**EXPERIMENT NO.1**

Q. Create a webpage using HTML with proper structure and semantic elements.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>My Webpage</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<!-- Header section with a navigation bar -->

<header>

<nav>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#about">About</a></li>

<li><a href="#services">Services</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

</nav>

</header>

<!-- Main content section -->

<main>

<!-- Hero section with a welcoming message -->

<section id="home">

<h1>Welcome to My Webpage</h1>

<p>This is a simple webpage that uses semantic HTML elements to structure the content effectively.</p>

</section>

<!-- About section -->

<section id="about">

<h2>About Us</h2>

<p>We are a company focused on delivering quality services to our customers.</p>

</section>

<!-- Services section -->

<section id="services">

<h2>Our Services</h2>

<article>

<h3>Service 1</h3>

<p>Description of the first service offered.</p>

</article>

<article>

<h3>Service 2</h3>

<p>Description of the second service offered.</p>

</article>

<article>

<h3>Service 3</h3>

<p>Description of the third service offered.</p>

</article>

</section>

</main>

<!-- Footer section with contact information -->

<footer>

<section id="contact">

<h2>Contact Us</h2>

<address>

Email: <a href="mailto:info@mywebsite.com">info@mywebsite.com</a><br>

Phone: +123-456-7890

</address>

</section>

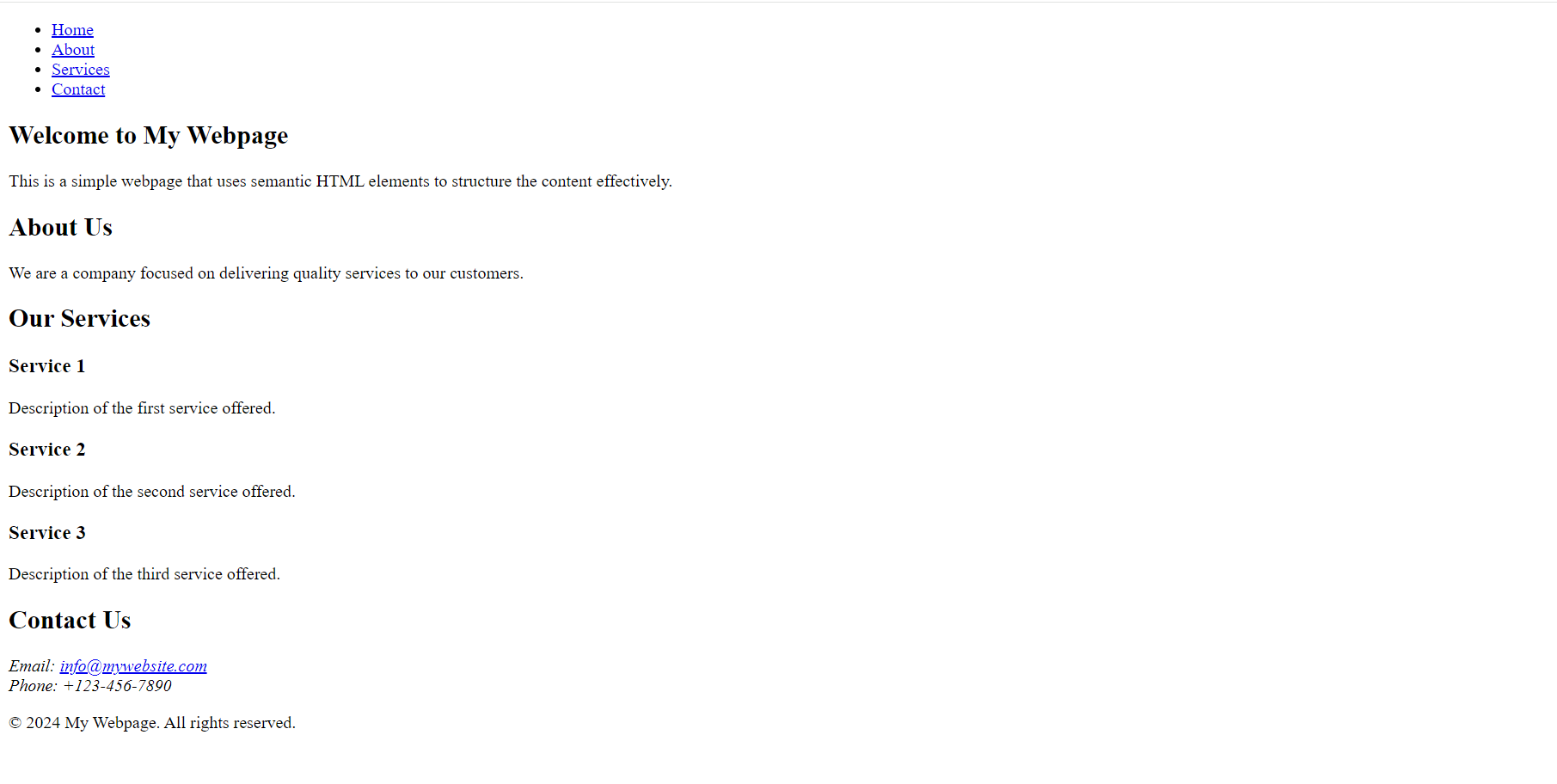
<p>&copy; 2024 My Webpage. All rights reserved.</p>

