PMCA601L – FULL STACK WEB DEVELOPMENT

MODULE – 3 FRONT END FRAMEWORK ANGULAR JS

GETTING STARTED WITH ANGULAR – MODULES, DIRECTIVES, DATA BINDING, SERVICES, CREATING A BASIC ANGULAR APPLICATION – ANGULAR COMPONENTS – BUILDING TEMPLATE, USING EXTERNAL TEMPLATES – INJECTING DIRECTIVES – EXPRESSIONS – USING BASIC EXPRESSIONS – INTERACTING WITH THE COMPONENT CLASS IN EXPRESSIONS – BUILT-IN DIRECTIVES – EVENT AND CHANGE DETECTION – USING BROWSER EVENTS – EMITTING CUSTOM EVENTS – IMPLEMENTING ANGULAR SERVICES IN WEB APPLICATIONS – BUILT-IN SERVICE, HTTP SERVICE, ROUTER SERVICE.

Angular JS – Introduction

- Angular JS is an open source JavaScript framework by Google to build web applications.
- AngularJS is a **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 a JavaScript framework written in JavaScript.
- AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

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

- It is an excellent framework for building single phase applications and line of business applications.

Angular JS – Advantages

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.

Directives, filters, modules, routes etc.

Angular JS – Directives

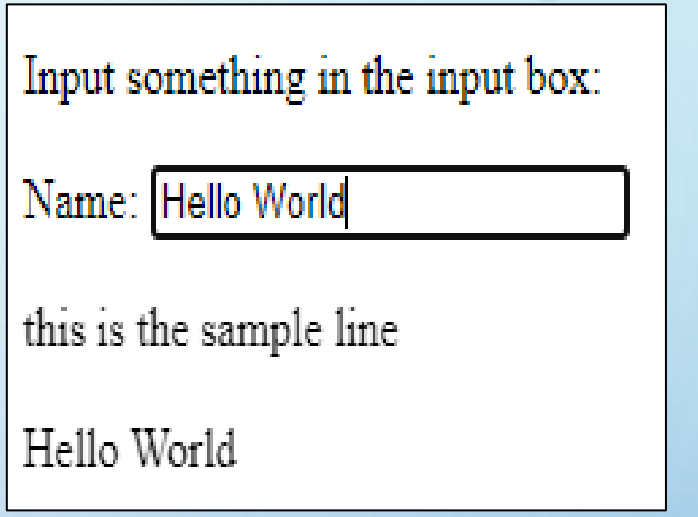
The AngularJS framework can be divided into three major parts –

- **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.
- **ng-init** → This directive initializes application data.
- **ng-repeat** → This directive repeats html elements for each item in a collection.

Directives list - https://www.javatpoint.com/angularjs-modules

Angular JS – Sample

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="name"></p>
this is the sample line<br>
<p ng-bind="name"></p>
</div> </body> </html>
```



The **ng-app** directive tells AngularJS that the element is the "owner" of an AngularJS **application.**

The **ng-model** directive binds the value of the input field to the application variable **name.**

The **ng-bind** directive binds the content of the <p> element to the application variable **name.**

Angular JS – MVC Model

- AngularJS **modules** define AngularJS applications.
- AngularJS **controllers** contr ol AngularJS applications.
- The **ng-app** directive defines the application, the **ng-controller** directive defines the controller.

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

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

Angular JS – MVC Model

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

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

Module Part

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

View Part

```
First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}
```

Controller Part

```
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
  $scope.firstName= "John";
  $scope.lastName= "Doe";
});
```

Angular JS – Scope Object

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.

Angular JS – Controller

A controller is defined using **ng-controller** directive. A controller is a JavaScript object containing attributes/properties and functions. Each **controller** accepts $**scope** as a parameter which refers to the application/module that controller is to control.

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

First Name: <input type="text" ng-model="firstName"><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 = "Aryan";
    $scope.lastName = "Khanna";
});
</script>
```

- Here, the AngularJS application runs inside the is defined by ng-app="myApp".
- The AngularJS directive is ng-controller="myCtrl" attribute.
- The myCtrl function is a JavaScript function.
- AngularJS will invoke the controller with a $scope object.
- In AngularJS, $scope is the application object (the owner of application variables and functions).
- The controller creates two properties (variables) in the scope (firstName and lastName).
- The ng-model directives bind the input fields to the controller properties (firstName and lastName).

Angular JS – Module

A module defines an application. It is a **container** for the **different parts** of your **application** like controller, services, filters, directives etc.
A module is used as a **Main()** method. Controller always belongs to a module.
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>
```

Here, "myApp" specifies an HTML element in which the application will run.

Angular JS – Controller and Module

To **add a controller** to your application refer to the controller with the **ng-controller** directive.

```
<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
    $scope.firstName = "Ajeet";
    $scope.lastName = "Maurya";
});
</script>
```

Angular JS – 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**.
- Examples: {{ 5 + 5 }} , {{ firstName + " " + lastName }}

Angular JS – Expression Example

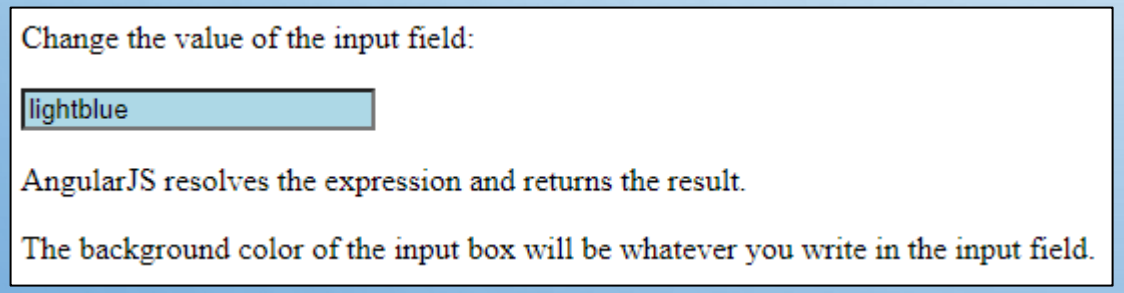
```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>

