WEB TECHNOLOGIES

Practical-01

Aim: Introduction & installation of node.js.

Theory:

Node.js:

Node.js is an open-source server environment. It is a back-end JavaScript runtime environment that runs on the V8 JavaScript Engine and executes JavaScript code outside a web browser. Node.js is cross-platform and runs on Windows, Linux, Unix and macOS.

Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The functionality of running scripts server-side produces dynamic web page content before the page is sent to the user's web browser. Node.js unifies web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications.

Advantages and Disadvantages of Node.js:

Advantages of Node.js:

1. Easy Scalability- One of the key advantages of Node.js is that developers find it easy to scale the applications in horizontal as well as the vertical directions.

2. Easy to learn- It is easier to learn Node.js and consumes less time to work with it.

3. High Performance- Node.js interprets the JavaScript code via Google's V8 JavaScript engine. This engine compiles the JavaScript code directly into the machine code. This makes it easier and faster to implement the code in an effective manner.

4. Large Community- Node.js has a large and active community of developers who keep on continuously contributing towards its further development and improvement.

5. Extended Support- With Node.js, the developers can get an extended support for the various commonly used tools.

6. Highly Extensible- The Node.js is known to be highly extensible, which means that you can customize and further extend Node.js as per their requirements.

Disadvantages of Node.js:

1. Unstable API- One of the key problems that most developers encounter is the Application Programming Interface (API) keeps on changing at frequent intervals and does not remain stable. At times, a new API appears having a number of backwards-incompatible changes. As a result the developers are forced to make changes in the code to match the compatibility with the latest version of the Node.js API.

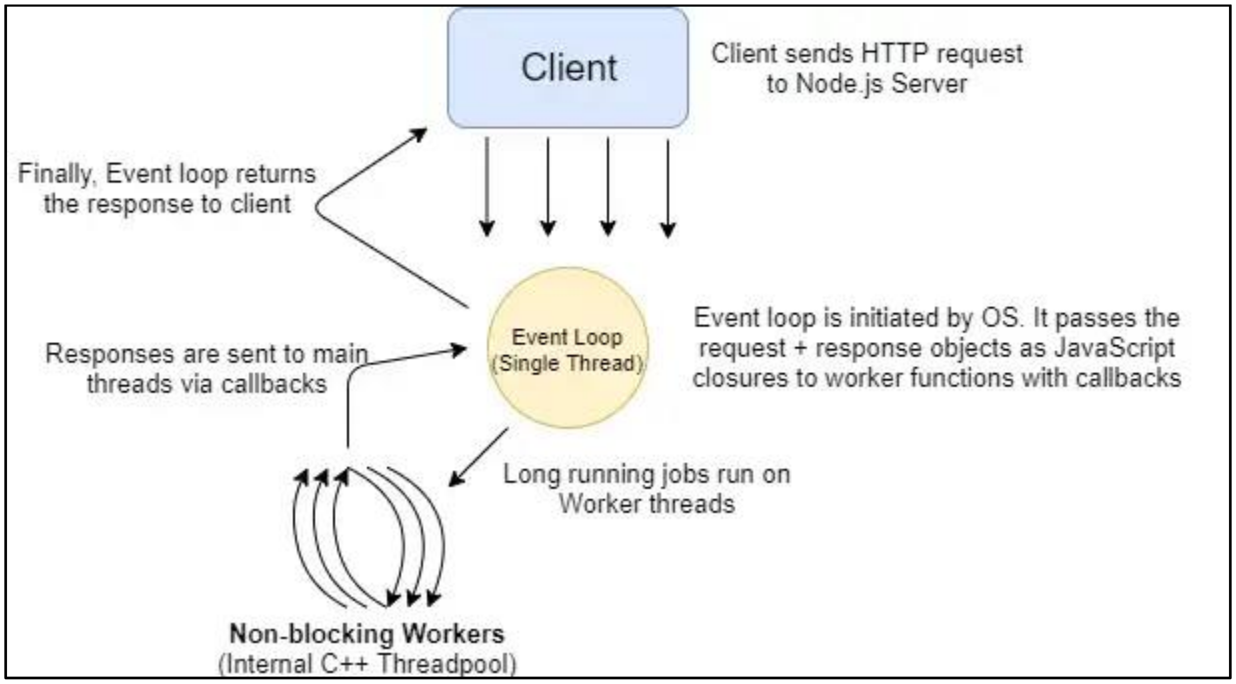
2. Lack of Library Support- In comparison to any other programming language, JavaScript lacks a well-equipped and robust library system. As a result, processes like Object-Relational Mapping (ORM), image processing, database operations, and XML parsing must be performed via a common library. This makes it difficult to use Node.js for even simple programming tasks.

3. Asynchronous Programming Model- To make the applications more scalable, the necessary requisite is adoption of the asynchronous programming model. However, many developers may find this programming model to be more difficult in comparison to the linear blocking I/O programming. Another disadvantage of asynchronous programming is that the code tends to become clunky, forcing programmers to rely on nested calls.

Node.js Process Model:

The Node.js process model differs from traditional web servers in that Node.js runs in a single process with requests being processed on a single thread. One advantage of this is that Node.js requires far fewer resources. When a request comes in, it will be placed in an event queue. Node.js uses an event loop to listen for events to be raised for an asynchronous job. The event loop continuously runs, receiving requests from the event queue.

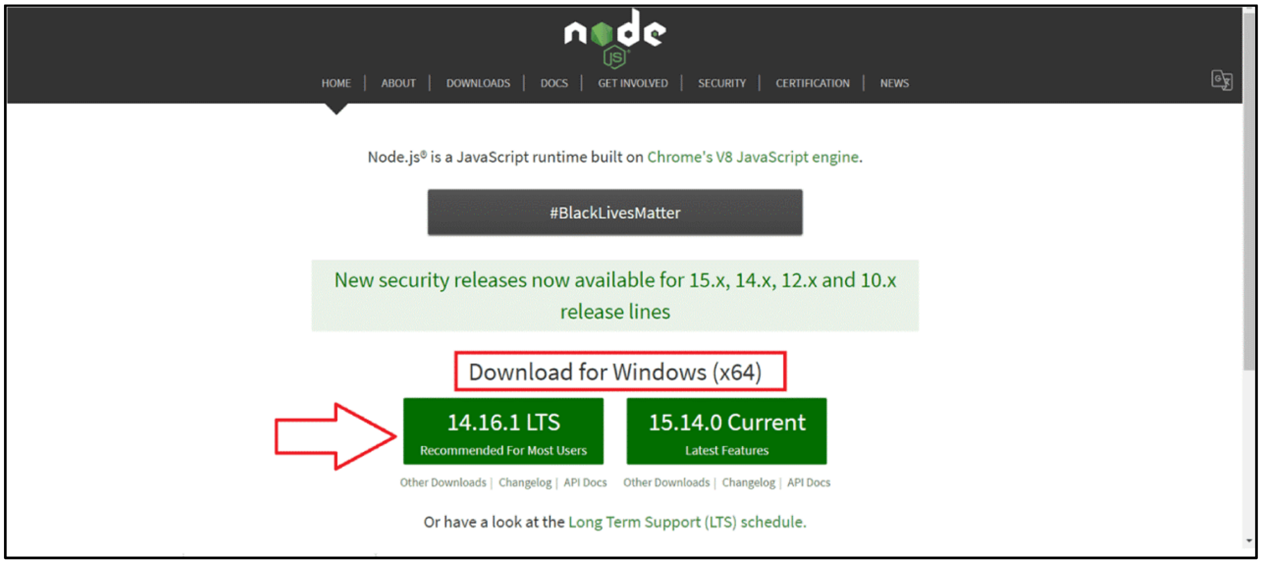
There are two scenarios that will occur depending on the nature of the request. If the request is non-blocking, it does not involve any long-running processes or data requests, the response will be immediately prepared and then sent back to the client. In the event the request is blocked, requiring I/O operations, the request will be sent to a worker thread pool. The request will have an associated call-back function that will fire when the request is finished and the worker thread can send the request to the event loop to be sent back to the client. In this way, when the single thread receives a blocking request, it hands it off so that the thread can process other requests in the meantime. In this way Node.js is inherently asynchronous.


Client
Client sends HTTP request to Node.js Server
Finally, Event loop returns the response to client
Event Loop (Single Thread)
Responses are sent to main threads via callbacks
Event loop is initiated by OS. It passes the request + response objects as JavaScript closures to worker functions with callbacks
Long running jobs run on Worker threads
Non-blocking Workers (Internal C++ Threadpool)

<u>Installation of Node.js:</u>

<u>Step 1: Download the Installer</u>-

Download the Windows Installer from NodeJs official website. Make sure you have downloaded the latest version of NodeJs. Here, we are choosing the 64-bit version of the Node.js installer.


HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS
#BlackLivesMatter
14.16.1 LTS
Recommended For Most Users
15.14.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

New security releases now available for 15.x, 14.x, 12.x and 10.x release lines

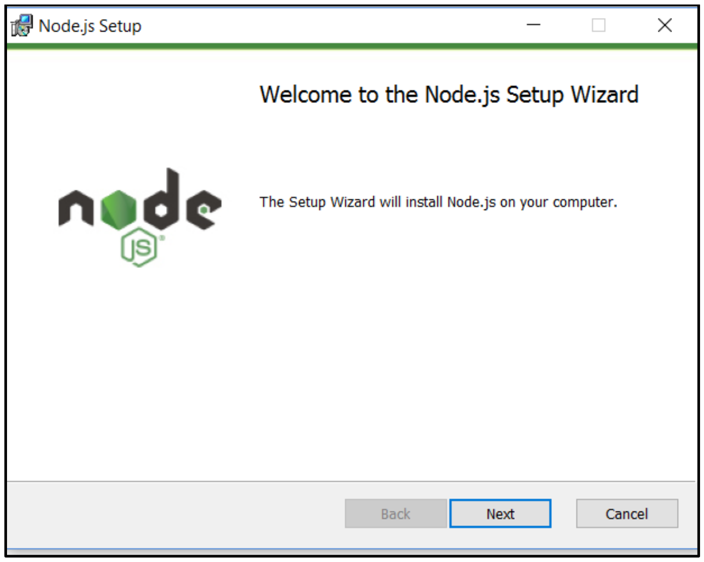
Download for Windows (x64)

Or have a look at the Long Term Support (LTS) schedule.

The LTS (Long-term Support) version is highly recommended. After the download of the installer package, install it with a double-click on it.

Now the .msi file will be downloaded to your browser. Choose the desired location for that.

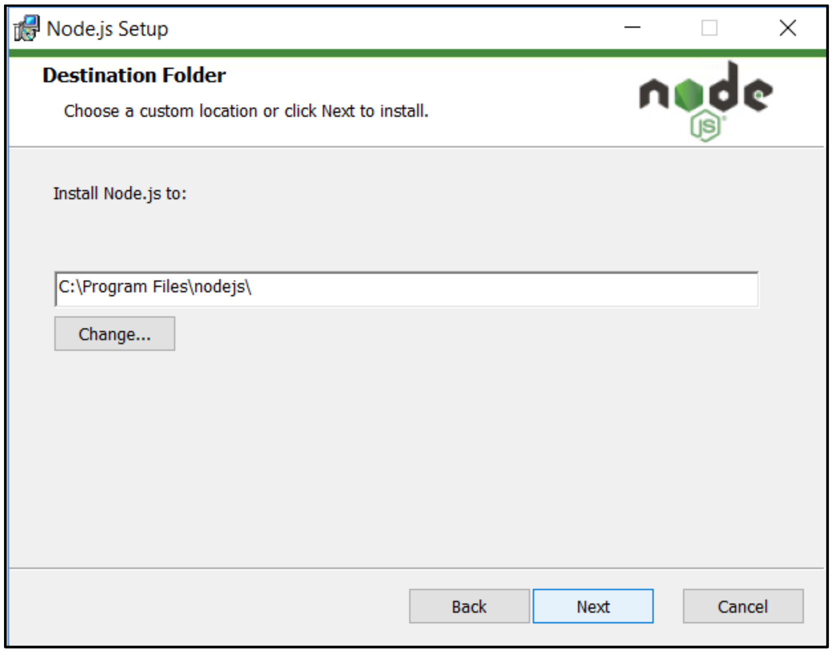
Step 2: Install Node.js-


Node.js Setup
Welcome to the Node.js Setup Wizard
The Setup Wizard will install Node.js on your computer.
Back
Next
Cancel

After choosing the path, double-click to install .msi binary files to initiate the installation process. Then give access to run the application.

You will get a welcome message on your screen and click the “Next” button. The installation process will start.

Choose the desired path where you want to install Node.js.

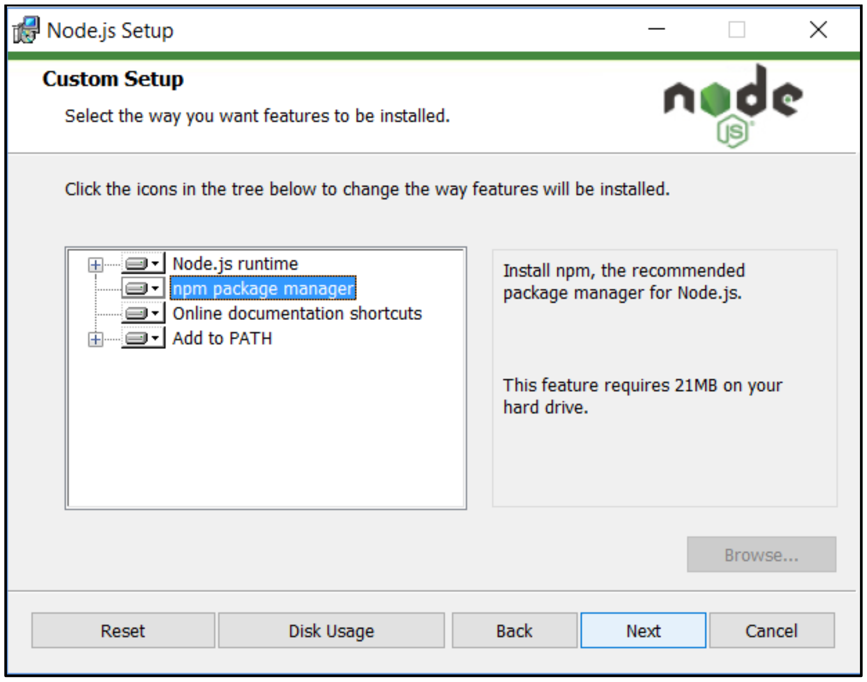


C:\Program Files\nodejs\

By clicking on the Next button, you will get a custom page setup on the screen. Make sure you choose npm package manager, this way, we can install Node and NPM simultaneously.

You should have 143MB of space to install Node.js and npm features.

The following features will be installed by default:



Custom Setup

Select the way you want features to be installed.

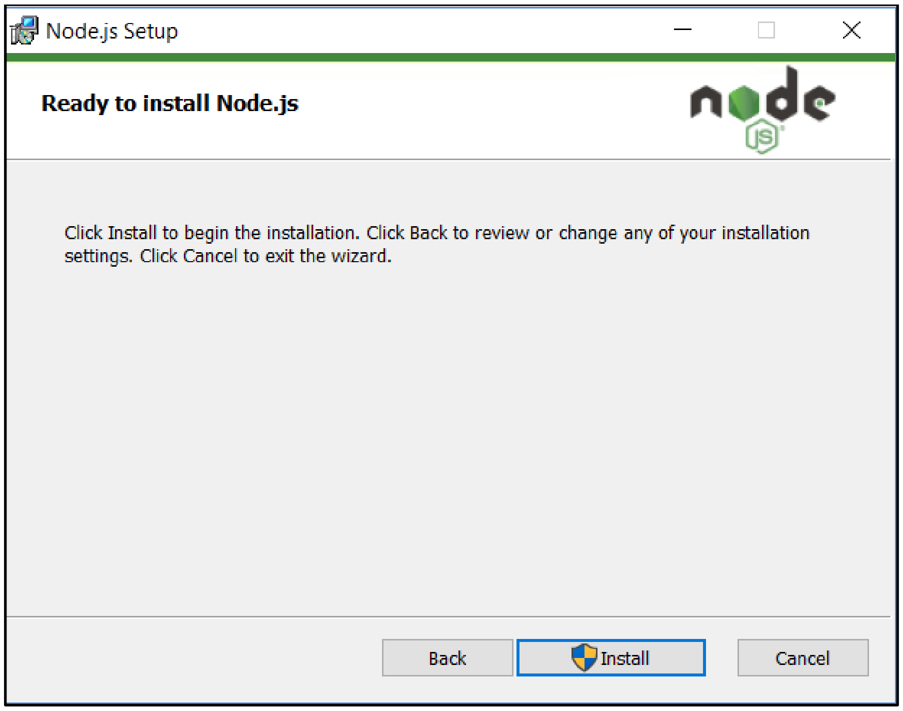
Click the icons in the tree below to change the way features will be installed.

- Node.js runtime
- npm package manager
- Online documentation shortcuts
- Add to PATH

Install npm, the recommended package manager for Node.js.

This feature requires 21MB on your hard drive.

The setup is ready to install Node and NPM, click in the Install Button to proceed.



Ready to install Node.js

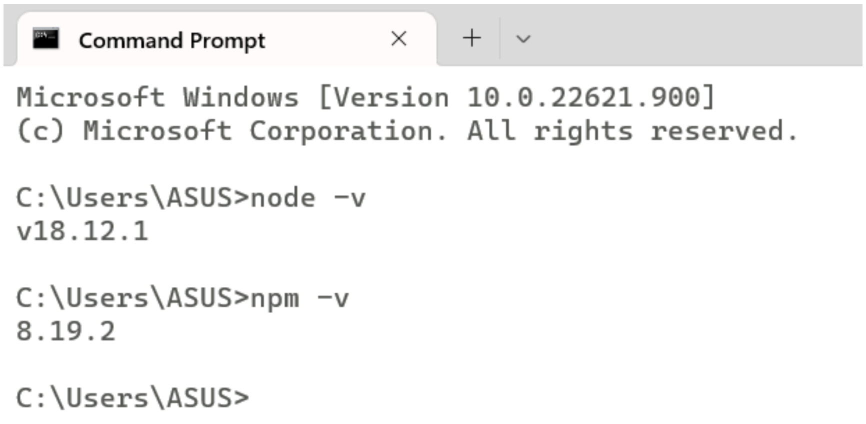
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Step 3: Check Node.js and NPM Version-

If you have a doubt whether you have installed everything correctly or not, you can verify it with “Command Prompt”.

To confirm Node installation, type ‘node -v’ command.

To confirm NPM installation, type ‘npm -v’ command.



```
Microsoft Windows [Version 10.0.22621.900]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>node -v
v18.12.1

C:\Users\ASUS>npm -v
8.19.2

C:\Users\ASUS>
```

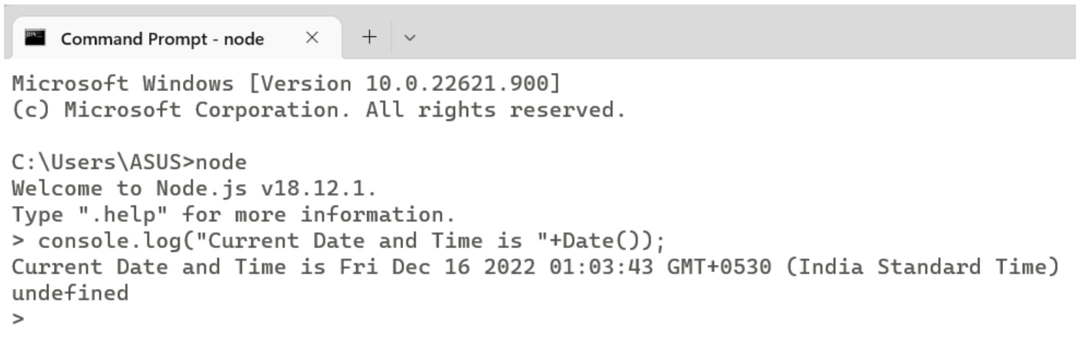
<u>Programs</u>:

Q1) Print today’s date and time using REPL.

