

**Batch:C1**

**Roll No.: 16010122221**

**Experiment / assignment / tutorial No 5**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

**Title: Demonstrate the Use of node.js**

**AIM:** To implement the node js Concepts based on the following topics.

**Problem Definition:**

**Problem statement:**

Consider the basic concepts, which are useful in the creation of an application.

Considering the following points, demonstrate the functionality of each with a simple script

**1) Basic Routing:**

1. Build First server application using http module
2. Basic routing: Demonstrate it using simple HTML/Json file
3. Demonstrate the callback in node.js

**2) File operation**

- Check Permissions of a File or Directory.
- Checking if a file or a directory exists.
- Determining the line count of a text file.
- Reading a file line by line.
- See the file content through browser.

**3) Building your custom modules**

-To demonstrate this use some mathematics function to create custom module.

**\$) Blocking and Non Blocking**

---

**Expected OUTCOME of Experiment:**

**CO 2:** Illustrate the concepts of various front-end, back-end web application development technologies & frameworks using different web development tools.

---

**Books/ Journals/ Websites referred:**

1. Shelly Powers Learning Node O' Reilly 2 nd Edition, 2016.

**Pre Lab/ Prior Concepts:**

**Write details about the following content**

- Node.js.  
Ans) Node.js is a powerful, open-source runtime environment built on Chrome's V8 JavaScript engine that allows you to run JavaScript on the server side. It is used to build scalable network applications due to its non-blocking, event-driven architecture, making it ideal for real-time applications like chat servers, APIs, and more.
- Basic Routing  
Ans) Routing in Node.js refers to directing incoming requests to specific endpoints of a server based on the request URL and HTTP method. Using the http module, you can create a server that listens to different routes (e.g., /home, /about) and responds with different content such as HTML or JSON.
- Custom module  
Ans) Custom modules in Node.js allow you to organize your code into separate files and reuse functionality. You can create your own modules using module.exports and require(). This modular approach promotes code reuse and better organization.
- Asynchronous Programming

Ans) Asynchronous programming in Node.js allows the server to handle multiple tasks concurrently without waiting for each task to complete. It uses non-blocking I/O operations with callbacks, promises, or async/await, making the application more efficient.

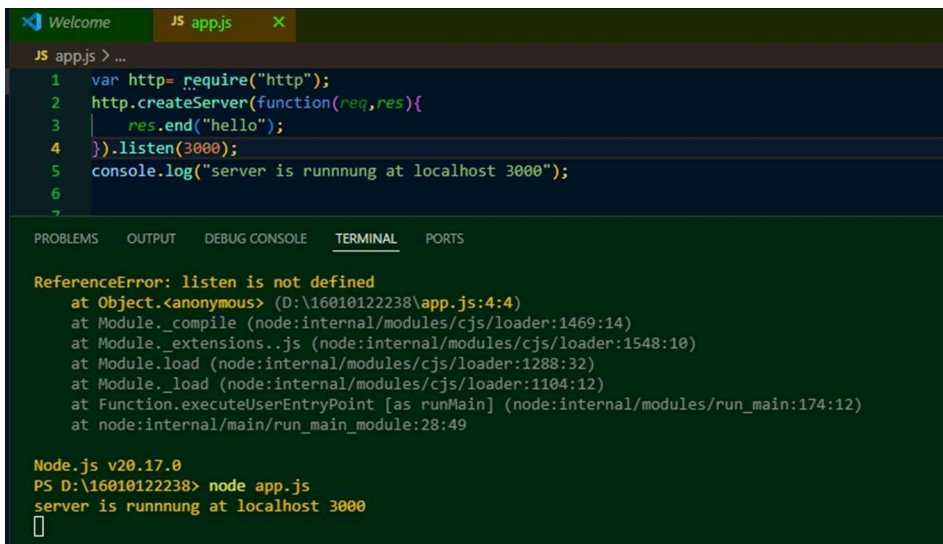
### Methodology:

- **Server Creation:** Uses the http module to create a basic server that listens on a specified port.
- **Basic Routing:** Handles requests based on the URL path, responding with different content (HTML, JSON, or error messages).
- **Custom Modules:** Demonstrates modular code by creating reusable components (e.g., mathematical functions) in separate files and importing them with require().
- **Asynchronous Programming:** Uses non-blocking I/O operations like fs.readFile() with callbacks, allowing the server to handle multiple tasks concurrently without waiting for each to finish.
- **Efficient and Scalable:** Combines routing, modular design, and asynchronous processing to create efficient, scalable server-side applications.