</footer>

</body>

</html>

**OUTPUT:**

****

**EXPERIMENT NO. 2**

Q. Style the web page created in Experiment 1 using CSS and Implement layout techniques such as Flexbox and Grid to create responsive and visually appealing layouts.

**1.HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>My Webpage</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<!-- Header section with a navigation bar -->

<header>

<nav>

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#about">About</a></li>

<li><a href="#services">Services</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

</nav>

</header>

<!-- Main content section -->

<main>

<!-- Hero section with a welcoming message -->

<section id="home">

<h1>Welcome to My Webpage</h1>

<p>This is a simple webpage that uses semantic HTML elements to structure the content effectively.</p>

</section>

<!-- About section -->

<section id="about">

<h2>About Us</h2>

<p>We are a company focused on delivering quality services to our customers.</p>

</section>

<!-- Services section -->

<section id="services">

<h2>Our Services</h2>

<div class="services-grid">

<article>

<h3>Service 1</h3>

<p>Description of the first service offered.</p>

</article>

<article>

<h3>Service 2</h3>

<p>Description of the second service offered.</p>

</article>

<article>

<h3>Service 3</h3>

<p>Description of the third service offered.</p>

</article>

</div>

</section>

</main>

<!-- Footer section with contact information -->

<footer>

<section id="contact">

<h2>Contact Us</h2>

<address>

Email: <a href="mailto:info@mywebsite.com">info@mywebsite.com</a><br>

Phone: +123-456-7890

</address>

</section>

<p>&copy; 2024 My Webpage. All rights reserved.</p>

</footer>

</body>

</html>

**2.CSS**

/\* General styles \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Header styling \*/

header {

background-color: #333;

padding: 1rem;

}

nav ul {

display: flex;

justify-content: center;

list-style: none;

margin: 0;

padding: 0;

}

nav ul li {

margin: 0 1rem;

}

nav ul li a {

color: white;

text-decoration: none;

font-weight: bold;

}

nav ul li a:hover {

color: #FFD700;

}

/\* Main content \*/

main {

padding: 2rem;

background-color: #f4f4f4;

text-align: center;

}

h1 {

color: #333;

}

h2 {

color: #555;

}

/\* Hero section \*/

#home {

padding: 2rem;

background-color: #ffebcd;

margin-bottom: 2rem;

border-radius: 10px;

}

/\* About section \*/

#about {

background-color: #eee;

padding: 2rem;

margin-bottom: 2rem;

border-radius: 10px;

}

/\* Services section - Grid layout \*/

.services-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 2rem;

margin-bottom: 2rem;

}

.services-grid article {

background-color: white;

padding: 1.5rem;

border-radius: 10px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.services-grid h3 {

color: #333;

}

/\* Footer styling \*/

footer {

background-color: #333;

color: white;

padding: 1rem;

text-align: center;

}

footer a {

color: #FFD700;

text-decoration: none;

}

footer a:hover {

text-decoration: underline;

}

footer p {

margin-top: 1rem;

