



BITE304L – WEB TECHNOLOGIES

DR.BRINDHA.K
ASSOCIATE PROFESSOR SENIOR
SCORE

MODULE – 3 :Web application and Angular JS

- Web applications
- Web application frame work
- MVC frame work
- Angular JS
- Introduction
- Data binding

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and in few words.

MODULE – 3 :Web application and Angular JS

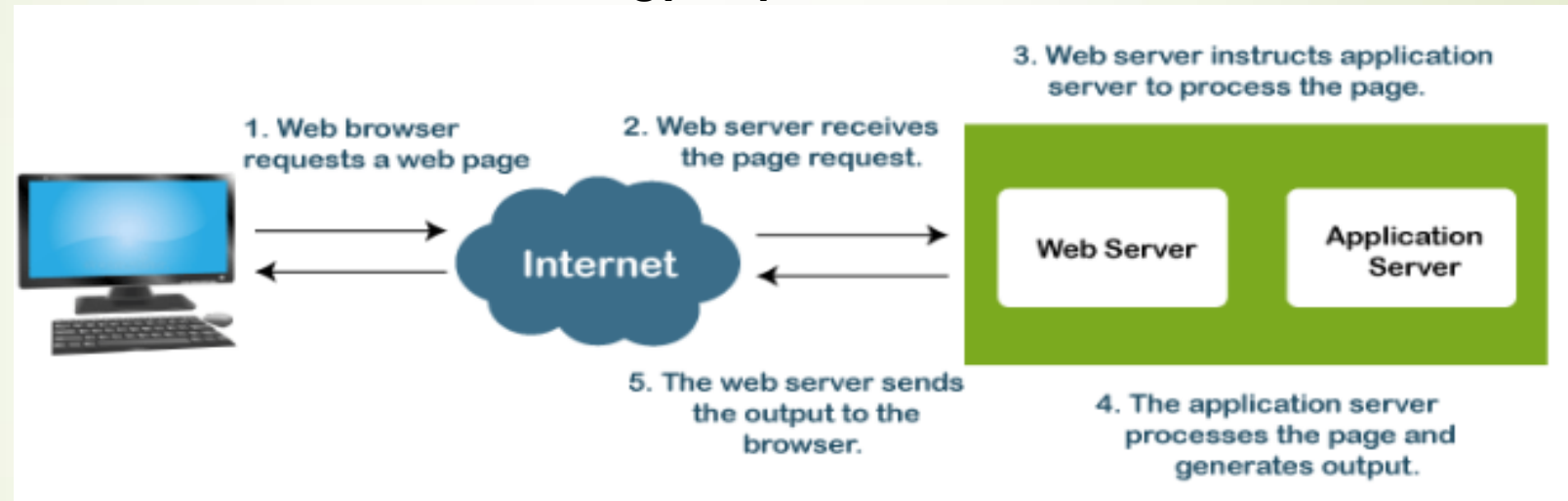
- Directives
- Modules
- Scopes
- Controllers
- Expressions
- Filters

MODULE – 3 :Web application and Angular JS

- Events
- Form – Single page application
- Multiple views and routing
- Service

Web application

- A web application is a Client-Server computer program that utilizes web browsers and web technology to perform tasks over the Internet.



- In general, a user sends a request to the web-server using web browsers such as Google Chrome, Microsoft Edge, Firefox, etc over the internet.
- Then, the request is forwarded to the appropriate web application server by the web-server.

Web application

- Web application server performs the requested operations/ tasks like processing the database, querying the databases; produces the result of the requested data.
- The obtained result is sent to the web-server by the web application server along with the requested data/information or processed data.
- The web server responds to the user with the requested or processed data/information and provides the result to the user's screen .
- **Merits**
- Any typical web application can run or accessible on any operating system such as the Windows, Mac, Linux as long as the browser is compatible.
- A web application is usually not required to install in the hard drive of the computer system, thus it eliminates all the issues related to the space limitation.
- Web applications are flexible. A user can work from any geographical location as long as he has a working internet connection.

Web application

➤ Demerits

- Internet connection is necessary to access any web application, and without an internet connection, anyone can't use any of the web applications. It is very typical to get an internet connection in our modern cities, still rural area internet connectivity not so well.
- A user must have to spend enough money to maintain the good condition of his web application.
- Speed-related issues are also affecting the web application's performance

➤ Example

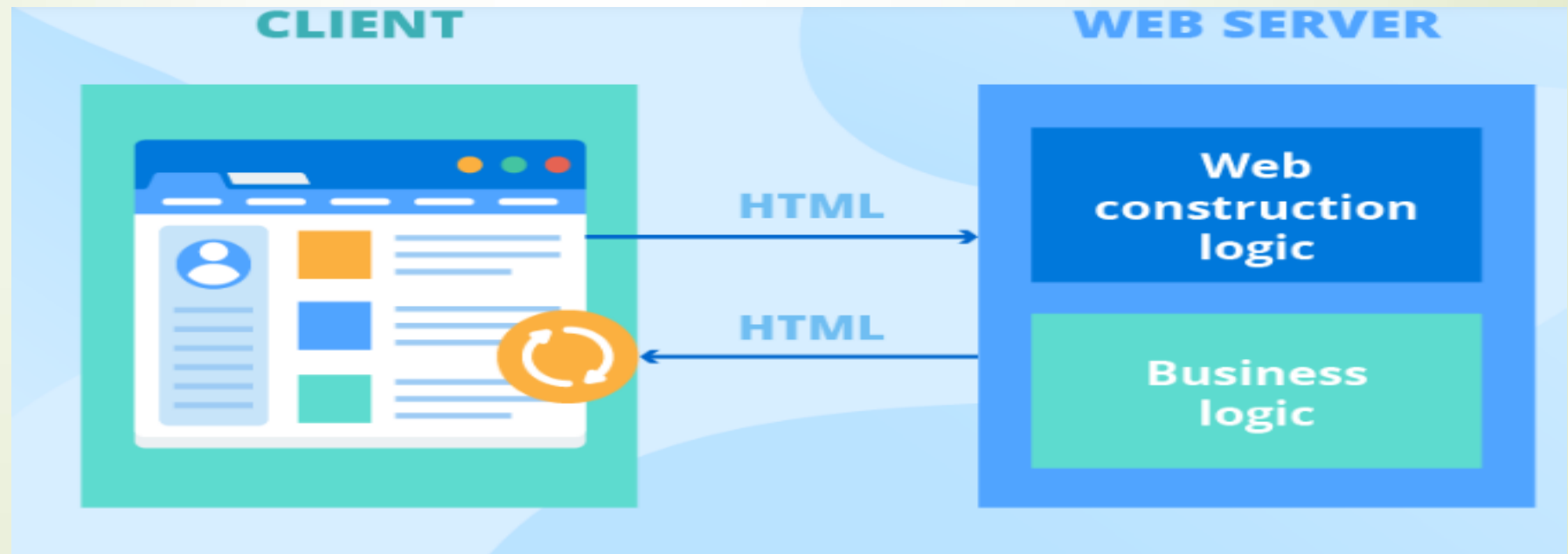
- Web applications include online forms, shopping carts, word processors, spreadsheets, video and photo editing, file conversion, file scanning, and email programs such as Gmail, Yahoo and AOL. Popular applications include Google Apps and Microsoft 365.

Web application Framework

- A web framework (WF) or web application framework (WAF) is a software framework that is **designed to support the development of web applications including web services, web resources, and web APIs.**
- Web frameworks provide **a standard way to build and deploy web applications.** Web frameworks aim to automate the overhead associated with common activities performed in web development.
- For example, many web frameworks **provide libraries for database access, templating frameworks, and session management,** and they often promote code reuse. Although they often target development of dynamic web sites, they are also applicable to static websites.

