**1. Create application to demonstrate Node.js Module**

**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.

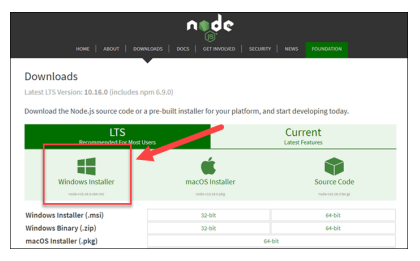
There are two scenarios that will occur depending on the nature of the request. If the request is non-blocking, it does not involve any longrunning processes or data requests, the response will be immediately prepared and then sent back to the client. In the event the request is blocking, requiring I/O 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.

**Traditional Web Server Model:**

The traditional web server model consists of a pool of threads which may process requests. Each time a new request comes in, it is assigned to a different thread in the pool. In the event a request is received and a thread is not available, the request will have to wait until a previous request finishes, a response is returned, and the thread is returned to the thread pool. In this way, the web server model is synchronous, or blocking.

**How to Install Node.js and NPM on Windows:**

In a web browser, navigate to https://nodejs.org/en/download/. Click the Windows Installer button to download the latest default version. At the time this article was written, version 10.16.0-x64 was the latest version. The Node.js installer includes the NPM package manager



**4 Step 2: Install Node.js and NPM from Browser**

1. Once the installer finishes downloading, launch it. Open the downloads link in your browser and click the file. Or, browse to the location where you have saved the file and double-click it to launch.
2. The system will ask if you want to run the software – click Run.
3. You will be welcomed to the Node.js Setup Wizard – click Next
4. On the next screen, review the license agreement. Click Next if you agree to the terms and install the software.
5. The installer will prompt you for the installation location. Leave the default location, unless you have a specific need to install it somewhere else – then click Next.
6. The wizard will let you select components to include or remove from the installation. Again, unless you have a specific need, accept the defaults by clicking Next.
7. Finally, click the Install button to run the installer. When it finishes, click Finish.

**Step 3: Verify Installation:**

Open a command prompt (or PowerShell), and enter the following: node -v

The system should display the Node.js version installed on your system. You can do the same for NPM: npm -v

**Node.js - REPL Terminal:**

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 datastructure, 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.

