Lesson 04 Demo 04

Blocking and Non-Blocking in Node.js

Objective: To read the file content synchronously and asynchronously using blocking and non-blocking methods

Tools required: Visual Studio Code and Node Package Manager

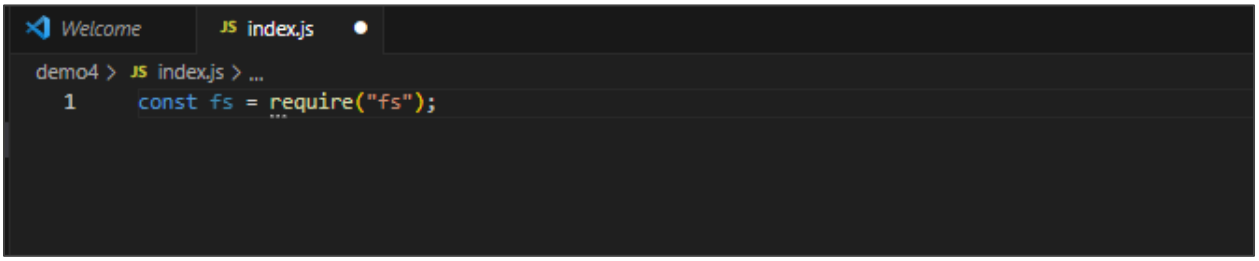
Prerequisites: Basic Linux and NPM commands

Steps to be followed:

1. Read the file content synchronously using the blocking method
2. Read the file content asynchronously using the non-blocking method

Step 1: Read the file content synchronously using the blocking method

1.1 Create a file named **index.js** inside the project directory and import the **node fs** module using the following code:
const fs = require("fs");



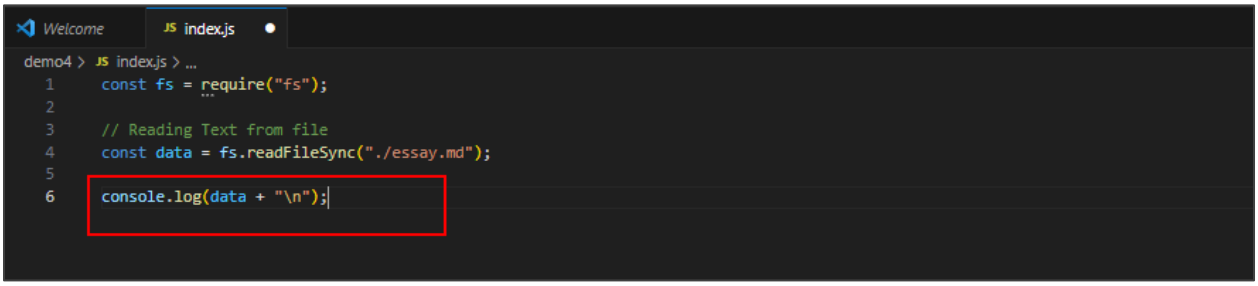
1.2 Use the **readFileSync** function to read the **essay.md** file synchronously:

```
// Reading Text from file
const data = fs.readFileSync("./essay.md");
```

> **Note:** Here the **essay.md** file is already created.

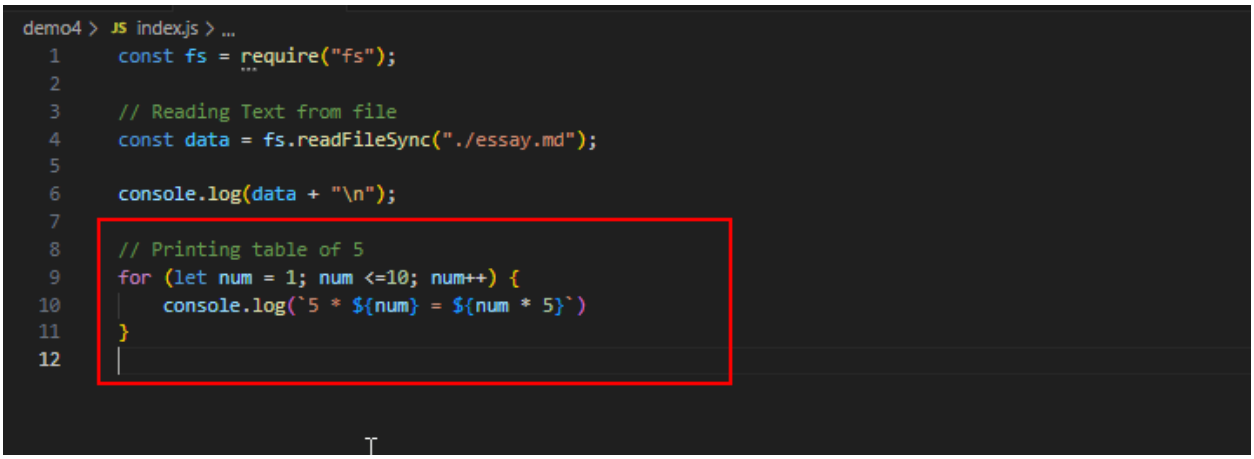
1.3 Use the following code to print the file content on the console:

```
console.log(data + "\n");
```



1.4 Use the following code to print the table of 5:

```
// Printing table of 5
for (let num = 1; num <=10; num++) {
   console.log(`5 * ${num} = ${num * 5}`)
}
```



Note: This code will not be executed until the above file read operation finishes. Therefore, the above file read is blocking the code to print the table of 5.