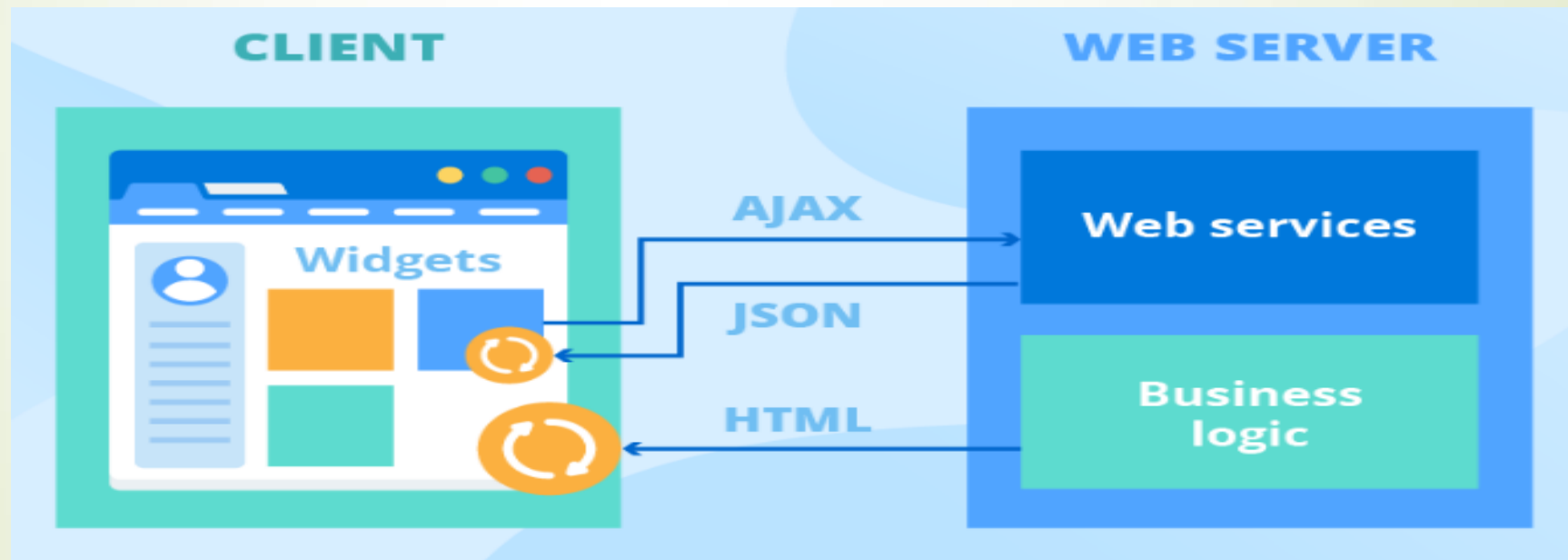
Types web architecture

- **Legacy HTML web app**
- According to the very basic web app architecture, a server, consisting of *web page construction logic* and *business logic* interacts with a client by sending out a complete HTML page.



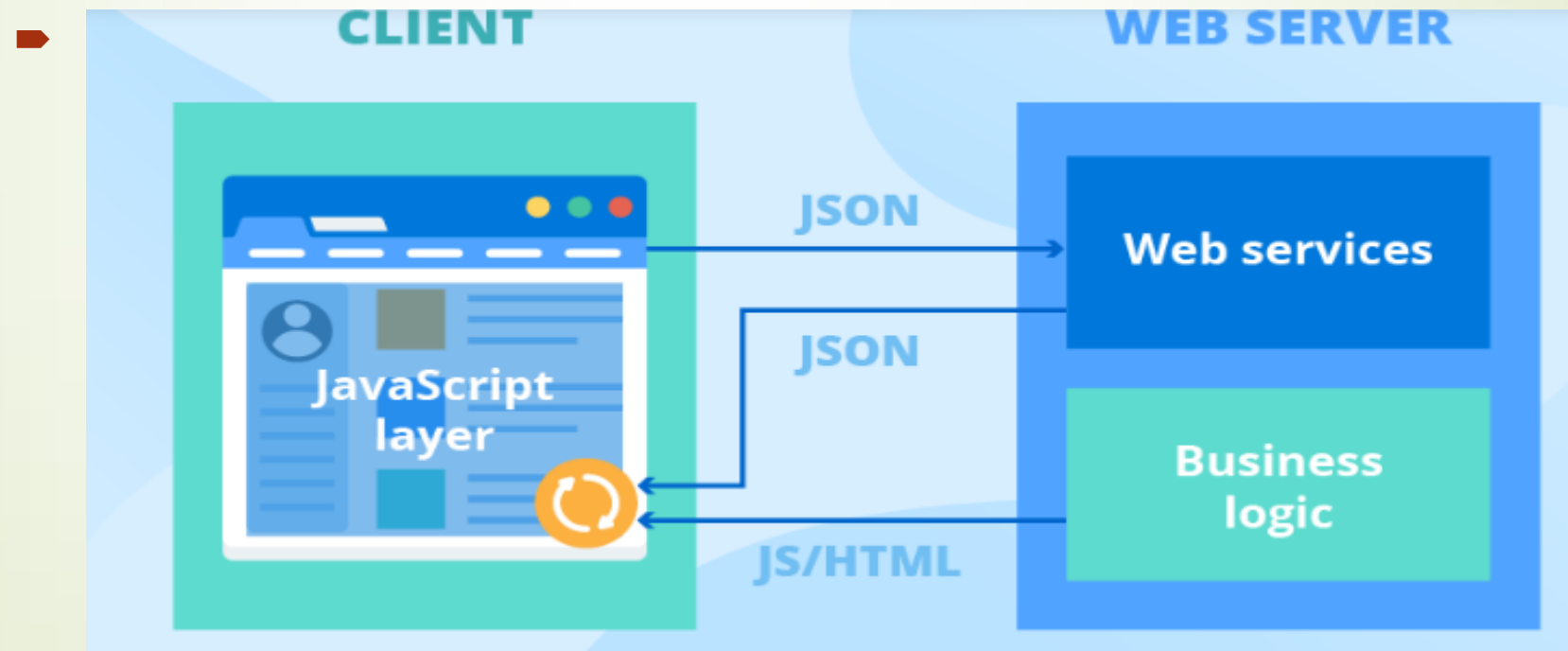
Widget web app

- In this type, the *web page construction logic* is replaced by *web services*, and each page on the client has separate entities called *widgets*. By sending *AJAX* queries to web services, widgets can receive chunks of data in *HTML* or *JSON* and display them without reloading the entire page.



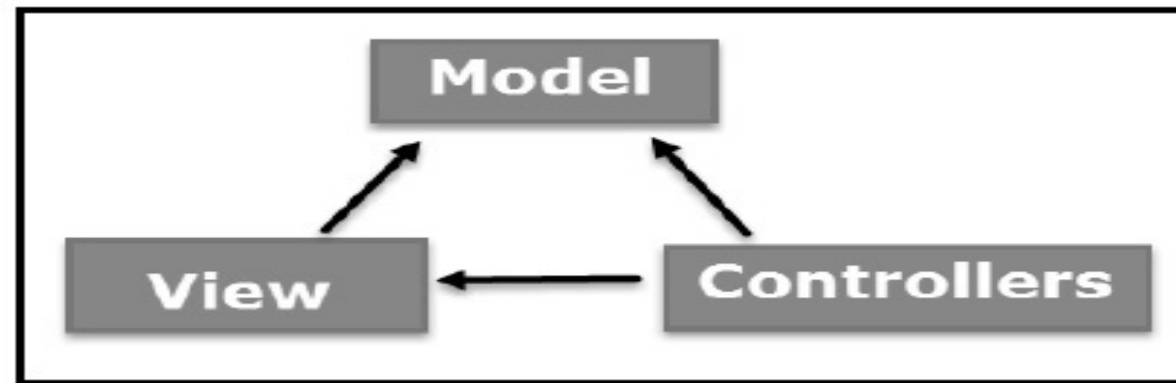
Single-page web app architecture

- ▶ With single-page applications (SPAs), you only download a single web page once. On the client side, this page has a JavaScript layer that can freely communicate with web services on the server and, using the data from web services, make real-time updates to itself.

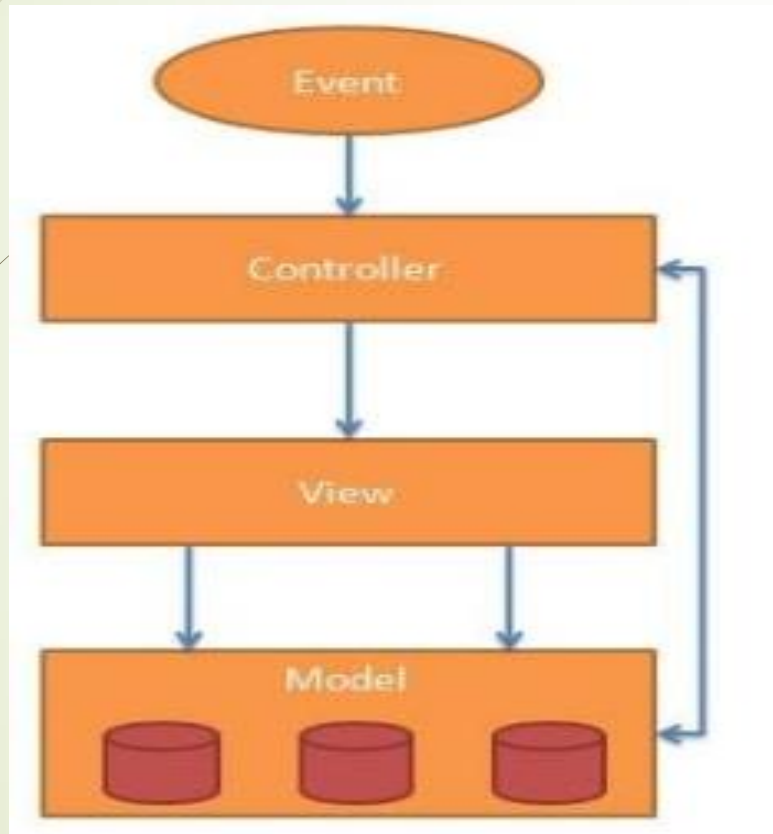


Model-View-Controller (MVC)

- The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.
- Each of these components are built to handle specific development aspects of an application
- MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.