<div ng-app>
<p>My first expression: {{ 5 + 5 }}</p>
</div>

</body>
</html>
```

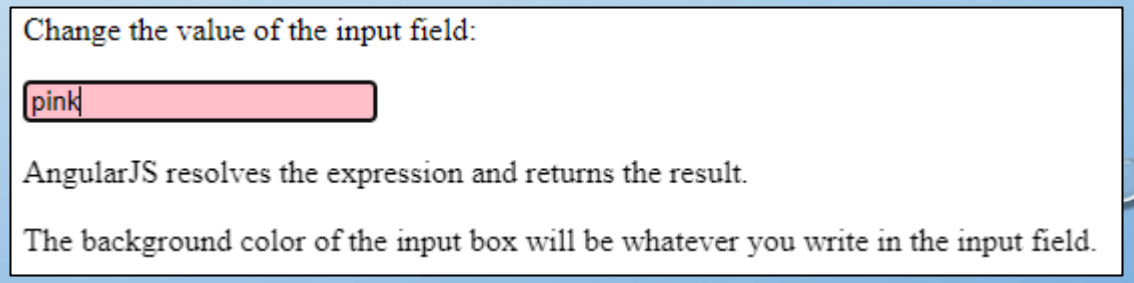
My first expression: 10

Angular JS – Expression Sample

A **web application** (web app) is an **application_program** that is stored on a

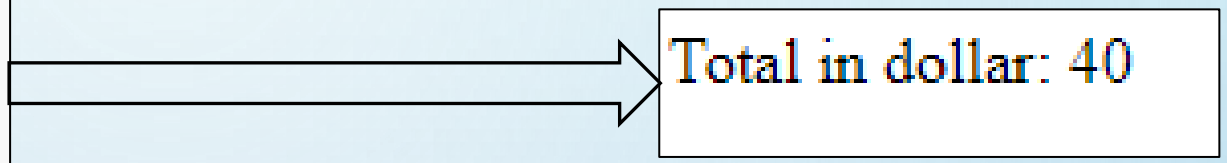
```
<p>Change the value of the input field:</p>

<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 will be whatever you write in the
input field.</p>
```



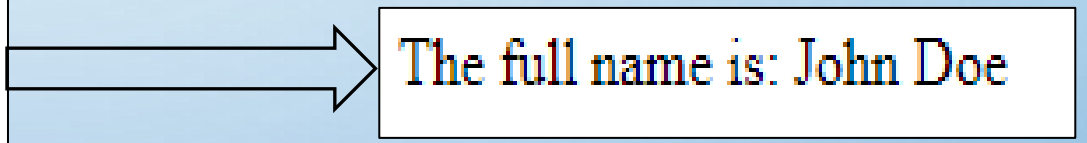


Angular JS – Expression Numbers and Strings

```
<div ng-app="" ng-init="quantity=8;cost=5">
<p>Total in dollar: {{ quantity * cost }}</p>
</div>
```

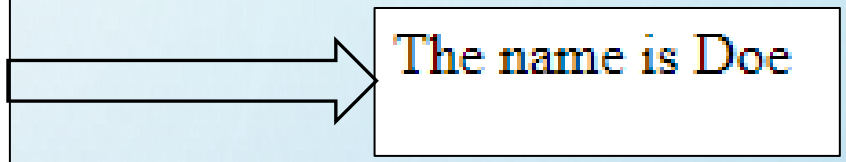



```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>The full name is: {{ firstName + " " + lastName }}</p>

</div>
```

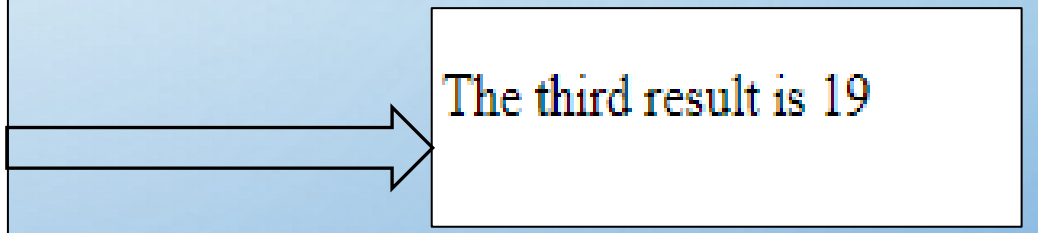


Angular JS – Expression Objects and Arrays

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>The name is {{ person.lastName }}</p>

</div>
```



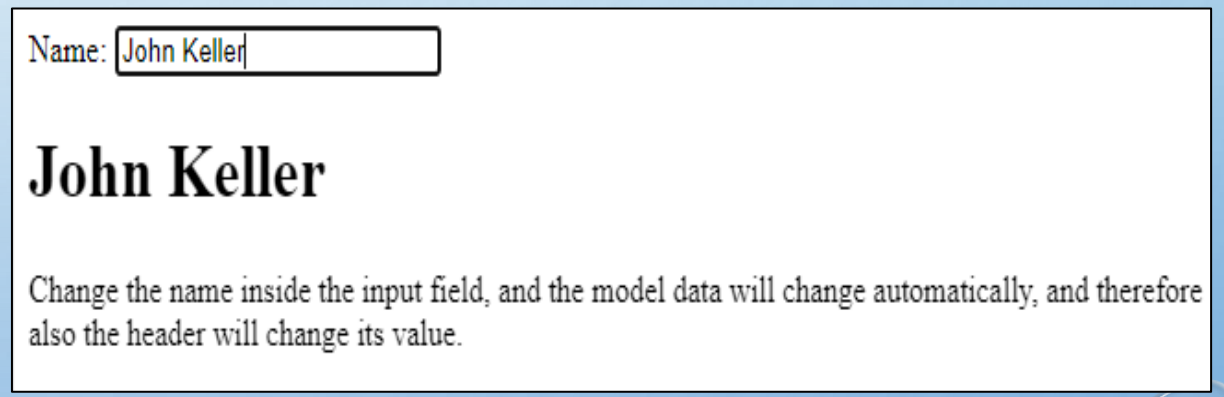

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>The third result is {{ points[2] }}</p>

