

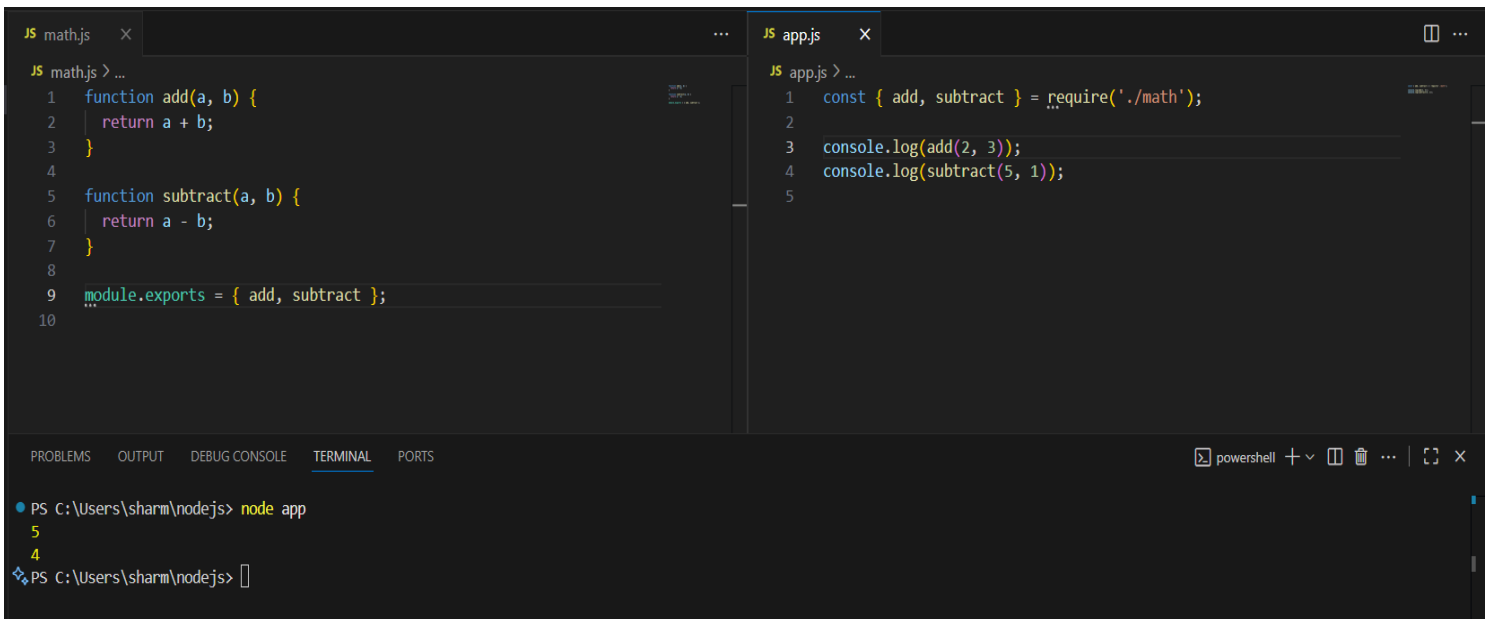
Practical – 3

1. Modules

- Task:

- Create a file called **math.js** to export add and subtract functions.
- Import these functions into **app.js** and use them.

Output:



The screenshot shows a Visual Studio Code editor with two open files: `math.js` and `app.js`. The `math.js` file defines two functions, `add` and `subtract`, and exports them. The `app.js` file imports these functions and uses them. The terminal window at the bottom shows the command `node app` being executed, resulting in the output `5` and `4`.

```
JS math.js > ...
1 function add(a, b) {
2   return a + b;
3 }
4
5 function subtract(a, b) {
6   return a - b;
7 }
8
9 module.exports = { add, subtract };
10

JS app.js > ...
1 const { add, subtract } = require('./math');
2
3 console.log(add(2, 3));
4 console.log(subtract(5, 1));
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\sharm\nodejs> node app
5
4
PS C:\Users\sharm\nodejs>
```

2. File System (Blocking vs Non-Blocking)

- **Task:**

- Write two scripts—one for blocking file read and one for non-blocking file read.

