Name: Manish Sanjay Bagul

Deccan Education Society's

Navinchandra Mehta Institute of Technology and Development

Class: FYMCA

Div : A

Roll No: C21008

Subject: Web Technologies

Deccan Education Society's

Navinchandra Mehta Institute of Technology and Development

C E R T I F I C A T E

This is to certify that Mr. / Miss. **Bagul Manish Sanjay** of M.C.A. Semester I with Roll No. **C21008** has completed. Practicals of **Web Technologies** under my supervision in this college during the year 2021-2022.

CO	R1 (Attendance)	R2 (Performance during lab session)	R3 (Innovation in problem-solving technique)	R4 (Mock Viva)	R5 (Variation in implementation of learned topics on projects)
CO1					
CO2					
CO3					
CO4					

Practical-in-charge

Head of Department
MCA Department
(NMITD)

Sr.No	Name of Practical	Date	Sign
1	Create an application to demonstrate Node.js Modules	08-02-2022	
2	Create an application to demonstrate various Node.js Events	09-02-2022	
3	Create an application to demonstrate Node.js Functions	16-02-2022	
4	Using File Handling demonstrate all basic file operations (Create, write, read, delete)	16-02-2022	
5	Create an HTTP Server and perform operations on it	22-02-2022	
6	Create an application to establish a connection with the MySQL database and perform basic database operations on it	22-02-2022	
7	Create an application using Filters	02-03-2022	
8	Create an application to demonstrate directives	02-03-2022	
9	Demonstrate controllers in Angular.js through an application	15-03-2022	
10	Demonstrate features of Angular.js forms with a program	15-03-2022	
11	Create a SPA (Single Page Application)	16-03-2022	

1. Create an application to demonstrate Node.js modules.

Solution :-

Module is set of functions that we wants to include in our application.
__dirname : The directory name of the current module.
__filename : The file name of the current module.
require.main : The module object representing the entry script loaded when the Node.js process launched.

JS nodejsModule1.js

```
1   console.log(__dirname);
2   console.log(__filename);
3   console.log(require.main);
4
```

Output :-

```
PS E:\Web Technologies\C21008> node nodejsModule1.js
E:\Web Technologies\C21008
E:\Web Technologies\C21008\nodejsModule1.js
Module {
  id: '.',
  path: 'E:\\Web Technologies\\C21008',
  exports: {},
  filename: 'E:\\Web Technologies\\C21008\\nodejsModule1.js',
  loaded: false,
  children: [],
  paths: [
    'E:\\Web Technologies\\C21008\\node_modules',
    'E:\\Web Technologies\\node_modules',
    'E:\\node_modules'
  ]
}
```

HTTP module :-

```
JS nodejsModule.js > ...
1   var http = require('http');
2   http.createServer(function (req, res) {
3     res.writeHead(200, {'Content-Type': 'text/html'});
4     res.write('Web Technology!');
5     res.end();
6   }).listen(3000);
7
```

Output :

Date module :

```
JS datemodule.js > ...
1   var dt = require('./dateFunction');
2   console.log(dt.myDatefunc());
```

```
JS dateFunction.js > myDatefunc > myDatefunc
1   exports.myDatefunc = function()
2   {
3       return Date();
4   }
```

Output :

```
PS E:\Web Technologies\C21008> node datemodule.js
Mon Feb 28 2022 17:14:13 GMT+0530 (India Standard Time)
PS E:\Web Technologies\C21008>
```

---

2. Create an application to demonstrate various Node.js events.

Solution :

Node.js Events :
In Node.js application, Events and Callbacks concepts are used to provide concurrency. As Node.js applications are single threaded and every API of Node.js are asynchronous, so its uses functions to maintain concurrency. Node uses observer pattern. Node thread keeps an event loop and after the completion of any task, it fires the corresponding event which signals the event listener function to get executed.

Event: 'newListener':
The EventEmitter instance will emit its own 'newListener' event before a listener is added to its internal array of listeners.
Listeners registered for the 'newListener' event are passed the event name and a reference to the listener being added.
The fact that the event is triggered before adding the listener has a subtle but important side effect: any additional listeners registered to the same name within the 'newListener' callback are inserted before the listener that is in the process of being added.

EventEmitter Class :
EventEmitter class lies in the events module. It is accessible via the following code −

```
// Import events module
var events = require('events');
// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance faces any error, it emits an 'error' event. When a new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired. EventEmitter provides multiple properties like on and emit. on property is used to bind a function with the event and emit is used to fire an event.

```
JS nodejsEvent.js > ...
1   const events = require("events");
2   const eventEmitter = new events.EventEmitter();
3   eventEmitter.on("connection", handleConnectionEvent);
4   eventEmitter.emit("connection");
5   eventEmitter.emit("connection");
6   eventEmitter.emit("connection");
7   eventEmitter.emit("connection");
8   function handleConnectionEvent() {
9       console.log("Connection Made !")
10  }
11  console.log("End of the program")
```

Output :

```
PS E:\Web Technologies\C21008> node nodejsEvent.js
Connection Made !
Connection Made !
Connection Made !
Connection Made !
End of the program
PS E:\Web Technologies\C21008>
```

Code :

```
JS nodejsEvent1.js > ...
1   const EventEmitter = require('events');
2   var eventEmitter = new EventEmitter();
3   eventEmitter.on('myEvent', (msg) => {
4       console.log(msg);
5   })
6   eventEmitter.emit('myEvent', "First event");
```

Output :

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\Web Technologies\C21008> node nodejsEvent1.js
First event
PS E:\Web Technologies\C21008>
```

Code :

JS nodejsEvent2.js

```
const events = require("events");
const eventEmitter = new events.EventEmitter();
function listener1() {
    console.log("Event received by Listener 1");
}
function listener2() {
    console.log("Event received by Listener 2");
}
eventEmitter.addListener("write", listener1);
eventEmitter.on("write", listener2);
eventEmitter.emit("write");
console.log(eventEmitter.listenerCount("write"));
eventEmitter.removeListener("write", listener1);
console.log("Listener 1 is removed");
eventEmitter.emit("write");
console.log(eventEmitter.listenerCount("write"));
```

Output :

```
PS E:\Web Technologies\C21008> node nodejsEvent2.js
Event received by Listener 1
Event received by Listener 2
2
Listener 1 is removed
Event received by Listener 2
1
PS E:\Web Technologies\C21008>
```

---

3 . Create an application to demonstrate Node.js functions.