AngularJS MVC Architecture



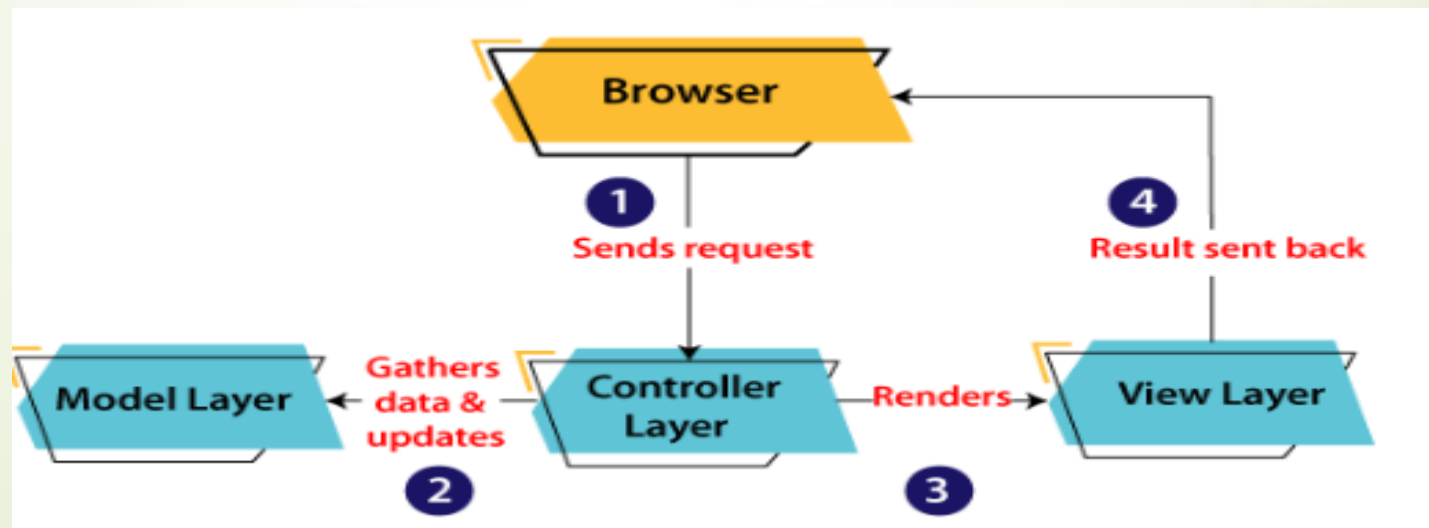
Controller: It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects

View: It is responsible for displaying all data or only a portion of data to the users.

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

Flow of MVC framework

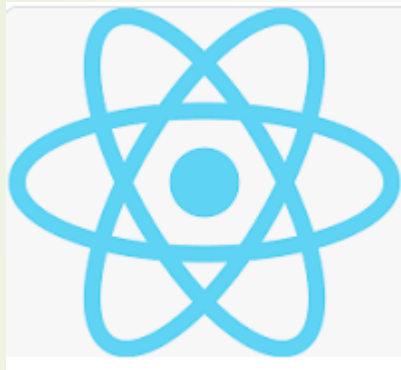
- A client (browser) sends a request to the controller on the server side, for a page.
- The controller then calls the model. It gathers the requested data.
- Then the controller transfers the data retrieved to the view layer.
- Now the result is sent back to the browser (client) by the view.



Merits of MVC framework

- MVC has the feature of scalability that in turn helps the growth of application.
- The components are easy to maintain because there is less dependency.
- A model can be reused by multiple views that provides reusability of code.
- The developers can work with the three layers (Model, View, and Controller) simultaneously.
- Using MVC, the application becomes more understandable.
- Using MVC, each layer is maintained separately therefore we do not require to deal with massive code.
- The extending and testing of application is easier.

Best front end Framework for web development – 2024





Angular JS

- AngularJS is an open source Model-View-Controller framework which is similar to the JavaScript framework.
- It can be added to an HTML page with a `<script>` tag. AngularJS extends HTML attributes with Directives, and binds data to HTML with Expressions.
- AngularJS is open source, completely free, and used by thousands of developers around the world. It is used in Single Page Application (SPA) projects.
- It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google
- AngularJS is a JavaScript framework written in JavaScript.

Key Features of Angular JS

- MVC framework
- Data Model Binding
- Writing less code
- Unit Testing ready
- AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

Download at <http://angularjs.org/>

(OR)

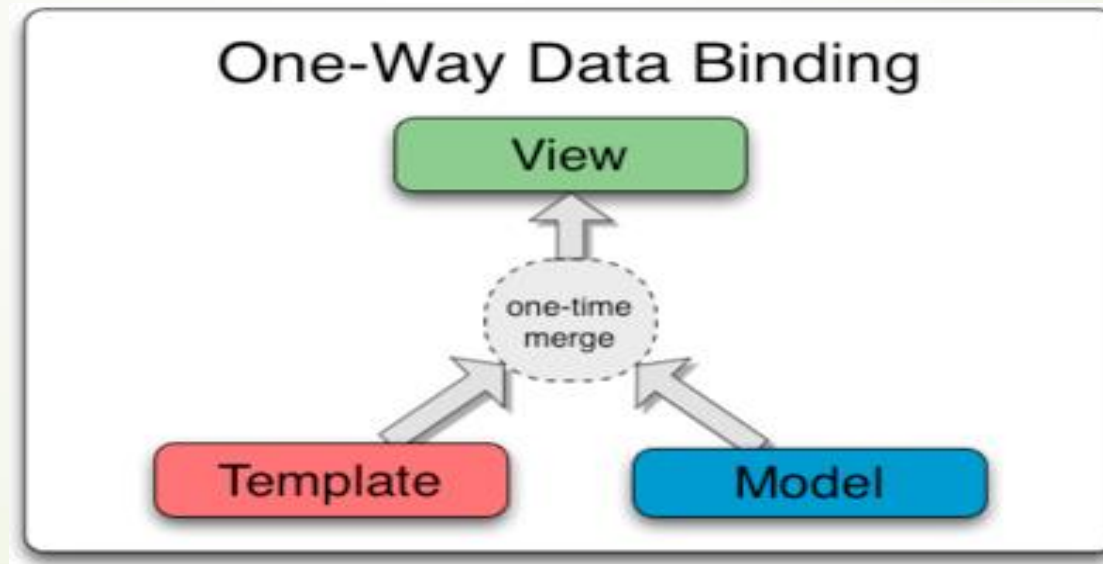
<https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js>

Angular JS Components

- Three main 3 directives are
- **ng-app** : This directive defines and links an AngularJS application to HTML.
- **ng-model** : This directive binds the values of AngularJS application data to HTML input controls.
- **ng-bind** : This directive binds the AngularJS application data to HTML tags.

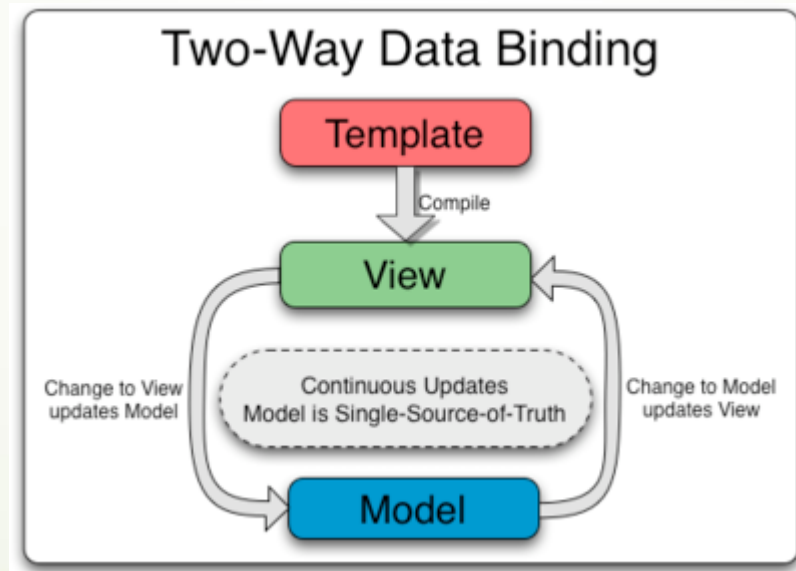
One way binding

- The one-way data binding is an approach where a value is taken from the data model and inserted into an HTML element.
- There is no way to update model from view. It is used in classical template systems. These systems bind data in only one direction.



Two way binding

- Data-binding in Angular apps is the automatic synchronization of data between the model and view components.
- The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.



Angular JS Directives

- Directives are classes that add additional behaviour to elements in your Angular applications. Simple AngularJS allows extending HTML with new attributes called Directives
- Most of the directives in AngularJS are starting with ng- where ng stands for Angular.