</div>
```



Angular JS – Data Binding

- Data binding in AngularJS is the **synchronization** between the **model** and the **view**. The data model is a collection of data available for the application.
- When data in the *model* changes, the *view* reflects the change, and when data in the *view* changes, the *model* is updated as well. This happens immediately and automatically

```
<div ng-app="myApp" ng-controller="myCtrl">
    Name: <input ng-model="firstname">
    <h1>{{firstname}}</h1>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstname = "John";
 });
</script>
```



Angular JS – ng-repeat

ng-repeat directive repeats html elements for each item in a collection.

```
<div ng-app="" ng-init="names=['Jani','Hege','Kai']">
  <p>Looping with ng-repeat:</p>
  <ul>
    <li ng-repeat="x in names">
      {{ x }}
    </li>
  </ul>
</div>
```

Looping with ng-repeat:

- Jani
- Hege
- Kai

```
<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>
```

Looping with objects:

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

Angular JS – Filters

Filters are used to **format** data. Following is a list of filters used for transforming data.

Can add filters to expressions by using the ***pipe character*** **|**, followed by a filter.

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. "-" sign can be used to sort in reverse order
Uppercase	It formats a string to upper case.

Angular JS – Filters

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

<p>The name is {{ lastName | uppercase }}</p>

</div>

<script>
angular.module('myApp', []).controller('personCtrl', function($scope) {
    $scope.firstName = "John",
    $scope.lastName = "Doe"
});
</script>
```

The name is DOE

Angular JS – Filters

```
<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>
```

Looping with objects:

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

Angular JS – Filters

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

<ul>
<li ng-repeat="x in customers | orderBy : '-city'">
    {{x.name + ", " + x.city}}
</li>
</ul>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('orderCtrl', function($scope) {
    $scope.customers = [
        {"name" : "Bottom-Dollar Marketse" ,"city" : "Tsawassen"},
        {"name" : "Alfreds Futterkiste", "city" : "Berlin"},
        {"name" : "Bon app", "city" : "Marseille"},
        {"name" : "Cactus Comidas para llevar", "city" : "Buenos Aires"},
        {"name" : "Bolido Comidas preparadas", "city" : "Madrid"},
        {"name" : "Around the Horn", "city" : "London"},
        {"name" : "B's Beverages", "city" : "London"}
    ];
});
</script>

<p>Use the minus sign to sort descending (-city).</p>
```

- Bottom-Dollar Marketse, Tsawassen
- Bon app, Marseille
- Bolido Comidas preparadas, Madrid
- Around the Horn, London
- B's Beverages, London
- Cactus Comidas para llevar, Buenos Aires
- Alfreds Futterkiste, Berlin

Use the minus sign to sort descending (-city).

Angular JS – Filters

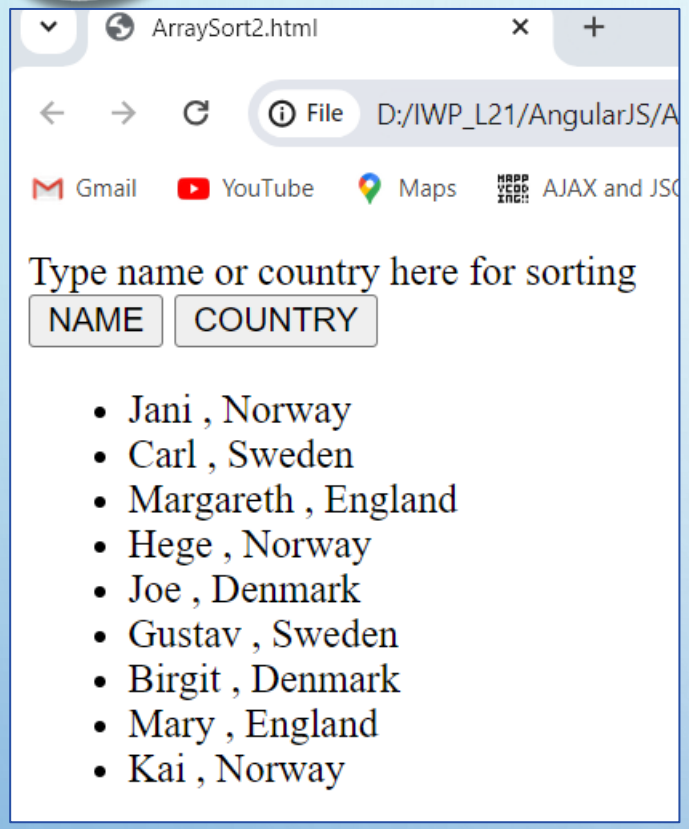
```
<div ng-app="myApp" ng-controller="orderCtrl">

