

| Title: Implementation of Node JS |
| --- |

**AIM:** To Implement the Concept of Node JS

**Problem Definition:**

-Demonstrate the Concept of Node JS With the help of Example.

\*(Students have to perform the task assigned within group and demonstrate the same).

**Resources used:**

* Google
* VS code

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO 1:**.Build full stack applications in Node using the MERN technologies.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

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

**Pre Lab/ Prior Concepts:**

Node.js is a powerful, open-source runtime environment that enables JavaScript to be used for server-side development. Built on Chrome's V8 JavaScript engine, Node.js allows developers to build scalable and high-performance applications using JavaScript. Its non-blocking, event-driven architecture makes it particularly well-suited for I/O-intensive tasks, such as handling multiple simultaneous connections. Node.js comes with a rich ecosystem of libraries and frameworks accessible through npm (Node Package Manager), which simplifies the development of web servers, real-time applications, and other networked software. Its single-threaded nature, combined with asynchronous processing, offers efficient performance and scalability, making Node.js a popular choice for modern web development.

**Implementation Details:**

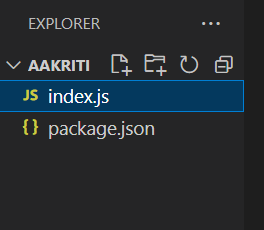
Students have to write stepwise details of implementation.

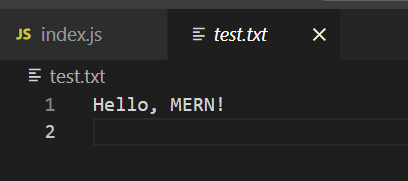
Problem statement:

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) File operation

- CRUD operations  
  




const fs = require('fs');

const path = './test.txt'

;

fs.writeFileSync(path, 'Hello, MERN!\n');

console.log('File created , written to test.txt.');

const data = fs.readFileSync(path, 'utf8');

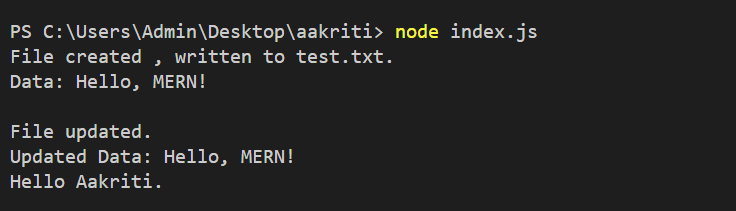
console.log('Data:', data);

fs.appendFileSync(path, 'Hello World.\n');

console.log('File updated.');

const updatedData = fs.readFileSync(path, 'utf8');

console.log('Updated Data:', updatedData);



const fs = require('fs');

const path = './test.txt';

fs.writeFileSync(path, 'Hello, MERN!\n');

console.log('File created , written to test.txt.');

const data = fs.readFileSync(path, 'utf8');

console.log('Data:', data);

fs.appendFileSync(path, 'Hello World.\n');

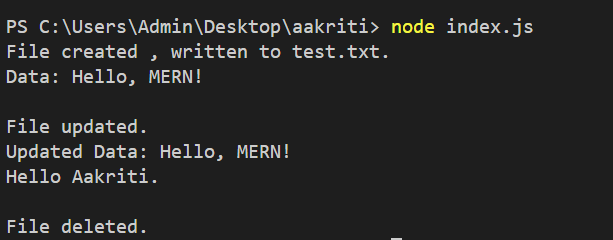
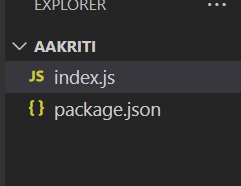
console.log('File updated.');

const updatedData = fs.readFileSync(path, 'utf8');

console.log('Updated Data:', updatedData);

fs.unlinkSync(path);

console.log('File deleted.');

- Check Permissions of a File or Directory.