Code:

```
console.log(“Current Date and Time is “+Date());
```

Output:



```
Microsoft Windows [Version 10.0.22621.900]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>node
Welcome to Node.js v18.12.1.
Type ".help" for more information.
> console.log("Current Date and Time is "+Date());
Current Date and Time is Fri Dec 16 2022 01:03:43 GMT+0530 (India Standard Time)
undefined
>
```

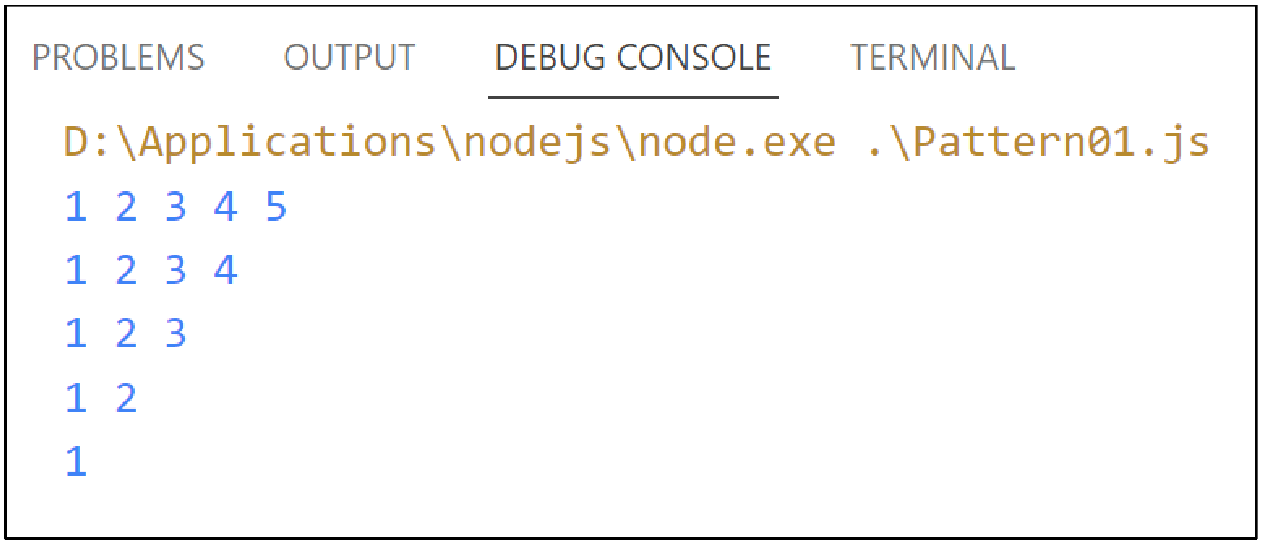
Q2) Write a program to print the following pattern.

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

Code:

```
        let str = "";
        for (let i=5; i>=1; i--)
        {
                for (let j=1; j<=i; j++)
    {
                        str += j+" ";
                }
                str += "\n";
        }
        console.log(str);
```

Output:



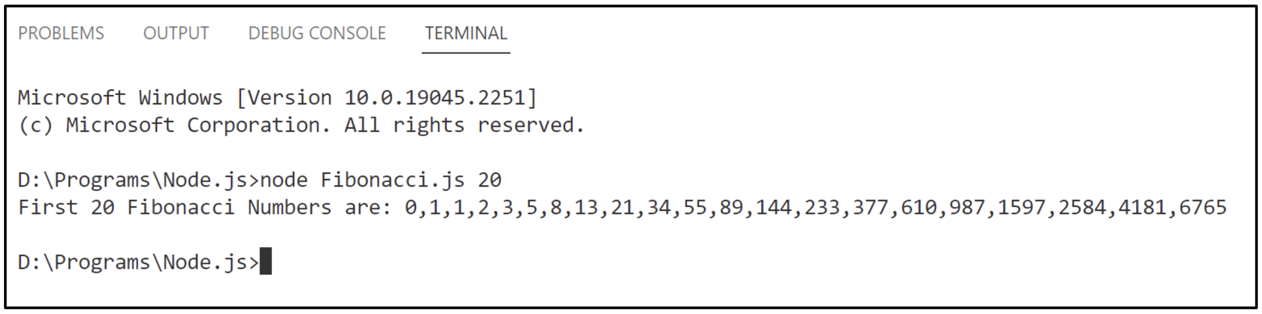
```
D:\Applications\nodejs\node.exe .\Pattern01.js
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Q3) Write a program to print first 20 Fibonacci Numbers (Take input through command line)

Code:

```
        var fib=function(n)
        {
                if(n==1)
     {
                        return[0,1];
                }
                 else
   {
                        var arr = fib(n-1);
                       arr.push(arr[arr.length-1]+arr[arr.length-2]);
                       return arr;
                }
        };
        userInput=process.argv[2];
        console.log("First "+userInput+" Fibonacci Numbers are: "+fib(userInput));
```

Output:



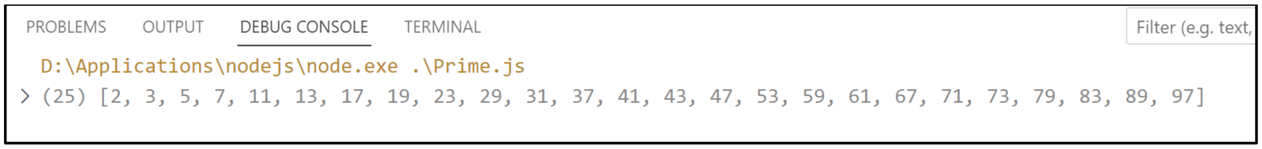
```
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

D:\Programs\Node.js>node Fibonacci.js 20
First 20 Fibonacci Numbers are: 0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765

D:\Programs\Node.js>
```

Q4) Write a program to print Prime Numbers between 1 to 100

Code:

```
const arr = []
for(var n=2; n<=100; n++)
{
    for(var i=2; i<n; i++)
    {
        if(n%i == 0)
          {
             break
          }
    }
       if(n == i)

   arr.push(n)
  }
}
console.log(arr);
```

Output:



```
D:\Applications\nodejs\node.exe .\Prime.js
> (25) [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Conclusion:

In this practical, we were introduced to Node.js. We performed the installation of Node.js and ran a few basic programs on it.

WEB TECHNOLOGIES

Practical-02

Aim: Implementation of Built-in and Custom Modules in Node,js

Theory:

What is Node.js Module:

Node.js modules are the blocks of encapsulated code that communicate with an external application on the basis of their related functionality. They are a simple or complex functionality organized in single or multiple JavaScript files which can be reused throughout the Node.js application. Each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope. The reason programmers are heavily reliant on modules is because of their re-usability as well as the ability to break down a complex piece of code into manageable chunks.
There are three types of Modules in Node.js:

- Built-in Modules
- Custom Modules
- Third Party Modules

Built-in Modules:

Node.js has many built-in modules that are part of the platform and comes with Node.js installation. These modules can be loaded into the program by using the **require** function.The **require()** function will return a JavaScript type depending on what the particular module returns. Some examples of Built-in modules are:

1. **os**: The OS module provides information about the computer's operating system.
2. **path**: The Path module provides a way of working with directories and file paths.
3. **http**: The HTTP module provides a way of making Node.js transfer data over HTTP.
4. **fs**: The File System module provides a way of working with the computer's file system.
5. **timer**: The Timers module provides a way scheduling functions to be called later at a given time.
6. **events**: The Events module provides a way of working with events.

Custom Modules:

We can define modules locally as custom/local modules. It consists of different functions declared inside a JavaScript object and we reuse them according to the requirement.
Custom modules must be written in a separate JavaScript file. In the separate file, we can declare a JavaScript object with different properties and methods.
Use the **exports** keyword to make properties and methods available outside the module file.
Some examples of Custom modules are:

1. **Calculator**: A custom module containing the basic functions of a calculator.

2. **Circle**: A custom module containing formulas for area and perimeter of a circle.

Third Party Modules:

Third-party modules are modules that are available online using the Node Package Manager(NPM). These modules can be installed in the project folder or globally.
Some of the popular third-party modules are mongoose, express, angular, and react.

NPM:

The Node Package Manager is an online repository for the publishing of open-source Node.js projects. It is a command-line utility for interacting with said repository that aids in package installation, version management, and dependency management. These applications can be searched for on https://www.npmjs.com/. Once you have a package you want to install, it can be installed with a single command-line command.

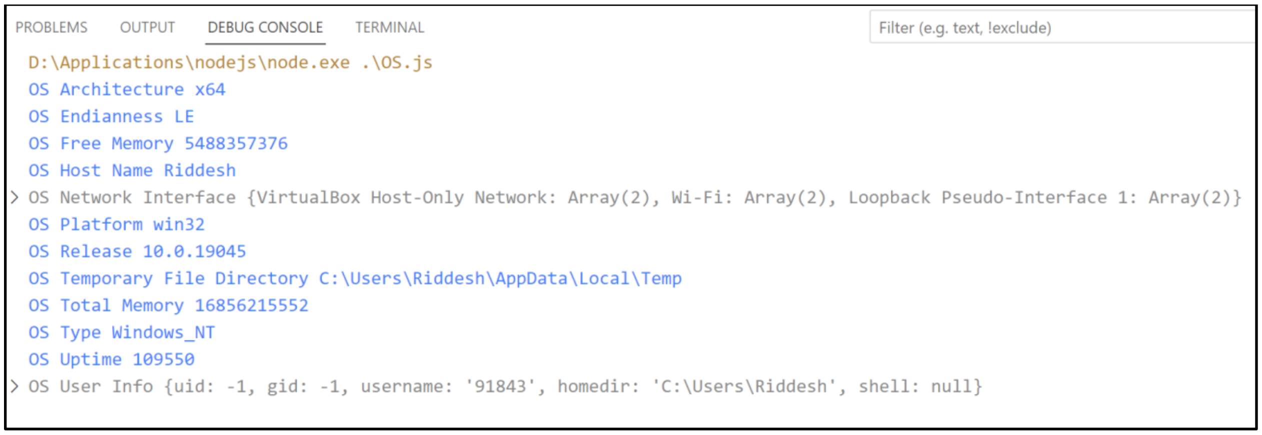
Programs:

Q1) Write a program to print information about the computer's operating system using OS module(use any 5 methods).

Code:

```
        let os = require('os');
console.log('OS Architecture' , os.arch());
console.log('OS Endianness' , os.endianness());
console.log('OS Free Memory' , os.freemem());
console.log('OS Host Name' , os.hostname());
console.log('OS Network Interface' , os.networkInterfaces());
console.log('OS Platform' , os.platform());
console.log('OS Release' , os.release());
console.log('OS Temporary File Directory' , os.tmpdir());
console.log('OS Total Memory' , os.totalmem());
console.log('OS Type' , os.type());
        console.log('OS Uptime' , os.uptime());
console.log('OS User Info' , os.userInfo());
```

Output:



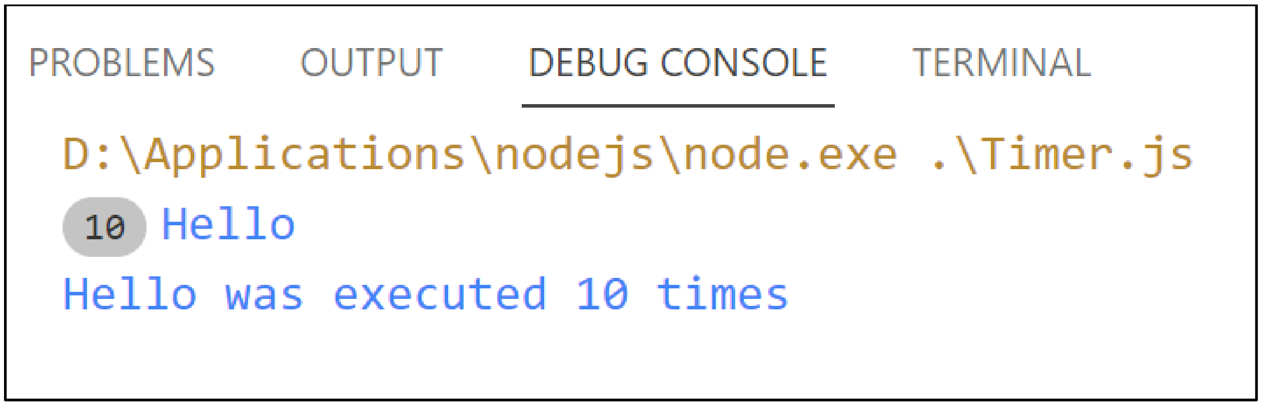
```
D:\Applications\nodejs\node.exe .\OS.js
OS Architecture x64
OS Endianness LE
OS Free Memory 5488357376
OS Host Name Riddesh
> OS Network Interface {VirtualBox Host-Only Network: Array(2), Wi-Fi: Array(2), Loopback Pseudo-Interface 1: Array(2)}
OS Platform win32
OS Release 10.0.19045
OS Temporary File Directory C:\Users\Riddesh\AppData\Local\Temp
OS Total Memory 16856215552
OS Type Windows_NT
OS Uptime 109550
> OS User Info {uid: -1, gid: -1, username: '91843', homedir: 'C:\Users\Riddesh', shell: null}
```

Q2) Print "Hello" every 500 milliseconds using Timer Module. The message should be printed exactly 10 times. Use SetInterval ,ClearInterval and SetTimeout methods.

Code:

```
let interval = setInterval(function(){
                        console.log('Hello');
            },500);

let timeout = setTimeout(function(){
                        clearInterval(interval);
                         console.log('Hello was executed 10 times');
},5500);
```

Output:



```
D:\Applications\nodejs\node.exe .\Timer.js
10 Hello
Hello was executed 10 times
```

Q3) Create a Calculator Node.js Module with functions add, subtract, multiply, Divide. And use the Calculator module in another Node.js file.

Code:

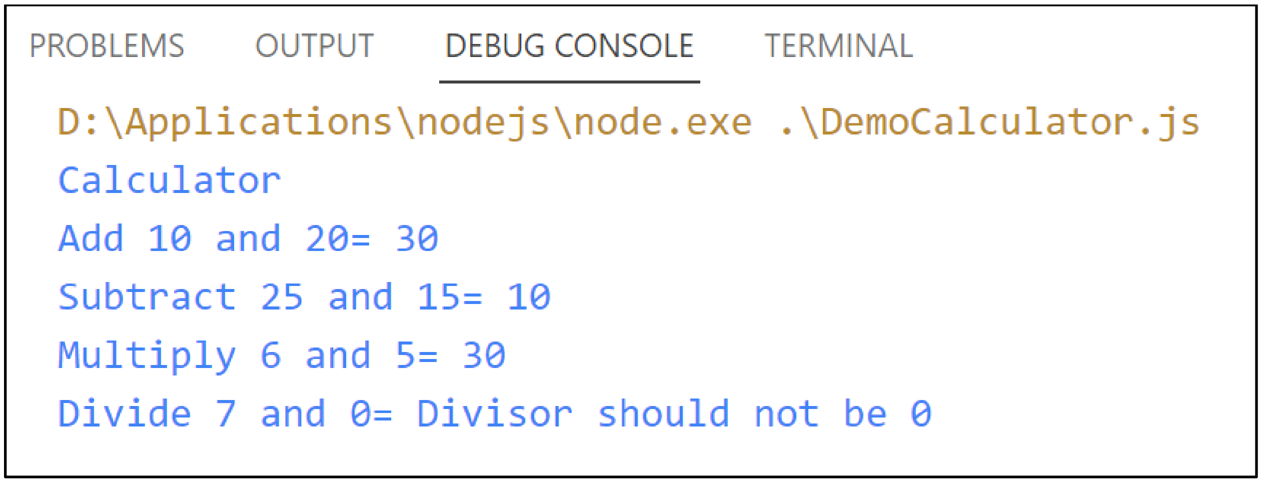
CustomCalculator.js:

```
        exports.message="Calculator";
exports.add=function(a,b){return a+b;}
exports.sub=function(a,b){return a-b;}
exports.mul=function(a,b){return a*b;}
exports.div=function(a,b){
                if(b==0){
                        return "Divisor should not be 0";
                }
                else{
                        return a/b;
                }
};
```

DemoCalculator.js:

```
var dt = require("./CustomCalculator");
console.log(dt.message);
console.log("Add 10 and 20= "+dt.add(10,20));
console.log("Subtract 25 and 15= "+dt.sub(25,15));
console.log("Multiply 6 and 5= "+dt.mul(6,5));
console.log("Divide 7 and 0= "+dt.div(7,0));
```

Output:



```
D:\Applications\nodejs\node.exe .\DemoCalculator.js
Calculator
Add 10 and 20= 30
Subtract 25 and 15= 10
Multiply 6 and 5= 30
Divide 7 and 0= Divisor should not be 0
```

Q4) Create a circle module with functions to find the area and perimeter of a circle and use it.

Code:

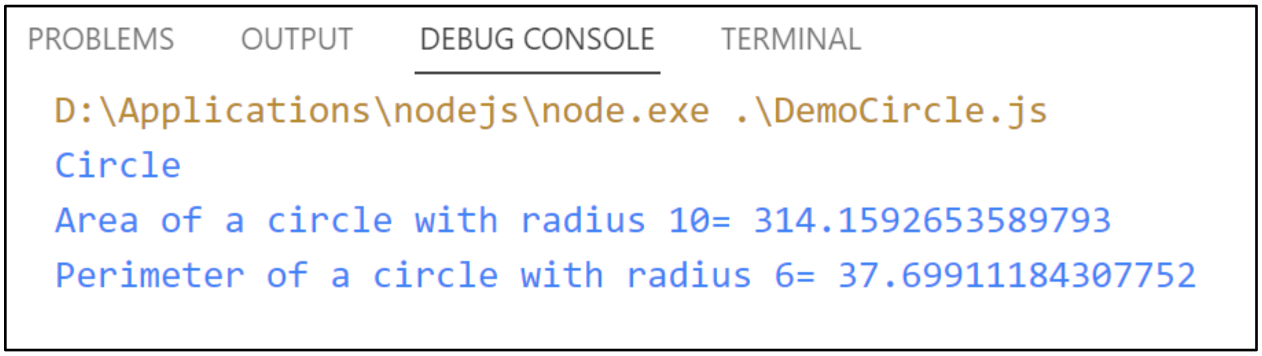
CustomCircle.js:

```
	exports.message="Circle"
exports.area=function(r){return  Math.PI*r*r;}
exports.peri=function(r){return  2*Math.PI*r;}
```

DemoCircle.js:

```
	var dt = require("./CustomCircle");
console.log(dt.message);
console.log("Area  of a circle with radius 10= "+dt.area(10));
console.log("Perimeter  of a circle with radius 6= "+dt.peri(6));
```

Output:



```
D:\Applications\nodejs\node.exe .\DemoCircle.js
Circle
Area of a circle with radius 10= 314.1592653589793
Perimeter of a circle with radius 6= 37.69911184307752
```

Conclusion:

In this practical, we learned about Node.js Modules and performed programs on Built-in and Custom Node.js Modules.

WEB TECHNOLOGIES

Practical-03

Aim: Implementation of Events in Node.js.

Theory:

Node.js Events:

Every action on a computer is an event. Like when a connection is made or a file is opened. Objects in Node.js can fire events, like the readStream object fires events when opening and closing a file

Events Module:

Node.js has a built-in module, called "Events", where you can create-, fire-, and listen for- your own events.
To include the built-in Events module use the **require()** method. In addition, all event properties and methods are an instance of an EventEmitter object. To be able to access these properties and methods, create an EventEmitter object:

```
var events = require('events');
var eventEmitter = new events.EventEmitter();
```

EventEmitter Object:

You can assign event handlers to your own events with the EventEmitter object.
To fire an event, use the **emit()** method.

EventEmitter Methods:

- **emitter.addListener(event, listener)**- Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added.

- **emitter.on(event, listener)**- Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. It can also be called as an alias of emitter.addListener()

- **emitter.once(event, listener)**- Adds a one time listener for the event. This listener is invoked only the next time the event is fired, after which it is removed.

- **emitter.removeListener(event, listener)**- Removes a listener from the listener array for the specified event. Caution: changes array indices in the listener array behind the listener.

- **emitter.removeAllListeners([event])**- Removes all listeners, or those of the specified event.

- **emitter.setMaxListeners(n)**- By default EventEmitters will print a warning if more than 10 listeners are added for a particular event.

- **emitter.getMaxListeners()**- Returns the current maximum listener value for the emitter which is either set by emitter.setMaxListeners(n) or defaults to EventEmitter.defaultMaxListeners.

- **emitter.listeners(event)**- Returns a copy of the array of listeners for the specified event.

- **emitter.emit(event[, arg1][, arg2][, ...])**- Raise the specified events with the supplied arguments.

- **emitter.listenerCount(type)**- Returns the number of listeners listening to the type of event.