}

/\* Responsive design \*/

@media (max-width: 600px) {

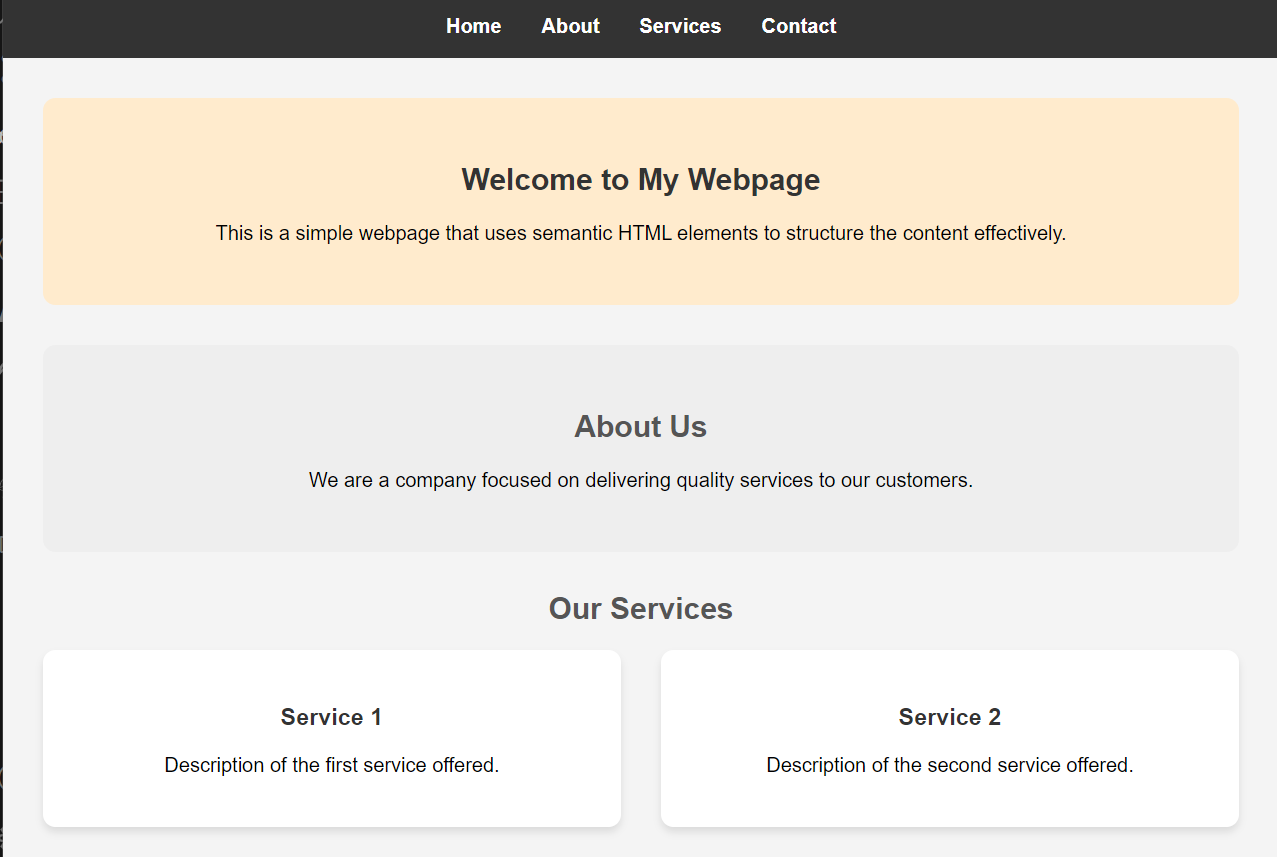
nav ul {

flex-direction: column;

}

}

**OUTPUT:**

****

**EXPERIMENT NO. 3**

Q. Implement event handling to respond to user interactions. Explore and compare popular client-side frameworks like React and Vue.js and Build a simple application using one of the chosen frameworks to understand its basic structure and functionality.

**App.js**

import React, { useState } from 'react';

import './App.css';

function App() {

// Declare a state variable 'count' and a method to update it 'setCount'

const [count, setCount] = useState(0);

return (

<div className="App">

<header className="App-header">

<h1>Simple Counter</h1>

<p>Count: {count}</p>

<button onClick={() => setCount(count + 1)}>

Increase

</button>

<button onClick={() => setCount(count - 1)}>

Decrease

</button>

<button onClick={() => setCount(0)}>

Reset

</button>

</header>

</div>

);

}

export default App;

**App.css**

.App {

text-align: center;

margin-top: 50px;

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

button {

padding: 10px 20px;

margin: 10px;

font-size: 16px;

cursor: pointer;

border: none;

border-radius: 5px;

background-color: #61dafb;

color: #282c34;

transition: background-color 0.3s;

}

button:hover {

background-color: #21a1f1;

}

**Vue.js Application**

<template>

<div class="hello">

<h1>Simple Counter</h1>

<p>Count: {{ count }}</p>

<button @click="increaseCount">Increase</button>

<button @click="decreaseCount">Decrease</button>

<button @click="resetCount">Reset</button>

</div>

</template>

<script>

export default {

data() {

return {

count: 0

};

},

methods: {

increaseCount() {

this.count++;

},

decreaseCount() {

this.count--;

},

resetCount() {

this.count = 0;

}

}

};

</script>

<style scoped>

.hello {

text-align: center;

margin-top: 50px;