Important built-in Angular JS Directives

Directive	Description
ng-app	Auto bootstrap AngularJS application.
ng-init	Initializes AngularJS variables
ng-model	Binds HTML control's value to a property on the \$scope object.
ng-controller	Attaches the controller of MVC to the view.
ng-bind	Replaces the value of HTML control with the value of specified AngularJS expression.
ng-repeat	Repeats HTML template once per each item in the specified collection.
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.

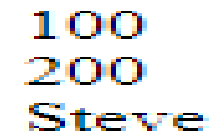
Important built-in Angular JS Directives

Directive	Description
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-init

- The ng-init directive can be used to initialize variables in AngularJS application.
- The following example demonstrates ng-init directive that initializes variable of string, number, array, and object.

```
<!DOCTYPE html>
<html >
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/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>
```



100
200
Steve

ng-model

- The ng-model directive is used for two-way data binding in AngularJS. It binds <input>, <select> or <textarea> elements to a specified property on the \$scope object.
- Two-way data binding is Angular applications' most often used data binding technique. The user types provide or changes any control value on one side. The same immediately updates into component variables and vice versa;
- In one-way data binding information flows in only one direction, and is when the information is displayed, but not updated.
- The property set via ng-model can be accessed in a controller using \$scope object.

ng-bind

- The ng-bind directive binds the model property declared via \$scope or ng-model directive or the result of an expression to the HTML element.

```
<!DOCTYPE html>
```

```
<html >
```

```
<head>
```

```
  <script
```

```
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
```

```
</head>
```

```
<body ng-app="">
```

```
  <div>
```

```
    5 + 5 = <span ng-bind="5 + 5"></span> <br />
```

```
    Enter your name: <input type="text" ng-model="name" /><br />
```

```
    Hello <span ng-bind="name"></span>
```

```
  </div>
```

```
</body>
```

```
</html>
```

5 + 5 = 10

Enter your name: shyam

Hello shyam

ng-repeat

- The ng-repeat directive repeats HTML once per each item in the specified array collection.

- Example

```
<body ng-app="" ng-init="students=['Bill','Steve','Ram']">  
  <ol>  
    <li ng-repeat="name in students">  
      {{name}}  
    </li>  
  </ol>  
  <div ng-repeat="name in students">  
    {{name}}  
  </div>
```

ng-if, ng-readonly, ng-disabled

- The ng-if directive creates or removes an HTML element based on the Boolean value returned from the specified expression. If an expression returns true then it recreates an element otherwise removes an element from the HTML document.
- ng-readonly
- The ng-readonly directive makes an HTML element read-only, based on the Boolean value returned from the specified expression. If an expression returns true then the element will become read-only, otherwise not.
- ng-disabled
- The ng-disabled directive disables an HTML element, based on the Boolean value returned from the specified expression. If an expression returns true the element will be disabled, otherwise not.

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.



Scope

- ▶ we can attach properties and methods to the `$scope` object inside controller function. The view can display `$scope` data using an expression, `ng-model`, or `ng-bind` directive.
- ▶ AngularJS creates and injects a different `$scope` object to each controller in an application.
- ▶ So, the data and methods attached to `$scope` inside one controller cannot be accessed in another controller. With the nested controller, child controller will inherit the parent controller's scope object.
- ▶ Therefore, child controller can access properties added in parent controller but parent controller cannot access properties added in child controller.
- ▶ The `ng-model` directive is used for two-way data binding. It transfers the data from controller to view and vice-versa. An expression and `ng-bind` directive transfers data from controller to view but not vice-versa.

Scope demo

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body ng-app="myNgApp">
<h1>AngularJS $scope Demo: </h1>
  <div ng-controller="myController">
    Message: <br />
    {{message}}<br />
    <span ng-bind="message"></span> <br />
    <input type="text" ng-model="message" />
  </div>
  <script>
    var ngApp = angular.module('myNgApp', []);

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

AngularJS \$scope Demo:

Message:
Hello World!
Hello World!

Angular JS Module

- A module in AngularJS is a container of the different parts of an application such as controller, service, filters, directives, factories etc. It supports separation of concern using modules.
- An AngularJS application must create a top level application module. This application module can contain other modules, controllers, filters, etc.
- The `angular.module()` method creates an application module, where the first parameter is a module name which is same as specified by `ng-app` directive. The second parameter is an array of other dependent modules [].
- The `angular.module()` method returns specified module object if no dependency is specified. Therefore, specify an empty array even if the current module is not dependent on other module.

Angular JS module demo

- we have created a controller named "myController" using `myApp.controller()` method. Here, `myApp` is an object of a module, and `controller()` method creates a controller inside "myApp" module.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
```

```
</head>
```

```
<body ng-app="myApp">
```

```
<h1>AngularJS Module Demo: </h1>
```

```
  <div ng-controller="myController">
```

```
    {{message}}
```

```
  </div>
```

```
<script>
```

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

```
  myApp.controller("myController", function ($scope) {
```

```
    $scope.message = "Hello Angular World!";
```

```
  });
```

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

Angular JS Controller

- The controller in AngularJS is a JavaScript function that maintains the application data and behavior using \$scope object.
- You can attach properties and methods to the \$scope object inside a controller function, which in turn will add/update the data and attach behaviours to HTML elements.
- The \$scope object is a glue between the controller and HTML.
- The ng-controller directive is used to specify a controller in HTML element, which will add behavior or maintain the data in that HTML element and its child elements.

Angular JS Controller demo

Hello World!

```
<!DOCTYPE html>
<html >
<head>
  <title>AngularJS Controller</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/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>
```

Creating Angular JS Application

- Step 1: Load framework Being a pure JavaScript framework, it can be added using
- `<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>`
- Step 2: Define AngularJS application using ng-app directive.

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

- Step 3: Define a model name using ng-model directive.

```
<p>Enter your Name: <input type="text" ng-model="name"></p>
```

- Step 4: Bind the value of above model defined using ng-bind directive.

```
<p>Hello <span ng-bind="name"></span>!</p>
```

- Executing AngularJS Application

Angular JS sample code

```
<html>
<title>AngularJS First Application</title>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">
</script>
<body>
<h1>Sample Application</h1>
<div ng-app="">
  <p>Enter your Name: <input type="text" ng-model="name"></p>
  <p>Hello <span ng-bind="name"></span>!</p>
</div>
</body>
</html>
```

Sample Application

Enter your Name:

Hello RAMA!

MVC demo

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
First Name: <input type="text" ng-model="firstName"><br><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>
</body>
</html>
```

First Name:

Last Name:

Full Name: John Doe

MVC demo 2

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
First Name: <input type="text" ng-model="firstName"><br><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Fullname : {{ firstName + " " +lastName }}
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName = $scope.firstName;
    $scope.lastName = $scope.lastName;
});
</script>
</body>
</html>
```

First Name:

Last Name:

Fullname : raj ram

Angular JS expression

- AngularJS expression is like JavaScript expression surrounded with braces - `{{ expression }}`. AngularJS evaluates the specified expression and binds the result data to HTML.
- AngularJS expression can contain literals, operators and variables like JavaScript expression. For example, an expression `{{2/2}}` will produce the result 1 and will be bound to HTML.
- AngularJS expression contains literals of any data type.
- AngularJS expression can contain arithmetic operators which will produce the result based on the type of operands, similar to JavaScript:
- AngularJS expression can contain variables declared via `ng-init` directive. The `ng-init` directive is used to declare AngularJS application variables of any data type.

Difference between JavaScript and Angular JS expression

- AngularJS expression cannot contain conditions, loops, exceptions or regular expressions e.g. if-else, ternary, for loop, while loop etc.
- AngularJS expression cannot declare functions.
- AngularJS expression cannot contain comma or void.
- AngularJS expression cannot contain return keyword.

Expression demo - operators

```
<!DOCTYPE html>
<html >
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body >
  <h1>AngularJS Expression Demo:</h1>
  <div ng-app>
    2 + 2 = {{2 + 2}} <br />
    2 - 2 = {{2 - 2}} <br />
    2 * 2 = {{2 * 2}} <br />
    2 / 2 = {{2 / 2}}
  </div>
</body>
</html>
```

AngularJS Expression Demo:

2 + 2 = 4
2 - 2 = 0
2 * 2 = 4
2 / 2 = 1

Expression demo – literals of any type

```
<html>
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></scri
pt>
</head>
<body >
  <h1>AngularJS Expression Demo:</h1>
  <div ng-app>
    {{ "Hello World" }}<br />
    {{ 100 }}<br />
    {{ true }}<br />
    {{ 10.2 }}
  </div>
</body>
</html>
```

AngularJS Expression Demo:

Hello World
100
true
10.2

Expression demo 3