<u>Programs</u>:

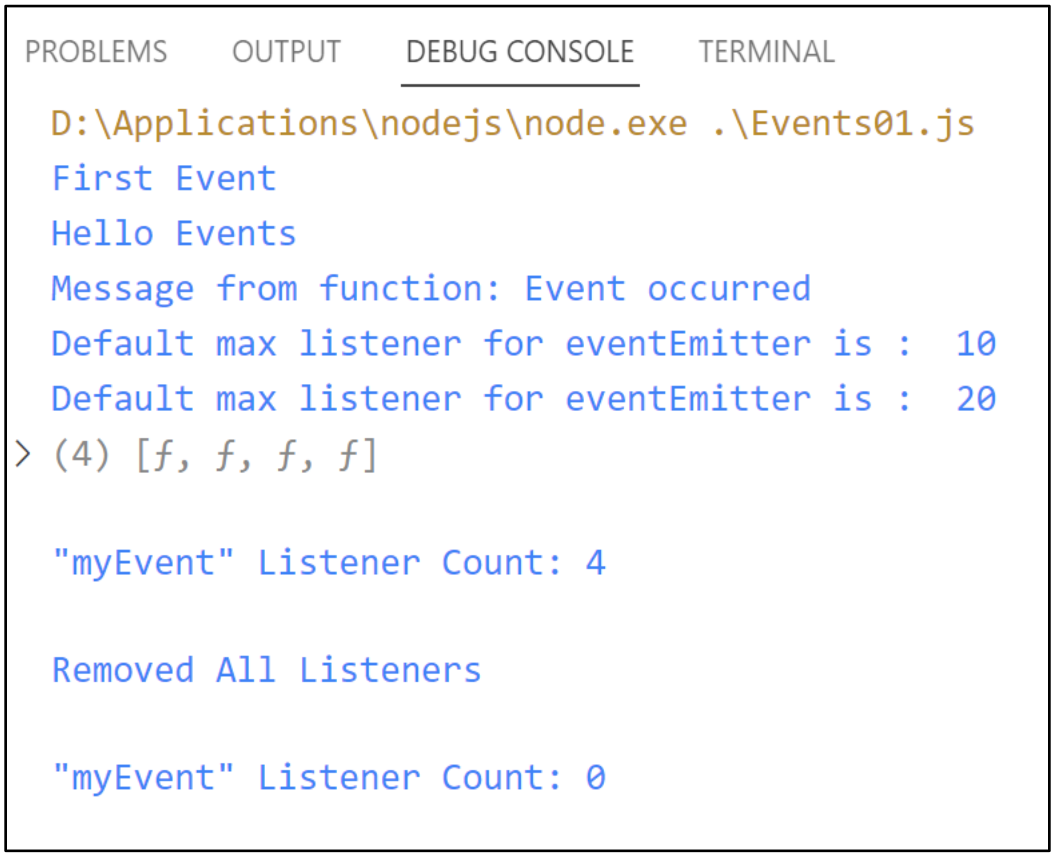
Q1) Create an application to demonstrate various Node.js Events in event emitter class.

Code:

```
var eventDemo = require('events');
var eventEmitter = new eventDemo.EventEmitter();

eventEmitter.on('myEvent',()=>{
            console.log("First Event");
});

var myEventHandler = ()=>{
             console.log('Hello Events');
}

eventEmitter.on('myEvent',myEventHandler);

var fun = (msg) => {
             console.log('Message from function: '+msg);
};

eventEmitter.on('myEvent',fun);

eventEmitter.emit('myEvent',"Event  occurred");

console.log('Default max listener for eventEmitter is : ',eventEmitter.getMaxListeners());

eventEmitter.setMaxListeners(20);

console.log('Default max listener for eventEmitter is : ',eventEmitter.getMaxListeners());

let msg = function() {
            console.log('Hello World');
}

var fun = function() {
           console.log('Message from function: '+msg);
};

eventEmitter.addListener('myEvent',fun)
```

```
console.log(eventEmitter.listeners('myEvent'));
console.log('\n"myEvent"  Listener Count: '+eventEmitter.listenerCount('myEvent'));

eventEmitter.removeAllListeners('myEvent');
console.log('\nRemoved  All Listeners');
console.log('\n"myEvent"  Listener Count: '+eventEmitter.listenerCount('myEvent'));
```

Output:



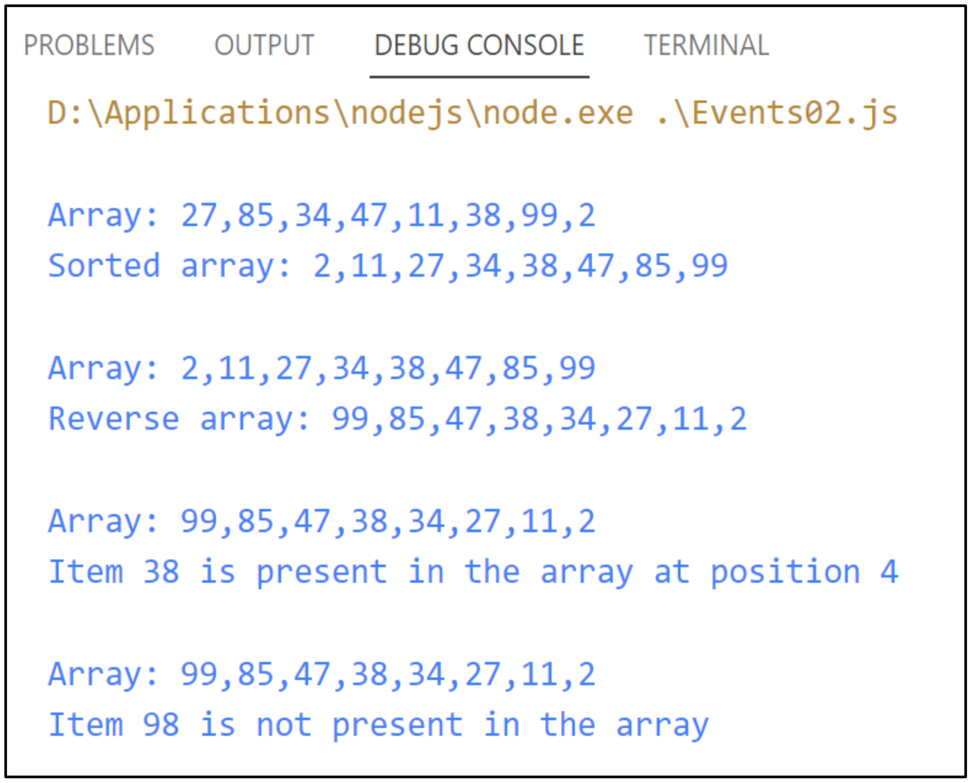
```
D:\Applications\nodejs\node.exe .\Events01.js
First Event
Hello Events
Message from function: Event occurred
Default max listener for eventEmitter is :  10
Default max listener for eventEmitter is :  20
> (4) [ƒ, ƒ, ƒ, ƒ]

"myEvent" Listener Count: 4

Removed All Listeners

"myEvent" Listener Count: 0
```

Q2) Create functions to sort,reverse and search for an element in an array. Register and trigger these functions using events.

Code:

```
        let events = require('events');
let eventEmitter = new events.EventEmitter();

let arr = [27,85,34,47,11,38,99,2];

function printArr(arr) {
                console.log("\nArray:  "+ arr);
}

function sortArr(arr) {
                arr.sort(function(n1,n2) {return n1-n2});
                console.log('Sorted array: ' + arr)
}

function reverseArr(arrRev) {
                arrRev.reverse();
                console.log('Reverse array: ' + arrRev)
}

function searchArr(arr, item) {
                let present = false;
                arr.forEach((element, index) => {
                        if (element === item) {
                        console.log('Item  ' + item + ' is present in the array at position '
                            + (index+1));
                                present = true;
                        }
                });
                if(!present) {
                        console.log('Item  ' + item + ' is not present in the array');
                }
}

eventEmitter.on('sortEvent', printArr);
eventEmitter.on('sortEvent', sortArr);
eventEmitter.on('reverseEvent', printArr);
eventEmitter.on('reverseEvent', reverseArr);
eventEmitter.on('searchEvent', printArr);
eventEmitter.on('searchEvent', searchArr);

eventEmitter.emit('sortEvent', arr);
eventEmitter.emit('reverseEvent', arr);
eventEmitter.emit('searchEvent', arr, 38);
eventEmitter.emit('searchEvent', arr, 98);
```

Output:



```
D:\Applications\nodejs\node.exe .\Events02.js

Array: 27,85,34,47,11,38,99,2
Sorted array: 2,11,27,34,38,47,85,99

Array: 2,11,27,34,38,47,85,99
Reverse array: 99,85,47,38,34,27,11,2

Array: 99,85,47,38,34,27,11,2
Item 38 is present in the array at position 4

Array: 99,85,47,38,34,27,11,2
Item 98 is not present in the array
```

<u>Conclusion</u>:

In this practical, we learned about the Node.js Events Module and implemented programs to demonstrate the various event functions successfully.

WEB TECHNOLOGIES

Practical-04

Aim: Implementation of Timers Module in Node.js

Theory:

Timers:

The Timers module provides a way of scheduling functions to be called later at a given time. The Timers module in Node.js contains various functions that allow us to execute a block of code or a function after a set period of time. The Timers module is global, so we do not need to use require() to import it.
Functions in timers can be used to delay or repeat the execution of other functions, which they receive as arguments.

Phases of event loop:

Event Loop consists of the following phases.

- **Timers**- in this phase callbacks of expired timers added using **setTimeout** or interval functions added using **setInterval** are executed.
- **Pending Callbacks**- executes I/O callbacks deferred to the next loop iteration.
- **Idle Handlers**- perform some libuv internal stuff, used internally.
- **Prepare Handlers**- perform some prep-work before polling for I/O, used internally.
- **I/O Polls**- retrieve new I/O events; execute I/O related callbacks.
- **Check Handlers**- **setImmediate()** callbacks are invoked here.
- **Close Callbacks**- execute close handlers.

Timers Module Functions:

Scheduling Timers: It is used to call a function after a set period of time.

- **setImmediate()-** When you want to execute some piece of code asynchronously, but as soon as possible, one option is to use the setImmediate() function provided by Node.js.

- **setInterval()-** If there is a block of code that should execute multiple times, setInterval() can be used to execute that code. setInterval() takes a function argument that will run an infinite number of times with a given millisecond delay as the second argument.

- **setTimeout()-** setTimeout() can be used to schedule code execution after a designated amount of milliseconds. setTimeout() accepts a function to execute as its first argument and the millisecond delay defined as a number as the second argument. Additional arguments may also be included and these will be passed on to the function.

Canceling Timers: It is used to cancel the scheduled timer. By passing the timer object setTimeout(), setImmediate(), and setInterval() return into the respective clear function, execution of that object will be halted completely. The respective functions are:

- **clearImmediate()**

- **clearInterval()**

- **clearTimeout()**

Programs:

Q1) Create an application to demonstrate Node.js timer functions.

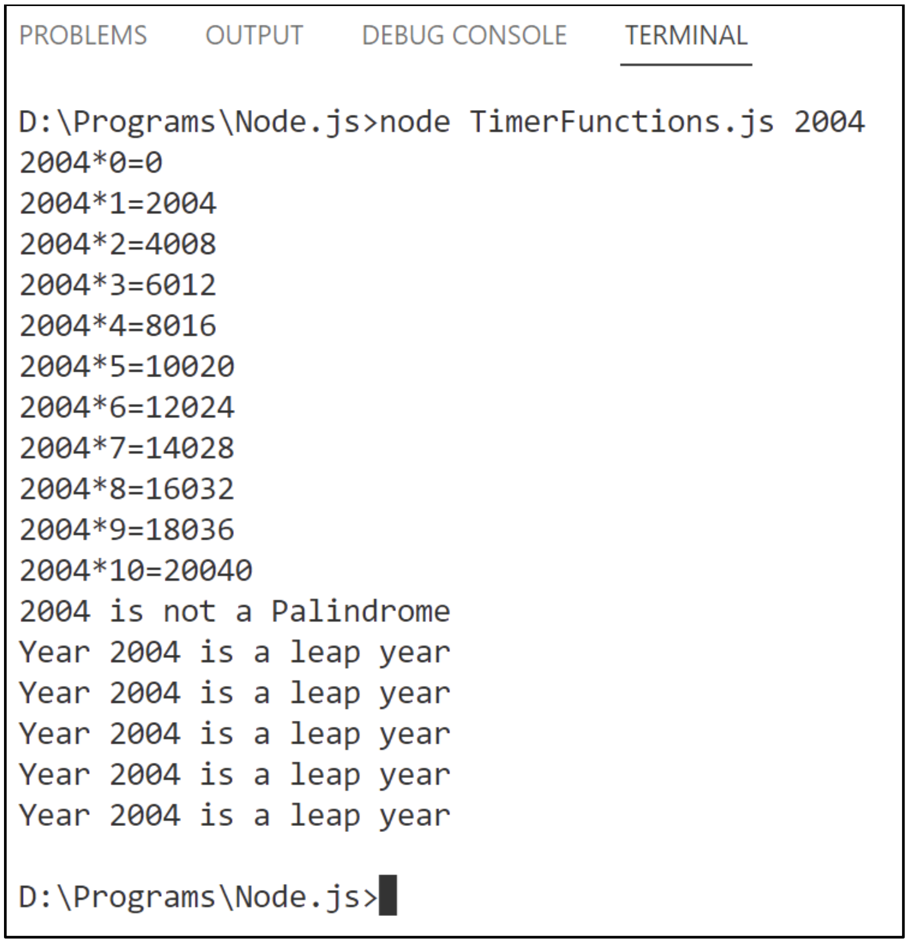
1. Print multiplication table for a number.
2. Find whether the entered year is a leap year or not.
3. Check whether the given string is palindrome or not.

using various timer functions(process.nextTick,setImmediate,setInterval and setTimeout).

Code:

```
let num = process.argv[2]

//Multiplication Table
process.nextTick(function(){
            for(let i=0; i<=10; i++){
                  console.log(num+"*"+i+"="+i*num);
            }
    });

//Leap Year
var leap = setInterval(function(){
            if((num%4==0)&&(num%100!=0)||(num%400==0)){
                  console.log("Year "+num+" is a leap year");
             }
            else{
                  console.log("Year "+num+" is not a leap year");
            }
},1000);

let timeout = setTimeout(function(){
            clearInterval(leap);
},6000);

//Palindrome
function reverseString(num){
             let rev = "";
            for(let i=num.length; i>=0; i--){
                  rev+=num.charAt(i);
             }
             return rev;
}
let revStr=reverseString(num);
var pal = setImmediate(function(){
            if(num==revStr){
                  console.log(num+' is a Palindrome');
            }
            else{
                  console.log(num+' is not a Palindrome');
            }
});
```

Output:



```
D:\Programs\Node.js>node TimerFunctions.js 2004
2004*0=0
2004*1=2004
2004*2=4008
2004*3=6012
2004*4=8016
2004*5=10020
2004*6=12024
2004*7=14028
2004*8=16032
2004*9=18036
2004*10=20040
2004 is not a Palindrome
Year 2004 is a leap year
Year 2004 is a leap year
Year 2004 is a leap year
Year 2004 is a leap year
Year 2004 is a leap year

D:\Programs\Node.js>
```

Conclusion:

In this practical, we learned about the Node.js Timers Module and made an application to demonstrate the various timer functions successfully.

WEB TECHNOLOGIES
PRACTICAL NO: 05

AIM: Implementation of fs modules in JavaScript.
THEORY:
File System Module:
To handle file operations like creating, reading, deleting, etc., Node.js provides an inbuilt module called FS (File System). Node.js gives the functionality of file I/O by providing wrappers around the standard POSIX functions. All file system operations can have synchronous and asynchronous forms depending upon user requirements. To use this File System module, use the require() method:
Syntax:- var fs = require('fs');

Synchronous Approach:
They are called **blocking functions** as it waits for each operation to complete, only after that, it executes the next operation, hence blocking the next command from execution i.e. a command will not be executed until & unless the query has finished executing to get all the result from previous commands.

Asynchronous approach:
They are called non-blocking functions as it never waits for each operation to complete, rather it executes all operations in the first go itself. The result of each operation will be handled once the result is available i.e. each command will be executed soon after the execution of the previous command. While the previous command runs in the background and loads the result once it is finished processing the data.

Functions in node.js fs modules are:
1) Open Method:
To create file, to write to a file or to read a file fs.open() method is used.
fs.readFile() is only for reading the file.
fs.writeFile() is only for writing to a file, whereas fs.open() method does several operations on a file.
Syntax:
fs.open(filename, flags, mode, callback)

2) Read Method:
The read() method of fs package reads the file using a file descriptor. In order to read files without file descriptor the readFile() method of fs package can be used.
Syntax:
fs.readFile(filename, encoding, callback_function)

3) Write Method:
The fs.writeFile() method is used to asynchronously write the specified data to a

file. By default, the file would be replaced if it exists. The ‘options’ parameter can be used to modify the functionality of the method.
Syntax:
fs.writeFile(file, data, options, callback)

4) Append Method:
The fs.appendFile() method is used to asynchronously append the given data to a file. A new file is created if it does not exist. The options parameter can be used to modify the behavior of the operation.
Syntax:
fs.appendFile(path, data[, options], callback)

5) Rename Method:
The fs.rename() method is used to asynchronously rename a file at the given old path to a given new path. It will overwrite the destination file if it already exists.
Syntax:
fs.rename(oldPath, newPath, callback).

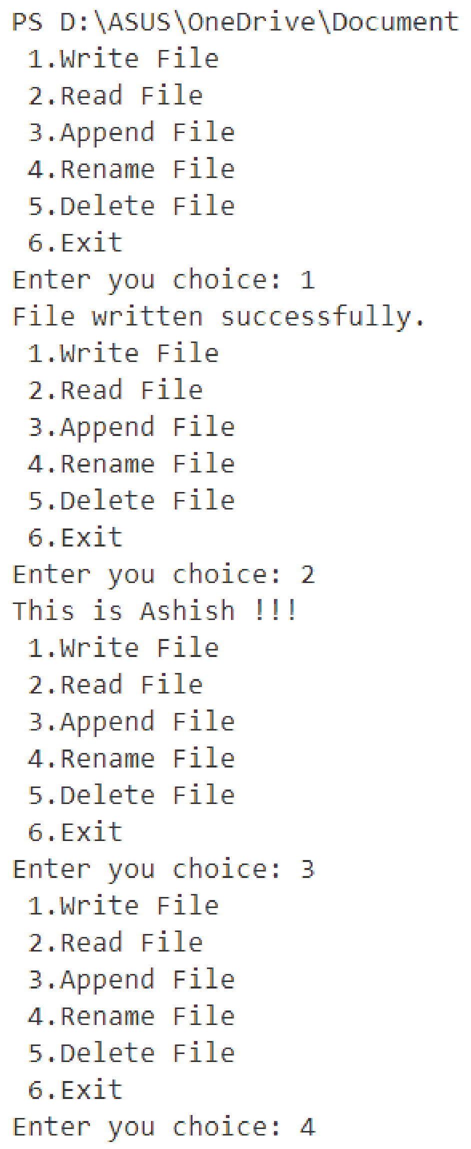
6) Unlink Method:
The fs.unlink() method is used to remove a file or symbolic link from the filesystem. This function does not work on directories, therefore it is recommended to use fs.rmdir() to remove a directory.
Syntax:
fs.unlink(path, callback)

A) Write a menu driven program to implement the following using Synchronous file functions.