const fs = require('fs');

const path = './test.txt';

fs.access(path, fs.constants.F\_OK, (err) => {

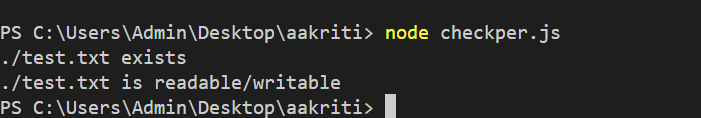
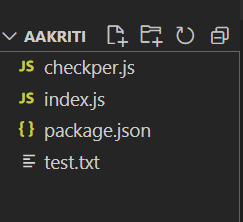
console.log(`${path} ${err ? 'does not exist' : 'exists'}`);

});

fs.access(path, fs.constants.R\_OK | fs.constants.W\_OK, (err) => {

console.log(`${path} ${err ? 'is not readable/writable' : 'is readable/writable'}`);

});



const fs = require('fs');

const path = './example.txt';

fs.access(path, fs.constants.F\_OK, (err) => {

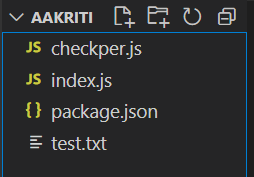
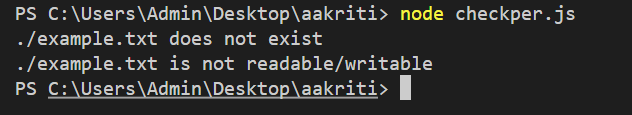
console.log(`${path} ${err ? 'does not exist' : 'exists'}`);

});

fs.access(path, fs.constants.R\_OK | fs.constants.W\_OK, (err) => {

console.log(`${path} ${err ? 'is not readable/writable' : 'is readable/writable'}`);

});

- Checking if a file or a directory exists.

const fs = require('fs');

const fileOrDirPath = './testDir';

fs.access(fileOrDirPath, fs.constants.F\_OK, (err) => {

console.log(`${fileOrDirPath} ${err ? 'does not exist' : 'exists'}`);

});

- Determining the line count of a text file.

const readline = require('readline');

const fs = require('fs');

const filePath = './test.txt';

const readStream = fs.createReadStream(filePath);

const rl = readline.createInterface({

input: readStream,

crlfDelay: Infinity

});

let lineCount = 0;

rl.on('line', () => {

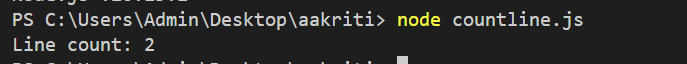
lineCount++;

});

rl.on('close', () => {

console.log(`Line count: ${lineCount}`);

});



- Reading a file line by line.

const readline = require('readline');

const fs = require('fs');

const filePath = './test.txt';

const readStream = fs.createReadStream(filePath);

const rl = readline.createInterface({

input: readStream,

crlfDelay: Infinity

});

rl.on('line', (line) => {

console.log(`${line}`);

});

- See the file content through browser.

const http = require('http');

const fs = require('fs');

const path = require('path');

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

if (req.method === 'GET' && req.url === '/') {

const filePath = path.join(\_\_dirname, 'test.txt');

fs.readFile(filePath, 'utf8', (err, data) => {

if (err) {

res.writeHead(500, { 'Content-Type': 'text/plain' });

res.end('Error reading file');

return;

}

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

res.end(data);

});

} else {

res.writeHead(404, { 'Content-Type': 'text/plain' });

res.end('Not Found');

}

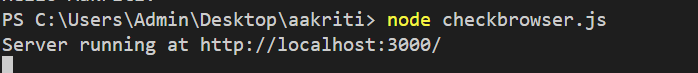
});

const PORT = 3000;

server.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}/`);

});



2) Building your custom modules

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

};

app.js

const math = require('./math');

const a = 10;