Solution :

Set timer :

settimer.js

```
const message = function() {
    console.log("Hello, I am Manish");
}
setTimeout(message, 3000);

setTimeout( () => {
    console.log("Hello, Manish here");
}, 3000);
```

Output :

```
PS E:\Web Technologies\C21008> node settimer.js
Hello, I am Manish
Hello, Manish here
PS E:\Web Technologies\C21008>
```

Callback function :
Node. js, being an asynchronous platform, doesn't wait around for things like file I/O to finish - Node. js uses callbacks. A callback is a function called at the completion of a given task; this prevents any blocking, and allows other code to be run in the meantime.

Callback function Arrow function :
Using arrow function as callback function can reduce lines of code.
The default syntax for arrow function is : ( ) => { }

This can be used as callbacks.

Code :

```
JS callback.js > ...
1   function displayResult(x) {
2       console.log(x);
3   }
4   function calculate(x, y, mycallback) {
5       let sum = x + y;
6       mycallback(sum);
7   }
8   calculate(25, 30, displayResult)
9
```

Output :

```
PS E:\Web Technologies\C21008> node callback.js
55
PS E:\Web Technologies\C21008>
```

---

4. Using file handling demonstrate all basic file operations (create , write , read, delete)

Solution :

Node.js File system :

The Node.js file system module allows you to work with the file system on our computer. To include the File System module, use the require() method: var fs = require('fs');

Create file :

```
createFile.js > ...
1   var fs = require('fs');
2   fs.writeFile('file1.txt', 'In this way, you can create file.', function (err) {
3       if (err)
4           console.log(err);
5       else
6           console.log("File has been created !!");
7   });
```

Output :

```
file1.txt
1   In this way, you can create file.
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\Web Technologies\C21008> node createFile.js
File has been created !!
PS E:\Web Technologies\C21008>
```

Write file :

Use fs.writFile() method to write data to a file. If file already exists then it overwrites the existing content otherwise it creates a new file and writes data into it.

Code :

writeFile.js > ...

```
1   var fs = require('fs');
2   fs.writeFile('file2.txt', 'Web Technology !!', function (err) {
3       if (err)
4           console.log(err);
5       else
6           console.log("Write oparation completed !!");
7   });
```

Output :

file2.txt

```
1   Web Technology !!
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\Web Technologies\C21008> node writeFile.js
Write oparation completed !!
PS E:\Web Technologies\C21008>
```

Read file :
Use fs.readFile() method to read the physical file asynchronously.

Code :

```
readFile.js > ...
1   var fs = require('fs');
2   var data = fs.readFileSync('readDemo.txt', 'utf-8');
3   console.log(data);
```

Output :

```
PS E:\Web Technologies\C21008> node readFile.js
My name is Manish, I like to do coding.
PS E:\Web Technologies\C21008>
```

Delete file :
Use fs.unlink() method to delete an existing file.

Code :

```
deleteFile.js > ...
1   var fs = require('fs');
2   fs.unlink('readDemo.txt', function() {
3       console.log("File deleted successfully !!");
4   });
```

Output :

```
PS E:\Web Technologies\C21008> node deleteFile.js
File deleted successfully !!
PS E:\Web Technologies\C21008>
```

---

5. Create an HTTP server and perform operations on it.

Solution :

Node.js HTTP module : Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).
To include the HTTP module:

Var http = require('http');

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client. Use the createServer() method to create an HTTP server.

Code :

Server.js > ...

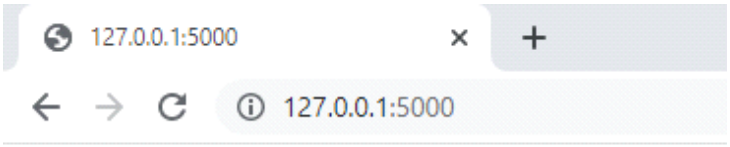
```
1  var http = require('http');
2  var server = http.createServer(function(req, res)
3  {
4      res.write("Web Technology !!");
5      res.end();
6  });
7  server.listen(5000);
8  console.log('Node.js web server at port 5000 is running....');
```

Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\Web Technologies\C21008> node Server.js
Node.js web server at port 5000 is running....
```

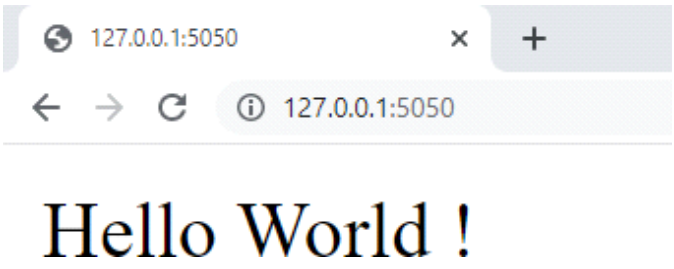


Web Technology !!

Code :

