.js **(for controller)**

What is Scope?

- The Scope is an object that is specified as a binding part between the HTML (view) and the JavaScript (controller).
- It plays a role of joining controller with the views. It is available for both the view and the controller.
- To make a controller in AngularJS, you have to pass the $scope object as an argument.

```
<div ng-app="myApp" ng-controller="myCtrl">
<h1>{{carname}}</h1>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.carname = "Volvo";
});
</script>
```

AngularJS Filters

Filter	Description
Currency	It formats a number to a currency format.
Date	It formats a date to a specified format.
Filter	It select a subset of items from an array.
Json	It formats an object to a Json string.
Limit	It is used to limit an array/string, into a specified number of elements/characters.
Lowercase	It formats a string to lower case.
Number	It formats a number to a string.
Orderby	It orders an array by an expression.
Uppercase	It formats a string to upper case.

How to add filters to expressions

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="personCtrl">
<p>The name is {{ firstName | uppercase }}</p>
</div>
<script>
angular.module('myApp', []).controller('personCtrl', function($scope) {
    $scope.firstName = "nmims",
    $scope.lastName = "navi mumbai"
});
</script>
</body>
</html>
```

AngularJS Validation

- AngularJS provides client-side form validation. It checks the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

- It also holds the information about whether the input fields have been touched, or modified, or not.

- Following directives are generally used to track errors in an AngularJS form:

  $dirty - states that value has been changed.
  $invalid - states that value entered is invalid.
  $error - states the exact error.

Example: AJS Validation

Required field Validation

```
<input name = "firstname" type = "text" ng-model = "firstName" required>
<span style = "color:red" ng-show = "studentForm.firstname.$dirty && studentForm.firstname.$invalid">
      <span ng-show = "studentForm.firstname.$error.required">First Name is required.</span>
 </span>
```

Email Validation

```
<input name = "email" type = "email" ng-model = "email" length = "100" required>
<span style = "color:red" ng-show = "studentForm.email.$dirty && studentForm.email.$invalid">
      <span ng-show = "studentForm.email.$error.required">Email is required.</span>
      <span ng-show = "studentForm.email.$error.email">Invalid email address.</span>
</span>
```

Click for Working Example