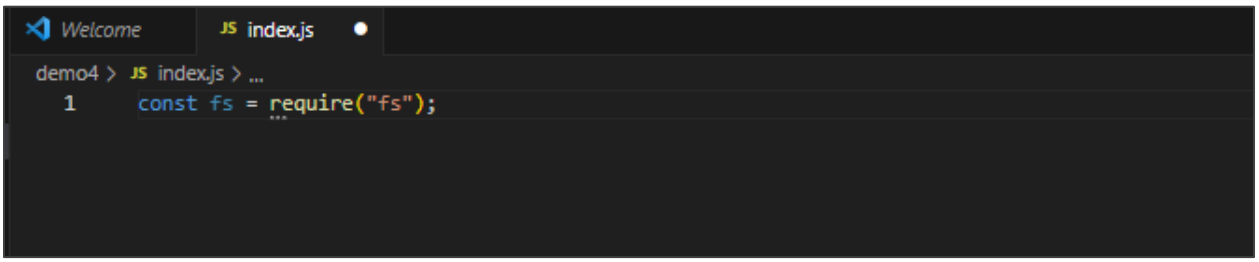
1.5 Execute the **index.js** using the following command:
node index.js

```
demopython1yopm@ip-172-31-16-204:~/Desktop/nodeProjec/demo4$ node index.js
Mechanical Engineering Student Sample
In the first sample essay from mechanical engineering, what stands out immediate
ly are the length and the photographs. In this case, the student was applying fo
r an engineering scholarship, so he was given room to flesh out technical materi
al as well as address issues such as personal motivations one would expect to re
ad in a personal statement.

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Step 2: Read the file content asynchronously using the non-blocking method

2.1 Import the **node fs** module using the following code:
const fs = require("fs");

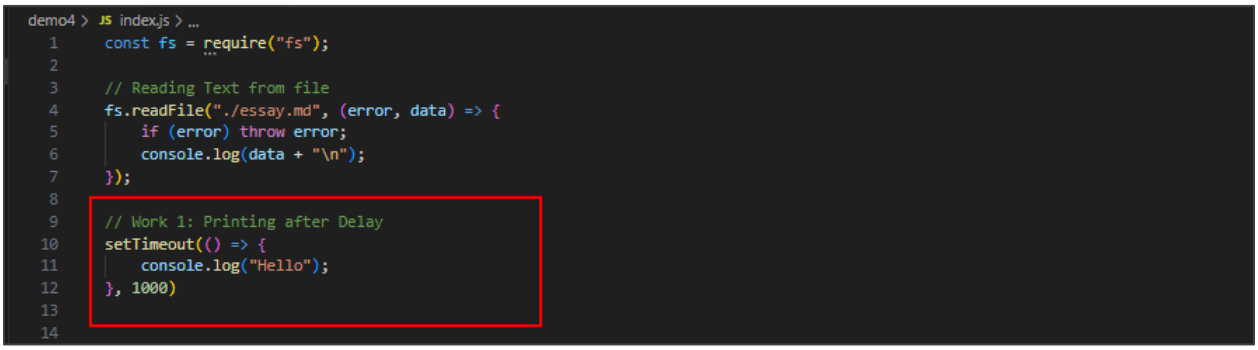


2.2 Use the **readFile** function to read the **essay.md** file asynchronously:

```
// Reading Text from file
fs.readFile("./essay.md", (error, data) => {
   if (error) throw error;
   console.log(data + "\n");
});
```

2.3 Write the following code to demonstrate the non-blocking method:

```
// Work 1: Printing after Delay
setTimeout(() => {
   console.log("Hello");
}, 1000)
```



Note: Since this code is executed without getting blocked, the above file is read using the non-blocking code method.

2.4 Use this code to print a table of number 5:

```
// Work 2: Printing table of 5
for (let num = 1; num <=10; num++) {
   console.log(`5 * ${num} = ${num * 5}`)
}
```

```
demo4 > JS index.js > ...
1   const fs = require("fs");
2
3   // Reading Text from file
4   fs.readFile("./essay.md", (error, data) => {
5       if (error) throw error;
6       console.log(data + "\n");
7   });
8
9   // Work 1: Printing after Delay
10  setTimeout(() => {
11      console.log("Hello");
12  }, 1000)
13
14  // Work 2: Printing table of 5
15  for (let num = 1; num <=10; num++) {
16      console.log(`5 * ${num} = ${num * 5}`)
17  }
```

2.5 Execute the **index.js** using the following command:

node index.js

```
demopython1yopm@ip-172-31-16-204:~/Desktop/nodeProjec/demo4$ node index.js
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
Mechanical Engineering Student Sample
In the first sample essay from mechanical engineering, what stands out immediate
ly are the length and the photographs. In this case, the student was applying fo
r an engineering scholarship, so he was given room to flesh out technical materi
al as well as address issues such as personal motivations one would expect to re
ad in a personal statement.

Hello
```

By following these steps, you have successfully read the file content synchronously and asynchronously using blocking and non-blocking methods, respectively.