```
JS Server2.js > ...
1   var http = require('http');
2   http.createServer(function(req, res) {
3       res.writeHead(200, {'Content-Type' : 'text/html'});
4       res.write('Hello World !');
5       res.end();
6   }).listen(5050);
```

Output :

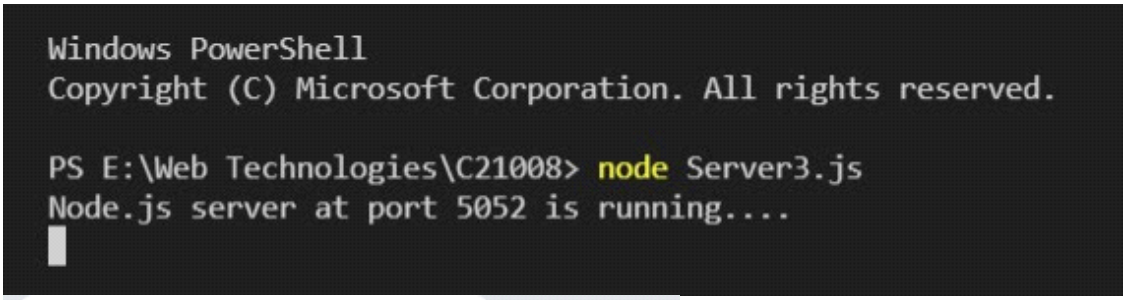


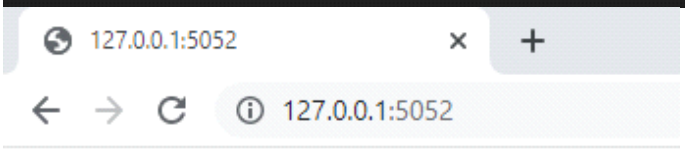
Code :

Server3.js > ...

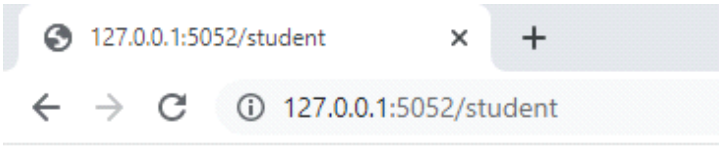
```
var http = require('http');
var server = http.createServer(function(req, res) {
    if (req.url == '/') {
        res.writeHead(200, {'Content-Type' : 'text/html'});
        res.write('<html><body><p>This is Home page.</p></body></html>');
        res.end();
    }
    else if(req.url =='/student') {
        res.writeHead(200, {'Content-Type' : 'text/html'});
        res.write('<html><body><p>This is Student page.</p></body></html>')
    }
    else if(req.url =='/admin') {
        res.writeHead(200, {'Content-Type' : 'text/html'});
        res.write('<html><body><p>This is Admin page.</p></body></html>')
    }
    else
        res.end('Invalid request !');
});
server.listen(5052);
console.log('Node.js server at port 5052 is running....')
```

Output :

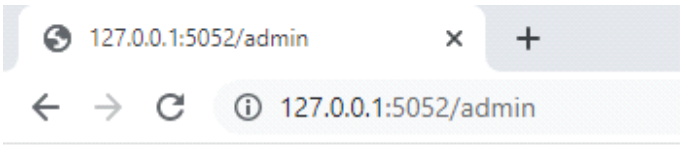




This is Home page.



This is Student page.



This is Admin page.

---

6. Create an application to establish a connection with the MySQL database and perform basic database operations on it.

Solution :

MySQL Database

To be able to experiment with the code examples, you should have MySQL installed on your computer.

You can download a free MySQL database at https://www.mysql.com/downloads/.

Install MySQL Driver

Once you have MySQL up and running on your computer, you can access it by using Node.js.

To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "mysql" module, downloaded from NPM.

To download and install the "mysql" module, open the Command Terminal and execute the following:

C:\Users\Your Name>npm install mysql

Node.js can use this module to manipulate the MySQL database:

var mysql = require('mysql');

- **Create Connection**
  Start by creating a connection to the database.
  Use the username and password from your MySQL database.

demo_db_connectioin.js

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\Web Technologies\C21008\NODE-sql> node demo_db_connection.js
Connected !!
PS E:\Web Technologies\C21008\NODE-sql>
```

Creating a Database

To create a database in MySQL, use the "CREATE DATABASE" statement

create_db.js

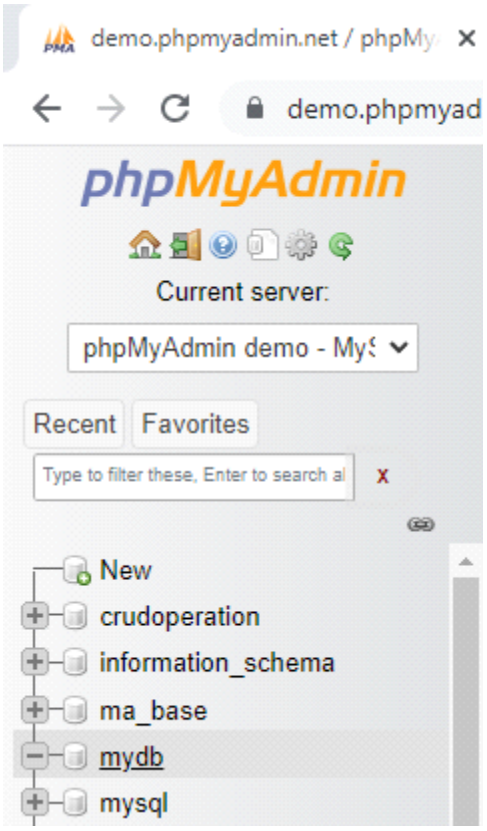
```
1  var mysql = require('mysql');
2
3  var con = mysql.createConnection({
4    host: "localhost",
5    user: "root",
6    password: " "
7  });
8
9  con.connect(function(err) {
10   con.query("CREATE DATABASE mydb", function (err, result) {
11     if (err) throw err;
12     console.log("Database created");
13   });
14 });
```

Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\Web Technologies\C21008\NODE-sql> node create_db.js
Database Created !!
PS E:\Web Technologies\C21008\NODE-sql>
```



Creating a Table

To create a table in MySQL, use the "CREATE TABLE" statement.

Make sure you define the name of the database when you create the connection

