**Batch: D - 1 Roll No.: 16010122096**

**Experiment / assignment / tutorial No. 02**

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

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

|  |
| --- |
| Title: Implementation of Node.js |

**AIM:** To Implement the Concept of Node.js

**Problem Definition:**

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

- 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.

2) Building your custom modules

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

3) 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

4) Blocking and Non Blocking

**Resources used:**

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

**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:**

Before starting with Node.js, ensure you understand JavaScript basics, especially asynchronous programming (callbacks and promises). Familiarize yourself with file system operations and HTTP protocols. Knowledge of creating and using modules, along with synchronous and asynchronous operations, will be crucial for effectively implementing Node.js functionalities.

**Implementation Details:**

**1] File Operations:**

const fs = require('fs').promises;

const readline = require('readline');

const fsSync = require('fs');

*// Read file*

async function readFile(*filePath*) {

    try {

        const data = await fs.readFile(filePath);

        console.log("File Content:", data.toString());

    } catch (err) {

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

    }

}

*// Write file*

async function writeFile(*filePath*, *content*) {

    try {

        await fs.writeFile(filePath, content);

        console.log("Data written successfully");

    } catch (err) {

        console.error("Error writing file:", err);

    }

}

*// Append file*

async function appendToFile(*filePath*, *content*) {

    try {

        await fs.appendFile(filePath, content);

        console.log("Update operation!");

    } catch (err) {

        console.error("Error appending to file:", err);

    }

}

*// Delete file*

async function deleteFile(*filePath*) {

    try {

        await fs.unlink(filePath);

        console.log("File deleted successfully!");

    } catch (err) {

        console.error("Error deleting file:", err);

    }

}

*// Check permissions*

async function checkPermissions(*path*) {

    try {

        await fs.access(path, fs.constants.R\_OK | fs.constants.W\_OK);

        console.log(`${path} is readable and writable`);

    } catch (err) {

        if (err.code === 'EACCES') {

            console.log(`${path} is not accessible (read or write permissions missing)`);

        } else {

            console.error(`Error checking permissions for ${path}:`, err.message);

        }

    }

}

*// Display permissions*

async function displayPermissions(*path*) {

    try {

        const stats = await fs.stat(path);

        const mode = stats.mode;

        console.log(`Permissions for ${path}:`);

        console.log(`- Read: ${Boolean(mode & fs.constants.S\_IRUSR)}`);

        console.log(`- Write: ${Boolean(mode & fs.constants.S\_IWUSR)}`);

        console.log(`- Execute: ${Boolean(mode & fs.constants.S\_IXUSR)}`);

    } catch (err) {

        console.error(`Error retrieving permissions for ${path}:`, err.message);

    }

}

*// Change permissions*

async function changePermissions(*path*, *mode*) {

    try {

        await fs.chmod(path, mode);

        console.log(`Permissions for ${path} changed to ${mode.toString(8)}`);

    } catch (err) {

        console.error(`Error changing permissions for ${path}:`, err.message);

    }

}

*// Check if file or directory exists*

async function checkIfExists(*path*) {

    try {

        await fs.access(path);

        console.log(`${path} exists`);

    } catch (err) {

        console.error(`${path} does not exist:`, err.message);

    }

}

*// Count lines*

async function countLines(*filePath*) {

    try {

        const data = await fs.readFile(filePath, 'utf-8');

        const lines = data.split('\n');

        lines.forEach((*line*, *index*) => {

            console.log(`Line ${index + 1}: ${line}`);

        });

        console.log(`\nTotal number of lines: ${lines.length}`);

    } catch (err) {

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

    }

}

*// Read lines*

async function readLines(*filePath*) {

    return new Promise((*resolve*, *reject*) => {

        const fileStream = fsSync.createReadStream(filePath);

        const rl = readline.createInterface({

            input: fileStream,

            crlfDelay: Infinity

        });

        rl.on('line', (*line*) => {

            console.log(`Line from file: ${*line*}`);

        });

        rl.on('close', () => {

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

            resolve(); *// Resolve the promise when reading is complete*

        });

        rl.on('error', (*err*) => {

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

            reject(*err*); *// Reject the promise if an error occurs*

        });

    });