|  |
| --- |
| . **Aim 1: Write a node.js program to demonstrate the concept of Local Modules.**  **CODE:** addition.js function Add(a,b) {  return(a+b)  }  function Sub(a,b) {  return(a-b)  }  module.exports = {Add, Sub} index.js const lib = require('./add')  const sum = lib.Add(5,8);  const sub = lib.Sub(9,8);  console.log("Addition of a and b is ",sum);  console.log("Subtraction of a and b is ",sub);  **OUTPUT:**  **WhatsApp Image 2024-12-12 at 1.12.15 PM**  **Aim2: Write a node.js program to demonstrate the concept of Non-Global Core Module**  **CODE:** file.js const fs = require('fs');  const path = require('path');  const filePath = path.join(\_\_dirname, "fileCreated.txt");  fs.writeFile(filePath, "Hello Bachchoo", (err)=>{  if(err) throw err;  console.log("File Created with the help of fs Module");  })  **OUTPUT:**  1.2 (2)    **Aim3: Write a node.js program to demonstrate the concept of Third-Party Module.**  **CODE:**  const express = require('express');  const app = express();  app.get('/', (req, res) => {  res.send('This is Express Demo Page');  });  app.get('/AboutUs', (req, res) => {  res.send('This is About Us page');  });  app.listen(4000, () => {  console.log('Server is running on http://localhost:4000');  });  **OUTPUT:**     Create an application to demonstrate various Node.js EventsAim 1: Write a program to demonstrate Event emitter class using properties like on and emit. **CODE:**  let events = require('events');  let eventEmitter = new events.EventEmitter();  let myEventHandler = () => {  console.log("Order Received! Baking a pizza...");  };  eventEmitter.on('order-pizza', myEventHandler);  eventEmitter.emit('order-pizza');  eventEmitter.removeListener('order-pizza', myEventHandler);  console.log("Listener Removed");  eventEmitter.removeAllListeners('order-pizza');  console.log("All listeners removed");  **OUTPUT:**   Aim2: Write a program to demonstrate the concept of event emitter object using function. **CODE:**  let events = require('events');  let eventEmitter = new events.EventEmitter();  eventEmitter.on('order-pizza', () => {  console.log("Order Received! Baking a pizza...");  });  eventEmitter.emit('order-pizza');  **OUTPUT:**   Aim3: Write a program to demonstrate the concept of event emitter object with multiple event handlers. **Code:**  let events = require('events');  var eventEmitter = new events.EventEmitter();  eventEmitter.on('order-pizza', (size, topping) => {  console.log(`Order Received! Baking a ${size} pizza with ${topping}...`);  });  eventEmitter.on('order-pizza', (size) => {  if (size === 'large') {  console.log("Serving Complimentary Sprite bottle!");  }  });  eventEmitter.emit('order-pizza', 'large', 'Black Olives');  **OUTPUT:**   3: Create an application to demonstrate Node.js Functions.Aim 1: Write a node.js program to demonstrate the Named function. **Code:**  function greetUser(name) {  return `Hello, It's ${name}!`;  }  function calculateSum(a, b) {  return a + b;  }  const userName = "Named Function";  console.log(greetUser(userName));  const num1 = 10;  const num2 = 20;  console.log(`The sum of ${num1} and ${num2} is:- ${calculateSum(num1, num2)}`);  **OUTPUT:**   Aim 2: Write a node.js program to demonstrate Function Expressions. **CODE:**  const greetUser = function(name) {  return `Hello, It's ${name}!`;  };  const calculateSum = function(a, b) {  return a + b;  };  const userName = "Function Expression";  console.log(greetUser(userName));  const num1 = 15;  const num2 = 25;  console.log(`The sum of ${num1} and ${num2} is: ${calculateSum(num1, num2)}`);  **OUTPUT:**   Aim 3: Write a node.js program to demonstrate Arrow Function. **CODE:**  const addArrow = (a, b) => {  return a+b;  }  const result1 = addArrow(5, 3);  console.log('Result using arrow function:', result1);  **OUTPUT:**   Aim4 : Write a node.js program to demonstrate Higher order function. **CODE:**  function greeting(message) {  return function(name) {  console.log(`${message}, ${name}!