```
create_table.js  create_db.js
create_table.js > ...
1   var mysql = require('mysql');
2
3   var con = mysql.createConnection({
4     host: "localhost",
5     user: "yourusername",
6     password: "yourpassword",
7     database: "mydb"
8   });
9
10  con.connect(function(err) {
11    var sql = "CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))";
12    con.query(sql, function (err, result) {
13      if (err) throw err;
14      console.log("Table created");
15    });
16  });
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\Web Technologies\C21008\NODE-sql> node create_table.js
Table created
PS E:\Web Technologies\C21008\NODE-sql>
```

Table structure Relation view

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	Name	varchar(255)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	2	Address	varchar(255)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More

Insert Into Table

To fill a table in MySQL, use the "INSERT INTO" statement.

insert_record.js

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  var sql = "INSERT INTO customers (name, address) VALUES ('Manish', 'Ambarnath')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
});
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\Web Technologies\C21008\NODE-sql> node insert_record.js
1 record inserted
PS E:\Web Technologies\C21008\NODE-sql>
```

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

```
SELECT * FROM `customers`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Name	Address
Manish	Ambarnath
Manish	Ambarnath

---

7. Create an application using Filters

Solution :

Filters : AngularJS Filters allow us to format the data to display on UI without changing original format.

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

AngularJS provides following filters :

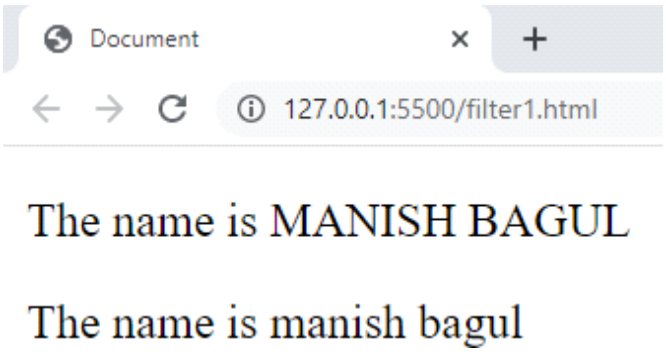
- currency - Format a number to a currency format.
- date - Format a date to a specified format.
- filter - Select a subset of items from an array.
- json - Format an object to a JSON string.
- limitTo - Limits an array/string, into a specified number of elements/characters.
- lowercase - Format a string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

Uppercase and lowercase filter :

filter1.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src = "angular-min.js"></script>
</head>
<body>
    <div ng-app = "myApp" ng-controller = "uppercaseCtrl">
        <p>The name is {{firstName+ ' ' + lastName | uppercase}}</p>
        <p>The name is {{firstName+ ' ' + lastName | lowercase}}</p>
    </div>
    <script>
        angular.module('myApp',[]).controller('uppercaseCtrl',function($scope) {
            $scope.firstName = 'Manish',
            $scope.lastName = 'Bagul'
        });
    </script>
</body>
</html>
```

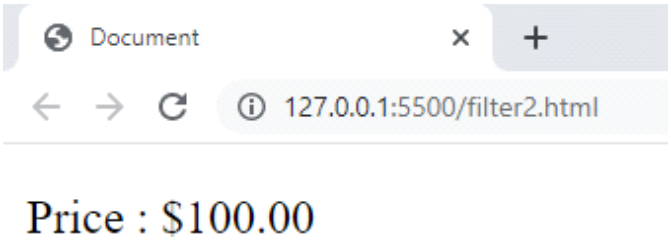
Output :



AngularJS currency filter :

```
filter2.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src = "angular-min.js"></script>
</head>
<body ng-app = "myApp" ng-controller="currencyCtrl">
    <p>Price : {{price | currency}}</p>
</body>
<script>
    angular.module("myApp",[]).controller("currencyCtrl", function($scope) {
        $scope.price = 100;
    });
</script>
</html>
```

Output :

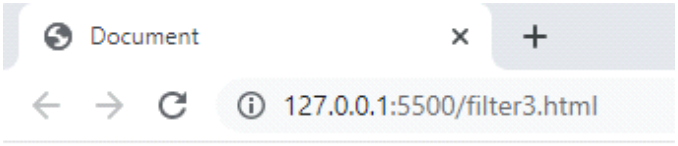


OrderBy filter

filter3.html > ...

```
<body>
    <div ng-app = "myApp" ng-controller = "namesCtrl">
        <p>Looping with objects...</p>
        <ul>
            <li ng-repeat = " x in name | orderBy: 'name'">
                {{ x.name + ',  ' + x.country }}
            </li>
        </ul>
    </div>
    <script>
        angular.module('myApp', []).controller('namesCtrl', function($scope) {
            $scope.name = [
                {name: 'Manish',country: 'India'},
                {name: 'Jay',country: 'Australia'},
                {name: 'Pushkar',country: 'Japan'},
                {name: 'Rohit',country: 'USA'}
            ]
        })
    </script>
</body>
```

Code :



Looping with objects...

- Jay, Australia
- Manish, India
- Pushkar, Japan
- Rohit, USA

AngularJS filter Filter

filter4.html > ...

```
<!DOCTYPE html>
<html>
<script src="angular-min.js"></script>
<body>
<div ng-app="myApp" ng-controller="namesCtrl">
<ul>
  <li ng-repeat="x in names | filter : 'h'">
    {{ x }}
  </li>
</ul>
</div>
<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = ['Manish', 'Utkarsh', 'Pushkar', 'Rohit', 'Jay', 'Virat',
                    'Sushil', 'Omkar', 'Ajay'];
});
</script>
<p>This example displays only the names containing the letter "h".</p>
</body>
</html>
```