}

*// Main function*

async function main() {

    const filePath = 'file.txt';

    console.log('Checking if file exists:');

    await checkIfExists(filePath);

    console.log('\nWriting to file:');

    await writeFile(filePath, 'Sample text');

    console.log('\nAppending to file:');

    await appendToFile(filePath, '\nHello Sir, How are you?');

    console.log('\nReading file content:');

    await readFile(filePath);

    console.log('\nReading file line by line:');

    await readLines(filePath);

    console.log('\nCounting lines in file:');

    await countLines(filePath);

    console.log('\nChecking initial permissions:');

    await displayPermissions(filePath);

    await checkPermissions(filePath);

    console.log('\nChanging permissions to read and write for owner only (mode 0o644):');

    await changePermissions(filePath, 0o644);

    console.log('\nChecking permissions after change:');

    await displayPermissions(filePath);

    await checkPermissions(filePath);

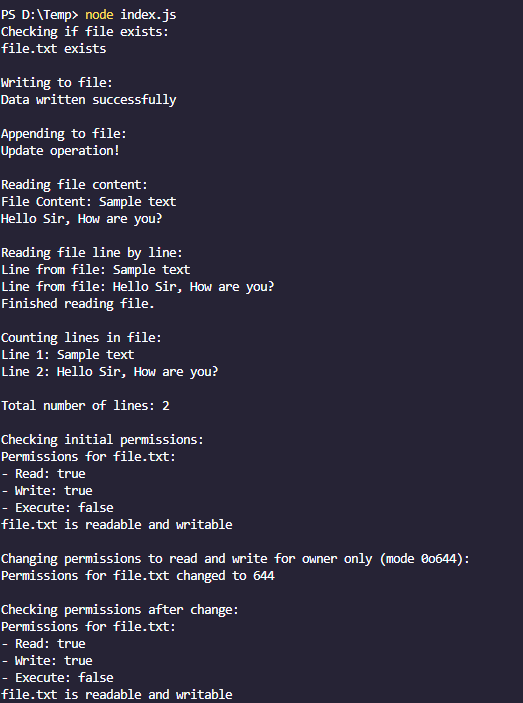
    console.log('\nDeleting file:');

    await deleteFile(filePath);

}

main();

**Output:**

****

**2] Custom Modules:**

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

};

const mathUtils = require('./mathModules');

const a = 10;

const b = 5;

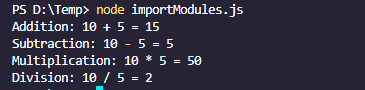
console.log(`Addition: ${a} + ${b} = ${mathUtils.add(a, b)}`);

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

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

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

**Output:**

****

**3] Basic Routing and Server creation:**

const http = require('http');

const fs = require('fs').promises;

const path = require('path');

const PORT = 3000;

const HTML\_FILE\_PATH = path.join(\_\_dirname, 'home.html');

const JSON\_FILE\_PATH = path.join(\_\_dirname, 'data.json');

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

    if (*req*.method === 'GET') {

        if (*req*.url === '/') {

            try {

                const data = await fs.readFile(HTML\_FILE\_PATH, 'utf-8');

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

*res*.end(data);

            } catch (err) {

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

*res*.end('Error reading HTML file');

            }

        } else if (*req*.url === '/data') {

            try {

                const data = await fs.readFile(JSON\_FILE\_PATH, 'utf-8');

*res*.writeHead(200, { 'Content-Type': 'application/json' });

*res*.end(data);

            } catch (err) {

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

*res*.end('Error reading JSON file');

            }

        } else {

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

*res*.end('Not Found');

        }

    } else {

*res*.writeHead(405, { 'Content-Type': 'text/plain' });

*res*.end('Method Not Allowed');

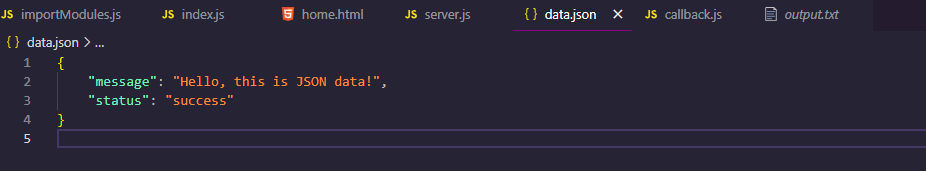
    }

});

server.listen(PORT, () => {

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

});