}

button {

padding: 10px 20px;

margin: 10px;

font-size: 16px;

cursor: pointer;

border: none;

border-radius: 5px;

background-color: #42b983;

color: white;

transition: background-color 0.3s;

}

button:hover {

background-color: #2c8f6c;

}

</style>

**App.vue**

<template>

<div id="app">

<HelloWorld />

</div>

</template>

<script>

import HelloWorld from './components/HelloWorld.vue';

export default {

name: 'App',

components: {

HelloWorld

}

};

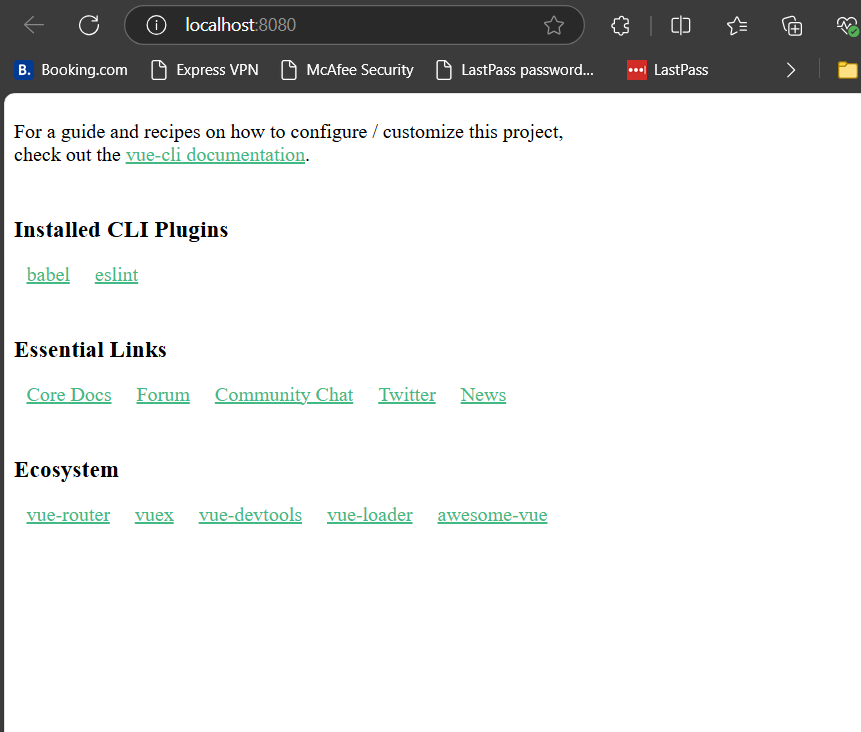
</script>

<style>

/\* You can keep the default styles or modify them as needed \*/

</style>

**OUTPUT:**

****

**EXPERIMENT NO. 4**

Q. Set up a basic Node.js server to handle HTTP requests and responses.

**Step 1: Install Node.js**

If you haven’t installed Node.js yet, follow these steps:

1. **Download and install Node.js** from the official website: [https://nodejs.org](https://nodejs.org/). Install the **LTS version**.
2. **Verify installation**:
   * Open your terminal (Command Prompt on Windows) and run:

bash

Copy code

node -v

npm -v

* + This should show the version numbers of Node.js and npm, confirming the installation.

**Step 2: Create a New Directory for the Server**

1. **Navigate to your desired location** (like the desktop) in your terminal:

bash

Copy code

cd Desktop

1. **Create a new directory** for your server:

bash

Copy code

mkdir my-node-server

1. **Move into the directory**:

bash

Copy code

cd my-node-server

**Step 3: Initialize a Node.js Project**

1. **Run npm init** to set up a new Node.js project:

bash

Copy code

npm init

1. You will be asked a series of questions (like project name, version, description, etc.). You can press **Enter** to accept the default values.

At the end, npm will create a package.json file which holds project information and dependencies.

**Step 4: Create a Basic Server Script**

1. **Create a new file** called server.js:
   * In your terminal:

bash

Copy code

touch server.js

1. Or manually create a new file named server.js in your my-node-server directory.
2. **Open the server.js file** in your text editor (like Visual Studio Code).
3. **Write the following code** inside server.js to create a basic HTTP server:

javascript

Copy code

// Import the HTTP module

const http = require('http');

// Create an HTTP server

const server = http.createServer((req, res) => {

// Set the response header (status code 200 and content type text)

res.writeHead(200, { 'Content-Type': 'text/plain' });

// Send a response to the client

res.end('Hello, World!\n');

});

// Set the server to listen on port 3000

server.listen(3000, () => {

console.log('Server is running on http://localhost:3000');

});

**Step 5: Run the Node.js Server**

1. **Save the server.js file** and go back to the terminal.
2. **Run the server** using Node.js:

bash

Copy code

node server.js

You should see a message in the terminal: Server is running on http://localhost:3000.