`);  }};  // Passing a function as an argument  function performAction(action, value) {  action(value);  }  const sayHello = greeting('Hello');  const sayGoodbye = greeting('Goodbye');  // Calling the higher-order function  performAction(sayHello, 'Raj');  performAction(sayGoodbye, 'John');  **OUTPUT:**   4: Using File Handling demonstrates all basic file operations (Create, write, read, delete).Aim 1:Write a node.js program to show how to use path.extname() method to get the file extension from a file path. **CODE:**  const path = require('path');  const filePath = '/home/user/Documents/file.txt';  const extname = path.extname(filePath);  console.log(`File extension: ${extname}`);  **OUTPUT:**   Aim 2: Write a node.js program to show how to use the path.resolve() method to resolve a sequence of paths or path segments into an absolute path. **CODE:**  const path = require('path');  const absolutePath = path.resolve('folder1', 'folder2', 'file.txt');  console.log(`Absolute path: ${absolutePath}`);  **OUTPUT:**   Aim 3:Write a node.js program to demonstrate the use of path.join() method. **CODE:**  const path = require('path');  const joinedPath = path.join('folder1', 'folder2', 'file.txt');  console.log(`Joined path: ${joinedPath}`);  **OUTPUT:**   Aim 4:Write a program to create a new empty file using open() method. **CODE:**  const fs = require('fs');  fs.open('newfile.txt', 'w', (err, file) => {  if (err) throw err;  console.log('New empty file created!');  });  **OUTPUT:**   Aim 5:Create a new file using the appendFile() method in node.js. **CODE:**  const fs = require('fs');  fs.appendFile('newfile.txt', 'Hello, this is some text!', (err) => {  if (err) throw err;  console.log('Text has been appended to the file!');  });  **OUTPUT:**     Aim 6:Create a Node.js file that reads the HTML file, and return the content. **CODE:**  const fs = require('fs');  fs.readFile('index.html', 'utf8', (err, data) => {  if (err) throw err;  console.log('HTML Content:');  console.log(data);  }); **OUTPUT:**   Aim 7:Write a program to read content of file in node.js synchronously and asynchronously. **CODE: Asynchronous**  const fs = require('fs');  fs.readFile('file.txt', 'utf8', (err, data) => {  if (err) throw err;  console.log('Asynchronous content:', data);  }); **OUTPUT:**    **CODE: Synchronous**  const fs = require('fs');  try {  const data = fs.readFileSync('file.txt', 'utf8');  console.log('Synchronous content:', data);  } catch (err) {  console.error(err);  }  **OUTPUT:**   Aim 8:Write a program to delete a file with the File System module (use the fs.unlink() method). **CODE:**  const fs = require('fs');  fs.unlink('newfile.txt', (err) => {  if (err) throw err;  console.log('File deleted!');  });  **OUTPUT:**   Aim 9:Write a program to rename a file with the File System module. **CODE:**  const fs = require('fs');  fs.rename('file.txt', 'newfile.txt', (err) => {  if (err) throw err;  console.log('File renamed!');  });  **OUTPUT:**   5: Create an HTTP Server and perform operations on it.Aim 1:Write a program to create an HTTP server that listens to server ports and gives a response back to the client using http module. **CODE:**  let http = require('http');  const server = http.createServer((req, res) => {  res.write('Response From Server: ABC');  res.end();  });  server.listen(3030, 'localhost', () => { console.log("Server running at http://localhost:3030");  });  **OUTPUT:**   Aim 2:Write a program which demonstrates how to set the content-type of HTTP Response? **CODE:**  const http = require('http');  const PORT = 3000;  const server = http.createServer((req, res) => {  res.writeHead(200, { 'Content-Type': 'text/plain' });  res.end('Hello, this is a response from your HTTP server!\n');  });  server.listen(PORT, () => { console.log(`Server is running and listening on [http://localhost:${PORT](about:blank)}`);  });  **OUTPUT:**   Aim 3:Write a program to shut down the http server. **CODE:**  let http = require('http');  const server = http.createServer((req, res) => {  res.setHeader('Content-Type', 'text/plain');  res.write('Response From Server:');  res.end();  });  server.listen(3030, 'localhost', () => {  console.log("Server running at http://localhost:3030");  server.close(() => { console.log("Server Closed");  })  });  **OUTPUT:**   Q6- Create an application to establish a connection with the MySQL database and perform basic database operations on it.Aim 1:Write a node.js program to connect with MySQL Database. **CODE:**  const mysql = require("mysql");  const connection = mysql.createConnection({  