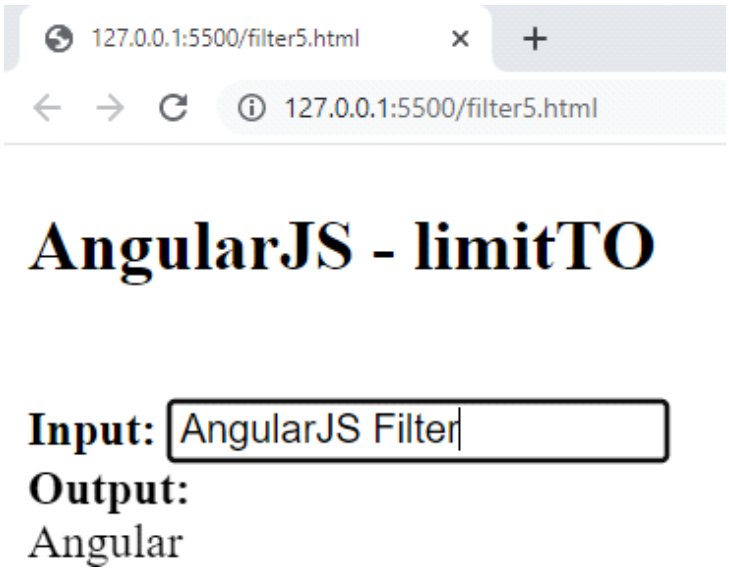
Output :

limitTo Filter

filter5.html > ...

```
<!DOCTYPE html>
<html>
<script src="angular-min.js"></script>
<body>
    <h2>AngularJS - limitTO</h2>
    <br>
    <div ng-app="myApp" ng-controller="myCtrl">
        <strong>Input:</strong>
        <input type="text" ng-model="string">
        <br>
        <strong>Output:</strong>
        <br>
        {{string|limitTo:7}}
    </div>
    <script>
        var app = angular.module('myApp', []);
        app.controller('myCtrl', function($scope) {
            $scope.string = "";
        });
    </script>
</body>
</html>
```

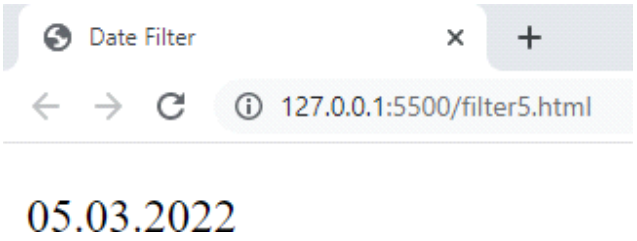
Output :



Date Filter :

```
<> filter5.html > ...
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <title>Date Filter</title>
5   <script src="angular-min.js"></script>
6   </head>
7   <body>
8       <div ng-app="myApp" ng-controller="dateCntrl">
9           <p>{{ today | date : "dd.MM.y" }}</p>
10      </div>
11  <script>
12      var app = angular.module('myApp', []);
13      app.controller('dateCntrl', function($scope) {
14      $scope.today = new Date();
15      });
16  </script>
17  </body>
18  </html>
```

Output :



---

8. Create an application to demonstrate directives.

Solution :

Directives :
AngularJS lets you extend HTML with new attributes called Directives.
AngularJS has a set of built-in directives which offers functionality to your applications.
AngularJS also lets you define your own directives.

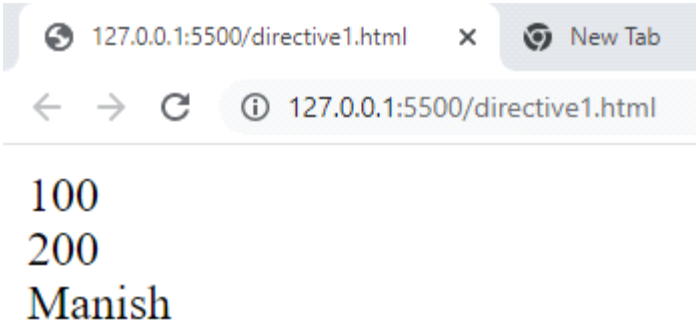
AngularJS directives are extended HTML attributes with the prefix ng-.
The ng-app directive initializes an AngularJS application.
The ng-init directive initializes application data.
The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

ng-init :
The ng-init directive can be used to initialize variables in AngularJS application.

directive1.html

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

Output :



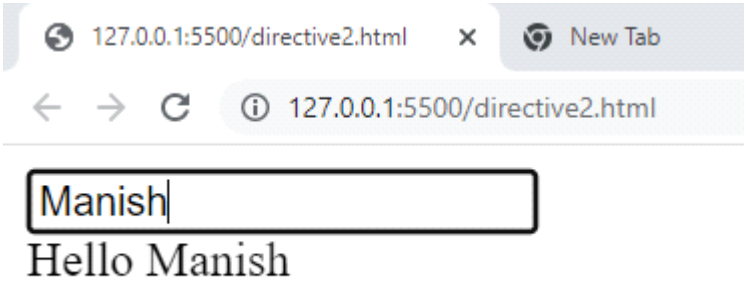
ng-model :

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. So, the value of the element will be the value of a property and vica-versa.

directive2.html

```
<!DOCTYPE html>
<html >
<head>
    <script src="angular-min.js"></script>
</head>
<body ng-app>
    <input type="text" ng-model="name" />
    <div>
        Hello {{name}}
    </div>
</body>
</html>
```

Output :



ng-bind

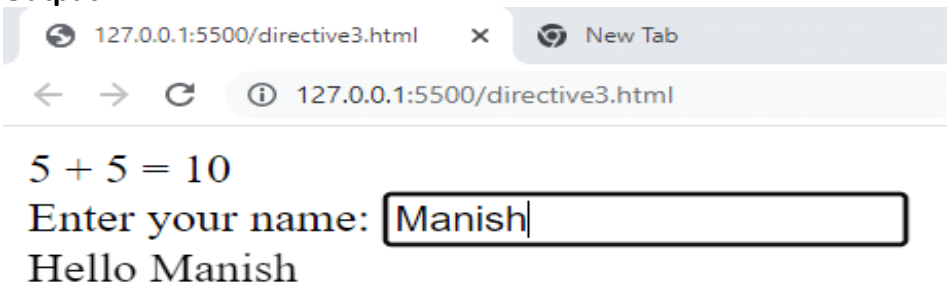
The ng-bind directive binds the model property declared via $scope or ng-model directive or the result of an expression to the HTML element. It also updates an element if the value of an expression changes.