Output:

```
JS Blocking.js X
JS Blocking.js > ...
1  const fs = require('fs');
2  const data = fs.readFileSync('test.txt', 'utf8');
3  console.log(data);
4  console.log('Read Complete');
5

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

• PS C:\Users\sharm\nodejs> node Blocking
hello world
Read Complete
❖ PS C:\Users\sharm\nodejs> 
```

```
JS Non-Blocking.js X
JS Non-Blocking.js > ...
1  const fs = require('fs');
2  fs.readFile('test.txt', 'utf8', (err, data) => {
3    if (err) throw err;
4    console.log(data);
5  });
6  console.log('Read Initiated');
7

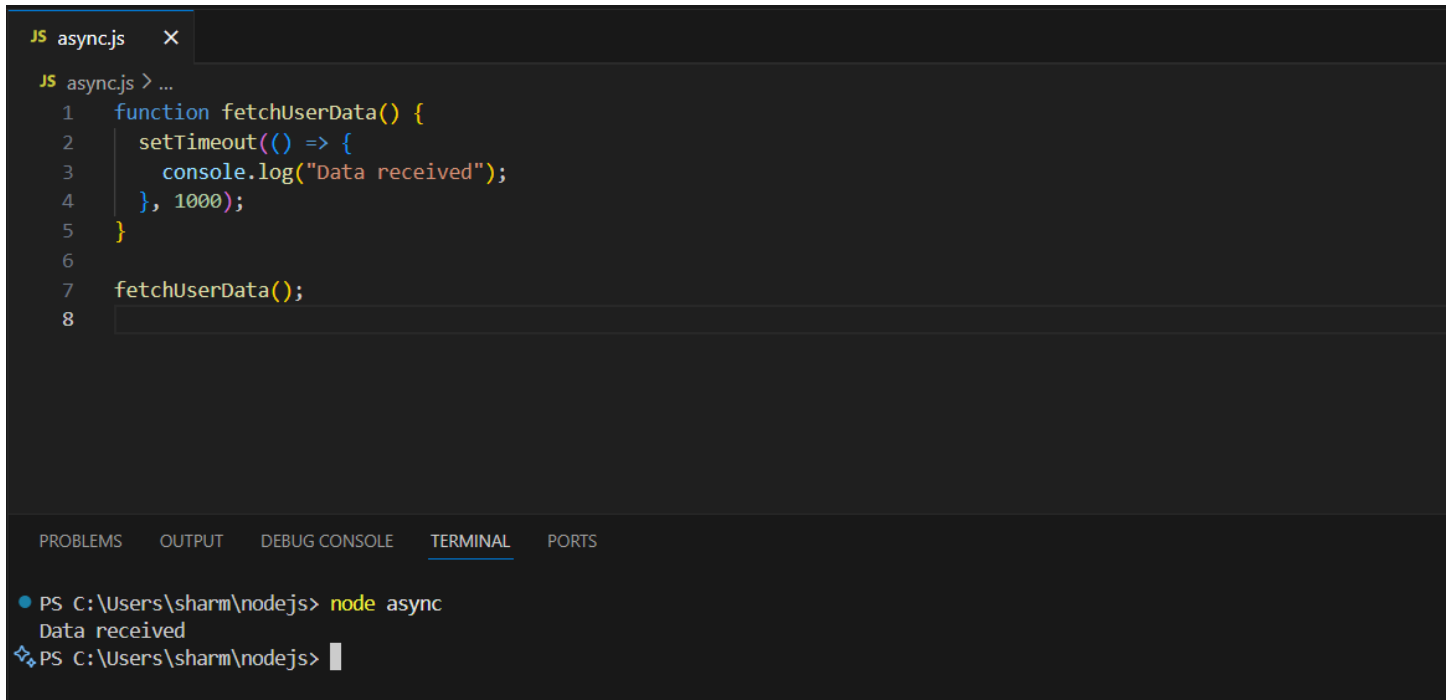
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

• PS C:\Users\sharm\nodejs> node Non-Blocking
Read Initiated
hello world
❖ PS C:\Users\sharm\nodejs> 
```

3. Asynchronous Programming

- **Task:**
 - Write a function that fetches user data (simulated with `setTimeout`) and logs "Data received".

Output:



The screenshot shows a VS Code editor with a file named `async.js`. The code defines a function `fetchUserData()` that uses `setTimeout` to simulate an asynchronous operation. The function logs "Data received" after a 1000ms delay. The function is then called. The terminal output shows the command `node async` being executed, followed by the output "Data received".

```
JS async.js X
JS async.js > ...
1 function fetchUserData() {
2   setTimeout(() => {
3     console.log("Data received");
4   }, 1000);
5 }
6
7 fetchUserData();
8

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS C:\Users\sharm\nodejs> node async
Data received
❖ PS C:\Users\sharm\nodejs>
```