<ul>
<li ng-repeat="x in customers | orderBy :['city','-name']">
    {{x.name + ", " + x.city}}
</li>
</ul>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('orderCtrl', function($scope) {
    $scope.customers = [
        {"name" : "Bottom-Dollar Marketse" ,"city" : "Tsawassen"},
        {"name" : "Alfreds Futterkiste", "city" : "Berlin"},
        {"name" : "Bon app", "city" : "Marseille"},
        {"name" : "Cactus Comidas para llevar", "city" : "Buenos Aires"},
        {"name" : "Bolido Comidas preparadas", "city" : "Madrid"},
        {"name" : "Around the Horn", "city" : "London"},
        {"name" : "B's Beverages", "city" : "London"}
    ];
});
</script>
```

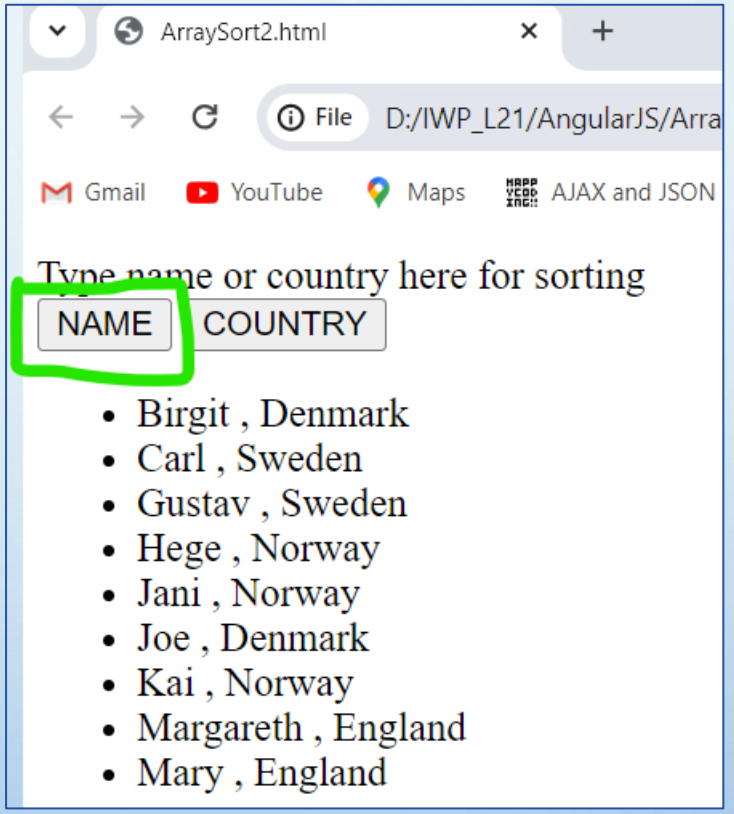
- Alfreds Futterkiste, Berlin
- Cactus Comidas para llevar, Buenos Aires
- B's Beverages, London
- Around the Horn, London
- Bolido Comidas preparadas, Madrid
- Bon app, Marseille
- Bottom-Dollar Marketse, Tsawassen

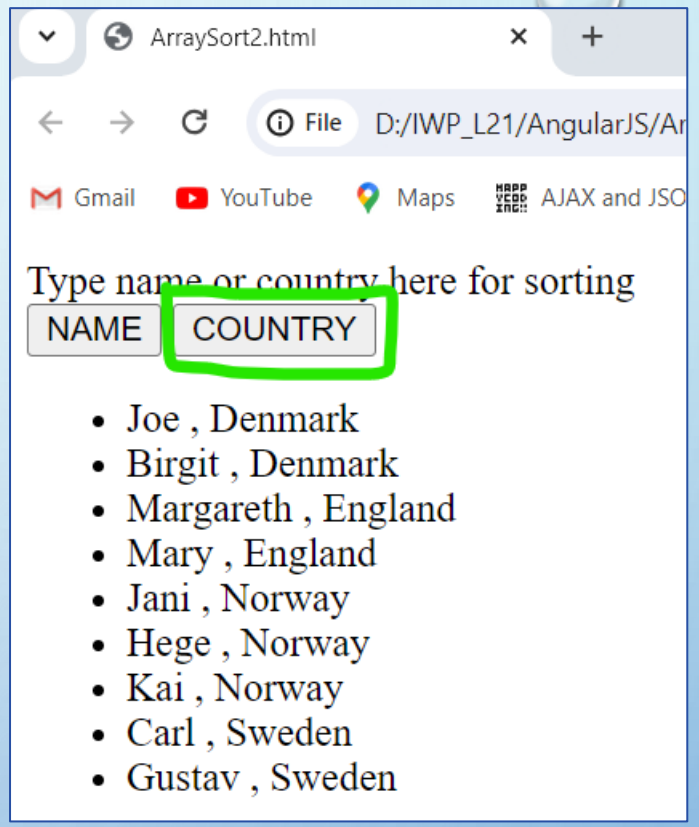
Angular JS – Filters