host: "localhost",  user: "root",  password: "password",  database: "MCA"  });  connection.connect(function(error){  if(error) throw error;  console.log("Database Connected");  });  **OUTPUT:**   Aim 2:Write a node.js program to select table data from MySQL database. **CODE:**  const mysql = require("mysql");  const connection = mysql.createConnection({  host: "localhost",  user: "root",  password: "password",  database: "MCA"  });  connection.connect(function(error){  connection.query('select \* from student', function(error, result){  if(error) throw error;  console.log(result);  })  })  **OUTPUT:**   Aim 3:Write node.js program to select only one record from MySQL database. **CODE:**  const mysql = require("mysql");  const connection = mysql.createConnection({  host: "localhost",  user: "root",  password: "password",  database: "MCA"  });  connection.connect(function(error){  connection.query('select \* from student', function(error, result){  if(error) throw error;  console.log(result[0]);  })  })  **OUTPUT:**   Aim 4:Write node.js program to select name of employee having id=1 from MySQL database. **CODE:**  const mysql = require("mysql");  const connection = mysql.createConnection({  host: "localhost",  user: "root",  password: "password",  database: "MCA"  });  connection.connect(function(error){  connection.query('select \* from student', function(error, result){  if(error) throw error;  console.log(result[0].StudentName);  })  })  **OUTPUT:**   Aim 5:Write a node.js program to create a table ‘emp’; **CODE:**  const mysql = require("mysql");  const connection = mysql.createConnection({  host: "localhost",  user: "root",  password: "password",  database: "MCA"  });  connection.connect(function(error) {  if (error) throw error;  console.log("Connected to the database");  connection.query(`CREATE TABLE emp(id INT, ename VARCHAR(10))`, function(error, results) {  if (error) throw error;  console.log("Table created successfully");  });  });  **OUTPUT:**   Aim 6:Write a node.js program to insert a record in the emp table. **CODE:**  const mysql = require("mysql");  const connection = mysql.createConnection({  host: "localhost",  user: "root",  password: "password",  database: "MCA"  });  connection.connect(function(error) {  if (error) throw error;  console.log("Connected to the database");  connection.query(`Insert into emp values(2, "MCA")`, function(error, results) {  if (error) throw error;  console.log("Inserted successfully");  });  });  **OUTPUT:**   7: Create an application in React JS to implement component life cycle. **Aim:**  **CODE:**  import React from "react";  import ReactDOM from "react-dom/client";  export default class App extends React.Component {  constructor(props) {  super(props);  this.state = { hello: "Hello MCA Students!" };  }  componentDidMount() {  console.log("componentDidMount()");  }  changeState() {  this.setState({ hello: "This is a node JS Practiacl!" });  }  render() {  return (  <div className="bg-gray-500 flex flex-col gap-7 h-96 p-4">  <h1>  React JS Component Life Cycle <br/>  {this.state.hello}  </h1>  <h2>  <a className="text-red-400 bg-green-200 p-4 mt-6 rounded-lg" onClick={this.changeState.bind(this)}>  Press Here!  </a>  </h2>  </div>  );  }  shouldComponentUpdate(nextProps, nextState) {  console.log("shouldComponentUpdate()");  return true;  }  componentDidUpdate() {  console.log("componentDidUpdate()");  }  }  **OUTPUT:**     8: Create an application to implement class and functional components in ReactJS. **Aim 1:Write a node.js program to implement class component**  **CODE:** Welcome.js import React, { Component } from "react";  class Welcome extends Component {  render() {  return (  <div>  <h1>This is Class Component..</h1>  <p>WEB Practical with class Baised Components</p>  </div>  )  }  }  export default Welcome;  **OUTPUT:**    **Aim 2:Write a node.js program to implement functional componen**t  **CODE:** functionComponents.js import React from "react";  const functionComponents = () => {  return (  <div>  <h1>Welcome To the Functional Component</h1>  <h3>This is Functional Component..