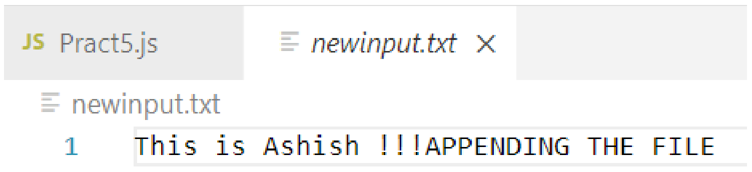
Code:-

```
var rl = require('readline-sync');
do {
   console.log(" 1.Write File\n 2.Read File\n 3.Append File\n 4.Rename File\n 5.Delete File\n
6.Exit");
   var ch = parseInt(rl.question("Enter  you choice: "));
   switch (ch) {
      case 1:
         var fs = require('fs');
         var content = "This is Ashish !!!"
         fs.writeFileSync('input.txt',  content); console.log('File written successfully.');
         break;
      case 2:
         var fs = require('fs');
         var data = fs.readFileSync('input.txt');  console.log(data.toString());
         break;
      case 3:
         var fs = require('fs');
         var content = "APPENDING THE FILE"; fs.appendFileSync('input.txt',  content);
console.log("File Appended Successfully");
         break;
      case 4:
         var fs = require('fs'); fs.renameSync('input.txt',  'newinput.txt'); console.log('File renamed
successfully');
         break;
      case 5:
         var fs = require("fs");
         var filename = 'newinput.txt'; fs.unlinkSync(filename);
         console.log('The file is deleted successfully!!')
         break;
      case 6:
         console.log("Thank you! BYE!!")
         break;
   }
}
while (ch != 6);
```

OUTPUT:-



```
PS D:\ASUS\OneDrive\Document
 1.Write File
 2.Read File
 3.Append File
 4.Rename File
 5.Delete File
 6.Exit
Enter you choice: 1
File written successfully.
 1.Write File
 2.Read File
 3.Append File
 4.Rename File
 5.Delete File
 6.Exit
Enter you choice: 2
This is Ashish !!!
 1.Write File
 2.Read File
 3.Append File
 4.Rename File
 5.Delete File
 6.Exit
Enter you choice: 3
 1.Write File
 2.Read File
 3.Append File
 4.Rename File
 5.Delete File
 6.Exit
Enter you choice: 4
```

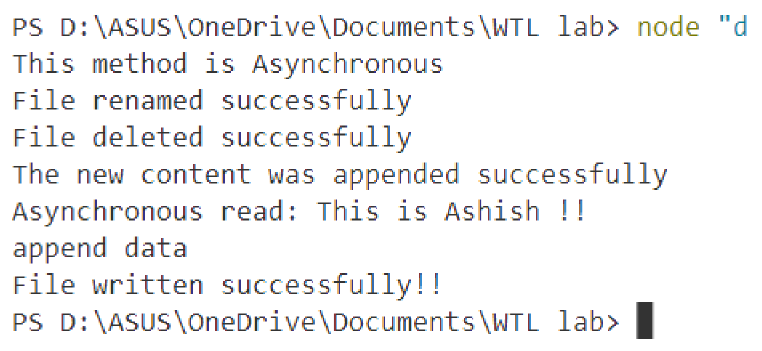


```
This is Ashish !!!APPENDING THE FILE
```

B) Implement the previous program using Asynchronous File functions.
Code:-

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

//Write a File using Nodejs
var content= "This is Ashish !!";
fs.writeFile('input.txt', content , (err) => {
if (err)
throw err;
console.log('File written successfully!!');
});
// Read a File using Nodejs var fs = require("fs");
fs.readFile('input.txt', function (err, data) { if (err) {
   return console.error(err);
   }
   console.log("Asynchronous read: " + data.toString())
   });

//append file
   new_data="append data";
   fs.appendFile('input.txt','\n' + new_data, (err) => {
   if(err)
   throw err;
   console.log('The new content was appended successfully');
   });
//RENAME
   var fs = require('fs');
   fs.rename('input.txt', 'newinput.txt', (err) => {
   if (err)
   throw err;
   console.log('File renamed successfully');
   });
   console.log("This method is Asynchronous");
//DELETE
   var fs = require('fs');
   var filename = 'newinput.txt';
   fs.unlink(filename, (err) => {
   if (err)
   throw err;
   console.log('File deleted successfully');
   });
```

OUTPUT:-



```
PS D:\ASUS\OneDrive\Documents\WTL lab> node "d
This method is Asynchronous
File renamed successfully
File deleted successfully
The new content was appended successfully
Asynchronous read: This is Ashish !!
append data
File written successfully!!
PS D:\ASUS\OneDrive\Documents\WTL lab> 
```

<u>WEB TECHNOLOGIES</u>
<u>PRACTICAL NO: 06</u>

AIM: Create a HTTP Server and serve HTML,CSV,JSON, PDF Files, video using piping and render html pages using Routing.

THEORY:-
Web Server:-
A web server is software that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World Wide Web. The main job of a web server is to display website content through storing, processing and delivering webpages to users. Besides HTTP, web servers also support SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol), used for email, file transfer and storage. Web servers are used in web hosting, or the hosting of data for websites and web-based applications -or web applications.

Http Request and Response:-

When a client makes a request, the HTTP server emits a request event, passing in HTTP request and HTTP response objects.The HTTP request object allows you to query some of the request properties. The HTTP response allows you to build an HTTP
response that will be sent to the client.

req.url: This property contains the requested URL as a string. It does not contain the schema, hostname, or port, but it contains everything after that. (res.end(req.url);)

req.method: This contains the HTTP method used on the request. It can be, for example, GET, POST, DELETE, or HEAD.

req.headers: This contains an object with a property for
every HTTP header on the request.
[var util = require('util');
res.end(util.inspect(req.headers));]

Piping: Its purpose is to attach a writeable stream to a readable stream allowing to pass the readable stream data to the writeable stream.

Routing: A router is a JavaScript object that maps URLs to functions. The router calls a function based on the URL. In the past, a web application was a series of interconnected pages. This could either be static pages or dynamic pages that are generated on the server.

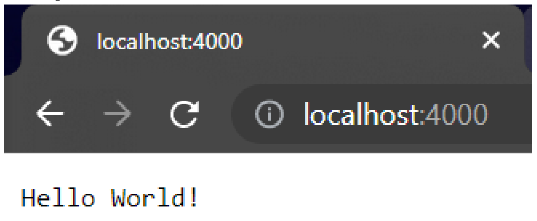
i) Create a HTTP Server and serve a HTML,CSV,JSON and PDF Files.

a)CREATE SERVER , PRINT HELLO WORLD.

Code:-

```
var http=require('http');
var server=http.createServer();
server.on('request',function(req,res){
   res.writeHead(200,{'Content-Type':'text/plain'});
   res.write('Hello World!');
   res.end();
});
server.listen(4000);
```

Output:-



```
Hello World!
```

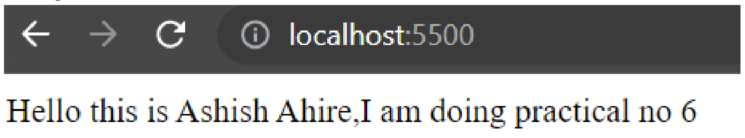
b) SERVE A HTML FILE
Code:-
htmlServer.js(//File Name)

```
let http = require('http');
let fs = require('fs');
let handleRequest = (request, response) => {
   response.writeHead(200, { 'Content-Type': 'text/html' });
   fs.readFile('./text.html', null, function (error, data) {
      if (error) {
         response.writeHead(404);
         response.write('Sorry!File not Found!');
      } else {
         response.write(data);
      }
      response.end();
   });
};
http.createServer(handleRequest).listen(5500);
```

text.html(//File Name)

```
<html>

<head>
   <title>
      Serving Html
   </title>

<body>
   <p>Hello this is Ashish Ahire,I am doing practical no 6</p>
</body>
</head>

</html>
```

Output:-



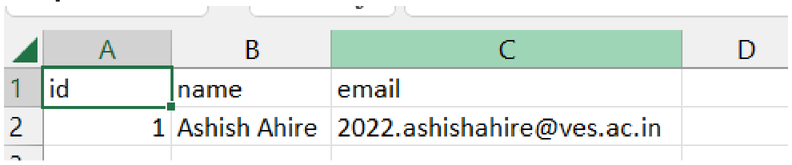
```
Hello this is Ashish Ahire,I am doing practical no 6
```

c)SERVE A CSV FLE
Code:-
csvServer.js(//File Name)

```
let http = require('http');
const requestListener = function (req, res) {
   res.setHeader("Content-Type", "text/csv");
   res.setHeader("Content-Disposition", "attachment;filename=input.csv");
   res.writeHead(200);
   res.write('id,name,email\n1,Ashish Ahire,2022.ashishahire@ves.ac.in,');
   res.end();
};
http.createServer(requestListener).listen(4000);
```

create a blank input.csv file

Output:

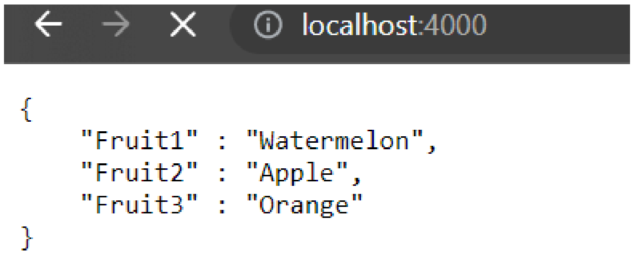


	A	B	C	D
1	id	name	email	
2	1	Ashish Ahire	2022.ashishahire@ves.ac.in	

d) SERVE JSON FILE
Code:-

```
let http = require('http');
let fs = require('fs');
const requestListener = function (req, res) {
   res.setHeader("Content-Type", "application/json");
   res.writeHead(200);
   fs.readFile('./omkar.json', null, function (error, data) {
      if (error) {
         res.writeHead(404);
         res.write('Whoops!File not Found!');

      } else {
         res.write(data);
      }
   });
}
http.createServer(requestListener).listen(4000);
```

Output:-



```
{
    "Fruit1" : "Watermelon",
    "Fruit2" : "Apple",
    "Fruit3" : "Orange"
}
```

e) SEREVE PDF FILE
Code:-

```
var http = require('http');
var fs = require('fs');
http.createServer((req, res) => {
   res.writeHead(200, { "Content-Type": "application/pdf" });
   fs.readFile('application.pdf', (error, data) => {
      if (error) {
         res.writeHead(404);
         res.write('Whoops!File not Found!');

      } else {
         res.write(data);
         res.end();
      }
   });
}).listen(4000);
```

Output:-

ii) Create a HTTP Server and stream a video file using piping.
CODE:

```
var fs=require('fs');
require('http').createServer(function(req,res){
   res.writeHead(200,{'Content-Type':'Video/mp4'});
   var rs=fs.createReadStream('test.mp4');
   rs.pipe(res);
}).listen(2000);
```

Output:

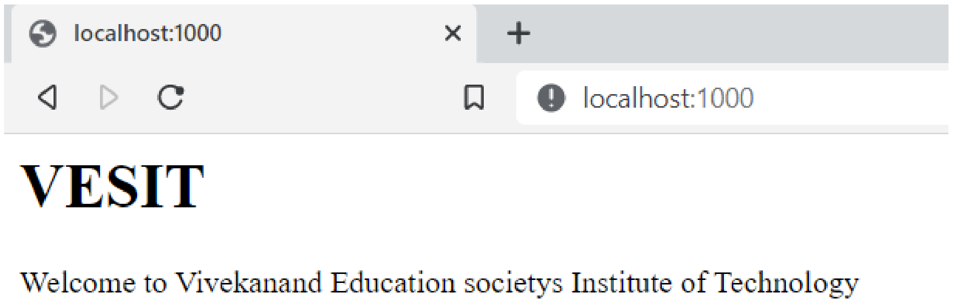
iii) Develop 3 HTML Web Pages for college home page(write about college & dept),about me and contact. Create a server and render these pages using Routing.

Code:-

collegeServer.js(//File Name)

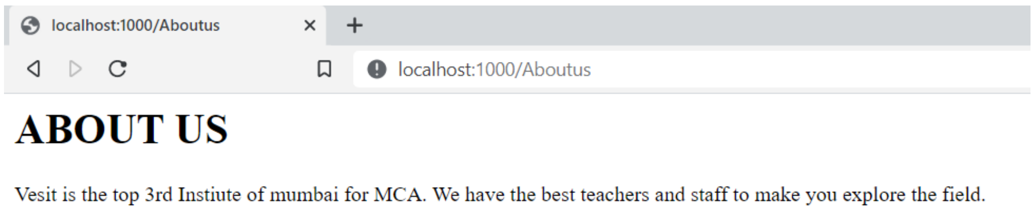
```
var http = require('http');
var server = http.createServer(function (req, res) {
   if (req.url == '/') {
      res.writeHead(200, { 'content-type': 'text/html' });
      res.write('<html><body><h1>VESIT</h1><p> Welcome to Vivekanand Education societys
Institute of Technology </body></html>');
      res.end();
   }
   else if (req.url == '/Aboutus') {
      res.writeHead(200, { 'content-type': 'text/html' });
      res.write('<html><body><h1>ABOUT US</h1><p>Vesit is the top 3rd Instiute of mumbai
for MCA. We have the best teachers and staff to make you explore the
field.</p></body></html>');
      res.end();
   }
   else if (req.url == '/contactus') {
      res.writeHead(200, { 'content-type': 'text/html' });
      res.write('Institute is in Chembur, Contact number- 3482934729');
      res.end();
   }
   else
      res.end('Invalid Request');
});
server.listen(1000);
console.log("Node js is running.....");
```

Output:



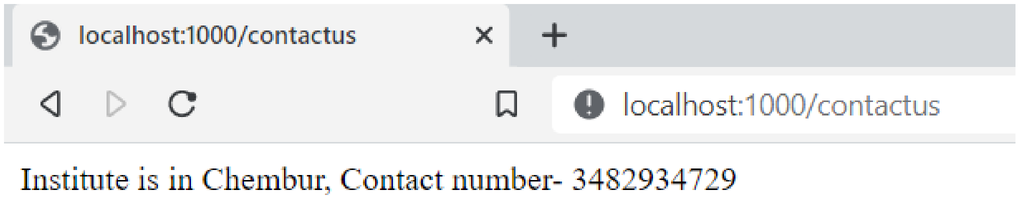
VESIT

Welcome to Vivekanand Education societys Institute of Technology


localhost:1000/Aboutus

ABOUT US

Vesit is the top 3rd Instiute of mumbai for MCA. We have the best teachers and staff to make you explore the field.


localhost:1000/contactus

Institute is in Chembur, Contact number- 3482934729

Conclusion:
I have learnt to create a HTTP Server and serve HTML,CSV,JSON, PDF Files. I have also learnt to display video using piping and render html pages using Routing.

WEB TECHNOLOGIES
PRACTICAL NO: 07
AIM: Implement Mysql query in node js.

PREREQUISITE:-
1.Open cmd from the path in which we are going to do practical and write npm init -y
2.Write npm install mysql in terminal.
3.Open xampp and start mysql and apache.

THEORY:-
Node.js can be used in database applications. One of the most popular databases is MySQL .You can access MySql 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.
Open the Command Terminal and execute the following:
```
npm install mysql
```

Now you have downloaded and installed a mysql database driver.
Node.js can use this module to manipulate the MySQL database:
```
var mysql = require('mysql');
```

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

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

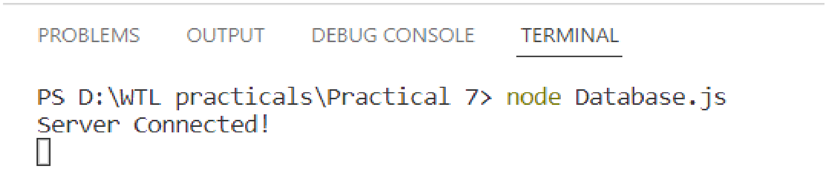
Use SQL statements to read from (or write to) a MySQL database. This is also Called “to query” the database.
```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: "+ result);
  });
```

1)Create an application to establish a connection with the MySQL database and perform basic database operations on it(student consisting rollno, name, address, Contact No,Dept),insert 10 records, update a particular student's record, delete a record.Display all Records, Order by Student name.

Database.js (//File Name)
Code:-

```
var mysql = require('mysql');
var con = mysql.createConnection({
   host: "localhost",
   user: "root",
   password: ""
});
con.connect(function (err) {
   if (err) throw err;
   console.log("Server Connected!");
});
```

Output:-

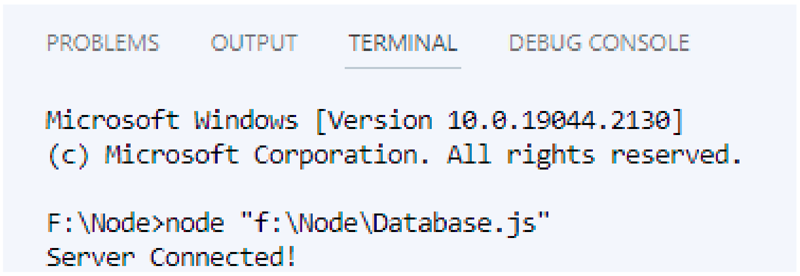


```
PS D:\WTL practicals\Practical 7> node Database.js
Server Connected!
```

Createtable.js (// File Name)
Code:-

```
var mysql = require('mysql');
var con = mysql.createConnection({
   host: "localhost",
   user: "root",
   password: ""
});
con.connect(function (err) {
   if (err) throw err;
   console.log("Server Connected!");
   con.query("CREATE DATABASE ashishrr",
      function (err, result) {
         if (err) throw err;
         console.log("Database ashish created");
      });
});
```

Output:-



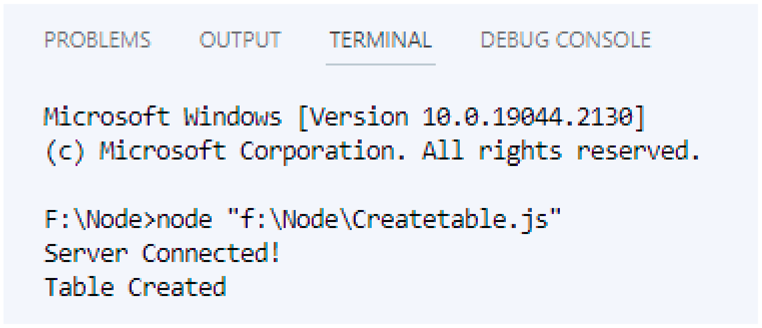
```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

F:\Node>node "f:\Node\Database.js"
Server Connected!
```

Createtable.js (// File Name)

Code:-

```
var mysql = require('mysql');
const { VARCHAR } = require('mysql/lib/protocol/constants/types');
var con = mysql.createConnection({
   host: "localhost",
   user: "root",
   password: "",
   database: "ashishrr"
});
con.connect(function (err) {
   if (err) throw err;
   console.log("Server Connected!");
   var sql = "CREATE TABLE studentashish(Roll_No INTEGER(10),Name
VARCHAR(10),Address VARCHAR(25),Contact INTEGER(10),Dept VARCHAR(15))";
   con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("Table Created");
   });

});
```

Output:-



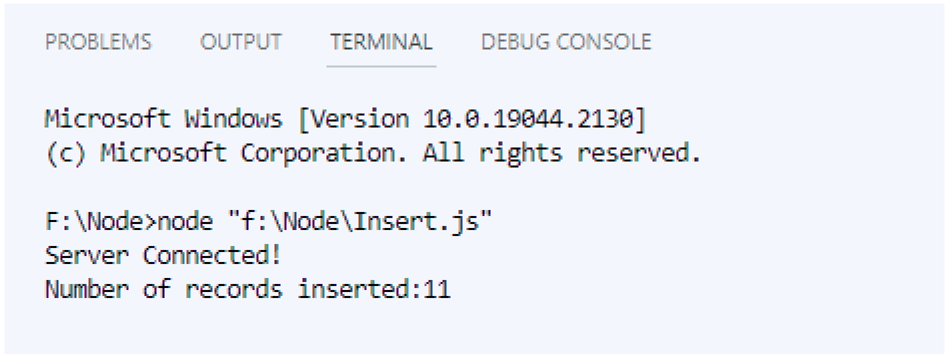
```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

F:\Node>node "f:\Node\Createtable.js"
Server Connected!
Table Created
```

Insert.js (//File Name)

Code:-

```
var mysql = require('mysql');
const { VARCHAR } = require('mysql/lib/protocol/constants/types');
var con = mysql.createConnection({
   host: "localhost",
   user: "root",
   password: "",
   database: "ashishrr"
});
con.connect(function (err) {
   if (err) throw err;
   console.log("Server Connected!");
   var sql = "INSERT INTO studentashish(Roll_No,Name,Address,Contact,Dept)VALUES ?";
   var values = [
      [10, Ashish, 'Panvel', 9846532256, 'MCA'],
      [11, 'Sailee', 'Goregaon', 9856721348, 'Mtech'],
      [12, 'Anish', 'Andheri', 9357382934, 'BSc'],
      [13, Mohit, 'Pune', 8945275630, 'BCom'],
      [14, 'Avantika', 'Chembur', 9645284658, 'BCA'],
      [15, 'Shivanee', 'UP', 3462846573, 'Mtech'],
      [16, 'Shubham', 'Devrukh', 8734021745, 'Mtech'],
      [17, 'Pranit', 'Ratnagiri', 8956346783, 'MCA'],
      [18, 'Suyog', 'Kanpur', 3498263547, 'MCA'],
      [19, 'Jayesh', 'Vashi', 4562094621, 'Mtech'],
      [20, 'Kanika', 'Chiplun', 6529465125, 'BSc'],
   ];
   con.query(sql, [values], function (err, result) {
      if (err) throw err;
      console.log("Number of records inserted:" + result.affectedRows);

   });
});
```

Output:-



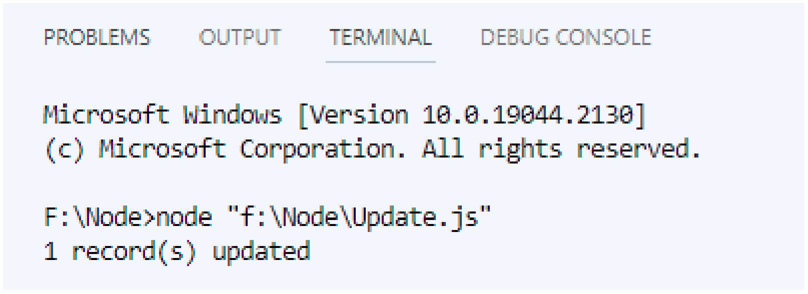
```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

F:\Node>node "f:\Node\Insert.js"
Server Connected!
Number of records inserted:11
```

Update.js (//File Name)

Code:-

```
var mysql = require('mysql');
var con = mysql.createConnection({
   host: "localhost",
   user: "root",
   password: "",
   database: "ashishrr"
});
con.connect(function (err) {
   if (err) throw err;
   var sql = "UPDATE studentashish SET name='Kashi' WHERE Name='Shivanee'";
   con.query(sql, function (err, result) {
      if (err) throw err;
      console.log(result.affectedRows + " record(s) updated");