```
<div ng-app="myApp" ng-controller="namesCtrl">
Type name or country here for sorting
<button ng-click="sort('name')">NAME</button>
<button ng-click="sort('country')">COUNTRY</button>
<ul>
  <li ng-repeat ="x in names|orderBy:sortName">
  {{ 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'}
        ];
    $scope.sort = function(x){
      $scope.sortName=x;
    }
});
</script>
```

Angular JS – Filters



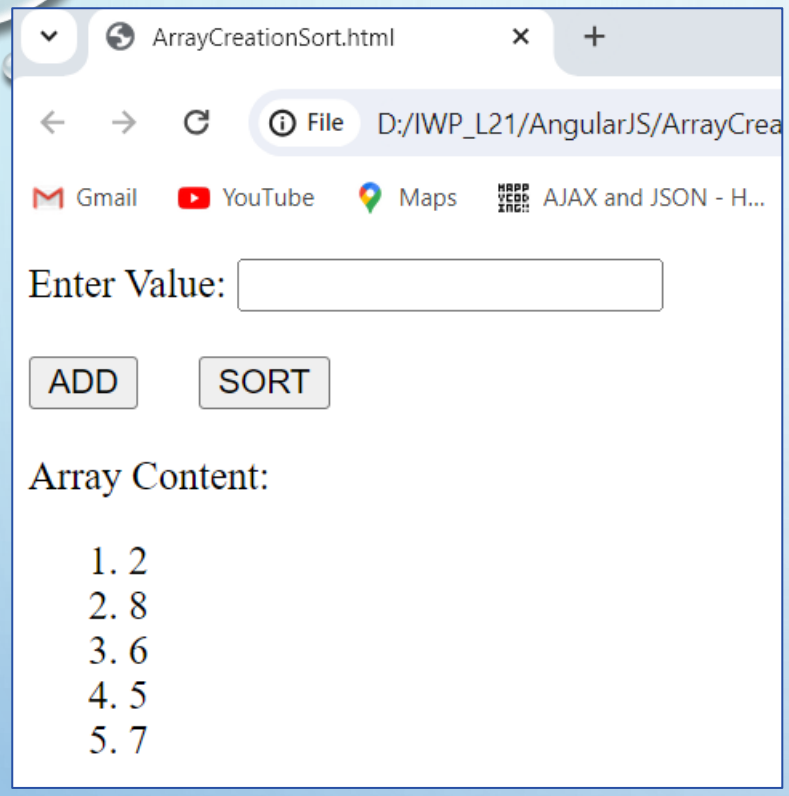


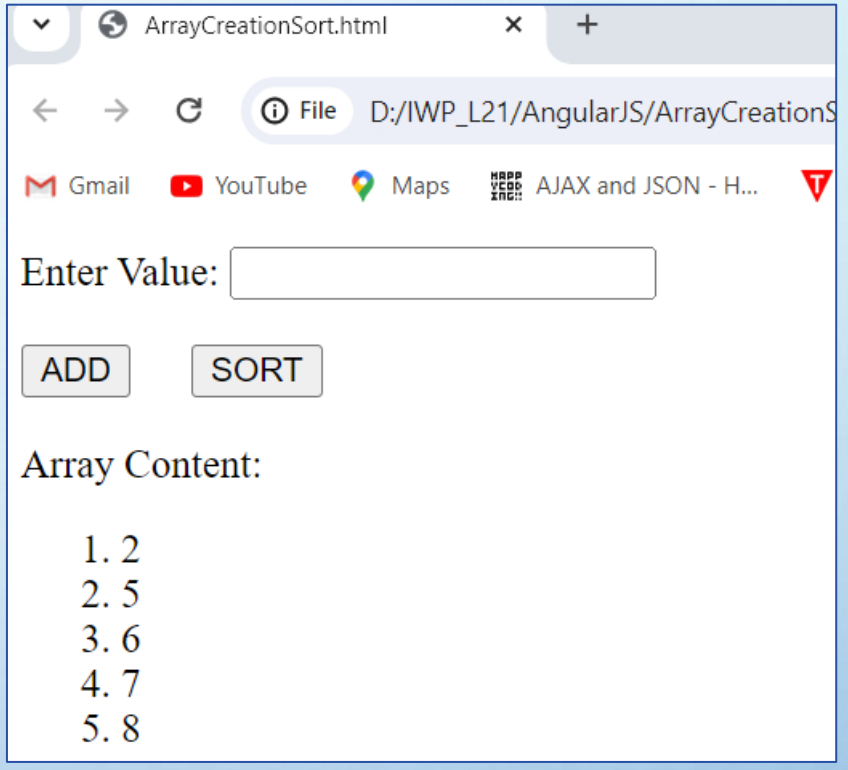


Angular JS – Dynamic Array Creation & Sort

```
<script>
var app = angular.module("arrayCreation",[]);
app.controller("arrCreation",function($scope){
   $scope.arr=[];
   $scope.add = function(){
      $scope.arr.push($scope.arrayValue);
      $scope.arrayValue=""; }
 $scope.sortArray = function(){
      $scope.arr.sort(); }
});
</script>
<div ng-app="arrayCreation" ng-controller ="arrCreation">
Enter Value: <input type=text ng-model="arrayValue"><br><br>
<button ng-click="add()">ADD</button>    
<button ng-click = "sortArray()"> SORT </button><br>
<br>Array Content:
<ol>
   <li ng-repeat="x in arr">{{ x }}
</ol>
</div>
```

Angular JS – Dynamic Array Creation & Sort





Angular JS – Events

A **web application** (web app) is an **application_program** that is stored on a

AngularJS has its own HTML events directives. Can add AngularJS event listeners to the HTML elements by using one or more of these directives:

- ng-blur
- ng-change
- ng-click
- ng-copy
- ng-cut
- ng-dblclick
- ng-focus
- ng-keydown
- ng-keypress

- ng-keyup
- ng-mousedown
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-mouseup
- ng-paste

Angular JS – Mouse Move Event

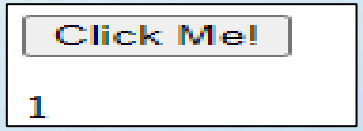
Increase the count variable when the mouse moves over the H1 element:

```
<body>

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

<h1 ng-mousemove="count = count + 1">Mouse Over Me!</h1>

<h2>{{ count }}</h2>

</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.count = 0;
});
</script>
```

Angular JS – Mouse Click

The **ng-click** directive defines AngularJS code that will be executed when the element is being clicked.

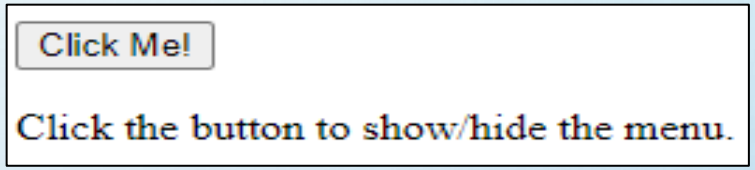
```
<body>