directive3.html

```
<!DOCTYPE html>
<html >
<head>
    <script src="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>
```

Output :



ng-repeat

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

directive4.html

```
<!DOCTYPE html>
<html>
<head>
    <script src="angular-min.js"></script>
    <style>
        div {
            border: 1px solid green;
            width: 100%;
            height: 50px;
            display: block;
            margin-bottom: 10px;
            text-align:center;
            background-color: yellow;
        }
    </style>
</head>
<body ng-app="" ng-init="students=['Manish','Sanjay','Bagul']">
    <ol>
        <li ng-repeat="name in students">
            {{name}}
        </li>
    </ol>
    <div ng-repeat="name in students">
        {{name}}
    </div>
</body>
</html>
```

Output :

---

9. Demonstrate controllers in Angular.js through an application.

Solution :

Angular.js : AngularJS extends HTML with new attributes. AngularJS is perfect for Single Page Applications (SPAs). 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>

Controller :
AngularJS controllers control the data of AngularJS applications. AngularJS controllers are regular JavaScript Objects.

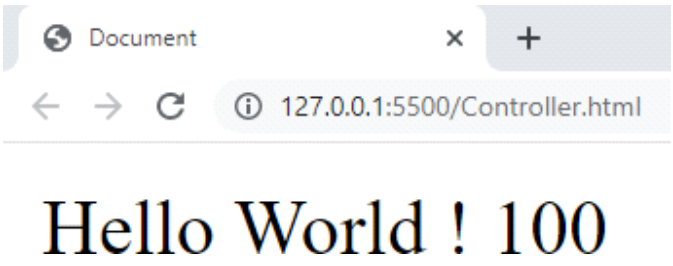
The ng-controller directive defines the application controller.
A controller is a JavaScript Object, created by a standard JavaScript object constructor.
The ng-controller directive defines the application controller.

Code :

Controller.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="angular-min.js"></script>
</head>
<body ng-app ="myApp">
    <div ng-controller="myController"> <!---view-->
        {{message}}
        {{price}}
    </div>
</body>
<script>
    var ngApp=angular.module("myApp",[]);
    ngApp.controller("myController",function($scope)
    {
        $scope.message = "Hello World !";
        $scope.price=100;
    });
</script>
</html>
```

Output :

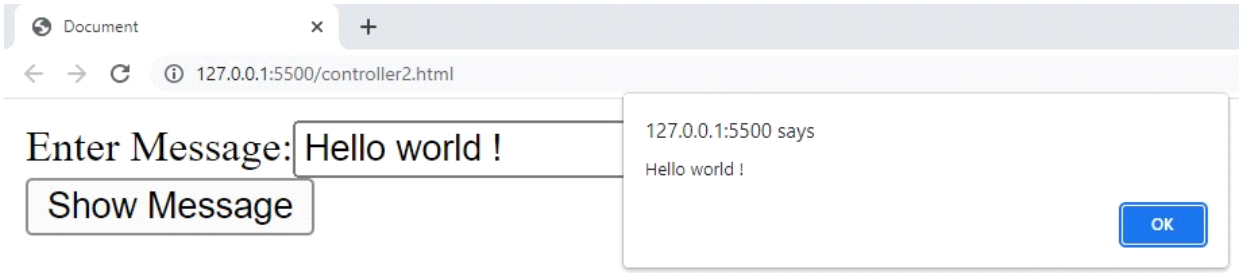


Code :

controller2.html > ...

```
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width =device-width,intial-scale=1.0">
<title>Document</title>
<script src="angular-min.js"></script>
</head>
<body ng-app="myngapp">
    <div ng-controller="mycontroller">
        Enter Message:<input type="text" ng-model="message"/><br>
        <button ng-click="showmsg(message)">
            Show Message
        </button>
    </div>
<script>
    var ngapp=angular.module('myngapp',[]);
    ngapp.controller('mycontroller',function($scope){
        $scope.message="Hello world !";
        $scope.showmsg=function(msg)
        {
        alert(msg);
        }
        });
</script>
</body>
</html>
```

Output :

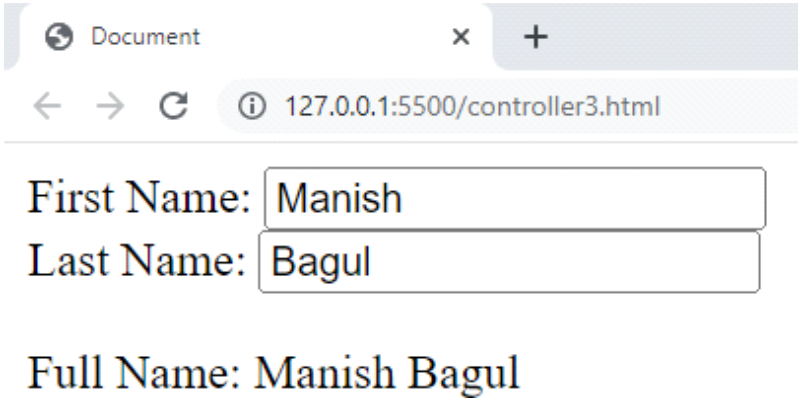


Code :

controller3.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
    <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 = "";
            $scope.lastName="";
        });
    </script>
</body>
</html>
```

Output :



---

10. Demonstrate features of Angular.JS forms with a program.

Solution :

AngularJS 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. A form is a collection of controls for the purpose of grouping related controls together.