   });
});
```

Output:-

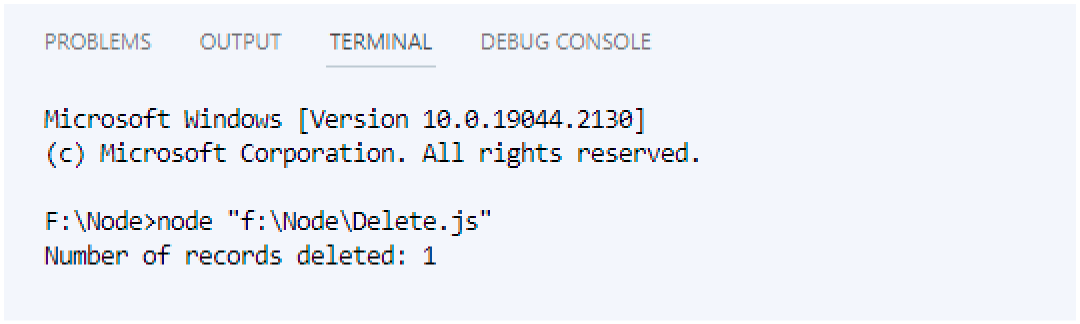


```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

F:\Node>node "f:\Node\Update.js"
1 record(s) updated
```

Delete.js (//File Name)
Code:-

```
var mysql = require('mysql');
var con = mysql.createConnection({
   host: "localhost",
   user: "root",
   password: "",
   database: "ashishrr"
});
con.connect(function (err) {
   if (err) throw err;
   var sql = "DELETE FROM studentashish WHERE Name='Kanika'";
   con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("Number of records deleted: " + result.affectedRows);
   });
});
```

Output:-

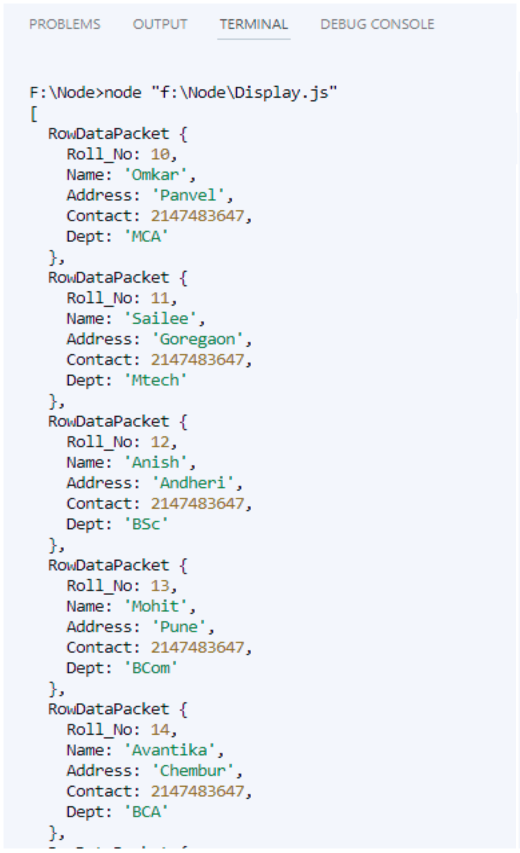


```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

F:\Node>node "f:\Node\Delete.js"
Number of records deleted: 1
```

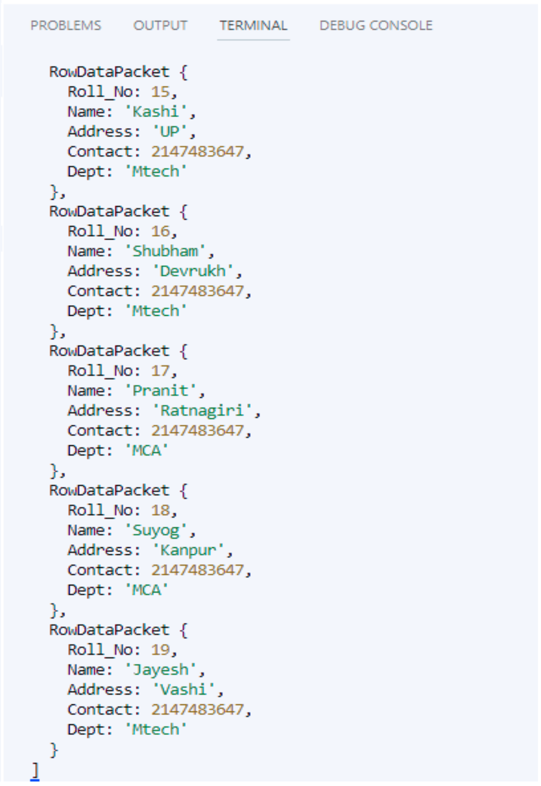
Display.js (//File Name)
Code:-

```
var mysql = require('mysql');
var con = mysql.createConnection({
   host: "localhost",
   user: "root",
   password: "",
   database: "ashishrr"
});
con.connect(function (err) {
   if (err) throw err;
   con.query("SELECT * FROM studentashish", function (err, result, fields) {
      if (err) throw err;
      console.log(result);
   });
});
```

Output:-



```
F:\Node>node "f:\Node\Display.js"
[
  RowDataPacket {
    Roll_No: 10,
    Name: 'Omkar',
    Address: 'Panvel',
    Contact: 2147483647,
    Dept: 'MCA'
  },
  RowDataPacket {
    Roll_No: 11,
    Name: 'Sailee',
    Address: 'Goregaon',
    Contact: 2147483647,
    Dept: 'Mtech'
  },
  RowDataPacket {
    Roll_No: 12,
    Name: 'Anish',
    Address: 'Andheri',
    Contact: 2147483647,
    Dept: 'BSc'
  },
  RowDataPacket {
    Roll_No: 13,
    Name: 'Mohit',
    Address: 'Pune',
    Contact: 2147483647,
    Dept: 'BCom'
  },
  RowDataPacket {
    Roll_No: 14,
    Name: 'Avantika',
    Address: 'Chembur',
    Contact: 2147483647,
    Dept: 'BCA'
  },
```



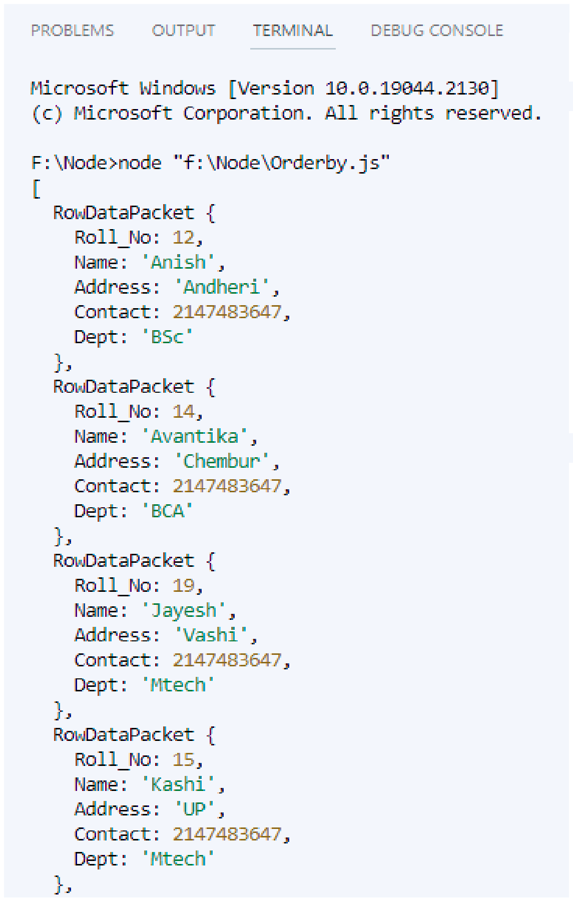
```
  RowDataPacket {
    Roll_No: 15,
    Name: 'Kashi',
    Address: 'UP',
    Contact: 2147483647,
    Dept: 'Mtech'
  },
  RowDataPacket {
    Roll_No: 16,
    Name: 'Shubham',
    Address: 'Devrukh',
    Contact: 2147483647,
    Dept: 'Mtech'
  },
  RowDataPacket {
    Roll_No: 17,
    Name: 'Pranit',
    Address: 'Ratnagiri',
    Contact: 2147483647,
    Dept: 'MCA'
  },
  RowDataPacket {
    Roll_No: 18,
    Name: 'Suyog',
    Address: 'Kanpur',
    Contact: 2147483647,
    Dept: 'MCA'
  },
  RowDataPacket {
    Roll_No: 19,
    Name: 'Jayesh',
    Address: 'Vashi',
    Contact: 2147483647,
    Dept: 'Mtech'
  }
]
```

Orderby.js (//File Name)

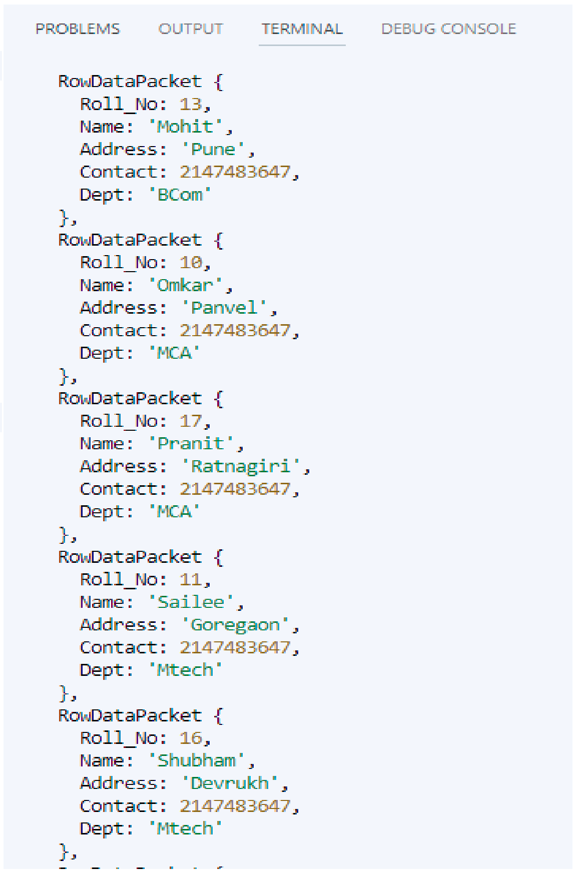
Code:-

```
var mysql = require('mysql');
var con = mysql.createConnection({
   host: "localhost",
   user: "root",
   password: "",
   database: "ashishrr"
});
con.connect(function (err) {
   if (err) throw err;
   con.query("SELECT * FROM studentashish ORDER BY Name;", function (err, result, fields) {
      if (err) throw err;
      console.log(result);
   });
});
```

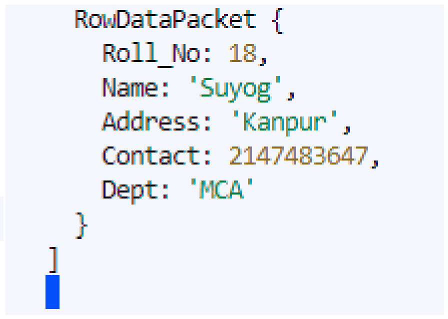
Output:-



```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

F:\Node>node "f:\Node\Orderby.js"
[
  RowDataPacket {
    Roll_No: 12,
    Name: 'Anish',
    Address: 'Andheri',
    Contact: 2147483647,
    Dept: 'BSc'
  },
  RowDataPacket {
    Roll_No: 14,
    Name: 'Avantika',
    Address: 'Chembur',
    Contact: 2147483647,
    Dept: 'BCA'
  },
  RowDataPacket {
    Roll_No: 19,
    Name: 'Jayesh',
    Address: 'Vashi',
    Contact: 2147483647,
    Dept: 'Mtech'
  },
  RowDataPacket {
    Roll_No: 15,
    Name: 'Kashi',
    Address: 'UP',
    Contact: 2147483647,
    Dept: 'Mtech'
  },
```



```
  RowDataPacket {
    Roll_No: 13,
    Name: 'Mohit',
    Address: 'Pune',
    Contact: 2147483647,
    Dept: 'BCom'
  },
  RowDataPacket {
    Roll_No: 10,
    Name: 'Omkar',
    Address: 'Panvel',
    Contact: 2147483647,
    Dept: 'MCA'
  },
  RowDataPacket {
    Roll_No: 17,
    Name: 'Pranit',
    Address: 'Ratnagiri',
    Contact: 2147483647,
    Dept: 'MCA'
  },
  RowDataPacket {
    Roll_No: 11,
    Name: 'Sailee',
    Address: 'Goregaon',
    Contact: 2147483647,
    Dept: 'Mtech'
  },
  RowDataPacket {
    Roll_No: 16,
    Name: 'Shubham',
    Address: 'Devrukh',
    Contact: 2147483647,
    Dept: 'Mtech'
  },
```

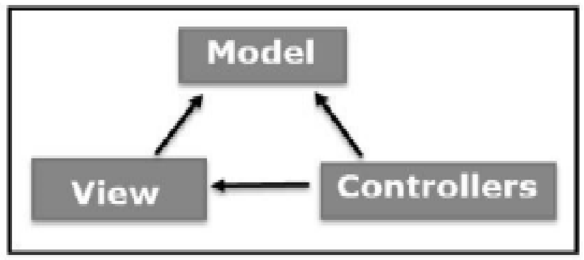


```
  RowDataPacket {
    Roll_No: 18,
    Name: 'Suyog',
    Address: 'Kanpur',
    Contact: 2147483647,
    Dept: 'MCA'
  }
]
```

Conclusion:-
In this practical I learned about creating connection , creating database , creating table, inserting, updating deleting data, displaying multiple data.

WEB TECHNOLOGIES
PRACTICAL NO: 08
AIM: Implementation of Angular JS.
THEORY:
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 succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. AngularJS teaches the browser new syntax through a construct we call directives.

MVC Pattern stands for Model-View-Controller Pattern.
Model - This layer represents an object for carrying data and may hold data logic to update controller in case of data changes.
View - This layer holds the data presentation that the model contains. **Controller -** This is a middleman between model and view and controls the data
Flow into data object and updates the view whenever the data is updated

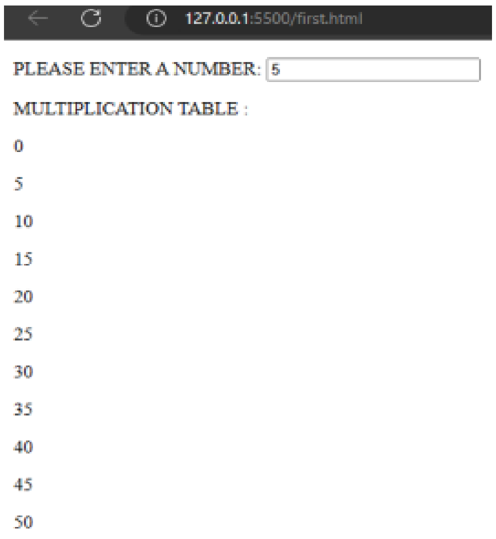


In Angular Js, expressions are classified as the following types:
1. Number Expressions
If any expression is using the number and the operators like, -, *, % etc. those expressions we will call it as number expressions.
2. String Expressions
The string expression is used to perform operations on string values.
3. Object Expressions
The object expressions basically used to hold object properties and evaluates at the view where they are used.
4. Array Expressions
The array expressions are used to hold an array of object values.

1. Create an Angular JS application to accept a number from user and display its multiplication table (Use ng-init and ng-repeat).
CODE:

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>  <body>
<div ng-app="" ng-init="multiply=[0,1,2,3,4,5,6,7,8,9,10]">
<p>PLEASE ENTER A NUMBER: <input type="text" ng-model="number"></p>
<p>MULTIPLICATION TABLE :</p>
<p ng-repeat="x in multiply">
{{number * x }} </p>
</div>
</body>
</html>
```

OUTPUT:



127.0.0.1:5500/first.html

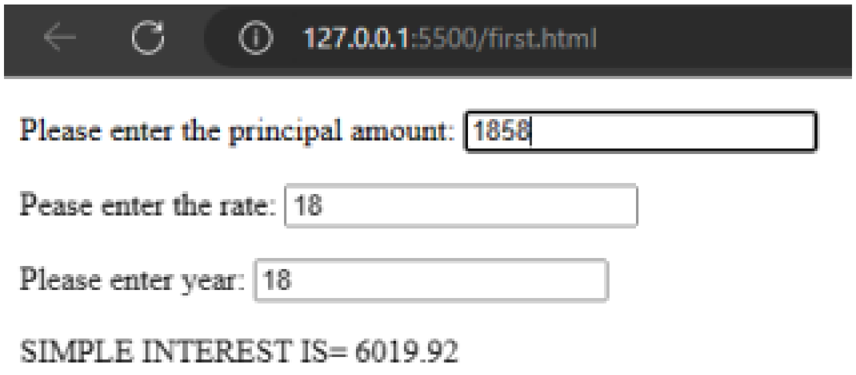
```
PLEASE ENTER A NUMBER: 5
MULTIPLICATION TABLE :
0
5
10
15
20
25
30
35
40
45
50
```

2. Create an Angular JS application to calculate Simple Interest .Accept a principal amount, Rate of interest and year from user.

CODE:

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>  <body>
<div ng-app="">
<p>Please enter the principal amount: <input type="text" ng-model="principal"></p>  <p>Pease
enter the rate: <input type="text" ng-model="rate"></p>
<p>Please enter year: <input type="text" ng-model="year"></p>
<p>SIMPLE INTEREST IS=
{{(principal * rate * year) /100 }}
</p>
</div>
</body>
</html>
```

OUTPUT:



```
127.0.0.1:5500/first.html

Please enter the principal amount: 1858

Pease enter the rate: 18

Please enter year: 18

SIMPLE INTEREST IS= 6019.92
```

CONCLUSION:

I have learnt the implementation of Angular JS.

WEB TECHNOLOGIES

PRACTICAL NO: 09

AIM: Implementation of Angular JS Filters.

THEORY:

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. The following table lists important filters.

1.Number Filter

A number filter formats numeric data as text with comma and specified fraction size. Syntax:{{ number_expression | number:fractionSize}}
If a specified expression does not return a valid number then number filter displays an empty string.

2.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.
Syntax:{{ expression | currency : 'currency_symbol' : 'fraction'}}

3.Date filter

Formats date to string based on the specified format.
Syntax:{{ date_expression | date : 'format'}}

4.OrderBy filter

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

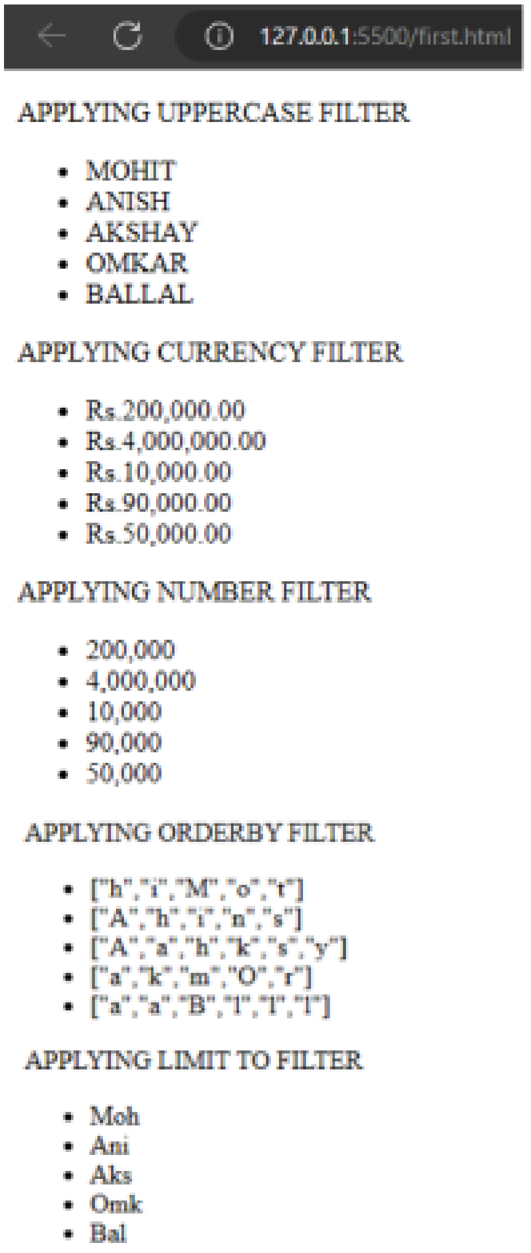
AngularJS Custom Filter

In angularjs, custom filters are used to create our own filters to format data and it will increase reusability of code. Suppose in angularjs application if we need common format of data in multiple pages then it's better to use custom filter. By creating custom filters we can reduce redundancy of code in angularjs applications.
Syntax of AngularJS Custom Filter :{{ customexpression | customfilter }}

1.Create an Array to store the information of 5 employees (empid, name, salary,doj) and apply relevant filters to:
i) convert name to uppercase
ii) apply currency and number filter
iii) order by
iv) LimitTo
CODE:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>    <div ng-app="" ng-init="emp=[
{id:1,name:'Mohit',salary:200000,DOJ:'MAY  31, 2002'},
{id:2,name:'Anish',salary:4000000,DOJ:'JULY  18, 2006'},
{id:3,name:'Akshay',salary:10000,DOJ:'JUNE  29, 2008'},
{id:4,name:'Omkar',salary:90000,DOJ:'MAR  16, 2002'},
{id:5,name:'Ballal',salary:50000,DOJ:'OCT  30, 2002'}]">
    <p>APPLYING UPPERCASE FILTER</p>
    <ul>
      <li ng-repeat="x in emp">
        {{x.name | uppercase }}
      </li>
    </ul>
    <p>APPLYING CURRENCY FILTER</p>
    <ul>
      <li ng-repeat="x in emp">
        {{x.salary | currency: 'Rs.' }}
      </li>
    </ul>
    <p>APPLYING NUMBER FILTER</p>
    <ul>
      <li ng-repeat="x in emp">
        {{x.salary | number }}
      </li>
    </ul>
    <p>APPLYING ORDERBY FILTER</p>
    <ul>
      <li ng-repeat="x in emp">
        {{x.name | orderBy }}
      </li>
    </ul>
    <p>APPLYING LIMIT TO FILTER</p>
```