<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>
```

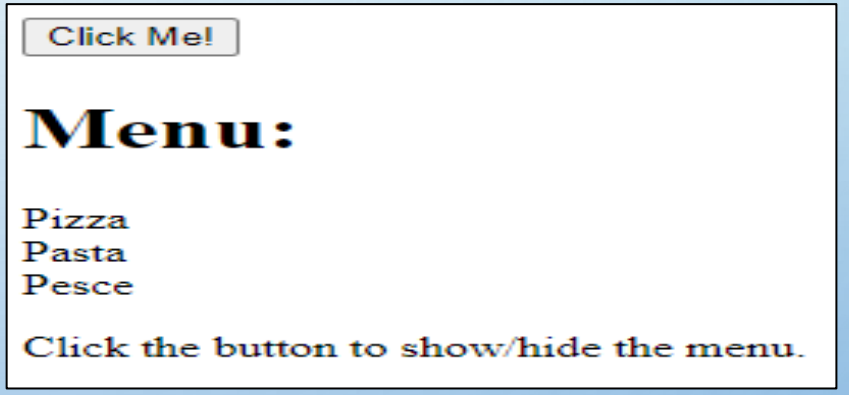


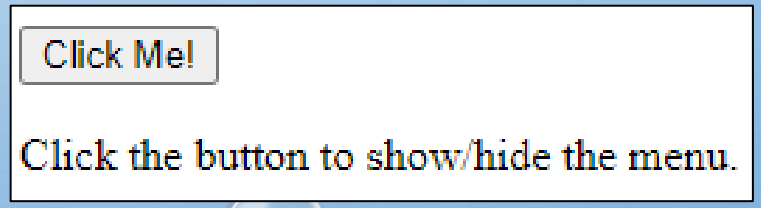
Angular JS – Toggle

If you want to show a section of HTML code when a button is clicked, and hide when the button is clicked again, like a dropdown menu, make the button behave like a toggle switch:

```
<div ng-app="myApp" ng-controller="myCtrl">
<button ng-click="myFunc()">Click Me!</button>

<div ng-show="showMe">
    <h1>Menu:</h1>
    <div>Pizza</div>
    <div>Pasta</div>
    <div>Pesce</div>
</div> </div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.showMe = false;
    $scope.myFunc = function() {
        $scope.showMe = !$scope.showMe;
    }
});
</script>

<p>Click the button to show/hide the menu.</p>
```







Angular JS – Forms

AngularJS facilitates you to create a form enriches with **data binding** and validation of input controls.

Input controls are ways for a user to enter data.

Following are the input controls used in AngularJS forms:

- input elements
- select elements
- button elements
- textarea elements

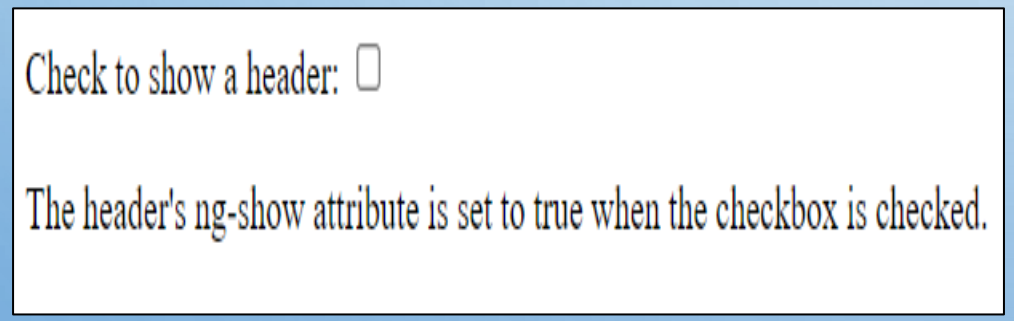
Following is a list of **events** supported in AngularJS:

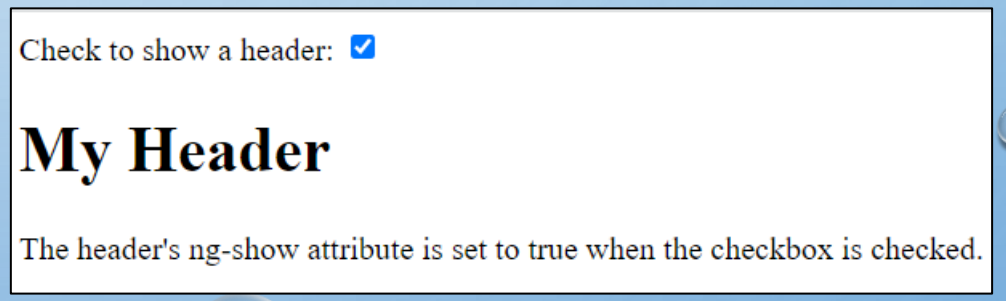
- ng-click
- ng-dbl-click
- ng-mousedown
- ng-mouseup
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-keydown
- ng-keyup
- ng-keypress
- ng-change

Angular JS – Form – Checkbox

A checkbox has a value true or false. The **ng-model** directive is used for a checkbox.

```
<div ng-app="">
  <form>
    Check to show a header:
    <input type="checkbox" ng-model="myVar">
  </form>
  <h1 ng-show="myVar">My Header</h1>
</div>

<p>The header's ng-show attribute is set to true when the checkbox is checked.</p>
```





Angular JS – Radio Button

Using **ng-switch directive to hide and show** HTML sections depending on the value of the radio buttons.

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

<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>
```

Pick a topic: ○ Dogs ○ Tutorials ○ Cars

The ng-switch directive hides and shows
HTML sections depending on the value of the radio buttons.

