

# **Web Technology**

## **Subject Code: MCAL14**

A Practical Journal Submitted in Fulfillment  
of the Degree of

**MASTER**

**In**

**COMPUTER APPLICATION**

**Year 2022-2023**

**By**

(Ravishankar Jaiswal)  
(172047))

Semester- 1

Under the Guidance of

**Prof. Mr. Abhinandan Sawant**



Institute of Distance and Open Learning  
Vidya Nagari, Kalina, Santacruz East – 400098.  
University of Mumbai

**PCP Center**

[Satish Pradhan Dyanasadhana College, Thane]



## **Institute of Distance and Open Learning,**

**Vidyanagari, Kalina, Santacruz (E) -400098**

### **CERTIFICATE**

This to certify that, (**Ravishankar Jaiswal**) appearing **Master in Computer Application (Semester I) Application ID: 172047**) has satisfactorily completed the prescribed practical of **MCA L14-Web Technology Lab** as laid down by the University of Mumbai for the academic year 2022-23

Teacher in charge

Examiners

Coordinator  
IDOL, MCA  
University of Mumbai

Date: -

01/04/2023

Place: - THANE

## What is node js?

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

## Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

- **Asynchronous and Event Driven** – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- **Very Fast** – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering** – Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** – Node.js is released under the [MIT license](#)

To demonstrate the use of REPL Terminal in node.js.

REPL stands for Read Eval Print Loop and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output

in an interactive mode. Node.js or **Node** comes bundled with a REPL environment. It performs the following tasks –

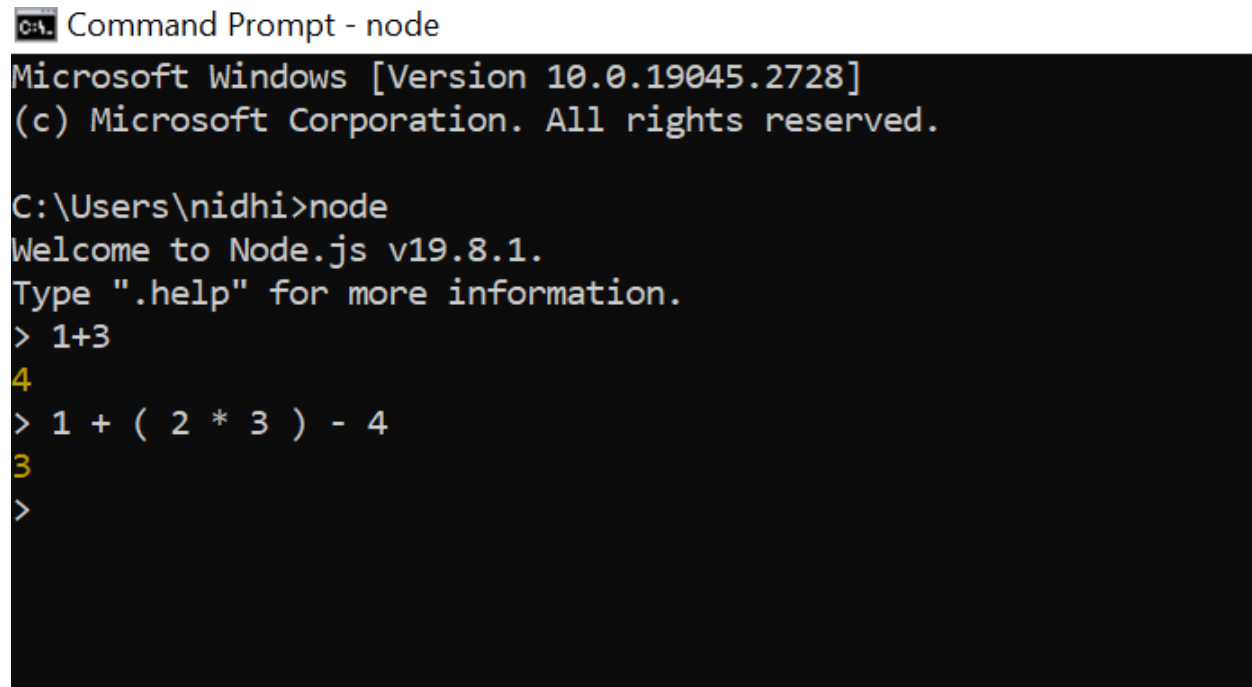
- **Read** – Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- **Eval** – Takes and evaluates the data structure.
- **Print** – Prints the result.
- **Loop** – Loops the above command until the user presses **ctrl-c** twice.