</h3>  </div>  );  }  export default functionComponents;  **OUTPUT:**   9: Create an application in React JS to import and export the files (components). **Aim:**  **CODE:**  **Main Components: App.js**  import Content from "../components/Content";  import Footer from "../components/Footer";  import Header from "../components/Header";  import "./styles.css";  export default function App() {  return (  <div className="App">  <Header />  <Content />  <Footer />  </div>  );  }  **Header Components: Header.js**  import React from "react";  const Header = () => {  return (  <div> <h3>Welcome to My Header Component</h3> </div>  )};  export default Header;  **Footer Components: Footer.js**  import React from "react";  const Footer = () => {  return <h3>Welcome to My Footer Component</h3>;  };  export default Footer;  **Content Components: Content.js**  import React from "react";  const Content = () => {  return (  <main>  <p> This is the main content of the application<br /> which is another Component.  </p>  </main>  )};  export default Content;  **OUTPUT:**     10: Create an application to implement state and props. **Aim:**  **CODE: State:- App.js**  import { useState } from "react";  import "./styles.css";  export default function App() {  const [value, setValue] = useState("Sunshine");  const ValueChange = () => {  setValue("Starlight");  };  return (  <div className="product-wrapper">  <h2>{`My Name is ${value}`}</h2>  <button onClick={ValueChange}>Click To Change</button>  </div>  )};  **OUTPUT:**    **After Clicking Button:-**    **CODE: Props:-**  const data = [  {  pname: "Samsung Galaxy S22",  price: 72900,  rating: 4.5,  features: ["Android", "6.1 inch display", "8GB Ram"],  additional: "A smartphone with difference",  },  {  pname: "SOnePlus Nord 2T 5G",  price: 33999,  rating: 4.2,  features: ["Android", "6.1 inch display", "8GB Ram"],  additional: "A smartphone with difference",  },  {  pname: "Realme C11 2021",  price: 8900,  rating: 3.2,  features: ["Android", "6.1 inch display", "8GB Ram"],  additional: "A smartphone with difference",  }  ];  export default data;  **App.js**  import "./styles.css";  import data from "./data";  import ProductCart from "./ProductCart";  export default function App() {  return (  <div className="App">  {  data.map((product) => (  <ProductCart  pname={product.pname}  price={product.price}  rating={product.rating}  features={product.features}  additional={product.additional}  />  ))}  </div>  )  }  **ProductCart.js**  import "./styles.css";  export default function ProductCart({additional,pname,price,rating, features,}){  return (  <div className="product-wrapper">  <h2>{pname}</h2>  <h3>{price}</h3>  <h4>{rating}</h4>  <div>{features.join(",")}</div>  <button onClick={(event) => alert(additional)}>  More Info  </button>  </div>  )  };  **style.css**  .App {  font-family: sans-serif;  text-align: center;  display: flex;  justify-content: center;  }  .product-wrapper {  border: solid black 1px;  margin: 10px;  padding: 10px;  width: 400px;  background-color: aquamarine;  }  .product-wrapper button {  font-size: 1em;  border: solid black 1px;  padding: 10;  background: none;  cursor: pointer;  margin: 10px 0;  background-color: blueviolet;  color: wheat;  padding: 10px;  }  **OUTPUT:**   11: Create an application in ReactJS to use DOM events. **Aim :**  **CODE:**  import React, { useState } from "react";  function App() {  const [clickCount, setClickCount] = useState(0);  const [hoverText, setHoverText] = useState("Hover over the box!");  const [keyPressed, setKeyPressed] = useState("");  const handleButtonClick = () => {  setClickCount(clickCount + 1);  };  const handleMouseEnter = () => {  setHoverText("You are hovering!");  };  const handleMouseLeave = () => {  setHoverText("Hover over the box!");  };  const handleKeyDown = (event) => {  setKeyPressed(`You pressed: ${event.key}`);  };  return (  <div className="w-full h-screen flex justify-center items-center bg-sky-700">  <div className="App font-sans text-center bg-gray-300 flex flex-col justify-center items-center w-[600px] p-24 rounded-lg shadow-lg">  <h1 className="text-3xl font-bold text-gray-800 mb-6">  React DOM Events Example  </h1>  {/\* Click Event \*/}  <button onClick={handleButtonClick}  className="px-6 py-2 bg-blue-600 text-white rounded shadow-md hover:bg-blue-700 transition" >  Click Me!  </button>  <p className="mt-2 text-gray-800 font-semibold text-xl">  Button clicked {clickCount} times  </p>  {/\* Hover Event \*/}  <div onMouseEnter={handleMouseEnter}  onMouseLeave={handleMouseLeave}  className="w-48 h-24 bg-blue-100 rounded-lg shadow-md mx-auto my-6 flex items-center justify-center text-blue-600 border border-blue-300">  {hoverText}  </div>  {/\* Keyboard Event \*/}  <input type="text"  placeholder="Type something..."  onKeyDown={handleKeyDown}  className="border border-gray-400 rounded-md px-4 py-2 w-3/4 max-w-md mx-auto text-gray-800"/>  <p className="mt-2 text-gray-600">{keyPressed}</p>  </div>  </div>  )};  export default App;  **OUTPUT:**     12: Create an application to implement React Hooks. **Aim 1: USE STATE**  **CODE:-**  import { useState } from "react";  function App() {  let [count, setCount] = useState(0);  const Incment = () => {  setCount(count + 1);  };  const Decment = () => {  if (count > 0) {  setCount(count - 1);  }};  return (  <div className="flex justify-center items-center h-screen bg-gray-600">  <div className="bg-green-300 shadow-lg rounded-lg p-8 w-[500px] h-60">  <h1 className="text-5xl text-center text-black py-6 font-medium">My Count is {count}</h1>  <div className="flex justify-center gap-4">  <button className="px-4 py-2 bg-blue-500 text-white font-medium rounded hover:bg-blue-600" onClick={Incment}>  Increment by One  </button>  <button className="px-4 py-2 bg-red-500 text-white font-medium rounded hover:bg-red-600" onClick={Decment}>  Decrement by One  </button>  </div>  </div>  </div>  )};  export default App;  **OUTPUT:**      **AIM 2: UseEffect:-**  **CODE:**  import React, { useState, useEffect } from "react";  function App() {  const [users, setUsers] = useState([]);  const [loading, setLoading] = useState(true);  useEffect(() => {  const fetchData = async () => {  try {  const response = await fetch( "https://jsonplaceholder.typicode.com/users" );  const data = await response.json();  setUsers(data);  } catch (error) {  console.error("Error fetching data:", error);  } finally {  setLoading(false);  }};  fetchData();  }, []);  return (  <div className="min-h-screen bg-gray-500 flex flex-col items-center py-8">  <div className="bg-green-200 w-[800px] p-4">  <h1 className="text-3xl font-bold text-red-500 px-7">User List</h1>  {loading ? ( <p className="text-gray-700 text-lg">Loading...</p>  ) : (  <ul className="w-full max-w-md p-4 grid grid-cols-2">  {users.map((user) => (  <li key={user.id} className="py-2 px-4 last:border-none flex justify-between items-center">  <span className="text-gray-800 font-medium">{user.username}</span>  </li>  ))}  </ul>)}  </div>  </div>  )};  export default App;  **OUTPUT: Data Fetched from Api**   13: Create SPA using React Router. **Aim:**  **CODE:**  **App.js**  import React from "react";  import { Routes, Route } from "react-router-dom";  import Manatee from "./Manatee";  import Narwhal from "./Narwhal";  import Whale from "./Whale";  import "./styles.css";  function App() {  function Home() {  return (  <>  <h1>Marine Mammals</h1>  <nav>  <ul>  <li>  <a href="/manatee">Manatee</a>  </li>  <li>  <a href="/narwhal">Narwhal</a>  </li>  <li>  <a href="/whale">Whale</a>  </li>  </ul>  </nav>  </>  );  }  return (  <>  <Routes>  <Route path="/" element={<Home />} />  <Route path="/manatee" element={<Manatee />} />  <Route path="/narwhal" element={<Narwhal />} />  <Route path="/whale" element={<Whale />} />  </Routes>  </>  );  }  export default App;  **Manatee.js**  import React from "react";  export default function Manatee() {  return <h2>Manatee</h2>;  }  **Narwhal.js**  import React from "react";  export default function Narwhal() {  return <h2>Narwhal</h2>;  }  **Whale.js**  import React from "react";  export default function Whale() {  return <h2>Whale</h2>;  }  **styles.css**  \* {  font-family: sans-serif;  text-align: center;  background-color: darkgoldenrod;  }  nav ul {  display: flex;  gap: 20px;  text-decoration: none;  align-items: center;  justify-content: center;  text-align: center;  }  ul li {  list-style-type: none;  text-align: center;  }  a {  text-decoration: none;  text-align: center;  color: red;  font-size: 20px;  background-color: aquamarine;  padding: 10px;  border-radius: 8px;  }  **OUTPUT:**  **/home**    **/manatee**    **/narwhal** |