1. **Open a browser** and go to http://localhost:3000. You should see the message Hello, World! displayed in the browser.

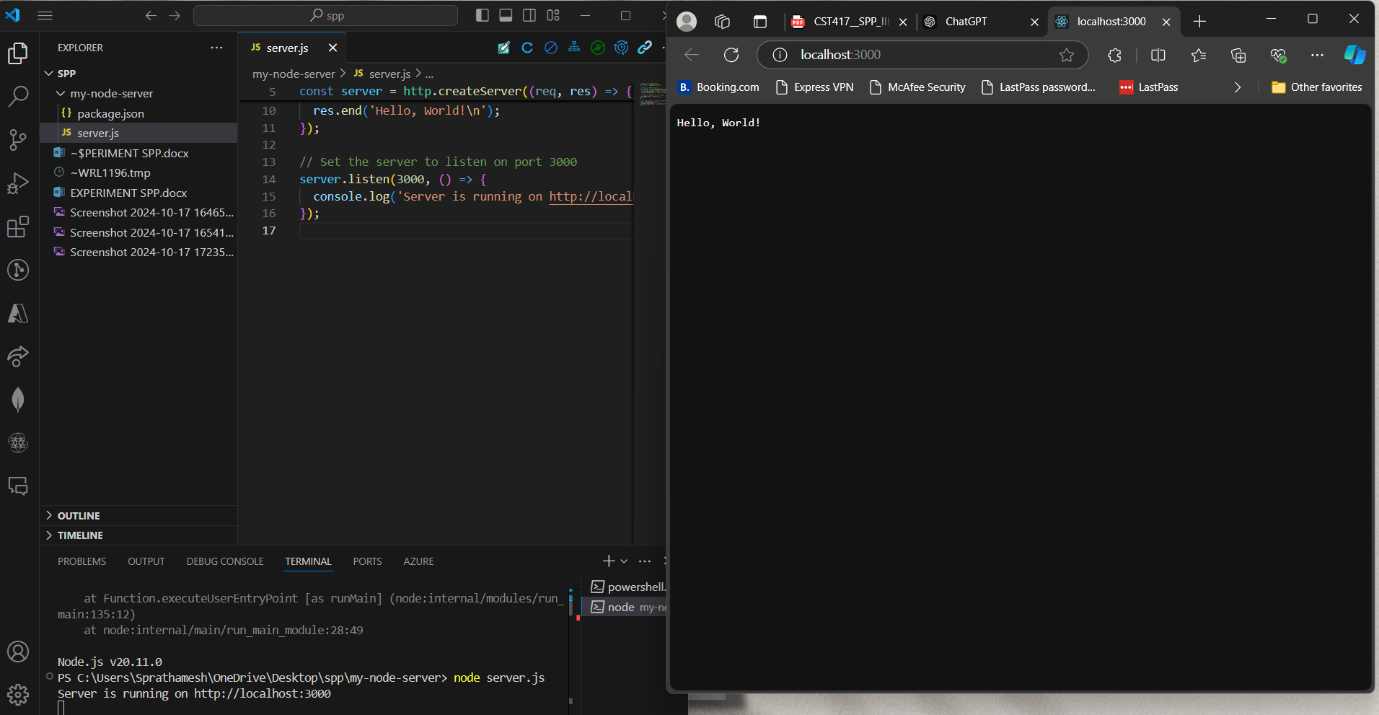
**Step 6: Understanding the Code**

* **const http = require('http');**: This imports Node’s built-in http module, which is used to create the server.
* **http.createServer((req, res) => {...})**: This creates an HTTP server. The (req, res) part represents the incoming request from the client (req) and the response that the server will send back (res).
* **res.writeHead(200, { 'Content-Type': 'text/plain' });**: This sets the response header, indicating that the response will be plain text and the status code is 200 (OK).
* **res.end('Hello, World!\n');**: This sends the actual response, which is Hello, World!.
* **server.listen(3000)**: This tells the server to listen for requests on port 3000.

**Step 7: Stop the Server**

* To stop the server, go back to the terminal where the server is running and press CTRL + C. This will stop the Node.js process.

**OUTPUT:**

****

**EXPERIMENT NO. 5**

Q. Implement routes to serve static files and handle dynamic content generation. Integrate a database (e.g., MongoDB) with the Node.js server.