The REPL feature of Node is very useful in experimenting with Node.js codes and to debug JavaScript codes.

## Online REPL Terminal

To simplify your learning, we have set up an easy to use Node.js REPL environment online, where you can practice Node.js syntax – [Launch Node.js REPL Terminal](#)

## Simple Expression



The screenshot shows a Windows Command Prompt window titled "Command Prompt - node". The text inside the window is as follows:

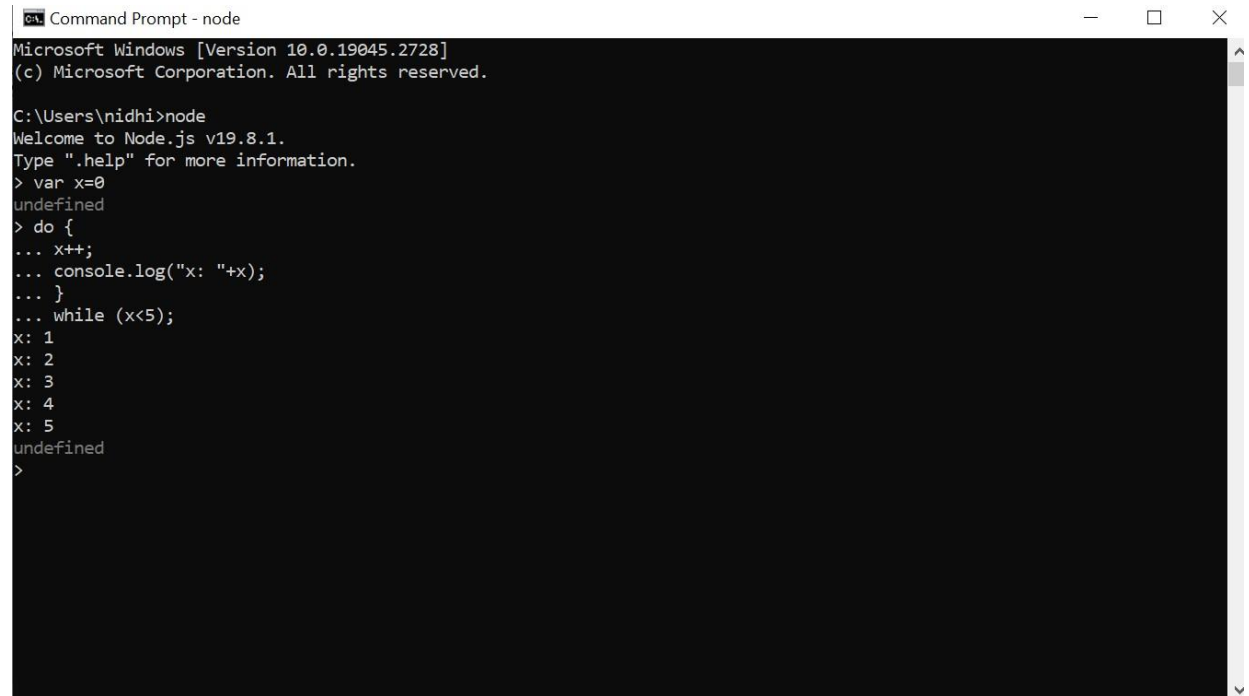
```
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nidhi>node
Welcome to Node.js v19.8.1.
Type ".help" for more information.
> 1+3
4
> 1 + ( 2 * 3 ) - 4
3
>
```