```
Hello World  
200  
1  
20.4
```

```
<!DOCTYPE html>  
<html >  
<head>  
  <script  
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></scri  
pt>  
</head>  
<body >  
  <div ng-app>  
    {{ "Hello" + " World" }}<br />  
    {{ 100 + 100 }}<br />  
    {{ true + false }}<br />  
    {{ 10.2 + 10.2 }}<br />  
  </div>  
</body>  
</html>
```

Expression demo 4

```
<!DOCTYPE html>
<html >
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body >
  <div ng-app ng-init="greet='Hello World!'; amount= 10000;rateOfInterest = 10.5;
duration=10; myArr = [100, 200]; person = { firstName:'Steve', lastName : 'Jobs'}">
    {{ (amount * rateOfInterest * duration)/100 }}<br />
    {{myArr[1]}} <br />
    {{person.firstName + " " + person.lastName}}
  </div>
</body>
</html>
```

Angular JS Filters

- **AngularJS Filters allow us to format the data to display on UI without changing original format.**
- **Filters can be used with an expression or directives using pipe | sign.**
- **{{expression | filterName:parameter }}**
- **Angular includes various filters to format data of different data types.**

List of Important Filters

Filter Name	Description
Number	Formats a numeric data as text with comma and fraction.
Currency	Formats numeric data into specified currency format and fraction.
Date	Formats date to string in specified format.
Uppercase	Converts string to upper case.
Lowercase	Converts string to lower case.
Filter	Filters an array based on specified criteria and returns new array.
orderBy	Sorts an array based on specified predicate expression.
Json	Converts JavaScript object into JSON string
limitTo	Returns new array containing specified number of elements from an existing array.

Number Filter

- A number filter formats numeric data as text with comma and specified fraction size.
- `{{ number_expression | number:fractionSize }}`
- If a specified expression does not return a valid number then number filter displays an empty string.
- The following example demonstrates how to use number filter with number expression or a model property.

Number filter demo

AngularJS Number Filter Demo:

```

<!DOCTYPE html>
<html>
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/an
</head>
<body ng-app >
<h1>AngularJS Number Filter Demo: </h1>
  Enter Amount: <input type="number" ng-model="amount" /> <br />
  100000 | number = {{100000 | number}} <br />
  amount | number = {{amount | number}} <br />
  amount | number:2 = {{amount | number:2}} <br />
  amount | number:4 = {{amount | number:4}} <br />
  amount | number = <span ng-bind="amount | number"></span>
</body>
</html>

```

Enter Amount:

100000		number = 100,000
amount		number = 55,000
amount		number:2 = 55,000.00
amount		number:4 = 55,000.0000
amount		number = 55,000

Currency Filter

- The currency filter formats a number value as a currency. When no currency symbol is provided, default symbol for current locale is used.
- `{{ expression | currency : 'currency_symbol' : 'fraction' }}`

Currency Filter demo

AngularJS currency Filter Demo:

Default currency: \$100,000.00
 Custom currency identifier: Rs.100,000.00
 No Fraction: Rs.100,000
 Fraction 2: GBP100,000.00

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body ng-app="myApp" >
<h1>AngularJS currency Filter Demo: </h1>
  <div ng-controller="myController">
    Default currency: {{person.salary | currency}} <br />
    Custom currency identifier: {{person.salary | currency:'Rs.'}} <br />
    No Fraction: {{person.salary | currency:'Rs.':0}} <br />
    Fraction 2: <span ng-bind="person.salary | currency:'GBP':2"></span>
  </div>
  <script>
    var myApp = angular.module('myApp', []);
    myApp.controller("myController", function ($scope) {
      $scope.person = { firstName: 'James', lastName: 'Bond', salary: 100000}
    });
  </script>
</body>
</html>
```

Date Filter

- Formats date to string based on the specified format.

- `{{ date_expression | date : 'format' }}`

```
<!DOCTYPE html>
```

```
<html>
```

```
<script
```

```
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
```

```
<body>
```

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

```
<p>Date = {{ today | date }}</p>
```

```
</div>
```

```
<script>
```

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

```
app.controller('datCtrl', function($scope) {
```

```
    $scope.today = new Date();
```

```
});
```

```
</script>
```

```
<p>The date filter formats a date object to a readable format.</p>
```

```
</body>
```

```
</html>
```

Date = Feb 16, 2023

The date filter formats a date object to a readable format.

Date Filter

- `<p>Date = {{ today | date }}</p>`
- `<p>Date = {{ today | date:"short" }}</p>` **"M/d/yy h:mm a"**
- `<p>Date = {{ today | date:"medium" }}</p>` **MMM d, y h:mm:ss a**
- `<p>Date = {{ today | date:"shortDate" }}</p>` **"M/d/yy"**
- `<p>Date = {{ today | date:"mediumDate" }}</p>` **"MMM d, y"**
- `<p>Date = {{ today | date:"longDate" }}</p>` **"MMMM d, y"**
- `<p>Date = {{ today | date:"fullDate" }}</p>` **"EEEE, MMMM d, y"**
- `<p>Date = {{ today | date:"shortTime" }}</p>` **"h:mm a"**
- `<p>Date = {{ today | date:"mediumTime" }}</p>` **"h:mm:ss a"**

Uppercase/lowercase filter

- The uppercase filter converts the string to upper case and lowercase filter converts the string to lower case.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <script
```

```
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
```

```
</head>
```

```
<body ng-app >
```

```
<h1>AngularJS Uppercase/Lowercase Filter Demo: </h1>
```

```
  <div ng-init="person.firstName='James';person.lastName='Bond'">
```

```
    Lower case: {{person.firstName + ' ' + person.lastName | lowercase}} <br />
```

```
    Upper case: {{person.firstName + ' ' + person.lastName | uppercase}}
```

```
  </div>
```

```
</body>
```

```
</html>
```

AngularJS Uppercase/Lowercase Filter Demo:

Lower case: james bond

Upper case: JAMES BOND

Filter based on search criteria

- Filter selects items from an array based on the specified criteria and returns a new array.
- `{{ expression | filter : filter_criteria }}`

Filter based on search criteria demo

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body ng-app="myApp" >
<h1>AngularJS Filter Demo: </h1>
  <div ng-controller="myController">
    Enter Name to search: <input type="text" ng-model="searchCriteria" /> <br />
    Result: {{myArr | filter:searchCriteria}}
  </div>
  <script>
    var myApp = angular.module('myApp', []);

    myApp.controller("myController", function ($scope) {
      $scope.myArr = ['Steve', 'Bill', 'James', 'Rob', 'Ram', 'Moin']
    });
  </script>
</body>
</html>
```

AngularJS Filter Demo:

Enter Name to search:

Result: ["Moin"]

orderBy filter

- The orderBy filter sorts an array based on specified expression predicate.
- `{{ expression | orderBy : predicate_expression : reverse }}`

Orderby filter demo

```
<!DOCTYPE html>
<html>
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body ng-app="myApp">
  <div ng-controller="myController">
    <select ng-model="SortOrder">
      <option value="+name">Name (asc)</option>
      <option value="-name">Name (dec)</option>
      <option value="+phone">Phone (asc)</option>
      <option value="-phone">Phone (dec)</option>
    </select>
    <ul ng-repeat="person in persons | orderBy:SortOrder">
      <li>{{person.name}} - {{person.phone}}</li>
    </ul>
  </div>
```

OrderBy filter demo

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

  myApp.controller("myController", function ($scope) {

    $scope.persons = [{ name: 'John', phone: '512-455-1276' },
                      { name: 'Mary', phone: '899-333-3345' },
                      { name: 'Mike', phone: '511-444-4321' },
                      { name: 'Bill', phone: '145-788-5678' },
                      { name: 'Ram', phone: '433-444-8765' },
                      { name: 'Steve', phone: '218-345-5678' }]

    $scope.SortOrder = '+name';
  });
</script>
</body>
</html>
```

Name (asc) ▼

- Bill - 145-788-5678
- John - 512-455-1276
- Mary - 899-333-3345
- Mike - 511-444-4321
- Ram - 433-444-8765
- Steve - 218-345-5678

Angular JS HTML DOM

- In AngularJS, some directives can be used to bind application data to attributes of HTML DOM elements.

Directive	Description
ng-disabled	It disables a given control.
ng-show	It shows a given control.
ng-hide	It hides a given control.
ng-click	It represents an AangularJS click event.

DOM demo

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
  <h2>AngularJS Sample Application</h2>
  <div ng-app = "">
    <table border = "0">
      <tr>
        <td><input type = "checkbox" ng-model = "enableDisableButton">Disable
Button</td>
        <td><button ng-disabled = "enableDisableButton">Click Me!</button></td>
      </tr>
      <tr>
        <td><input type = "checkbox" ng-model = "showHide1">Show Button</td>
        <td><button ng-show = "showHide1">Click Me!</button></td>
      </tr>
    </table>
  </div>
</body>
</html>
```

DOM demo

```

<tr>
  <td><input type = "checkbox" ng-model = "showHide2">Hide
Button</td>
  <td><button ng-hide = "showHide2">Click Me!</button></td>
</tr>
<tr>
  <td><p>Total click: {{ clickCounter }}</p></td>
  <td><button ng-click = "clickCounter = clickCounter + 1">Click
Me!</button></td>
</tr>
</table>
</div>
</body>
</html>

```

AngularJS Sample Application

☒ Disable Button
☒ Show Button
☒ Hide Button

Total click: 2

Angular JS Events

- AngularJS includes certain directives which can be used to provide custom behavior on various DOM events, such as click, dblclick, mouseenter etc.