Following are the input controls used in AngularJS forms:

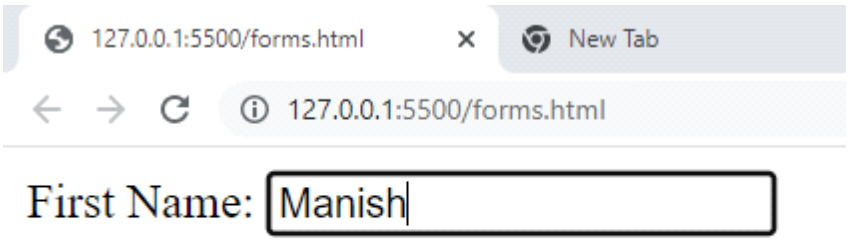
- input elements
- select elements
- button elements
- textarea elements

AngularJS provides multiple events that can be associated with the HTML controls. These events are associated with the different HTML input elements.

forms.html

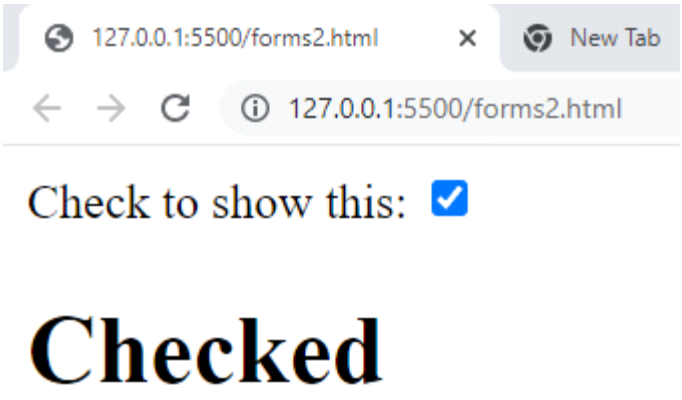
```
<!DOCTYPE html>
<html lang="en">
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="formCtrl">
  <form>
    First Name: <input type="text" ng-model="firstname">
  </form>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
    $scope.firstname = "Manish";
});
</script>
</body>
</html>
```

Output :



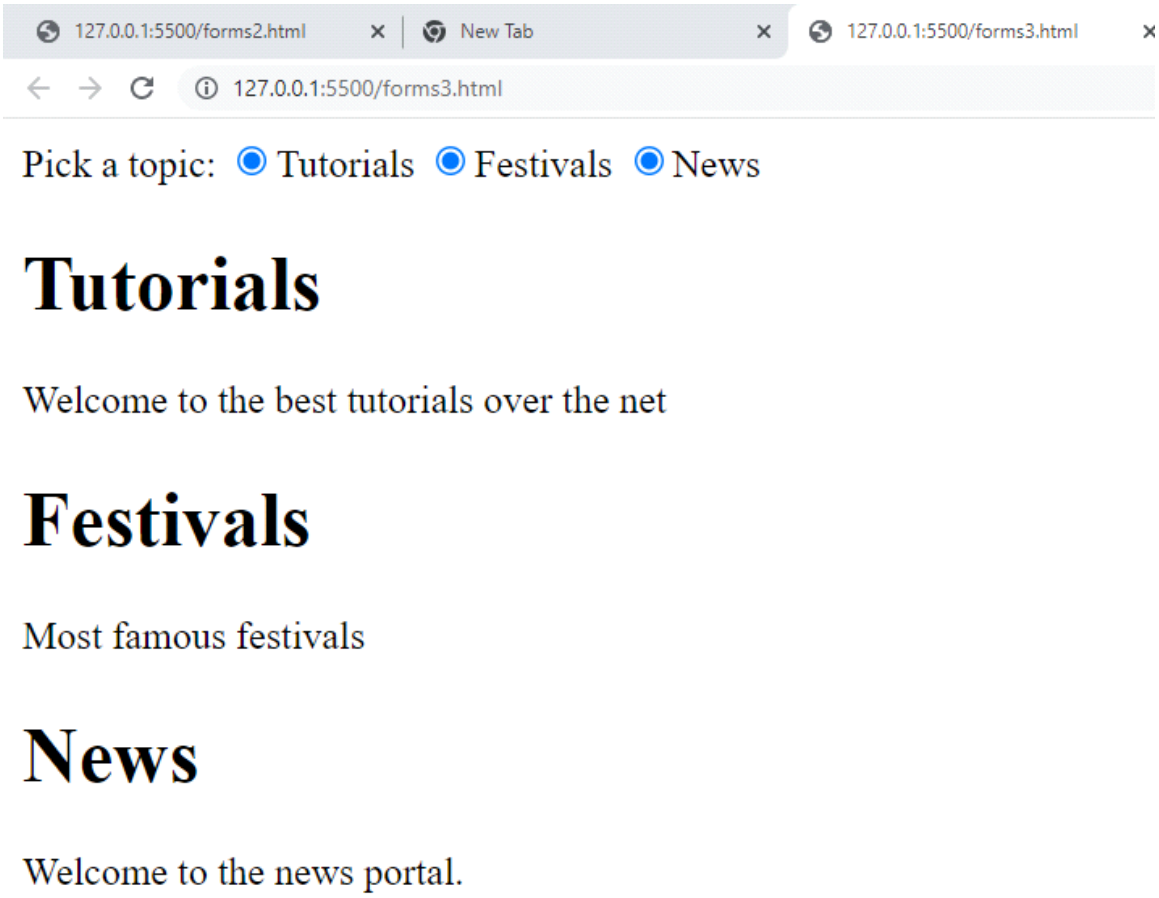
forms2.html

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="">
  <form>
    Check to show this:
    <input type="checkbox" ng-model="myVar">
  </form>
  <h1 ng-show="myVar">Checked</h1>
</div>
</body>
</html>
```



forms3.html

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="">
<form>
  Pick a topic:
  <input type="radio" ng-model="myVar" value="tuts">Tutorials
  <input type="radio" ng-model="myVar" value="fest">Festivals
  <input type="radio" ng-model="myVar" value="news">News
</form>
<div ng-switch="myVar">
  <div ng-switch-when="tuts">
    <h1>Tutorials</h1>
    <p>Welcome to the best tutorials over the net</p>
  </div>
  <div ng-switch-when="fest">
    <h1>Festivals</h1>
    <p>Most famous festivals</p>
  </div>
  <div ng-switch-when="news">
    <h1>News</h1>
    <p>Welcome to the news portal.</p>
  </div>
</div>
</body>
</html>
```