## Use Variables

```
> x=10
10
> var y=10
undefined
> x+y
20
> console.log("Hello World")
Hello World
undefined
>
```

## Multiline Expression



```
Command Prompt - node
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nidhi>node
Welcome to Node.js v19.8.1.
Type ".help" for more information.
> var x=0
undefined
> do {
... x++;
... console.log("x: "+x);
... }
... while (x<5);
x: 1
x: 2
x: 3
x: 4
x: 5
undefined
>
```

## Underscore Variable

```
> var x=10
undefined
> var y=20
undefined
> x+y
30
> var sum=_
undefined
> console.log(sum)
30
undefined
>
```

To demonstrate the use of standard callback pattern in node.js.

## Blocking Code Example

Create a text file named **input.txt** with the following content –

```
Tutorials Point is giving self learning content
to teach the world in simple and easy way!!!!
```

Create a js file named **main.js** with the following code –

```
var fs = require("fs");
var data = fs.readFileSync('input.txt');

console.log(data.toString());
console.log("Program Ended");
```

Now run the main.js to see the result –

```
$ node main.js
```

```
C:\Users\nidhi\callback>node main.js
Tutorials Point is giving self learning content
to teach the world in simple and easy way!!!!
Program Ended

C:\Users\nidhi\callback>
```

## Non-Blocking Code Example

Create a text file named input2.txt with the following content.

Hey There I m using Node JS to demonstrate Non-Blocking Code Example

Update main2.js to have the following code –

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

fs.readFile('input2.txt', function (err, data) {
  if (err) return console.error(err);
  console.log(data.toString());
});

console.log("Program Ended");
```

```
C:\Users\nidhi\callback>node main2.js
Program Ended
Hey There I m using Node JS to demonstrate Non-Blocking Code Example

C:\Users\nidhi\callback>
```

## To demonstrate the event emitter pattern in node.js

When an EventEmitter instance faces any error, it emits an 'error' event. When a new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

EventEmitter provides multiple properties like **on** and **emit**. **on** property is used to bind a function with the event and **emit** is used to fire an event.

Create a js file named main3.js with the following Node.js code –

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

var EventEmitter = new events.EventEmitter();

// listener #1

var listner1 = function listner1() {

    console.log('listner1 executed.');
```

```
    }

// listener #2

var listner2 = function listner2() {

    console.log('listner2 executed.');
```

```
    }

// Bind the connection event with the listner1 function

eventEmitter.addListener('connection', listner1);

// Bind the connection event with the listner2 function

eventEmitter.on('connection', listner2);
```



```

var eventListeners = require('events').EventEmitter.listenerCount
    (eventEmitter,'connection');

console.log(eventListeners + " Listner(s) listening to connection event");

// Fire the connection event

eventEmitter.emit('connection');

// Remove the binding of listner1 function

eventEmitter.removeListener('connection', listner1);

console.log("Listner1 will not listen now.");


// Fire the connection event

eventEmitter.emit('connection');

eventListeners = require('events').EventEmitter.listenerCount(eventEmitter,'connection');

console.log(eventListeners + " Listner(s) listening to connection event");

console.log("Program Ended.");

```

```

C:\Users\nidhi\callback>node main3.js
2 Listner(s) listening to connection event
listner1 executed.
listner2 executed.
listner1 will not listen now.
listner2 executed.
1 Listner(s) listening to connection event
Program Ended.

C:\Users\nidhi\callback>

```

## What is Angular JS?

AngularJS is a structural framework for dynamic web applications. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application components clearly and succinctly. Its data binding and dependency injection eliminate much of the code you

currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

## Core Features

The core features of AngularJS are as follows –

- **Data-binding** – It is the automatic synchronization of data between model and view components.
- **Scope** – These are objects that refer to the model. They act as a glue between controller and view.
- **Controller** – These are JavaScript functions bound to a particular scope.
- **Services** – AngularJS comes with several built-in services such as \$http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
- **Filters** – These select a subset of items from an array and returns a new array.
- **Directives** – Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc.
- **Templates** – These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using *partials*.
- **Routing** – It is concept of switching views.
- **Model View Whatever** – MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
- **Deep Linking** – Deep linking allows to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
- **Dependency Injection** – AngularJS has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.