Event Directive	Event Directive
ng-blur	ng-keypress
ng-change	ng-mousedown
ng-click	ng-mouseenter
ng-dblclick	ng-mouseleave
ng-focus	ng-mousemove
ng-keydown	ng-mouseover
ng-keyup	ng-mouseup

ng-click demo

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script
>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<button ng-click="count = count + 1">Click Me!</button>
<p>{{ count }}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.count = 0;
});
</script>
</body>
</html>
```



Mouse event demo

```
<!DOCTYPE html>
<html>
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"><
/script>
  <style>
    .redDiv {
      width: 100px;
      height: 100px;
      background-color: red;
      padding: 2px 2px 2px 2px;
    }
  </style>
</head>
<body>
  <div class="redDiv">
    <button class="btn btn-primary">Click Me!
  </div>
</body>
</html>
```

Mouse event demo

AngularJS Mouse Events Demo:

```
.yellowDiv {
  width: 100px;
  height: 100px;
  background-color: yellow;
  padding:2px 2px 2px 2px;
}
</style>
</head>
<body ng-app>
<h1>AngularJS Mouse Events Demo: </h1>
  <div ng-class="{redDiv: enter, yellowDiv: leave}" ng-
mouseenter="enter=true;leave=false;" ng-mouseleave="leave=true;enter=false">
    Mouse <span ng-show="enter">Enter</span> <span ng-
show="leave">Leave</span>
  </div>
</body>
</html>
```

Mouse Leave

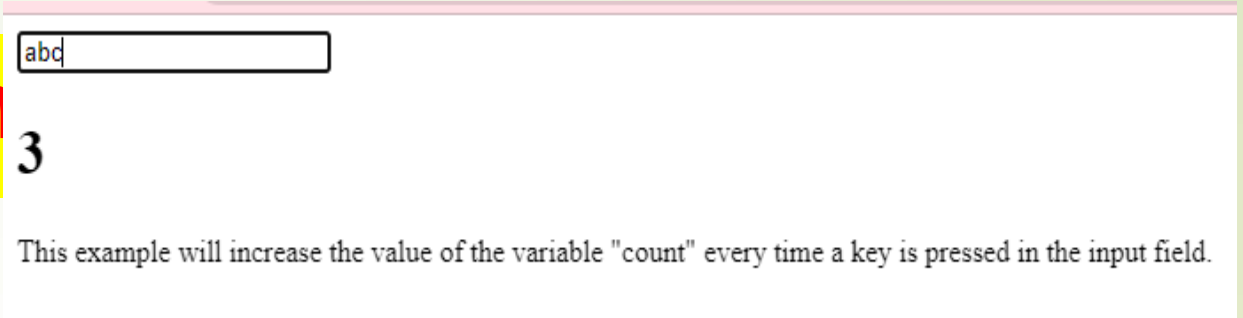


Key press Event

```

<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></s
cript>
<body ng-app="">
<input ng-keypress="count = count + 1" ng-init="count=0" />
<h1>{{count}}</h1>
<p>This example will increase the value of the variable "count" every time a
key is pressed in the input field.</p>
</body>
</html>

```



ng-focus demo

```
<!DOCTYPE html>
<html>
  <head>
    <title>ng-focus Directive</title>
    <script src=
"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
    </script>
  </head>
  <body ng-app="app" style="text-align:center;">
    <h1 style="color:lime">
      Focusing any form element
    </h1>
    <h2>ng-focus Directive</h2>
```

ng-focus demo

```
<div ng-controller="mycontrl"> Input:
  <input type="text"
    ng-focus="focused = true"
    ng-blur="focused = false" />
  <pre ng-show="focused">
    Input is focused.
  </pre>
</div>
<script>
  var app = angular.module("app", []);
  app.controller('mycontrl', ['$scope',
    function($scope) {}]);
</script>
</body>
</html>
```

Focusing any form element

ng-focus Directive

Input:

Input is focused.

Focusing any form element

ng-focus Directive

Input:

Angular JS – ng-repeat demo using table

```

<!DOCTYPE html>
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body ng-app="" ng-init="students=[
  {name:'ram',marks:70},
  {name:'shyam',marks:80},
  {name:'TOM',marks:65},
  {name:'KAM',marks:75},
  {name:'SAM',marks:67}
]">
  <table border="2">
    <tr> <td>Name of the students </td> <td> course mark </td> </tr>
    <tr ng-repeat="x in students">
      <td> {{x.name}} </td>
      <td> {{x.marks}} </td>
    </tr>
  </table>
</body>
</html>

```

Name of the students	course mark
ram	70
shyam	80
TOM	65
KAM	75
SAM	67

Angular JS Forms

- Forms in AngularJS provides data-binding and validation of input controls.
- Input controls provides data-binding by using the ng-model directive.
- Input controls are the HTML input elements:
 - input elements
 - select elements
 - button elements
 - textarea elements
- **Example – Text box**

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

```
<form>
```

```
  First Name: <input type="text" ng-model="firstname">
```

```
</form>
```

```
<h1>You entered: {{firstname}}</h1>
```

```
</div>
```


Angular JS Forms

■ Example – Check box / Radio button

```
<form>
```

Pick a topic:

```
<input type="radio" ng-model="myVar" value="dogs">Dogs
```

```
<input type="radio" ng-model="myVar" value="tuts">Tutorials
```

```
<input type="radio" ng-model="myVar" value="cars">Cars
```

```
</form>
```

```
<div ng-switch="myVar">
```

```
<div ng-switch-when="dogs">
```

```
<h1>Dogs</h1>
```

```
<p>Welcome to a world of dogs.</p>
```

```
</div>
```

```
<div ng-switch-when="tuts">
```

```
<h1>Tutorials</h1>
```

```
<p>Learn from examples.</p>
```

```
</div>
```

```
<div ng-switch-when="cars">
```

```
<h1>Cars</h1>
```

```
<p>Read about cars.</p>
```

```
</div>
```

```
</div>
```

```
<ANY ng-switch="expression">
```

```
<ANY ng-switch-when="matchValue1">...</ANY>
```

```
<ANY ng-switch-when="matchValue2">...</ANY>
```

```
<ANY ng-switch-default>...</ANY>
```

```
</ANY>
```

Angular JS Forms

■ Example – List box

```
<form>
  Select a topic:
  <select ng-model="myVar">
    <option value="">
    <option value="dogs">Dogs
    <option value="tuts">Tutorials
    <option value="cars">Cars
  </select>
</form>

<div ng-switch="myVar">
  <div ng-switch-when="dogs">
    <h1>Dogs</h1>
    <p>Welcome to a world of dogs.</p>
  </div>
  <div ng-switch-when="tuts">
    <h1>Tutorials</h1>
    <p>Learn from examples.</p>
  </div>
  <div ng-switch-when="cars">
    <h1>Cars</h1>
    <p>Read about cars.</p>
  </div>
</div>
```

Form validation in Angular JS

- AngularJS offers client-side form validation.
- AngularJS includes the following validation directives.

Directive	Description
ng-required	Sets required attribute on an input field.
ng-minlength	Sets minlength attribute on an input field.
ng-maxlength	Sets maxlength attribute on an input field. Setting the attribute to a negative or non-numeric value, allows view values of any length.
ng-pattern	Sets pattern validation error key if the ngModel value does not match the specified RegEx expression.

Form validation in Angular JS

- **State Properties**
- Angular includes properties which return the state of form and input controls.
- The state of the form and control changes based on the user's interaction and validation errors.
- These built-in properties can be accessed using form name or input control name.
- To check the form status **use `formName.propertyName`**,
- To check the state of input control, **use `formName.inputFieldName.propertyName`**.

Form validation in Angular JS

➤ State Properties

\$error	\$error object contains all the validation attributes applied to the specified element.
\$pristine	Returns true if the user has not interacted with control yet else returns false.
\$valid	Returns true if the model is valid
\$invalid	Returns true if the model is invalid
\$dirty	Returns true if user changed the value of model at least once
\$touched	Returns true if the user has tabbed out from the control.
\$untouched	Returns true if the user has not tabbed out from the control.

Angular JS validation CSS classes

- AngularJS includes following CSS classes to allow styling of form and input controls based on the state of form field.

CSS Class	Description
ng-valid	Angular will set this CSS class if the input field is valid without errors.
ng-invalid	Angular will set this CSS class if the input does not pass validations.
ng-pristine	Angular will set this CSS class if a user has not interacted with the control yet.
ng-dirty	Angular will set this CSS class if the value of form field has been changed.
ng-touched	Angular will set this CSS class if a user tabbed out from the input control.
ng-untouched	Angular will set this CSS class if a user has not tabbed out from the input control.
ng-submitted	Angular will set this CSS class if the form has been submitted.

Angular JS form validation without pattern

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<h2>Validation Example</h2>
<form ng-app="myApp" ng-controller="validateCtrl"
name="myForm" novalidate>
<p>Username:<br>
<input type="text" name="user" ng-model="user" required>
<span style="color:red" ng-show="myForm.user.$dirty && myForm.user.$invalid">
<span ng-show="myForm.user.$error.required">Username is required.</span>
</span>
</p>
<p>Email:<br>
<input type="email" name="email" ng-model="email" required>
<span style="color:red" ng-show="myForm.email.$dirty && myForm.email.$invalid">
<span ng-show="myForm.email.$error.required">Email is required.</span>
<span ng-show="myForm.email.$error.email">Invalid email address.</span>
</span>
```

Angular JS form validation without pattern

```

</p>
<p>
<input type="submit"
ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||
myForm.email.$dirty && myForm.email.$invalid">
</p>
</form>
<script>
var app = angular.module('myApp', []);
app.controller('validateCtrl', function($scope) {
    $scope.user = 'John Doe';
    $scope.email = 'john.doe@gmail.com';
});
</script>
</body>
</html>

```

Validation Example

Username: Username is required.