OUTPUT:

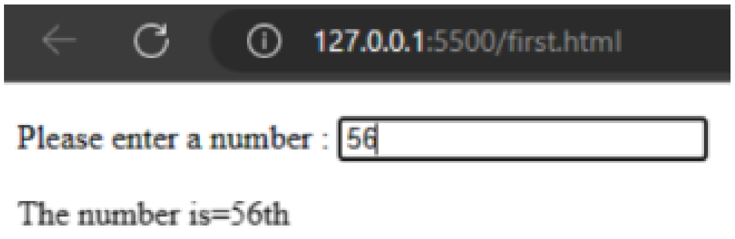


```
APPLYING UPPERCASE FILTER

  • MOHIT
  • ANISH
  • AKSHAY
  • OMKAR
  • BALLAL

APPLYING CURRENCY FILTER

  • Rs.200,000.00
  • Rs.4,000,000.00
  • Rs.10,000.00
  • Rs.90,000.00
  • Rs.50,000.00

APPLYING NUMBER FILTER

  • 200,000
  • 4,000,000
  • 10,000
  • 90,000
  • 50,000

APPLYING ORDERBY FILTER

  • ["h","i","M","o","t"]
  • ["A","h","i","n","s"]
  • ["A","a","h","k","s","y"]
  • ["a","k","m","O","r"]
  • ["a","a","B","l","l","l"]

APPLYING LIMIT TO FILTER

  • Moh
  • Ani
  • Aks
  • Omk
  • Bal
```

2. Create a custom filter ordinal and demonstrate its use
CODE:

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<body><div ng-app="myApp">
<p>Please enter a number : <input type="text" ng-model="number"></p> <p>The number
is={{number| ordinal}}</p></div><script>
var app = angular.module('myApp', []);
app.filter('ordinal', function() {
var app = angular.module('myApp', []);
return function(number) {
if(isNaN(number) || number < 1) {
return number;
} else { var lastDigit = number % 10;
if(lastDigit === 1) {
return number + 'st'
} else if(lastDigit === 2) {
return number + 'nd'
} else if (lastDigit === 3) {
return number + 'rd'
} else if (lastDigit > 3) {
return number + 'th'
}}}});
</script></body></html>
```

OUTPUT:



127.0.0.1:5500/first.html

Please enter a number : 56

The number is=56th

CONCLUSION:
I have understood the concept of filters and how to create one.

WEB TECHNOLOGIES
PRACTICAL NO: 10
AIM: Implementation of Angular Js Directives.
THEORY:
Directives:

1. Directives are markers(such as attributes, tags and class names) that tells Angular JS to attach a given behaviour to a DOM element (or transform it,replace it,etc.).
2. AngularJS lets you extend HTML with new attributes called **Directives**.
3. AngularJS has a set of built-in directives which offers functionality to your applications.
4. AngularJS also lets you define your own directives.

AngularJS directives are extended HTML attributes with the prefix ng-.

1. The **ng-app** directive initializes an AngularJS application.It also tells AngularJS that the <div> element is the "owner" of the AngularJS application.It defines the root element of an AngularJS application and auto bootstrap (automatically initialize) the application when a web page is loaded.
2. The **ng-init** directive initializes application data.
3. The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
4. The **ng-repeat** directive actually clones HTML elements once for each item in a collection.
5. The **ng-controller** directive defines which controller will be in charge of your view.

Ng-controller Directive
The ng-controller directive adds a controller to your application.
In the controller you can write code, and make functions and variables, which will be parts of an object, available inside the HTML element. In AngularJS this object is called a scope.
SCOPE
The scope is the binding part between the HTML (view) and the JavaScript (controller).
The scope is an object with the available properties and methods.
Ng-style directive

- The ng-style directive specifies the style attribute for the HTML element.
- The value of the ng-style attribute must be an object, or an expression returning an object.
- The object consists of CSS properties and values, in key value pairs.

Syntax
<element ng-style="expression"></element>
Mouse Event directives:
To perform mouse operations in AngularJS, we can use different mouse directives available.

1. **ng-mousedown** - executes when mouse is clicked on the element.
2. **ng-mouseenter** - executes when mouse enters into the element area.
3. **ng-mouseleave** - executes when mouse leaves the element area.
4. **ng-mousemove** - executes when mouse is moved on the element area.
5. **ng-mouseover** - executes when mouse is kept over the element.

6. **ng-mouseup** - executes when mouse is left after clicking the element.

Keyboard Event directives:
The **ng-keydown** directive tells AngularJS what to do when the keyboard is used on the specific HTML element.
The **ng-keydown** directive from AngularJS will not override the element's original onkeydown event, both will be executed.
Syntax
<element ng-keydown="expression"></element>

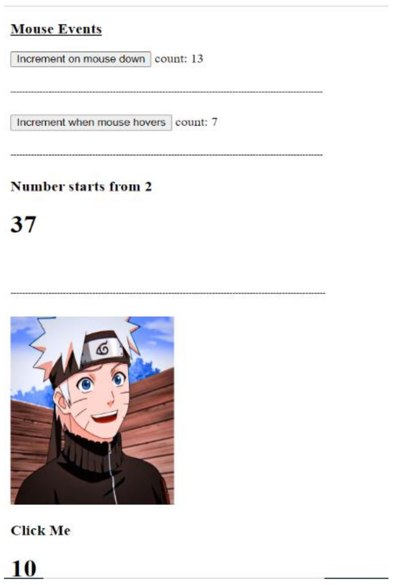
Restrictions:
You can restrict your directives to only be invoked by some of the methods.
The legal restrict values are:

- E for Element name
- A for Attribute
- C for Class
- M for Comment

By default the value is EA, meaning that both Element names and attribute names can invoke the directive.
Q.1] Create an application to demonstrate directives Style, mouse and keyboard directive
CODE:
```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body ng-app="myApp" ng-controller="myCtrl">
   <h3><u>Mouse Events</u></h3>
<button ng-mousedown = "count = count + 1" ng-init = "count = 0">
   Increment on mouse down
</button>
count: {{count}}
<br>
<h6>--------------------------------------------------------------------------------------------------------------</h6>
<button ng-mouseenter="increase = increase +1" ng-init="increase = 0">
   Increment when mouse hovers
</button>
count: {{increase}}
<br>
<h6>--------------------------------------------------------------------------------------------------------------</h6>
<div ng-mousemove="add=add+1"  ng-init="add = 2">
   <h3>Number starts from 2</h3>
</div>
<h1>{{add}}</h1>
<br>
```

```
<h6>------------------------------------------------------------------------------------------------------------</h6>
<div ng-mouseup = "element = element + 1" ng-init = "element = 10">
   <img src="Naruto.jpg" alt="naruto" width="200" height="250"> <br> <h3>Click Me</h3>
</div>
<h1>{{element}}</h1>
<br>
<h6>------------------------------------------------------------------------------------------------------------</h6>
<br>
<h1><u>Keyboard Events</u></h1>
<input ng-keydown = "num = num + 1" ng-init="num=0"/>
<h3>{{num}}</h3>
<p>num will be incremented everytime a key is pressed in the input field</p>
<br>
<h6>------------------------------------------------------------------------------------------------------------</h6>
<h1><u>Style events</u></h1>
<h3 ng-style="myObj">Welcome to the world of Angular JS</h3>
<br>
<h6>------------------------------------------------------------------------------------------------------------</h6>
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
  $scope.myObj = {
    "color" : "white",
    "background-color" : "coral",
    "font-size" : "60px",
    "padding" : "50px"
  }
});
</script>

</body>
</html>
```

OUTPUT:



Mouse Events

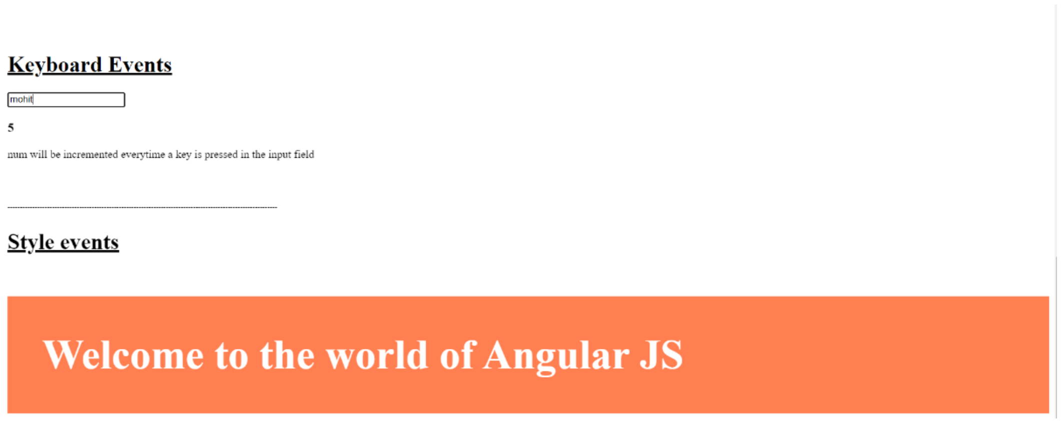
count: 13

count: 7

Number starts from 2

37

10



Keyboard Events

mohit

5

num will be incremented everytime a key is pressed in the input field

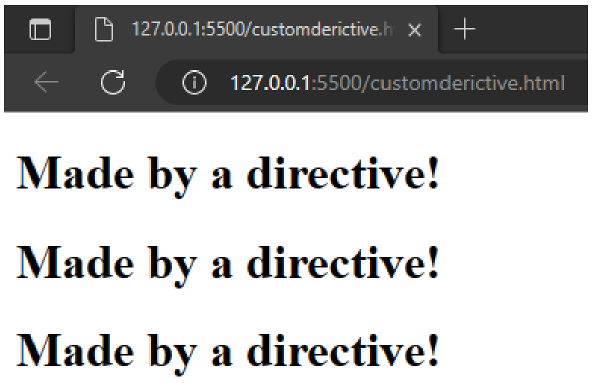
Style events

Welcome to the world of Angular JS

Q.2] create an application to demonstrate custom directive
CODE:

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body ng-app="myApp">
<my-test-directive></my-test-directive>
<div my-test-directive></div>
<!-- directive: my-test-directive -->
<script>
var app = angular.module("myApp",  []);
app.directive("myTestDirective",  function() {
return {
restrict : "MEA",
replace: "true",
template : "<h1>Made by a directive!</h1>"
};
});
</script>
</body>
</html>
</body>
</html>
```

OUTPUT:


127.0.0.1:5500/customderictive.h
127.0.0.1:5500/customderictive.html

```
Made by a directive!

Made by a directive!

Made by a directive!
```

CONCLUSION:
I have learnt to create application that demonstrates directives style, mouse and keyboard directives and custom directive.

WEB TECHNOLOGIES
PRACTICAL NO: 11

AIM: Implement Controllers in Angular Js.

THEORY:

Controllers:

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

Attach Behaviours:

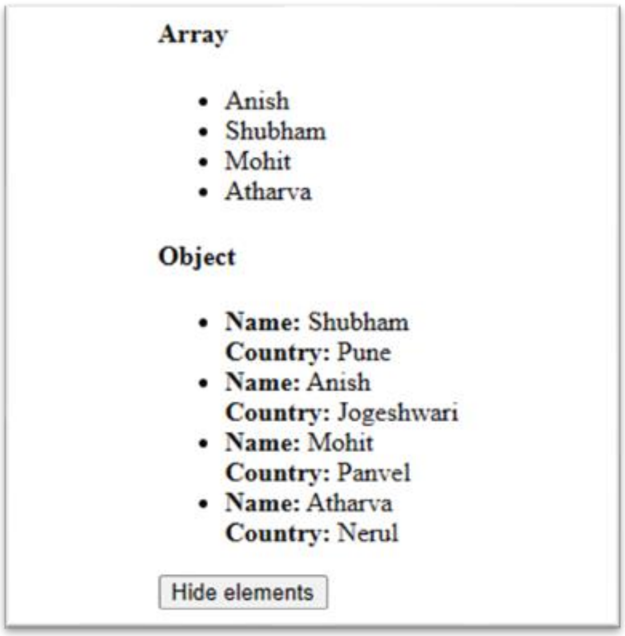
You can attach multiple methods to the scope object inside a controller, which can be used as an event handler or for other purposes.
Angular allows nested controllers.

Q.1] Create an array and an object as members in controller scope with some data in it. Add one behavior (Method) to a scope name as "Display" to print all the element from an array and object.

CODE:

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

<body ng-app="myApp" style="width: 40rem; margin: auto; padding: 2rem">

<div ng-controller="Controller">
<h4>Array</h4>
<ul>
<li ng-repeat="x in arrayPeople">{{x}}</li>
</ul>
<h4>Object</h4>
<ul>
<li ng-repeat="x in objectPeople">
<b>Name:</b> {{ x.name }} <br />
<b>Country:</b> {{ x.city }}
</li>
</ul>
<button ng-click="display()">
{{buttonValue}} elements
</button>
</div>
<script>
var app = angular.module("myApp", []);
app.controller("Controller", function ($scope) {
$scope.buttonValue = "Display";
$scope.display = function () {
if ($scope.buttonValue === "Display") {
$scope.buttonValue = "Hide";
$scope.arrayPeople = ["Anish", "Shubham", "Mohit", "Atharva"];
$scope.objectPeople = [
{ name: "Shubham", city: "Pune" },
{ name: "Anish", city: "Jogeshwari" },
{ name: "Mohit", city: "Panvel" },
{ name: "Atharva", city: "Nerul" },
];
} else {
$scope.buttonValue = "Display";
$scope.arrayPeople = [];
$scope.objectPeople = [];       }};})</script></body></html>
```

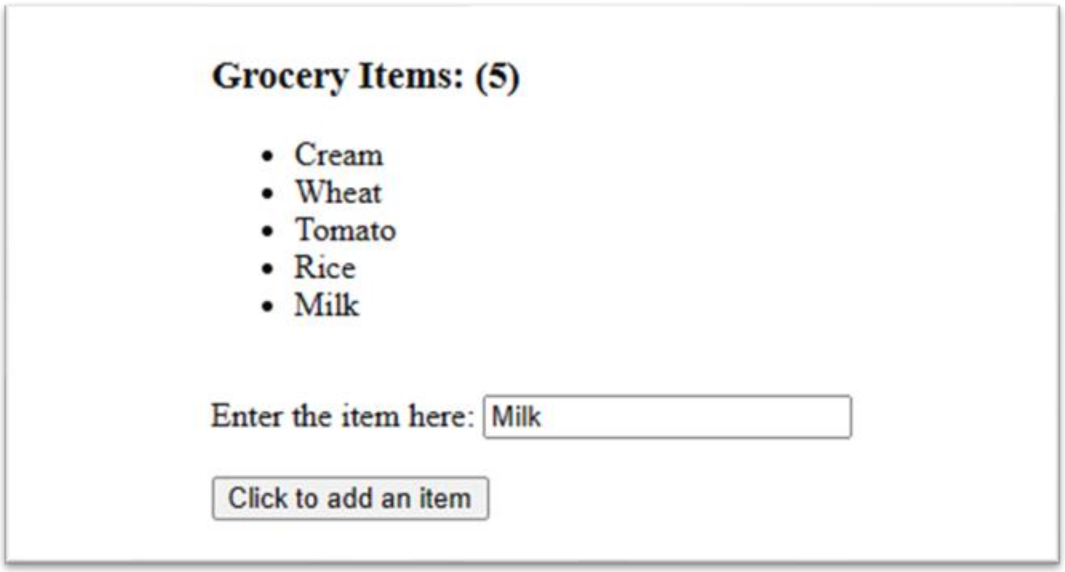
OUTPUT:



```
Array

  • Anish
  • Shubham
  • Mohit
  • Atharva

Object

  • Name: Shubham
    Country: Pune
  • Name: Anish
    Country: Jogeshwari
  • Name: Mohit
    Country: Panvel
  • Name: Atharva
    Country: Nerul
```

Q.2] Create a list of grocery items as a member to the scope object. Add the items entered by the user into the same list by calling "AddItem" method to the scope.
CODE:
```
<!DOCTYPE html>
<html>

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

</head>

<body ng-app="myApp" style="width: 40rem; margin: auto; padding: 2rem">

  <div ng-controller="Controller">
    <h3>Grocery Items: ({{grocerylist.length}})</h3>
    <ul>
      <li ng-repeat="x in grocerylist">{{x}}</li>
    </ul>
    <br>
    Enter the item here:
    <input type="text" name="arrItem" ng-model="item">
    <br><br>
    <button ng-click='push()'>
      Click to add an item
    </button>
    <br>
    <script>
      var app = angular.module("myApp", []);
      app.controller("Controller", function ($scope) {
        $scope.grocerylist = ["Cream", "Wheat", "Tomato", "Rice"];
        $scope.push = function () {
          if (!$scope.item) {
            alert("Please Add something in the textbox before clicking the Add Item button");
          } else if ($scope.grocerylist.includes($scope.item) === true) {
            alert($scope.item + " is already in the Grocery List");
          } else {
            $scope.grocerylist.push($scope.item);
          }
        }
      });
    </script>
</body>
</html>
```

OUTPUT:



```
Grocery Items: (5)

    • Cream
    • Wheat
    • Tomato
    • Rice
    • Milk

Enter the item here: Milk

Click to add an item
```

CONCLUSION:
I have learnt to implement Controllers in Angular Js.

WEB TECHNOLOGIES
PRACTICAL NO: 12

Aim: Demonstrate features of Angular.js forms with a program. Create a registration form for a college event containing fname, lName,gender, class, dept, type of event, contact, email. Put validation, on button click display.

Theory:

- 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:

1. input elements
2. select elements
3. button elements
4. textarea elements

Data-Binding:

- Input controls provides data-binding by using the ng-model directive. <input type="text" ng-model="firstname">
- The application does now have a property named firstname.
- The ng-model directive binds the input controller to the rest of your application.

Checkbox:
A checkbox has the value true or false. Apply the ng-model directive to a checkbox, and use its value in your application.

Radiobuttons:

- Bind radio buttons to your application with the ng-model directive.
- Radio buttons with the same ng-model can have different values, but only the selected one will be used.

Selectbox:

- Bind select boxes to your application with the ng-model directive.
- The property defined in the ng-model attribute will have the value of the selected option in the selectbox.
- 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.

CODE:

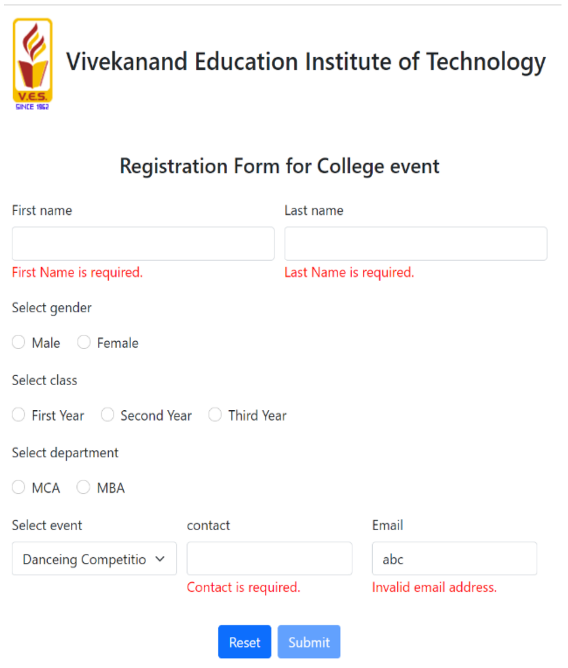
```
<!DOCTYPE html>
<html>

<head>
   <title>VESIT</title>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></scri  pt >
         <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
            <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js"
               integrity="sha384-
QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13"
               crossorigin="anonymous"></script>
</head>