Write a Simple program for multiplication in AngularJS.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

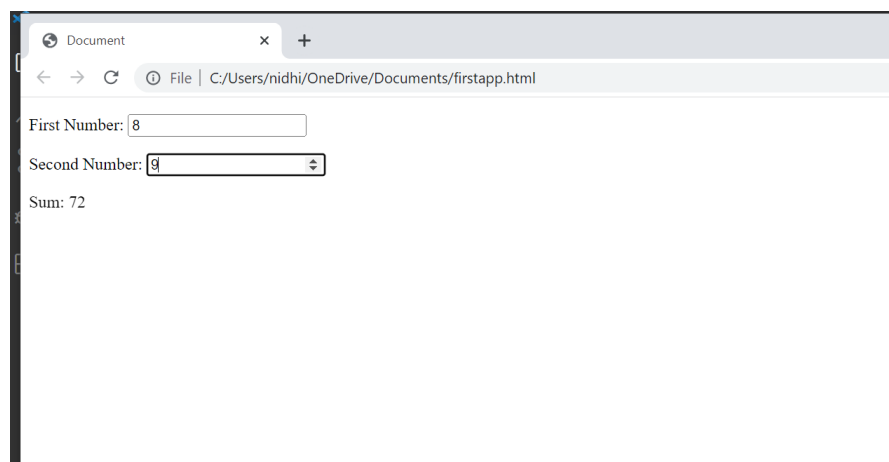
```

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></scri
pt>
  <title>Document</title>
</head>
<body ng-app>
  <div ng-app="">

    <p>First Number:
<input type="number" ng-model="num1" ng-init="num1=0" />
    </p>
    <p>Second Number:
<input type="number" ng-model="num2" ng-init="num2=0" />
    </p>
    <p>Sum: {{ num1 * num2 }}</p>
  </div>
</body>
</html>

```

## OUTPUT



Write a program to display your name with Welcome note Hello in Angular JS.

```

<!DOCTYPE html>
<html ng-app="app">
<head>
  <meta charset="utf 8">
  <title>Guru99</title>

```

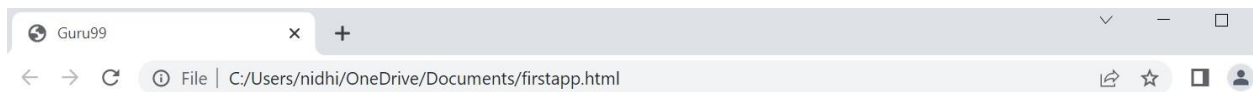
```

</head>
<body>
<h1 ng-controller="HelloWorldCtrl">{{message}}</h1>
<script src="https://code.angularjs.org/1.6.9/angular.js"></script>
<script>
    angular.module("app", []).controller("HelloWorldCtrl", function($scope) {
        $scope.message="This is Ankit Gupta Here. Hello Welcome to the World of
AngularJS"
    } )
</script>

</body>
</html>

```

## OUTPUT



**This is Ankit Gupta Here. Hello Welcome to the World of AngularJS**

Create Simple User Registration Form in AngularJS.