Email: Email is required.

Validation Example

Username:

Email:

Angular JS Form validation with pattern- Single form field demo

```

<!DOCTYPE html>
<html>
<head>
<title>
AngularJs ng-pattern Form Validation Example
</title>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<script type="text/javascript">
var app = angular.module('ngpatternApp', []);
app.controller('ngpatternCtrl', function ($scope) {
$scope.sendForm = function () {
$scope.msg = "Form Validated";
};
});
</script>

```

AngularJs ng-pattern Form Validation Example

Pin Code: Required!

Submit Form

AngularJs ng-pattern Form Validation Example

Pin Code:

Submit Form

Form Validated

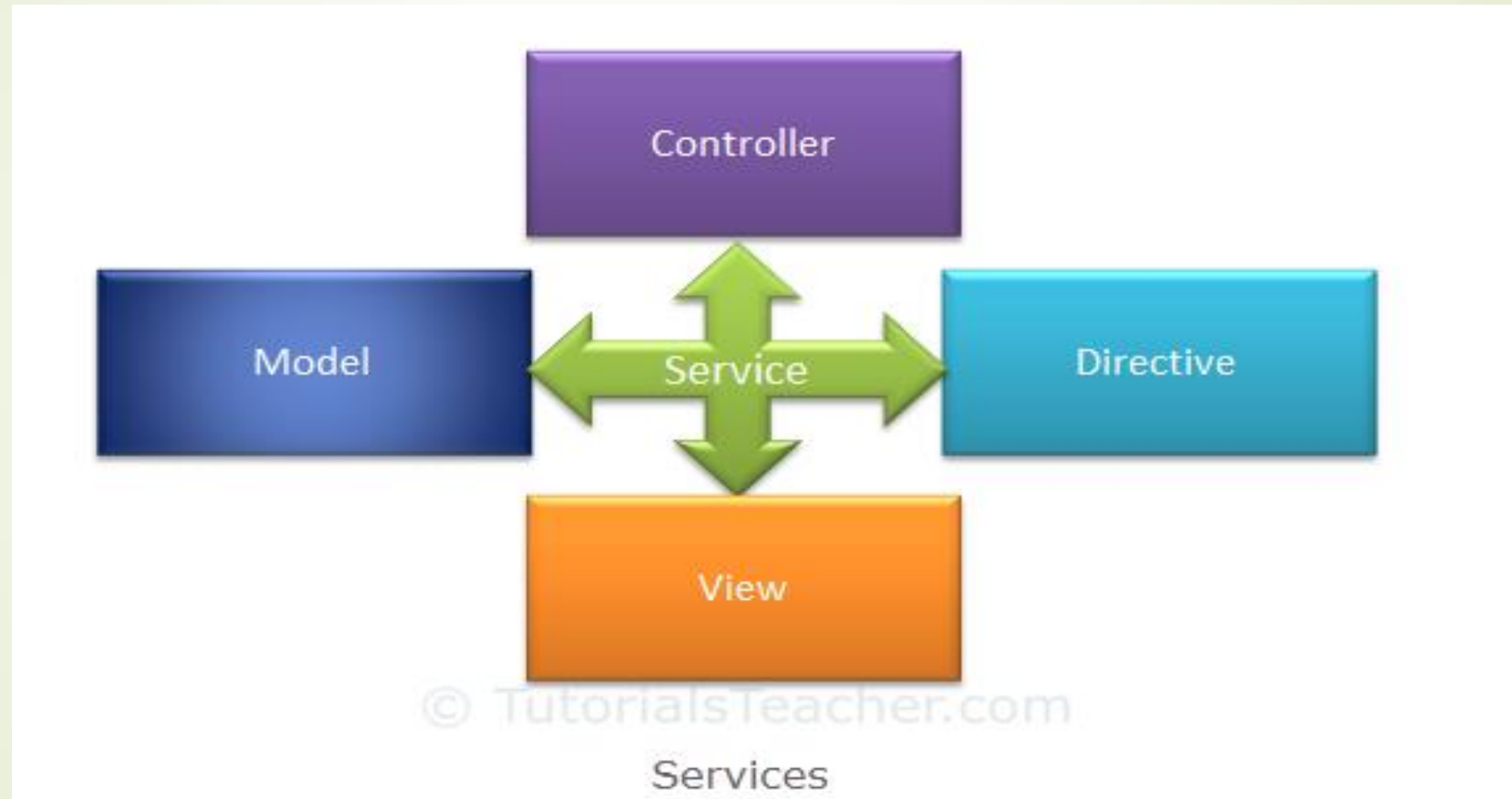
Angular JS Form validation with pattern- Single form field demo

```
</head>
<body>
<div ng-app="ngpatternApp" ng-controller="ngpatternCtrl">
<h3>AngularJs ng-pattern Form Validation Example</h3>
<form name="personForm" novalidate ng-submit="personForm.$valid &&sendForm()">
Pin Code:<input type="text" name="pincode" ng-model="txtpin" ng-pattern="/^[0-9]{1,6}$/" required />
<span style="color:Red" ng-show="personForm.pincode.$error.required"> Required!
</span>
<span style="color:Red" ng-
show="personForm.pincode.$dirty&&personForm.pincode.$error.pattern">Only Numbers
Allowed, Maximum 5 Characters</span>
<br /><br />
<button type="submit">Submit Form</button><br /><br />
<span>{{msg}}</span>
</form> </div> </body> </html>
```

Angular JS Services

- 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 built-in services starts with \$, same as other built-in objects.
- Most AngularJS services interact with the controller, model or custom directives. However, some services interact with view (UI) also for UI specific tasks.

Angular JS Services



\$location service

- In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application.
- The \$location service has methods which return information about the location of the current web page:

\$location service demo

```
!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<p>The url of this page is:</p>
<h3>{{myUrl}}</h3>
</div>
<p>This example uses the built-in $location service to get the absolute url of the page.</p>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $location) {
    $scope.myUrl = $location.absUrl();
});
</script>
</body>
</html>
```

The url of this page is:

<file:///D:/Winter%20Semester%202022-2023/Theory%20materials/Module%203/Angular/location%20service%20demo.html#!/>

This example uses the built-in \$location service to get the absolute url of the page.

\$http service

- The \$http service is used to send or receive data from the remote server using browser's XMLHttpRequest
- \$http is a service as an object

\$route service

- \$route is used for deep-linking URLs to controllers and views (HTML partials). It watches \$location.url() and tries to map the path to an existing route definition.
- Requires the ngRoute module to be installed.
- You can define routes through \$routeProvider's API.

\$ 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.

\$window service demo

```
<!DOCTYPE html>
<html>
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"><
  /script>
</head>
<body ng-app="myApp" ng-controller="myController">
<h1>AngularJS cancel $window Demo: </h1>
  <button ng-click="DisplayAlert('Hello World!')">Display Alert</button>
  <button ng-click="DisplayPrompt()">Display Prompt</button>
  <script>
    var myApp = angular.module('myApp', []);
```

\$window service demo

AngularJS cancel \$window Demo:

```
myApp.controller("myController", function ($scope, $window) {  
    $scope.DisplayAlert = function (message) {  
        $window.alert(message);  
    }  
    $scope.DisplayPrompt = function () {  
        var name = $window.prompt('Enter Your Name');  
  
        $window.alert('Hello ' + name);  
    }  
});  
</script>  
</body>  
</html>
```

\$timeout Service

The \$timeout service is AngularJS' version of the window.setTimeout function.

```
<!DOCTYPE html>
```

```
<html>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
```

```
<body>
```

```
<div ng-app="myApp" ng-controller="myCtrl">
```

```
<p>This header will change after two seconds:</p>
```

```
<h1>{{myHeader}}</h1> </div>
```

```
<p>The $timeout service runs a function after a specified number of milliseconds.</p>
```

```
<script>
```

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

```
app.controller('myCtrl', function($scope, $timeout) {
```

```
  $scope.myHeader = "Hello World!";
```

```
  $timeout(function () {
```

```
    $scope.myHeader = "How are you today?";
```

```
  }, 2000);
```

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

This header will change after two seconds:

How are you today?