### Implementation Details:

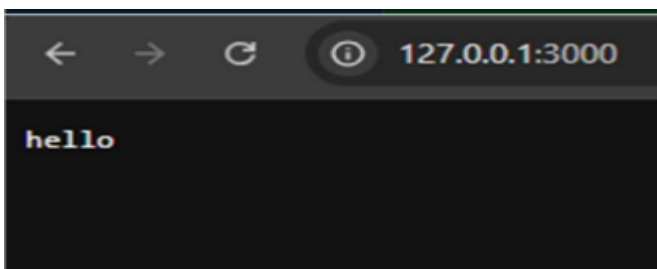
```
var http= require("http");
http.createServer(function(req,res){
  res.end("hello");
}).listen(3000);
console.log("server is runnung at localhost 3000");
```

#### 1.1 1) . Build First server application using http module



```
JS app.js > ...
1 var http= require("http");
2 http.createServer(function(req,res){
3   res.end("hello");
4 }).listen(3000);
5 console.log("server is runnung at localhost 3000");
6
ReferenceError: listen is not defined
    at Object.<anonymous> (D:\16010122238\app.js:4:4)
    at Module._compile (node:internal/modules/cjs/loader:1469:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1548:10)
    at Module.load (node:internal/modules/cjs/loader:1288:32)
    at Module._load (node:internal/modules/cjs/loader:1104:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:174:12)
    at node:internal/main/run_main_module:28:49

Node.js v20.17.0
PS D:\16010122238> node app.js
server is runnung at localhost 3000
```



## 1.2 Basic routing: Demonstrate it using simple HTML/Json file

App.js

```
const http = require('http');
const fs = require('fs');
const path = require('path');

const server = http.createServer((req, res) => {
  console.log(`Request for ${req.url} received.`);

  if (req.url === '/') {

    fs.readFile(path.join( dirname, 'index.html'), (err, data) => {
      if (err) {
        res.statusCode = 500;
        res.end('Error loading HTML file.');
```

```
        return;
      }
      res.statusCode = 200;
      res.setHeader('Content-Type', 'text/html');
      res.end(data);
    });
  } else if (req.url === '/data') {

    fs.readFile(path.join( dirname, 'data.json'), (err, data) => {
      if (err) {
        res.statusCode = 500;
        res.end('Error loading JSON file.');
```

```
        return;
      }
      res.statusCode = 200;
      res.setHeader('Content-Type', 'application/json');
      res.end(data);
    });
  } else {
    res.statusCode = 404;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Not Found');
```

```
  }
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

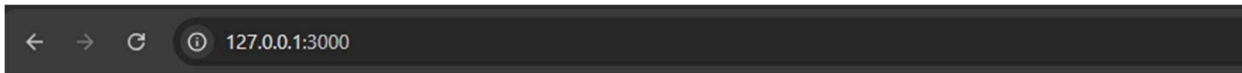
Data.html

```
{  
  "message": "Hello from Akshat!"  
}
```

Index.html

```
<!DOCTYPE html>  
<html>  
<head>  
<title>My Node.js Server</title>  
</head>  
<body>  
<h1>Hii ,Welcome to my Node.js Server!</h1>  
</body>  
</html>
```

Output



**Hii ,Welcome to my Node.js Server!**

### 1.3 [Demonstrate the callback in node.js](#)

```
// callback_example.js  
const fs = require('fs');  
  
function readFileCallback(err, data) {  
  if (err) {  
    console.error('Error reading file:', err);  
  }  
}
```

```

return;
}
console.log('File contents:', data.toString());
}

fs.readFile('example.txt', readFileCallback);

```

Example. Txt

EXP5

JS app.js

example.txt

example.txt

1 Hello, this is a sample text file!

2

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

PS C:\Users\Lenovo\Desktop\exp5> node app.js  
File contents: Hello, this is a sample text file!

2) Check Permissions of a File or Directory.