## Index.html

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>HTML Starter</title>

```

```

    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.26/angular.min.js"></scr
ipt>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
    <script src="app.js"></script>
</head>
<body>
    <div ng-app = "myApp" class = "container" style="width:550px">
        <div style="text-align:center;color:blue">
            <h3><b>User Registraion Form</b></h3>
        </div>
        <div ng-controller = "ContactController">
            <div align="right">
                <a href="#" ng-click="searchUser()">{{title}}</a>
            </div>
            <form role = "form" class="well" ng-hide="ifSearchUser">
                <div class = "form-group">
                    <label for = "name"> Name: </label>
                    <input type = "text" id = "name" class = "form-control"
placeholder = "Enter Name " ng-model = "newcontact.name">
                </div>
                <div class = "form-group">
                    <label for = "email"> Email: </label>
                    <input type = "email" id = "email" class = "form-control"
placeholder = "Enter Email " ng-model = "newcontact.email">
                </div>
                <div class = "form-group">
                    <label for = "password"> Password: </label>
                    <input type = "password" id = "password" class = "form-control"
placeholder = "Enter Password " ng-model = "newcontact.password">
                </div>
                <div class = "form-group">
                    <label for = "phone"> Phone: </label>
                    <input type = "text" id = "phone" class = "form-control"
placeholder = "Enter Phone " ng-model = "newcontact.phone">
                </div>
                <br>
                <input type="hidden" ng-model="newcontact.id">
                <input type="button" class="btn btn-primary" ng-click="saveContact()"
class="btn btn-primary" value = "Submit">
            </form>
        </div>
    </div>

```



```

        'name' : 'Steve John',
        'email' : 'john@gmail.com',
        'password': 'John123',
        'phone' : '911-91-199-999'}]];

// Save Service for saving new contact and saving existing edited contact.
this.save = function(contact)
{
    if(contact.id == null)
    {
        contact.id = uid++;
        contacts.push(contact);
    }
    else
    {
        for(var i in contacts)
        {
            if(contacts[i].id == contact.id)
            {
                contacts[i] = contact;
            }
        }
    }
};

// search for a contact

this.get = function(id)
{
    for(var i in contacts )
    {
        if( contacts[i].id == id)
        {
            return contacts[i];
        }
    }
};

//Delete a contact
this.delete = function(id)
{
    for(var i in contacts)
    {
        if(contacts[i].id == id)
        {

```

```

        contacts.splice(i,1);
    }
}

});
//Show all contacts
this.list = function()
{
    return contacts;
} ;
});

////Controller area .....

myApp.controller("ContactController" , function($scope , ContactService){
    console.clear();

    $scope.ifSearchUser = false;
    $scope.title ="List of Users";

    $scope.contacts = ContactService.list();

    $scope.saveContact = function()
    {
        console.log($scope.newcontact);
        if($scope.newcontact == null || $scope.newcontact == angular.undefined)
            return;
        ContactService.save($scope.newcontact);
        $scope.newcontact = {};
    };
    $scope.delete = function(id)
    {
        ContactService.delete(id);
        if($scope.newcontact != angular.undefined && $scope.newcontact.id ==
id)
        {
            $scope.newcontact = {};
        }
    };
    $scope.edit = function(id)
    {
        $scope.newcontact = angular.copy(ContactService.get(id));
    };
    $scope.searchUser = function(){
        if($scope.title == "List of Users"){
            $scope.ifSearchUser=true;

```



```

        $scope.title = "Back";
    }
    else
    {
        $scope.ifSearchUser = false;
        $scope.title = "List of Users";
    }
    };
});

```

## Output

The screenshot shows a web browser window with the title 'HTML Starter'. The address bar shows the file path 'C:/Users/nidhi/OneDrive/Documents/Index.html#'. The page content is divided into two main sections.

The first section is titled 'User Registration Form' in blue text. To its right, there is a link 'List of Users'. The form itself is a light gray box containing four input fields with labels 'Name:', 'Email:', 'Password:', and 'Phone:'. Each field has a placeholder text 'Enter Name', 'Enter Email', 'Enter Password', and 'Enter Phone' respectively. Below the input fields is a blue 'Submit' button.

The second section is titled 'Registered Users' in bold black text. It contains a table with the following data:

Name	Email	Phone	Action
Ankit	rg27101972@gmail.com	7387522338	<a href="#">edit</a> <a href="#">delete</a>