Pick a topic: ○ Dogs ◉ Tutorials ○ Cars

Tutorials

Learn from examples.

The ng-switch directive hides and shows
HTML sections depending on the value of the radio buttons.

Angular JS – Select

```
<form>
  Select a topic:
  <select ng-model="myVar">
    <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>
```

Select a topic:

The ng-switch directive hides and shows
HTML sections depending on the value of the dropdown list.

Select a topic: Dogs

Dogs

Welcome to a world of dogs.

The ng-switch directive hides and shows
HTML sections depending on the value of the dropdown list.

Angular JS – Form

```
<body>

<div ng-app="myApp" ng-controller="formCtrl">
  <form novalidate>
    First Name:<br>
    <input type="text" ng-model="user.firstName"><br>
    Last Name:<br>
    <input type="text" ng-model="user.lastName">
    <br><br>
    <button ng-click="reset()">RESET</button>
  </form>
  <p>form = {{user}}</p>
  <p>master = {{master}}</p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
    $scope.master = {firstName:"John", lastName:"Doe"};
    $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
    };
    $scope.reset();
});
</script>
```

First Name:
John
Last Name:
Doe

RESET

form = {"firstName":"John","lastName":"Doe"}

master = {"firstName":"John","lastName":"Doe"}

First Name:
John
Last Name:
Keller

RESET

form = {"firstName":"John","lastName":"Keller"}

master = {"firstName":"John","lastName":"Doe"}

Angular JS – Application Explanation

The **ng-app** directive defines the AngularJS application.
The **ng-controller** directive defines the application controller.
The **ng-model** directive binds two input elements to the **user** object in the model.
The **formCtrl** controller sets initial values to the **master** object, and defines the **reset()** method.
The **reset()** method sets the **user** object equal to the **master** object.
The **ng-click** directive invokes the **reset()** method, only if the button is clicked.
The **novalidate** attribute is not needed for this application, but normally you will use it in AngularJS forms, to override standard HTML5 validation.

Angular JS – Form Validation – email

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

<p>Try writing an E-mail address in the input field:</p>

<form name="myForm">
<input type="email" name="myInput" ng-model="myInput">
</form>

<p>The input's valid state is:</p>
<h1>{{myForm.myInput.$valid}}</h1>
<p>Note that the state of the input field is "true" before you start writing in it, even
if it does not contain an e-mail address.</p>

</body>
```

Try writing an E-mail address in the input field:

The input's valid state is:

true

Note that the state of the input field
is "true" before you start writing in it, even if it does not contain an e-mail address.

Try writing an E-mail address in the input field:

xyz

The input's valid state is:

false

Angular JS – Form Validation

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

<p>Try leaving the first input field blank:</p>

<form name="myForm">
<p>Name:
<input name="myName" ng-model="myName" required>
<span ng-show="myForm.myName.$touched && myForm.myName.$invalid">The name is required.
</span>
</p>

<p>Address:
<input name="myAddress" ng-model="myAddress" required>
</p>

</form>
```

Try leaving the first input field blank:

Name:

Address:

Try leaving the first input field blank:

Name: The name is required.

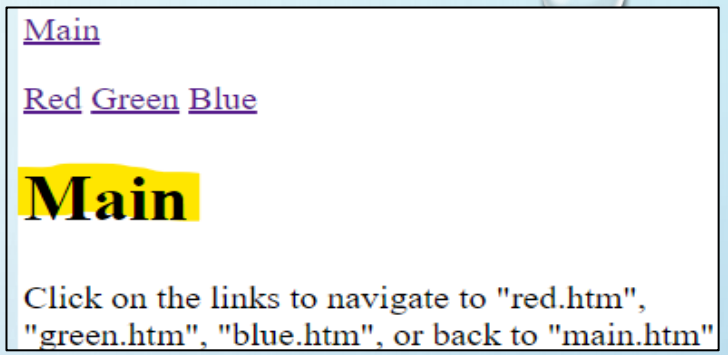
Address: asass

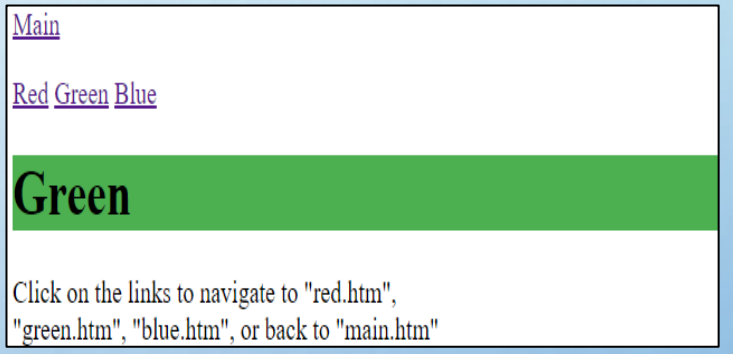
Angular JS – Routing

- To navigate to different pages in the application, but also want the application to be a SPA (Single Page Application), with no page reloading, can use the ngRoute module.

  The **ngRoute** module routes the application to **different pages without reloading** the entire application.

```
<body ng-app="myApp">
<p><a href="#/!">Main</a></p>
<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", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.htm"
    })
    .when("/red", {
        templateUrl : "red.htm"
    })
    .when("/green", {
        templateUrl : "green.htm"
    })
    .when("/blue", {
        templateUrl : "blue.htm"
    });
});
</script>

<p>Click on the links to navigate to "red.htm",<br>
"green.htm", "blue.htm", or back to "main.htm"</p>
</body>
```





Angular JS – Routing

Steps involved in Routing:

1. To make the applications ready for routing, must include the AngularJS Route module

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

2. Must add the ngRoute as a dependency in the application module

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

3. Now, the application has access to the route module, which provides the $routeProvider. Use the $routeProvider to configure different routes in your application:

```
app.config(function($routeProvider) {
  $routeProvider
  .when("/", {
    templateUrl : "main.htm"
  })
  .when("/red", {
    templateUrl : "red.htm"
  })
  .when("/green", {
    templateUrl : "green.htm"
  })
  .when("/blue", {
    templateUrl : "blue.htm"
  });
});
```

Angular JS – Routing

Steps Involved in Routing:

The application needs a container to put the content provided by the routing. This container is the ng-view directive.