The \$timeout service runs a function after a specified number of milliseconds.

Angular JS Routing

- We can build Single Page Application (SPA) with AngularJS. It is a web app that loads a single HTML page and dynamically updates that page as the user interacts with the web app.
- The ngRoute module helps your application to become a Single Page Application.
- If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.
- The ngRoute module *routes* your application to different pages without reloading the entire application.

Procedure to develop a single page application

- Creation of simple html page and add a simple layout with a navigation bar. We will also load angular and angular route library via CDN.
- Every angular application starts from creating a angular module. Module is a container that holds the different parts of your application such as controllers, service, etc
- The app variable is initialized as angular module and named as “myApp”, the ng-app directive is used to bind the angular application to a container. It is important to use the same module name to bind using ng-app directive.
- **Example**
- `var app = angular.module("myApp", ['ngRoute']);`

Procedure to develop a single page application

- Configure the routes
- The ngRoute module provides routing, deeplinking services and directives to our angular applications. We will be using \$routeProvider in app.config to configure our application routing.
- Example

```
app.config(function($routeProvider){  
    $routeProvider  
        .when("/", {template:"<p>AngularJS Single Page Application(SPA).</p>"})  
        .when("/about", {templateUrl : "about.html"})  
        .when("/services", {templateUrl : "services.html"})  
        .when("/projects", {templateUrl : "projects.html"})  
    });
```

Procedure to develop a single page application

- Writing HTML code inside template is very constraining and to counter it one can use `templateURL` and instead of writing HTML code directly we specify the HTML file that will be injected corresponding to route

- Example

```
1 <!-- about.html -->
2 <h2>About</h2>
3 <p>About Page Contents</p>
```

- we need to specify the container where content of each page will be loaded in our layout. In AngularJS, there is a `ng-view` directive that will injects the template file of the current route
- Add links to the HTML that will help in navigating to the configured pages

SPA – DEMO 1

```
!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script
>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-
route.js"></script>
<body ng-app="myApp">
<h1 style="color:red; background-color:yellow; font-size:50px;"> Single Page
Application without external HTML DOC </h1>
<p><a href="#/!">Main</a></p>
<a href="#!red">RED</a>
<a href="#!green">Green</a>
<p>Click on the links to change the content.</p>
<p>The HTML shown in the ng-view directive are written in the template property of
the $routeProvider.when method.</p>
<div ng-view></div>
```

SPA – DEMO 1

```
<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
  $routeProvider
    .when("/", {
      template : "<h1 style='background-color:lime'>Main</h1>"
    })
    .when("/red", {
      template : "<h1 style='background-color:red'>RED</h1>"
    })
    .when("/green", {
      template : "<h1 style='background-color:green'>GREEN</h1>"
    });
});
</script>
</body>
```

SPA DEMO 1 - OUPUT

Single Page Application without external HTML DOC

[Main](#)

[RED Green](#)

Click on the links to change the content.

The HTML shown in the ng-view directive are written in the template property of the \$routeProvider.when method.

Main

Single Page Application without external HTML DOC

[Main](#)

[RED Green](#)

Click on the links to change the content.

The HTML shown in the ng-view directive are written in the template property of the \$routeProvider.when method.

RED

Single Page Application without external HTML DOC

[Main](#)

[RED Green](#)

Click on the links to change the content.

The HTML shown in the ng-view directive are written in the template property of the \$routeProvider.when method.

GREEN

SPA demo 2

```
<!doctype html>
<html ng-app="myApp">
  <head>
    <script
      src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></s
    cript>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-
      route.js"></script>
  </head>
  <body>
    <script type="text/ng-template" id="home.html">
      <h1>Home</h1>
      <h3>{{message}}</h3>
    </script>
```

SPA demo 2

```
<script type="text/ng-template" id="blog.html">
  <h1>Blog</h1>
  <h3>{{message}}</h3>
</script>
<script type="text/ng-template" id="about.html">
  <h1>About</h1>
  <h3>{{message}}</h3>
</script>
<a href="#/">Home</a>
<a href="#/blog">Blog</a>
<a href="#/about">About</a>
<div ng-view></div>
<script>
  var app = angular.module('myApp', []);
  var app = angular.module('myApp', ['ngRoute']);
  app.config(function($routeProvider) {
    $routeProvider
```

SPA demo 2

```
.when('/', {
  templateUrl : 'home.html',
  controller : 'HomeController'
})
.when('/blog', {
  templateUrl : 'blog.html',
  controller : 'BlogController'
})
.when('/about', {
  templateUrl : 'about.html',
  controller : 'AboutController'
})
.otherwise({redirectTo: '/'});
});
app.controller('HomeController', function($scope) {
  $scope.message = 'Hello from HomeController';
});
app.controller('BlogController', function($scope) {
  $scope.message = 'Hello from BlogController';
});
app.controller('AboutController', function($scope) {
  $scope.message = 'Hello from AboutController';
});
</script> </body> </html>
```

[Home](#) [Blog](#) [About](#)

Home

Hello from HomeController

[Home](#) [Blog](#) [About](#)

Blog

Hello from BlogController

[Home](#) [Blog](#) [About](#)

About

Hello from AboutController

SPA DEMO 3 with external html doc

```
<!doctype html>
<html ng-app="myApp">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-route.js"></script>
  </head>
  <body>
    <a href="#/">Main</a>
    <a href="#/red">RED</a>
    <a href="#/green">GREEN</a>
    <a href="#/blue">BLUE</a>
    <div ng-view></div>
    <script>
var app = angular.module('myApp', []);
var app = angular.module('myApp', ['ngRoute']);
app.config(function($routeProvider) {
  $routeProvider
```

SPA DEMO 3 with external html doc

```
.when('/', {  
  templateUrl : 'main1.html',  
  controller : 'HomeController'  
})  
.when('/red', {  
  templateUrl : 'red.html',  
  controller : 'BlogController'  
})  
.when('/green', {  
  templateUrl : 'green.html',  
  controller : 'AboutController'  
})  
.when('/blue', {  
  templateUrl : 'blue.html',  
  controller : 'blueController'  
})  
.otherwise({redirectTo: '/'});  
});
```


SPA DEMO 3 with external html doc

```
app.controller('HomeController', function($scope) {  
    $scope.message = 'Main';  
});  
app.controller('BlogController', function($scope) {  
    $scope.message = 'RED';  
});  
app.controller('AboutController', function($scope) {  
    $scope.message = 'green';  
});  
app.controller('blueController', function($scope) {  
    $scope.message = 'blue';  
});  
    </script>  
    </body>  
</html>
```

External HTML WEB doc

- **Red.html**
- `<h1 style='background-color:red;'> {{message}}</h1>`
- **Green.html**
- `<h1 style='background-color:green;'> {{message}}</h1>`
- **Blue.html**
- `<h1 style='background-color:blue;'> {{message}}</h1>`
- **Main.html**
- `<h1 style='background-color:lime;'> {{message}}</h1>`

SPA demo 3 with external html doc

- This code run only in web server . You have to use any one of the web server



SPA demo 4

- Creation of shopping list
- Creation of list using ng-repeat directive
- Push method is used to add a new item in the list.
- Splice method is used to remove an item from the list.
- we will add a container for error messages, and write an error message when someone tries to add an existing item.

SPA demo 4

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<script>
var app = angular.module("myShoppingList", []);
app.controller("myCtrl", function($scope) {
    $scope.products = ["Milk", "Bread", "Cheese"];
    $scope.addItem = function () {
        $scope.errorText = "";
        if (!$scope.addMe) {return;}
        if ($scope.products.indexOf($scope.addMe) == -1) {
            $scope.products.push($scope.addMe);
        } else {
            $scope.errorText = "The item is already in your shopping list.";
        }
    }
}
```

SPA demo 4

```
$scope.removeItem = function (x) {  
    $scope.errortext = "";  
    $scope.products.splice(x, 1);  
}  
});  
</script>  
<div ng-app="myShoppingList" ng-controller="myCtrl">  
    <ul>  
        <li ng-repeat="x in products">{{x}}<span ng-click="removeItem($index)">x</span></li>  
    </ul>  
    <input ng-model="addMe">  
    <button ng-click="addItem()">Add</button>  
    <p>{{errortext}}</p>  
</div>  
<p>Try to add the same item twice, and you will get an error message.</p>  
</body>  
</html>
```

SPA demo 4

- Milk×
- Bread×
- Cheese×

Try to add the same item twice, and you will get an error message.

- Milk×
- Cheese×
- ghee×

Try to add the same item twice, and you will get an error message.