<body>
   <div class="container py-4">
      <div class="d-flex justify-content-center align-items-center mb-5">
         <img src="https://ves.ac.in/wp-content/uploads/2018/07/Logo.png" class="me-3"
alt="vesit-logo">
         <h3>Vivekanand Education Institute of Technology</h3>
      </div>
      <div ng-app="mainApp" ng-controller="studentController" class="w-50 mx-auto">
         <h4 class="text-center mb-4">Registration Form for College event</h4>
         <form name="studentForm" novalidate>
            <div class="row mx-auto">
               <div class="col mb-3 ps-0">
                  <label class="form-label">First name</label>
                  <input type="text" class="form-control" name="firstname" ng-model="firstname"
required>
                  <span style="color:red"
                     ng-show="studentForm.firstname.$dirty && studentForm.firstname.$invalid">
                     <span ng-show="studentForm.firstname.$error.required">First Name is
required.</span>
                  </span>
               </div>
               <div class="col mb-3 px-0">
                  <label class="form-label">Last name</label>
                  <input name="lastname" class="form-control" type="text" ng-model="lastname"
required>
```

```
                    <span style="color:red" ng-show="studentForm.lastname.$dirty &&
studentForm.lastname.$invalid">
                        <span ng-show="studentForm.lastname.$error.required">Last Name is
required.</span>
                    </span>
                </div>
            </div>
            <div class="mb-3">
                <label class="form-label d-block mb-3">Select gender</label>

                <div class="form-check d-inline-block me-3">
                    <input class="form-check-input" type="radio" name="gender" ng-model="gender"
required>
                    <label class="form-check-label">Male</label>
                </div>
                <div class="form-check d-inline-block">
                    <input class="form-check-input" type="radio" name="gender" ng-model="gender"
required>
                    <label class="form-check-label">Female</label>
                </div>
                <span style="color:red" ng-show="studentForm.gender.$dirty &&
studentForm.gender.$invalid">
                    <span ng-show="studentForm.gender.$error.required">Select gender is
required.</span>
                </span>
            </div>
            <div class="mb-3">
                <label class="form-label d-block mb-3">Select class</label>
                <div class="form-check d-inline-block me-3">
                    <input class="form-check-input" type="radio" name="class" ng-model="class"
required>
                    <label class="form-check-label">First Year</label>
                </div>
                <div class="form-check d-inline-block me-3">
                    <input class="form-check-input" type="radio" name="class" ng-model="class"
required>
                    <label class="form-check-label">Second Year</label>
                </div>
                <div class="form-check d-inline-block me-3">
                    <input class="form-check-input" type="radio" name="class" ng-model="class"
required>
                    <label class="form-check-label">Third Year</label>
                </div>
```

```
          <span style="color:red" ng-show="studentForm.class.$dirty &&
studentForm.class.$invalid">
            <span ng-show="studentForm.class.$error.required">Class is required.</span>
          </span>
        </div>
        <div class="mb-3">
          <label class="form-label d-block mb-3">Select department</label>
          <div class="form-check d-inline-block me-3">
            <input class="form-check-input" type="radio" name="department" ng-
model="department" required>
            <label class="form-check-label">MCA</label>
          </div>
          <div class="form-check d-inline-block me-3">
            <input class="form-check-input" type="radio" name="department" ng-
model="department" required>
            <label class="form-check-label">MBA</label>
          </div>
          <span style="color:red" ng-show="studentForm.department.$dirty &&
studentForm.department.$invalid">
            <span ng-show="studentForm.department.$error.required">Department is
required.</span>
          </span>
        </div>
        <div class="row mx-auto">
          <div class="col mb-3 ps-0">
            <label class="form-label">Select event</label>
            <select class="form-select" name="event" ng-model="event">
              <option value="option-1">Danceing Competition</option>
              <option value="option-2">Singing Competition</option>
              <option value="option-3">Game playing Competition</option>
              <option value="option-4">Sport Competition</option>
              <option value="option-5">Coding Competition</option>
            </select>
            <span style="color:red" ng-show="studentForm.event.$dirty &&
studentForm.event.$invalid">
              <span ng-show="studentForm.event.$error.required">Event is
required.</span>
            </span>
          </div>
          <div class="col mb-3 ps-0">
            <label class="form-label">contact</label>
            <input type="number" class="form-control" name="contact" ng-model="contact"
length="10"
              required>
```

```
                    <span style="color:red" ng-show="studentForm.contact.$dirty &&
studentForm.contact.$invalid">
                       <span ng-show="studentForm.contact.$error.required">Contact is
required.</span>
                       <span ng-show="studentForm.contact.$error.contact">Invalid contact.</span>
                    </span>
                 </div>
                 <div class="col mb-3">
                    <label class="form-label">Email</label>
                    <input name="email" class="form-control" type="email" ng-model="email"
length="100" required>
                    <span style="color:red" ng-show="studentForm.email.$dirty &&
studentForm.email.$invalid">
                       <span ng-show="studentForm.email.$error.required">Email is
required.</span>
                       <span ng-show="studentForm.email.$error.email">Invalid email
address.</span>
                    </span>
                 </div>
              </div>
              <div class="d-flex justify-content-center mt-3">
                 <button class="btn btn-primary me-2" ng-click="reset()">Reset</button>
                 <button class="btn btn-primary" ng-disabled="studentForm.firstname.$dirty &&
studentForm.firstname.$invalid ||
studentForm.lastname.$dirty && studentForm.lastname.$invalid || studentForm.gender.$dirty
&& studentForm.gender.$invalid || studentForm.class.$dirty && studentForm.class.$invalid ||
studentForm.department.$dirty && studentForm.department.$invalid ||
studentForm.event.$dirty && studentForm.event.$invalid || studentForm.contact.$dirty &&
studentForm.contact.$invalid ||
studentForm.email.$dirty && studentForm.email.$invalid" value="submit">Submit</button>
              </div>
           </form>
        </div>
    </div>
    <script>
        var mainApp = angular.module("mainApp", []); mainApp.controller('studentController',
function ($scope) {
       $scope.reset = function () {
       $scope.firstname = ""; $scope.lastname = ""; $scope.gender = ""; $scope.class = "";
         $scope.department = "";$scope.event = ""; $scope.contact = "";      $scope.email = "";
    }$scope.reset(); });
    </script>
</body>
</html>
```

OUTPUT:



Vivekanand Education Institute of Technology

Registration Form for College event

First name	Last name
First Name is required.	Last Name is required.

Select gender

Male Female

Select class

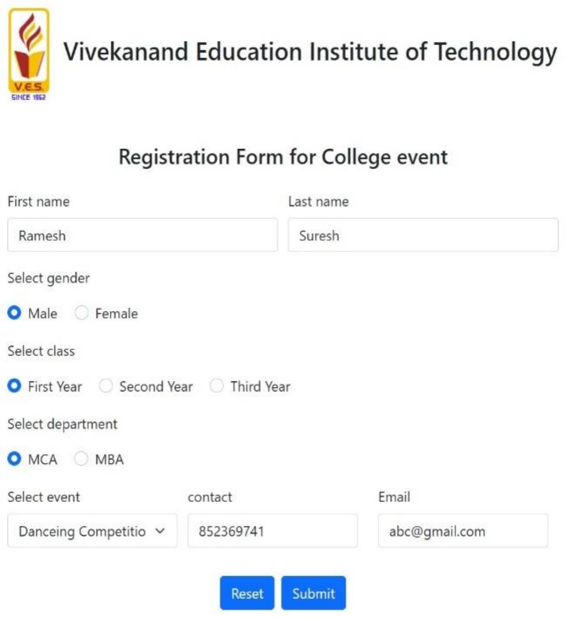
First Year Second Year Third Year

Select department

MCA MBA

Select event	contact	Email
Danceing Competitio		abc
	Contact is required.	Invalid email address.

Reset Submit



Vivekanand Education Institute of Technology

Registration Form for College event

First name	Last name
Ramesh	Suresh

Select gender

[✓ Male] Female

Select class

[✓ First Year] Second Year Third Year

Select department

[✓ MCA] MBA

Select event	contact	Email
Danceing Competitio	852369741	abc@gmail.com

Reset Submit

Conclusion: From this practical, I have learned how to use features of AngularJS Form.

WEB TECHNOLOGIES
PRACTICAL NO: 13

Aim: Create a SPA (Single Page Application)

Theory:
A Single Page Application (SPA) is a single web page, website, or web application that works within a web browser and loads just a single document. It does not need page reloading during its usage, and most of its content remains the same while only some of it needs updating. When the content needs to be updated, the SPA does it through JavaScript APIs.
This way, users can view a website withoutloading the entire new pageand data from the server. As a result, performance increases, and you feel like using a native application. It offers a more dynamic web experience to the users. SPAs help users be in a single,uncomplicatedweb space in easy, workable,and simple ways.

Examples of SPAs:
Gmail, Facebook, Trello, Google Maps, etc., all are Single Page Applications that offer an outstanding user experience in the browser with no page reloading. For instance, when you open your Gmail account, you can see nothing changes much during navigation. Its header and sidebar are the same in the inbox, and when a new email arrives, it reflects the change quickly by loading its content via JavaScript.

SPA work:
The architecture of Single page applications is simple. It involves client-side and server-side rendering technologies
Suppose you want to visit a specific web page. When you enter its address to request access from your browser, the browser sends this request to a server and comes with an HTML document in return.
Using a SPA, the server sends the HTML document only for the first request, and for subsequent requests, it sends JSON data. This means a SPA will rewrite the current page's content and not reload the whole web page. Hence, no need to wait extra for reloadingand faster performance. This capability makes a SPA behave like a native application.

A Single Page Application is different from multi-page applications (MPAs). The latter are web apps with multiple pages reloaded when a user requests new data. Furthermore, SPAs can take a while to load at first but offer faster performance and smooth navigation after loading. MPAs can be comparatively slow and needs top-class internet, especially with graphical elements. Examples of MPAs can be Amazon and Google Docs.

Benefits of SPAs:

Most resources, like HTML, JavaScript, and CSS that SPA needs, are loaded initially and don't require reloading when in use. Only data transmission can change, which makes it highly responsive. Let's find out what all benefits SPAs offer.

Better speed

Web applications must offer faster speed and not waste users' time; otherwise, users can find other efficient venues. SPAs provide shorter response times as the entire page doesn't have to reload and only the data changes in the requested content parts. Thus, the web app's speed improves considerably.
Enhanced user experience

Better user experience is vital for the success of an application. Many reports suggest that users leave web pages that are slower and not easy to use. But with SPAs, users don't have to wait again to reload the full content only to gain a portion of it. Instead, they can gain the requested information faster, which improves their experience using a SPA.

Efficient caching

A Single Page App can cache data efficiently as it sends a request to a server for one time only and then updates the other data. This way, it can use this data to function even when you are offline. If a user's connectivity breaks, it can synchronize the local data with the server when the connection is established.

Simplified development

Developing a SPA is easier as the developers don'tneed to spendmore time writing code and rendering the web pages on a server. Instead, they can reuse the server-side code and decouple the SPA from the frontend user interface. It implies that frontend and backend teams can concentrate on their jobs without worries.

But how?

The frontend development becomes effortless in SPAs due to their decoupled architecture, where frontend display separates from backend services. As a business's critical backend functionalities don't change much, your customers can have a consistent experience using your application, registering by filling a form, etc. You can keep the same content as well, but change how it looks.

When the backend logic and data decouples from how it is presented, you turn the app into service, enabling the developers to create multiple frontend ways and show that service. It also lets developers build, experiment, and deploy the frontend without worrying about the backend technology.

Easy to debug

Debugging an application is vital to ensure nothing can stop it from performing optimally by detecting and removing bugs and errors that can slow down its performance.
Single Page Applications are easy to debug with Google Chrome as they are built using popular frameworks like React, Angular, Vue.js, etc. You can easily monitor and investigate page elements, data, and network operations.
In addition, debugging in SPA is easier than MPAs because SPAs have their own developer tools for Chrome. Developers can examine the JS code rendering from the browser and debug SPAs instead of going through hundreds and thousands of lines of code. The Chrome debugging tools also view page elements, data requests from the server, and data caching.

Less resource consumption

Single Page Apps consume less bandwidth as they load the pages just once. They also work in areas with a slower internet connection, hence, convenient to use for everyone. Besides, they work offline, saving your data, so you don’t have to supply them withconsistent internetto accessandwork onthem, unlike MPAslike Google Docs.

Cross-platform compatibility

Developers can build applications that can run on any operating system, device, and browser easily using a single codebase. Hence, it adds to customer experience as well because they can use the SPA anywhere they want.

In addition, developers can also build feature-rich apps quite effortlessly. For instance, they can include real-time analytics while developing a content editing application.
However, there are also some negatives associated with SPAs.

Disadvantages of SPAs

Poor SEO performance

The SPAs architecture involves just a single page with a single URL. It limits the SPAs’ ability to benefit from search engine optimization (SEO). SEO techniques help improve your site’s ranking inthe searchengine results, as there’s highcompetition out there.

As there is just a single URL with no changes or exceptional addresses, optimizing it for SEO is tricky. It lacks indexation, good analytics, unique links, metadata, etc. Such pages get tough luck to be scanned by the search bots, so optimization becomes difficult.

Online threats

SPAs are more vulnerable to online threats such as cross-site scripting (XSS) than MPAs. Attackers can utilize XSS to inject client-side scripts into a web app and compromise it. In addition, access control is not tight at the operational level. It can expose sensitive data and functions if the developers don't take precautions.
Initial load times

Although SPAs are praised for showcasing great performance and speed, it takes a while to load the complete site. It may irritate some users who may not open the app again.

Browser history

SPAs don't store browser history. If you check the history for any valuable data, you only see the SPA's link to the entire website. Also, you can't move back and forth in the SPA. If you press the back button, you end up on a previously loaded web page, not the previous state. However, this drawback can be neutralized by using the HTML5 History API.

Code Server.js

```
const express = require('express'); const path = require('path');
const app = express(); app.use(express.json()); app.use(express.static("express")); // default
URL for website app.use('/', function (req, res) {
res.sendFile(path.join(__dirname + '/index.html')); });
app.listen(3000)
```

Index.html

```
<!DOCTYPE html> <html lang="en"> <title>SPA</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1"> <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css"> <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Lato"> <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<style> body {
font-family: "Lato", sans-serif }.mySlides { display: none
} </style>


<body>
<!-- Navbar -->
<div class="w3-top">
<div class="w3-bar w3-black w3-card">
```

```
<a class="w3-bar-item w3-button w3-padding-large w3-hide-medium w3-hide-large w3-
right" href="javascript:void(0)" onclick="myFunction()" title="Toggle Navigation Menu"><i
class="fa fa-bars"></i></a>
<a href="#" class="w3-bar-item w3-button w3-padding-large">HOME</a> <a href="#band"
class="w3-bar-item w3-button w3-padding-large w3-
hidesmall"> BAND</a>
<a href="#tour" class="w3-bar-item w3-button w3-padding-large w3-hidesmall">
TOUR</a>
<a href="#contact" class="w3-bar-item w3-button w3-padding-large w3-hidesmall">
CONTACT</a>
<div class="w3-dropdown-hover w3-hide-small">
<button class="w3-padding-large w3-button" title="More">MORE <i class="fa facaret-
down"></i></button>
<div class="w3-dropdown-content w3-bar-block w3-card-4">
<a href="#" class="w3-bar-item w3-button">Merchandise</a> <a href="#" class="w3-bar-item
w3-button">Extras</a>
<a href="#" class="w3-bar-item w3-button">Media</a> </div>
</div>
<a href="javascript:void(0)" class="w3-padding-large w3-hover-red w3-hide-small
w3-right"><i class="fa fa-search"></i></a> </div>
</div>
<!-- Navbar on small screens (remove the onclick attribute if you want the navbar to
always show on top of the content when clicking on the links) -->
<div id="navDemo" class="w3-bar-block w3-black w3-hide w3-hide-large w3-hide-medium
w3-top" style="margin-top:46px">
<a href="#band" class="w3-bar-item w3-button w3-padding-large"
onclick="myFunction()">BAND</a>
<a href="#tour" class="w3-bar-item w3-button w3-padding-large"
onclick="myFunction()">TOUR</a>
<a href="#contact" class="w3-bar-item w3-button w3-padding-large"
onclick="myFunction()">CONTACT</a>
<a href="#" class="w3-bar-item w3-button w3-padding-large"
onclick="myFunction()">MERCH</a>
</div>
<!-- Page content -->
<div class="w3-content" style="max-width:2000px;margin-top:46px"> <!-- Automatic Slideshow
Images -->
<div class="mySlides w3-display-container w3-center"> <img
src="https://www.w3schools.com/w3images/la.jpg"
style="width:100%">
<div class="w3-display-bottommiddle w3-container w3-text-white w3-padding-32 w3-
hide-small">
<h3>Los Angeles</h3>
<p><b>We had the best time playing at Venice Beach!</b></p> </div>
```

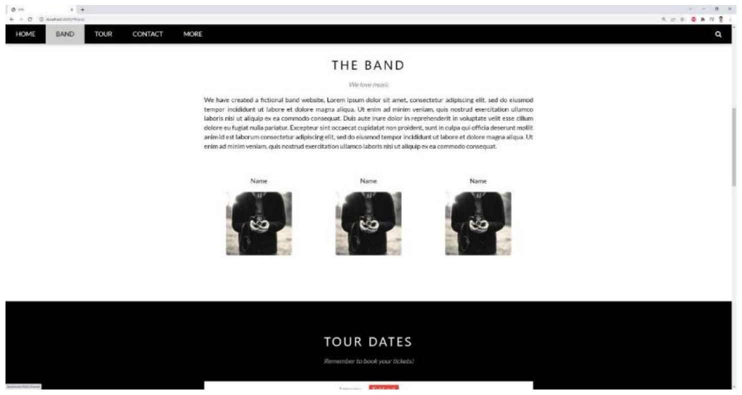
```
<div class="mySlides w3-display-container w3-center">
<img src="https://www.w3schools.com/w3images/ny.jpg" style="width:100%">
<div class="w3-display-bottommiddle w3-container w3-text-white w3-padding-32 w3-
hide-small">
<h3>New York</h3>
<p><b>The atmosphere in New York is lorem ipsum.</b></p> </div>
</div>
<div class="mySlides w3-display-container w3-center">
<img src="https://www.w3schools.com/w3images/chicago.jpg" style="width:100%">
<div class="w3-display-bottommiddle w3-container w3-text-white w3-padding-32 w3-
hide-small"> <h3>Chicago</h3>
<p><b>Thank you, Chicago - A night we won't forget.</b></p> </div>
</div>
<!-- The Band Section -->
<div class="w3-container w3-content w3-center w3-padding-64" style="max-width:800px"
id="band">
<h2 class="w3-wide">THE BAND</h2>
<p class="w3-opacity"><i>We love music</i></p>
<p class="w3-justify">We have created a fictional band website. Lorem ipsum dolor
sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip
ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt mollit anim id est laborum consectetur
adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip
ex ea commodo consequat.</p>
<div class="w3-row w3-padding-32"> <div class="w3-third">
<p>Name</p> <img
src="https://www.w3schools.com/w3images/bandmember.jpg" class="w3-round w3-margin-
bottom" alt="Random Name" style="width:60%">
</div>
<div class="w3-third"> <p>Name</p>
<img src="https://www.w3schools.com/w3images/bandmember.jpg" class="w3-round
w3-margin-bottom" alt="Random Name" style="width:60%"> </div>
<div class="w3-third"> <p>Name</p>
<img src="https://www.w3schools.com/w3images/bandmember.jpg" class="w3-round"
alt="Random Name"
style="width:60%"> </div>
</div> </div>
<!-- The Tour Section -->
<div class="w3-black" id="tour">
<div class="w3-container w3-content w3-padding-64" style="max-width:800px">
```

```
<h2 class="w3-wide w3-center">TOUR DATES</h2>
<p class="w3-opacity w3-center"><i>Remember to book your tickets!</i></p><br>
<ul class="w3-ul w3-border w3-white w3-text-grey">
<li class="w3-padding">January <span class="w3-tag w3-red w3-margin-left">Sold
out</span></li>
<li class="w3-padding">February <span class="w3-tag w3-red w3-marginleft">
Sold out</span></li>
<li class="w3-padding">March <span class="w3-badge w3-right w3-marginright">
3</span></li> </ul>
<div class="w3-row-padding w3-padding-32" style="margin:0 -16px"> <div class="w3-third w3-
margin-bottom">
<img src="https://www.w3schools.com/w3images/newyork.jpg" alt="New York"
style="width:100%"
class="w3-hover-opacity">
<div class="w3-container w3-white"> <p><b>New York</b></p>
<p class="w3-opacity">Fri 04 Mar 2022</p>
<p>Praesent tincidunt sed tellus ut rutrum sed vitae justo.</p> <button class="w3-button w3-
black w3-margin-bottom"
onclick="document.getElementById('ticketModal').style.display='block'">Buy Tickets</button>
</div> </div>
<div class="w3-third w3-margin-bottom">
<img src="https://www.w3schools.com/w3images/paris.jpg" alt="Paris" style="width:100%"
class="w3-hover-opacity">
<div class="w3-container w3-white"> <p><b>Paris</b></p>
<p class="w3-opacity">Sat 05 Mar 2022</p>
<p>Praesent tincidunt sed tellus ut rutrum sed vitae justo.</p> <button class="w3-button w3-
black w3-margin-bottom"
onclick="document.getElementById('ticketModal').style.display='block'">Buy Tickets</button>
</div> </div>
<div class="w3-third w3-margin-bottom">
<img src="https://www.w3schools.com/w3images/sanfran.jpg" alt="San Francisco"
style="width:100%" class="w3-hover-opacity"> <div class="w3-container w3-white">
<p><b>San Francisco</b></p>
<p class="w3-opacity">Sun 06 Mar 2022</p>
<p>Praesent tincidunt sed tellus ut rutrum sed vitae justo.</p> <button class="w3-button w3-
black w3-margin-bottom"


onclick="document.getElementById('ticketModal').style.display='block'">Buy Tickets</button>
</div> </div>
</div> </div>
</div>
<!-- Ticket Modal -->
<div id="ticketModal" class="w3-modal">
```