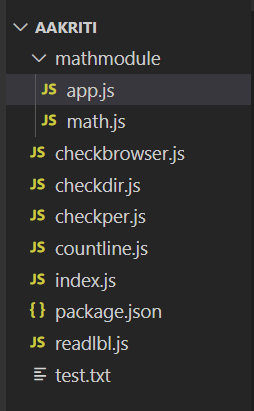
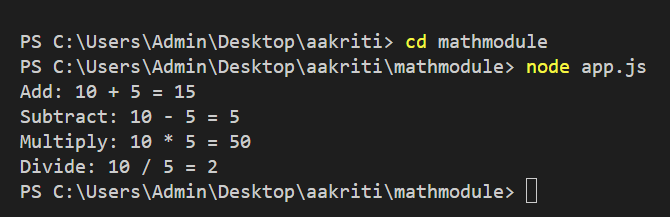
const b = 5;

console.log(`Add: ${a} + ${b} = ${math.add(a, b)}`);

console.log(`Subtract: ${a} - ${b} = ${math.subtract(a, b)}`);

console.log(`Multiply: ${a} \* ${b} = ${math.multiply(a, b)}`);

console.log(`Divide: ${a} / ${b} = ${math.divide(a, b)}`);

3) Basic Routing:

1. Build First server application using http module

const http = require('http');

const fs = require('fs');

const path = require('path');

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

let filePath = '.' + req.url;

if (filePath === './') filePath = './index.html';

const extname = path.extname(filePath);

let contentType = 'text/html';

switch (extname) {

case '.json':

contentType = 'application/json';

break;

case '.js':

contentType = 'text/javascript';

break;

case '.css':

contentType = 'text/css';

break;

}

fs.readFile(filePath, (err, content) => {

if (err) {

if (err.code === 'ENOENT') {

fs.readFile('./404.html', (error, errorContent) => {

res.writeHead(404, { 'Content-Type': 'text/html' });

res.end(errorContent, 'utf8');

});

} else {

res.writeHead(500);

res.end('Server Error: ' + err.code);

}

} else {

res.writeHead(200, { 'Content-Type': contentType });

res.end(content, 'utf8');

}

});

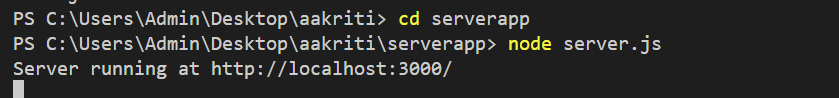
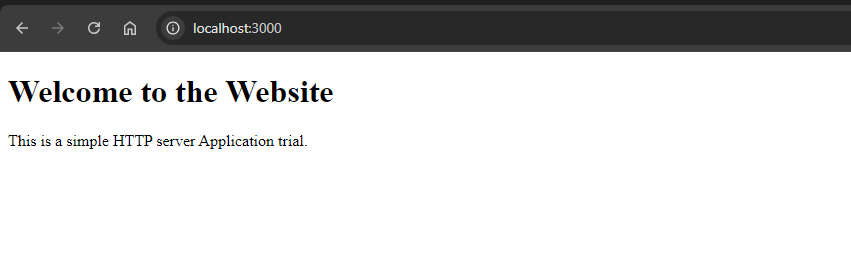
});

const PORT = 3000;

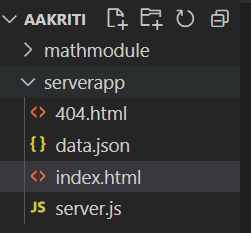
server.listen(PORT, () => {

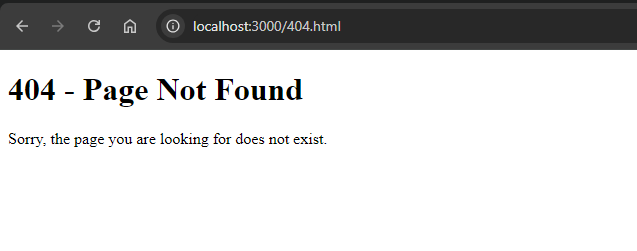
console.log(`Server running at http://localhost:${PORT}/`);