There are three different ways to include the ng-view directive in your application:

1
```
<div ng-view></div>
```

2
```
<ng-view></ng-view>
```

3
```
<div class="ng-view"></div>
```

Applications can only have **one ng-view directive**, and this will be the placeholder for all views provided by the route.

Angular JS – Routing

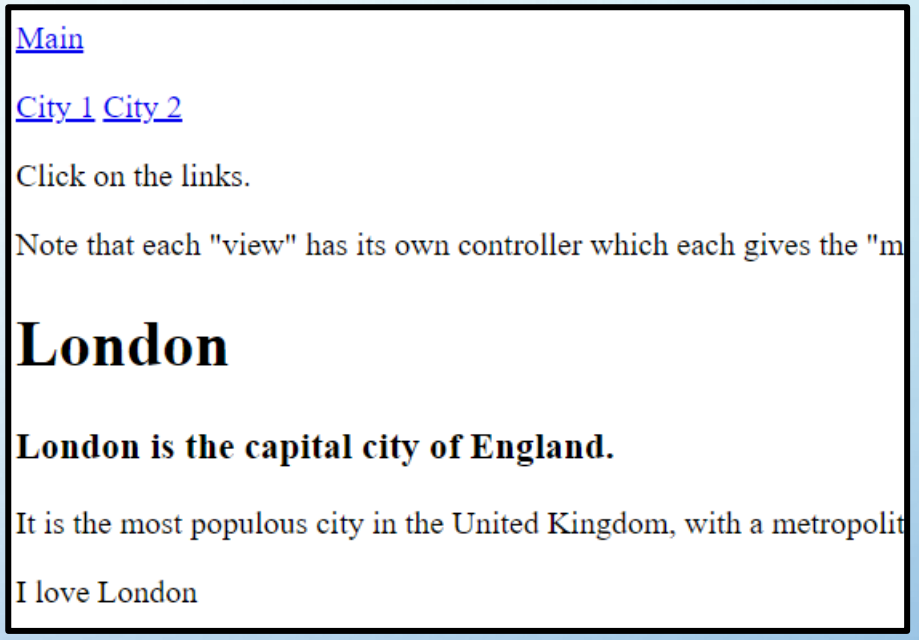
With the **$routeProvider** you can define what page to display when a user clicks a link.

```
<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.htm"
    })
    .when("/red", {
        templateUrl : "red.htm"
    })
    .when("/green", {
        templateUrl : "green.htm"
    })
    .when("/blue", {
        templateUrl : "blue.htm"
    });
});
```

Angular JS – Routing

App controller for each view

```
<body ng-app="myApp">
<p><a href="#/!">Main</a></p>
<a href="#!london">City 1</a>
<a href="#!paris">City 2</a>
<p>Click on the links.</p>
<p>Note that each "view" has its own controller
which each gives the "msg" variable a value.</p>

<div ng-view></div>
<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.htm",
    })
    .when("/london", {
        templateUrl : "london.htm",
        controller : "londonCtrl"
    })
    .when("/paris", {
        templateUrl : "paris.htm",
        controller : "parisCtrl"
    });
});
app.controller("londonCtrl", function ($scope) {
    $scope.msg = "I love London";
});
app.controller("parisCtrl", function ($scope) {
    $scope.msg = "I love Paris";
}); </script>
```



Angular JS – Template Property

In the previous examples, **templateUrl** property have used in the $**routeProvider.when** method.

Can also use the **template** property, which allows to write HTML directly in the property value, and not refer to a page.

```
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
  $routeProvider
  .when("/", {
    template : "<h1>Main</h1><p>Click on the links to change this content</p>"
  })
  .when("/banana", {
    template : "<h1>Banana</h1><p>Bananas contain around 75% water.</p>"
  })
  .when("/tomato", {
    template : "<h1>Tomato</h1><p>Tomatoes contain around 95% water.</p>"
  });
});
```

Angular JS – Otherwise

Otherwise method is the default route when none of the others get a match.

```
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
  $routeProvider
  .when("/banana", {
    template : "<h1>Banana</h1><p>Bananas contain around 75% water.</p>"
  })
  .when("/tomato", {
    template : "<h1>Tomato</h1><p>Tomatoes contain around 95% water.</p>"
  })
  .otherwise({
    template : "<h1>None</h1><p>Nothing has been selected</p>"
  });
});
```

Angular JS – Application

```
<script>
var app = angular.module("myShoppingList", []);
app.controller("myCtrl", function($scope) {
    $scope.products = ["Milk", "Bread", "Cheese"];
    $scope.addItem = function () {
        $scope.products.push($scope.addMe);
    }
    $scope.removeItem = function (x) {
        $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>
</div>

<p>Click the little x to remove an item from the shopping list.</p>
```

- Milk×
- Bread×
- Cheese×
- Apple×

Apple Add

Click the little x to remove an item from the shopping list.

- Milk×
- Bread×
- Apple×

Apple Add

Click the little x to remove an item from the shopping list.

References

1. www.javatpoint.com
2. www.tutorialspoint.com
3. www.w3schools.com
4. https://reintech.io/blog/how-to-set-up-an-angular-development-environment
5. https://docs.angularjs.org/guide/component