```

- const fs = require('fs');
-
- const filePath = 'example.txt';
-
- fs.access(filePath, fs.constants.R_OK | fs.constants.W_OK, (err) => {
-   if (err) {
-     console.error('No read/write permission for ${filePath}');
-   } else {
-     console.log(`${filePath} has read and write permissions`);
-   }
- });

```

```

PS C:\Users\Lenovo\Desktop\exp5> node app.js
example.txt has read and write permissions

```

- Checking if a file or a directory exists. App.js

```

const fs = require('fs');
const filePath = 'example.txt';

fs.stat(filePath, (err, stats) => {

  if (err) {
    console.log('File or directory does not exist: ${filePath}');
  } else {
    if (stats.isFile()) {

```

```
return;
}
console.log('File contents:', data.toString());
}

fs.readFile('example.txt', readFileCallback);
```

```
PS C:\Users\Lenovo\Desktop\exp5> node app.js
example.txt is a file.
```

- Determining the line count of a text file.

App.js

```
const fs = require('fs');
const filePath = 'example.txt';
fs.readFile(filePath, 'utf8', (err, data) => {
  if (err) {
    console.error(err);
    return;
  }
  const lines = data.split('\n').length;
  console.log(`Line count: ${lines}`);
});
```

PS C:\Users\Lenovo\Desktop\exp5> node app.js

Line count: 2

Department of Engineering,

Department of Computer

const fs = require('fs');

const readline = require('readline');

const filePath = 'example.txt';

const rl = readline.createInterface({  
 input: fs.createReadStream(filePath),  
 output: process.stdout,  
 terminal: false  
});

rl.on('line', (line) => {  
 console.log(`Line: \${line}`);  
});

Reading a file line by line.





```
PS C:\Users\Lenovo\Desktop\exp5> node app.js  
Line: Hello, this is a sample text file!
```

See the file content through browser.

```
const http = require('http');  
const fs = require('fs');  
  
const filePath = 'example.txt';  
  
const server = http.createServer((req, res) => {  
  if (req.url === '/') {  
    fs.readFile(filePath, 'utf8', (err, data) => {  
      if (err) {  
        res.statusCode = 500;  
        res.setHeader('Content-Type', 'text/plain');  
        res.end('Error reading file');  
        return;  
      }  
      res.setHeader('Content-Type', 'text/plain');  
      res.end(data);  
    });  
  } else {  
    res.statusCode = 404;  
    res.setHeader('Content-Type', 'text/plain');  
    res.end('Page not found');  
  }  
});  
  
const port = 3000;  
server.listen(port, () => {  
  console.log(`Server running at http://localhost:${port}/`);  
});
```

← → ↻ ⓘ localhost:3000

Hello, this is a sample text file!

3)

### Building your custom modules

-To demonstrate this use some mathematics function to create custom module.

```
// mathModule.js
function add(a, b) {
    return a + b;
}

function subtract(a, b) {
    return a - b;
}

function multiply(a, b) {
    return a * b;
}

function divide(a, b) {
    if (b === 0) {
        throw new Error("Cannot divide by zero");
    }
    return a / b;
}

module.exports = {
    add,
    subtract,
    multiply,
    divide
};
```

mathModule.js  
App.js

```
const math = require('./mathModule');
const num1 = 10;
const num2 = 5;

console.log(`Addition: ${num1} + ${num2} = ${math.add(num1, num2)}`);
console.log(`Subtraction: ${num1} - ${num2} = ${math.subtract(num1, num2)}`);
console.log(`Multiplication: ${num1} * ${num2} = ${math.multiply(num1, num2)}`);
console.log(`Division: ${num1} / ${num2} = ${math.divide(num1, num2)}`);
```



```
PS C:\Users\Lenovo\Desktop\exp5> node app.js
Addition: 10 + 5 = 15
Subtraction: 10 - 5 = 5
Multiplication: 10 * 5 = 50
Division: 10 / 5 = 2
```

```
const data = fs.readFileSync('example.txt', 'utf8');
console.log(data);

console.log('File read completed!');
```

```
No const fs = require('fs');

console.log('Start reading file...');

fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) {
    console.error(err);
    return;
  }
  console.log(data);
});

console.log('File read initiated, moving to next task!');
```

```
PS C:\Users\Lenovo\Desktop\exp5> node app.js
Start reading file...
File read initiated, moving to next task!
Hello, this is a sample text file!
```

**Conclusion: Thus , we were able to successfully execute and Demonstrate node.js concepts and execute them**