});

2. Basic routing: Demonstrate it using simple HTML/Json file

{



3. Demonstrate the callback in node.js   
  
server.js

const http = require('http');

const fs = require('fs');

const path = require('path');

// Helper function to read a file

function readFile(filePath, callback) {

fs.readFile(filePath, (err, content) => {

if (err) {

callback(err, null);

} else {

callback(null, content);

}

});

}

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

let filePath = '.' + req.url;

if (filePath === './') filePath = './index.html';

const extname = path.extname(filePath);

let contentType = 'text/html';

switch (extname) {

case '.json':

contentType = 'application/json';

break;

case '.js':

contentType = 'text/javascript';

break;

case '.css':

contentType = 'text/css';

break;

}

readFile(filePath, (err, content) => {

if (err) {

if (err.code === 'ENOENT') {

readFile('./404.html', (error, errorContent) => {

res.writeHead(404, { 'Content-Type': 'text/html' });

res.end(errorContent, 'utf8');

});

} else {

res.writeHead(500);

res.end('Server Error: ' + err.code);

}

} else {

res.writeHead(200, { 'Content-Type': contentType });

res.end(content, 'utf8');

}

});

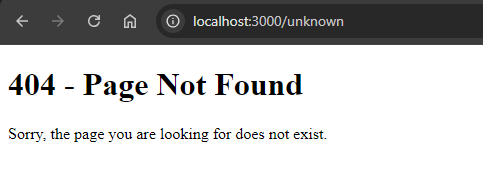
});

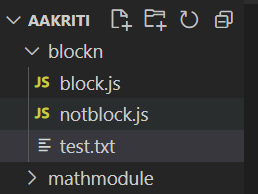
const PORT = 3000;

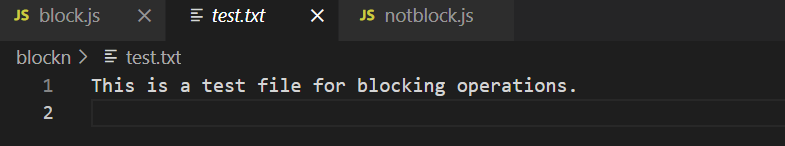
server.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}/`);

});



4) Blocking and Non Blocking  


Block:  


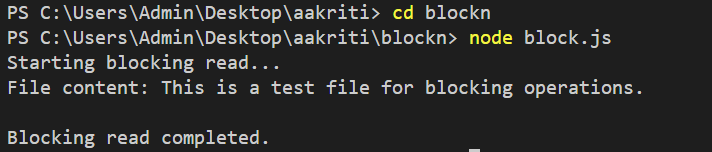
const fs = require('fs');

console.log('Starting blocking read...');

const data = fs.readFileSync('test.txt', 'utf8');

console.log('File content:', data);

console.log('Blocking read completed.');

  
  
Non-Block:

const fs = require('fs');

console.log('Starting non-blocking read...');

fs.readFile('test.txt', 'utf8', (err, data) => {

if (err) {

console.error('Error reading file:', err);

return;

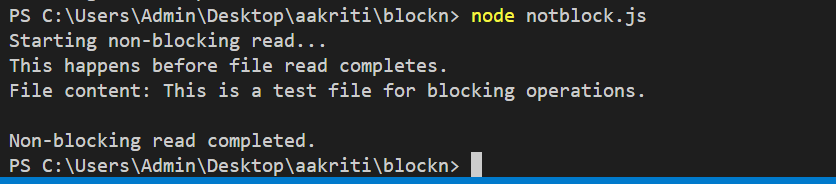
}

console.log('File content:', data);

console.log('Non-blocking read completed.');

});

console.log('This happens before file read completes.');



**Conclusion:**

Learned basic node commands to build MERN application.