---

11. Create SPA (Single page application)

Solution :

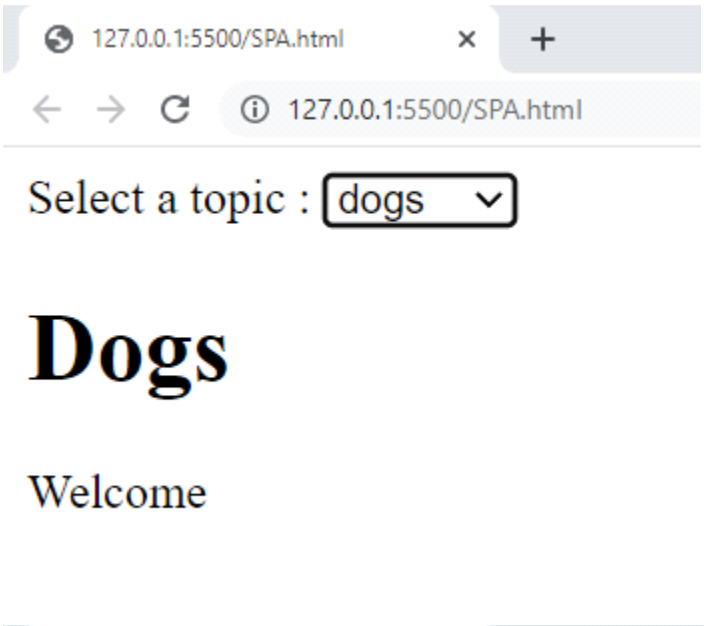
SPA :

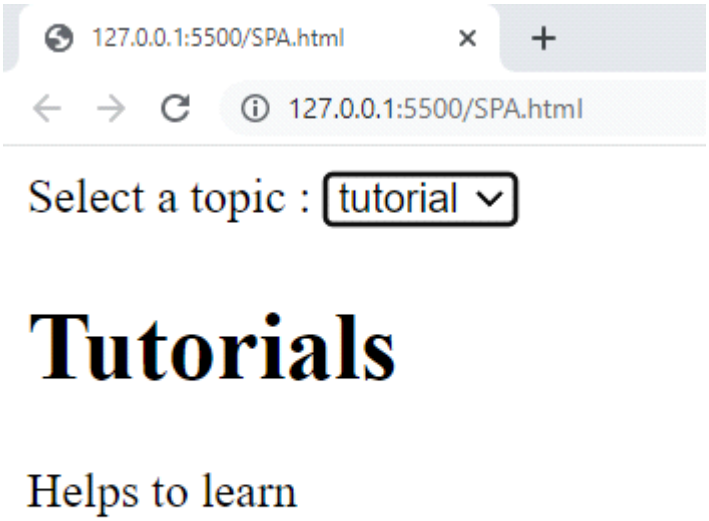
Single page application (SPA) is a web application that is contained in a single page. In a single page application all our code (JS, HTML, CSS) is loaded when application loads for the first time. Loading of the dynamic contents and the navigation between pages is done without refreshing the page. Single page application (SPA) is usually created using javascript based front-end web application framework.

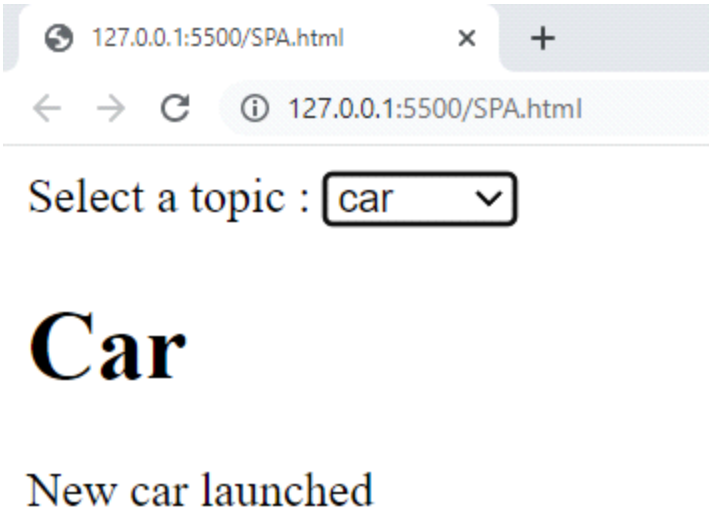
SPA.html

```
<html>
    <script src="angular-min.js"></script>
    <body ng-app = "">
        <form>
            Select a topic :
            <select ng-model = "myvar">
                <option value = "">
                <option value = "dogs">dogs
                <option value = "tutorial">tutorial
                <option value = "car">car
            </select>
        </form>
        <div ng-switch="myvar">
            <div ng-switch-when = "dogs">
                <h1>Dogs</h1>
                <p>Welcome</p>
            </div>
            <div ng-switch-when = "tutorial">
                <h1>Tutorials</h1>
                <p>Helps to learn</p>
            </div>
            <div ng-switch-when = "car">
                <h1>Car</h1>
                <p>New car launched</p>
            </div>
        </div>
    </body>
</html>
```

Output :


127.0.0.1:5500/SPA.html
127.0.0.1:5500/SPA.html
Select a topic : dogs
Dogs
Welcome


127.0.0.1:5500/SPA.html
127.0.0.1:5500/SPA.html
Select a topic : tutorial
Tutorials
Helps to learn


127.0.0.1:5500/SPA.html
127.0.0.1:5500/SPA.html
Select a topic : car
Car
New car launched