****

<!DOCTYPE *html*>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

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

    <title>Sample Server</title>

</head>

<body>

    <h1>Welcome to the Sample Server</h1>

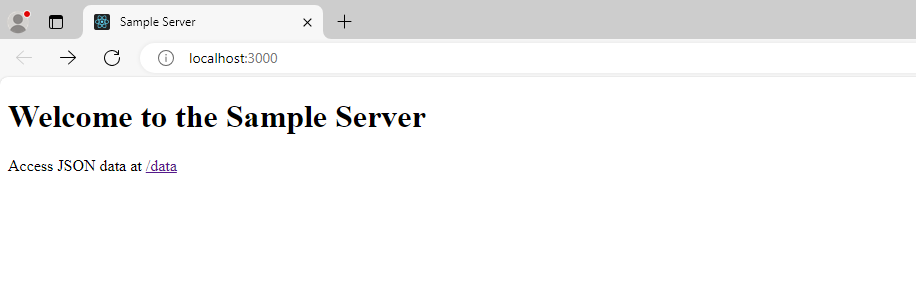
    <p>Access JSON data at <a *href*="/data">/data</a></p>

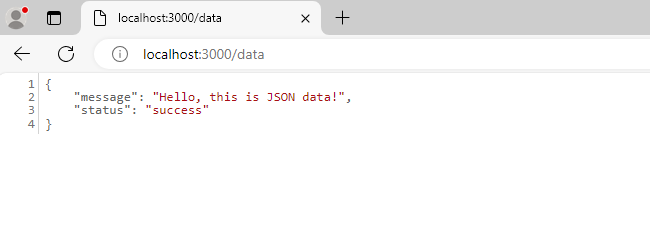
</body>

</html>

**Output:**

****

****

****

**Callback.js**

const fs = require('fs');

function readFile(*filePath*, *callback*) {

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

        if (*err*) {

            return callback(*err*);

        }

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

        callback(null, *data*);

    });

}

function writeFile(*filePath*, *content*, *callback*) {

    fs.writeFile(*filePath*, *content*, (*err*) => {

        if (*err*) {

            return callback(*err*);

        }

        console.log('Data written successfully');

        callback(null);

    });

}

function processFiles() {

    const readPath = 'input.txt';

    const writePath = 'output.txt';

    readFile(readPath, (*err*, *data*) => {

        if (*err*) {

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

            return;

        }

        writeFile(writePath, *data*, (*err*) => {

            if (*err*) {

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

                return;

            }

            readFile(writePath, (*err*, *newData*) => {

                if (*err*) {

                    console.error('Error reading the newly written file:', *err*);

                    return;

                }

                console.log('New file content read again:', *newData*);

            });

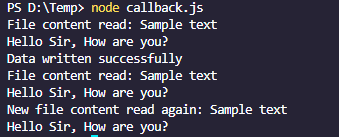
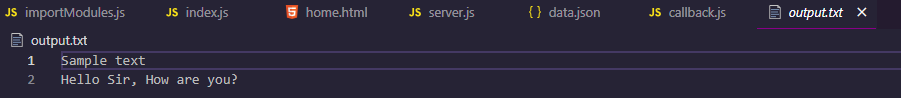
        });

    });

}

processFiles();

**Output:**

4] **Blocking & Non – Blocking:**

**Blocking**

const fs = require('fs');

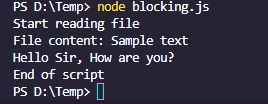
console.log('Start reading file');

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

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

console.log('End of script');

**Output:**



**Non – Blocking:**

const fs = require('fs');

console.log('Start reading file');

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

    if (*err*) {

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

        return;

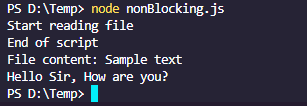
    }

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

});

console.log('End of script');

**Output:**

****

**Conclusion:**

Node.js enables efficient file operations, custom module creation, basic server routing, and demonstrates blocking vs. non-blocking behavior. Using non-blocking methods enhances performance and scalability.