```
<div class="w3-modal-content w3-animate-top w3-card-4"> <header class="w3-container w3-
teal w3-center w3-padding-32">
<span onclick="document.getElementById('ticketModal').style.display='none'"
class="w3-button w3-teal w3-xlarge w3-display-topright">×</span>
<h2 class="w3-wide"><i class="fa fa-suitcase w3-margin-right"></i>Tickets</h2>
</header>
<div class="w3-container">
<p><label><i class="fa fa-shopping-cart"></i> Tickets, $15 per person</label></p>
<input class="w3-input w3-border" type="text" placeholder="How many?">
<p><label><i class="fa fa-user"></i> Send To</label></p>
<input class="w3-input w3-border" type="text" placeholder="Enter email">
<button class="w3-button w3-block w3-teal w3-padding-16 w3-section w3-
right">PAY <i class="fa fa-check"></i></button> <button class="w3-button w3-red w3-section"
onclick="document.getElementById('ticketModal').style.display='none'">Close <i class="fa
fa-remove"></i></button>
<p class="w3-right">Need <a href="#" class="w3-text-blue">help?</a></p>
</div> </div>
</div>
<!-- The Contact Section -->
<div class="w3-container w3-content w3-padding-64" style="max-width:800px" id="contact">
<h2 class="w3-wide w3-center">CONTACT</h2>
<p class="w3-opacity w3-center"><i>Fan? Drop a note!</i></p> <div class="w3-row w3-
padding-32">
<div class="w3-col m6 w3-large w3-margin-bottom">
<i class="fa fa-map-marker" style="width:30px"></i> Chicago, US<br>
<i class="fa fa-phone" style="width:30px"></i> Phone: +00 151515<br>
<i class="fa fa-envelope" style="width:30px"> </i> Email: mail@mail.com<br>
</div>
<div class="w3-col m6">
<form action="/action_page.php" target="_blank">
<div class="w3-row-padding" style="margin:0 -16px 8px -16px"> <div class="w3-half">
<input class="w3-input w3-border" type="text" placeholder="Name" required
name="Name"> </div>
<div class="w3-half">
<input class="w3-input w3-border" type="text" placeholder="Email" required
name="Email"> </div>
</div>
<input class="w3-input w3-border" type="text" placeholder="Message" required
name="Message">
<button class="w3-button w3-black w3-section w3-right" type="submit">SEND</button>
</form> </div>
</div> </div>
<!-- End Page Content --> </div>
<!-- Image of location/map -->
```

```
<img src="https://www.w3schools.com/w3images/map.jpg"  class="w3-image w3-greyscalemin"
style="width:100%">
<!-- Footer -->
<footer class="w3-container w3-padding-64 w3-center w3-opacity w3-light-grey w3-
xlarge">
<i class="fa fa-facebook-official w3-hover-opacity"></i> <i class="fa fa-instagram w3-hover-
opacity"></i>
<i class="fa fa-snapchat w3-hover-opacity"></i> <i class="fa fa-pinterest-p w3-hover-
opacity"></i> <i class="fa fa-twitter w3-hover-opacity"></i>
<i class="fa fa-linkedin w3-hover-opacity"></i> <p class="w3-medium">Powered  by <a
href="https://www.w3schools.com/w3css/default.asp"  target="_blank">SPA.css</a></p>
</footer> <script>
// Automatic Slideshow - change image every 4 seconds var myIndex = 0;
carousel();
function carousel() { var i;
var x = document.getElementsByClassName("mySlides");  for (i = 0; i < x.length; i++) {
x[i].style.display = "none"; }
myIndex++;
if (myIndex > x.length) { myIndex = 1 } x[myIndex - 1].style.display = "block";
setTimeout(carousel,  4000);
}
// Used to toggle the menu on small screens when clicking on the menu button
function myFunction() {
var x = document.getElementById("navDemo");  if (x.className.indexOf("w3-show")  == -1) {
x.className += " w3-show"; } else {
x.className = x.className.replace("  w3-show", ""); }
}
// When the user clicks anywhere outside of the modal, close it var modal =
document.getElementById('ticketModal');  window.onclick = function (event) {
if (event.target == modal) { modal.style.display  = "none";
} }
</script>
 </body>
</html>
```

Output:



THE BAND

We love music

We have created a fictional band website. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

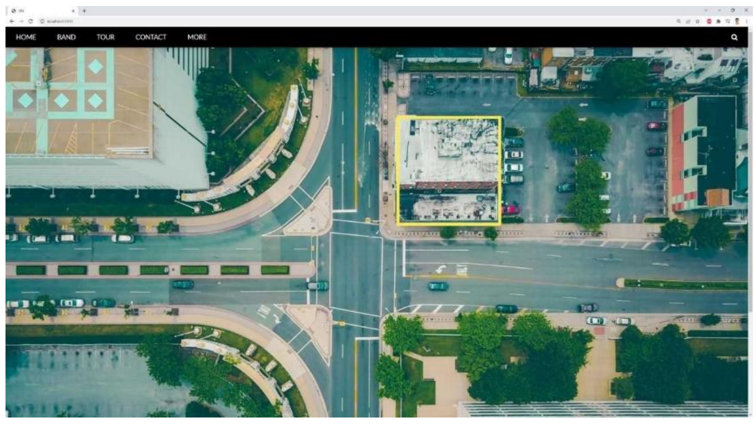
Name

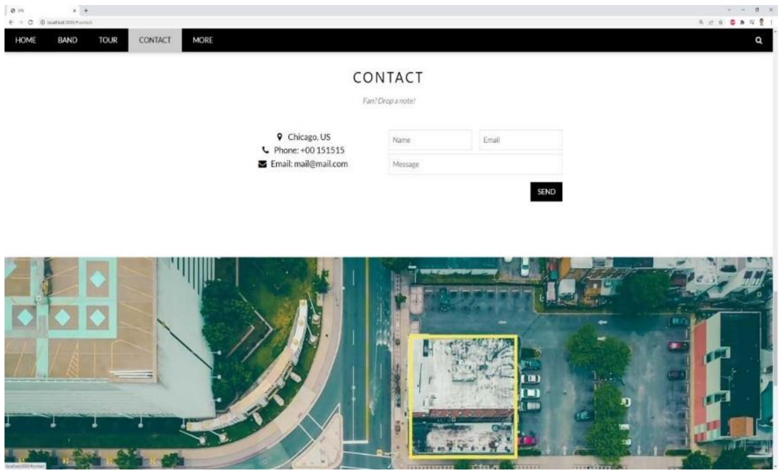
Name

Name

TOUR DATES

Remember to book your tickets!





CONTACT

Fan? Drop a note!

Chicago, US
Phone: +00 151515
Email: mail@mail.com

Conclusion:

From this practical, I have learned how to make SPA using Angular JS

Practical no: 13

Aim: Implementation of SPA (Single Page Application) in Angular js.

Theory:

INTRODUCTION TO SPA

Multi-page application (MPA)- with every click a new page would be loaded from the server

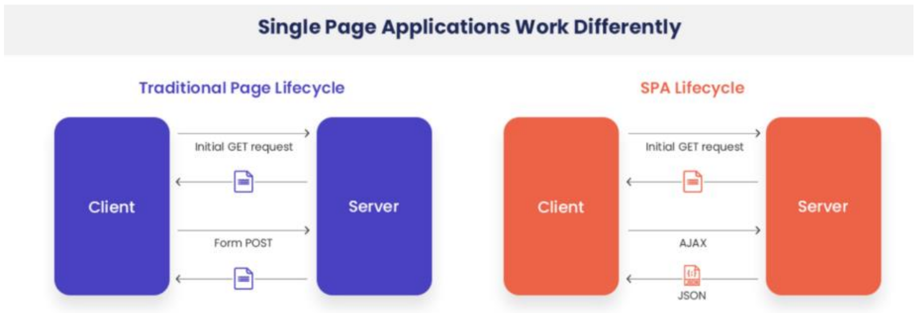
- Time consuming
- Increased the server load
- Made the website slower.

A Single-page Application (SPA)-loads a single html page and only a part of the page instead of the entire page gets updated with every click of the mouse

High performance and loading pages faster.

The goal is faster transitions that make the website feel more like a native app.

Web browser javascript frameworks and libraries, such as angularjs, ember.Js, extjs, knockout.Js, meteor.Js, react, vue.Js, and svelte have adopted SPA principles.



HOW IT WORKS

In the SPA, the whole data is sent to the client from the server at the beginning.

As the client clicks certain parts on the webpage, only the required part of the information is fetched from the server and the page is rewritten dynamically.

This results in a lesser load on the server and is cost-efficient.

SPAs use AJAX and HTML5 to create a fluid and responsive web applications
and most of the work happens on the client-side.

Popular applications such as Facebook, Gmail, Twitter, Google Drive, Netflix,
and many more are examples of SPA.

ANGULARJS ROUTING

The ngRoute module helps your application to become a single page application.

What is Routing in AngularJS?

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.

Advantages
Team collaboration
Single-page applications are excellent when more than one developer is working on the same project. It allows backend developers to focus on the API, while the frontend developers can focus on creating the user interface based on the backend API.
Caching
The application sends a single request to the server and stores all the received information in the cache. This proves beneficial when the client has poor network connectivity.
Fast and responsive
As only parts of the pages are loaded dynamically, it improves the website's speed.
Debugging is easier
Debugging single page applications with chrome is easier since such applications are developed using like ANGULARJS BATARANG and REACT developer tools.
Linear user experience
Browsing or navigating through the website is easy.

Disadvantages

SEO optimization
SPAs provide poor SEO optimization. This is because single-page applications operate on javascript and load data at once server. The URL does not change and different pages do not have a unique URL. Hence it is hard for the search engines to index the SPA website as opposed to traditional server-rendered pages.
Browser history

A SPA does not save the users' transition of states within the website. A browser saves the previous pages only, not the state transition. Thus when users click the back button, they are not redirected to the previous state of the website. To solve this problem, developers can equip their SPA frameworks with the HTML5 history API.

Security issues

Single-page apps are less immune to cross-site scripting (xss) and since no new pages are loaded, hackers can easily gain access to the website and inject new scripts on the client-side.

Memory consumption

Since the SPA can run for a long time sometimes hours at a time, one needs to make sure the application does not consume more memory than it needs. Else, users with low memory devices may face serious performance issues.

Disabled javascript

Developers need to chalk out ideas for users to access the information on the website for browsers that have javascript disabled.

1]Create a single page application for any one of these use cases(minimum 5-6 Webpages)

- **eCommerce Websites.**
- **Event Websites.**
- **Online Forums.**
- **Personal Websites.**
- **Blogs**

Code:

1]Home.html:

```
<!DOCTYPE html>
<html>

<head>
  <style>
  ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #47d7d2;
```

```
 }

 li {
   float: left;
 }

 li a {
   display: block;
   color: white;
   text-align: center;
   padding: 14px 16px;
   text-decoration: none;
 }

 li a:hover {
   background-color: #32a2c4;
 }
 </style>
 <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 ng-app="myApp">
<ul>
 <li><a class="active" href="#/!">Home</a></li>
 <li><a href="#!AboutUs">About Us</a></li>
 <li><a href="#!Contact">Contact Us</a></li>
 <li><a href="#!ourgoal">Our Goal</a></li>
 <li><a href="#!Achieve">Our Achievement</a></li>
</ul>
<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
   $routeProvider
   .when("/", {
       templateUrl : "Main.html"
   })
   .when("/AboutUs", {
       templateUrl : "About.html"
   })
   .when("/Contact", {
       templateUrl : "Contact.html"
```

```
    })
    .when("/ourgoal", {
        templateUrl : "Ourgoal.html"
    })
    .when("/Achieve", {
        templateUrl : "Ourachievement.html"
    });
});
</script>

</body>
</html>
```

2]Main.html:

```
<h1>Home page</h1>
<p><b>This is our home page</b></p>
```

3]About:

```
<h1>About Us</h1>
<p><b>Vivekanand Education Society's Institute of Technology, also known as VESIT or V. E.
S. Institute of Technology,
    <br><br> was established in 1984 as an engineering
    college affiliated with the University of Mumbai.</b>
</p>
```

4]Contact:

```
<h2>Address : Hashu Advani Memorial Complex, Collector’s Colony, <br>Chembur, Mumbai –
400 074. India.</h2>

<p><b>Tel :</b> +91-22-61532510 / 27 (Admission)</p>

<p><b>Fax :</b> +91-22-61532555</p>

<p><b>Email :</b> vesit@ves.ac.in<br>
            vesit.admission@ves.ac.in<br>
            vesit.website@ves.ac.in<br>
            exam.vesit@ves.ac.in<br>
            (Transcripts / Exam)<br>
            vesit.research@ves.ac.in  (R&D)</p>
```

5]Ourgoal:

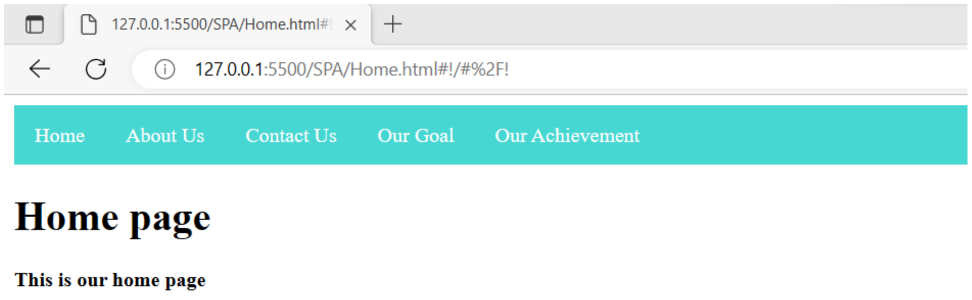
<h1>Our Goal is to provide Best Education in India.</h1>

6] Our Achievement:

<h2>Our Achievement</h2>
<h3>One of top 3 Mca college in Mumbai</h3>

Output:

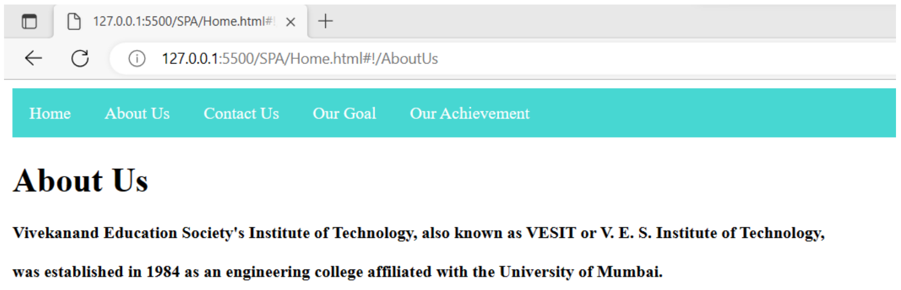
Home page:


127.0.0.1:5500/SPA/Home.html#
127.0.0.1:5500/SPA/Home.html#!/#%2F!
Home
About Us
Contact Us
Our Goal
Our Achievement

Home page

This is our home page

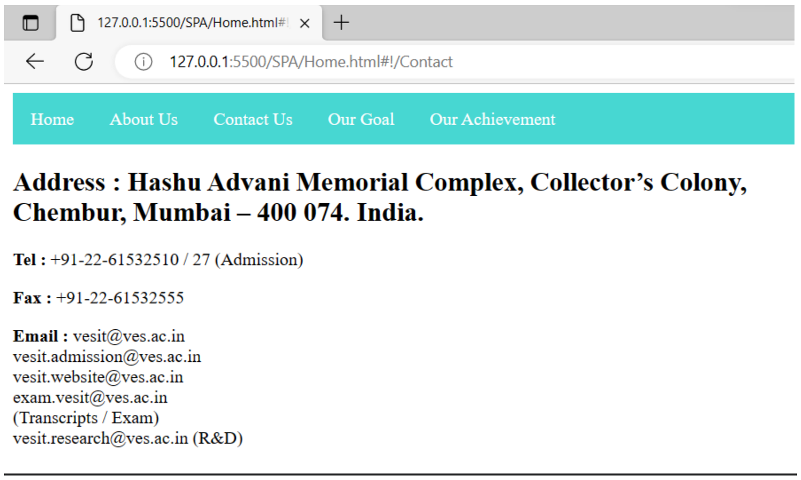
About us:


127.0.0.1:5500/SPA/Home.html#
127.0.0.1:5500/SPA/Home.html#!/AboutUs
Home
About Us
Contact Us
Our Goal
Our Achievement

About Us

Vivekanand Education Society's Institute of Technology, also known as VESIT or V. E. S. Institute of Technology, was established in 1984 as an engineering college affiliated with the University of Mumbai.

Contact us:


127.0.0.1:5500/SPA/Home.html#!
127.0.0.1:5500/SPA/Home.html#!/Contact
Home
About Us
Contact Us
Our Goal
Our Achievement

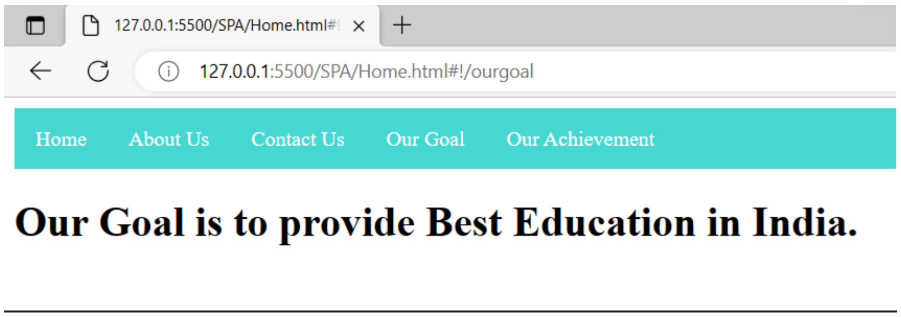
Address : Hashu Advani Memorial Complex, Collector's Colony, Chembur, Mumbai – 400 074. India.

Tel : +91-22-61532510 / 27 (Admission)

Fax : +91-22-61532555

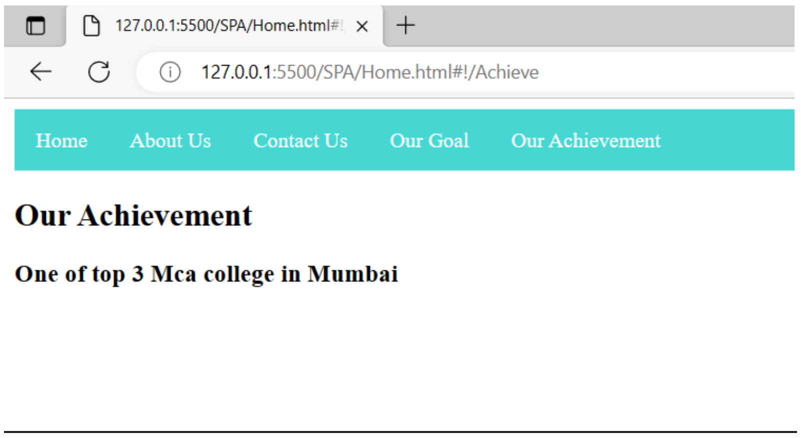
Email : vesit@ves.ac.in
vesit.admission@ves.ac.in
vesit.website@ves.ac.in
exam.vesit@ves.ac.in
(Transcripts / Exam)
vesit.research@ves.ac.in (R&D)

<u>Our Goal:</u>


127.0.0.1:5500/SPA/Home.html#!
127.0.0.1:5500/SPA/Home.html#!/ourgoal
Home
About Us
Contact Us
Our Goal
Our Achievement

Our Goal is to provide Best Education in India.

<u>Our Achievement:</u>


127.0.0.1:5500/SPA/Home.html#!
127.0.0.1:5500/SPA/Home.html#!/Achieve
Home
About Us
Contact Us
Our Goal
Our Achievement

Our Achievement

One of top 3 Mca college in Mumbai

Conclusion: From this Practical I Learned SPA (Single Page Application) in Angular js.