Practical 3

Part 1: Modules

Task:

Create a file math.js exporting add and subtract functions.
Import them into app.js and use them.

Code Of Match.js:

Practical 3 > 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 = {
10      add,
11      subtract
12  };
13
```

Code of App.js:

Practical 3 > JS app.js > ...

```
1   const { add, subtract } = require('Practical 3/Math.js');
2
3   const sum = add(10, 5);
4   const difference = subtract(10, 5);
5
6   console.log(`The sum of 10 and 5 is: ${sum}`);
7   console.log(`The difference of 10 and 5 is: ${difference}`);
```

Output:

```
The sum of 10 and 5 is: 15
The difference of 10 and 5 is: 5
```

Part 2:

File System (Blocking vs non-blocking)

Task:

Write two scripts:

Blocking file read

Non-blocking file read

1.Blocking File System:

Practical 4 > JS Blocking_fileread.js > ...

```
// Import the file system module.
const fs = require('fs');

// Path to the file we will read.
const filePath = 'temp.txt';

// Log a message to show the script has started.
console.log('Starting the blocking file read...');

try {
  // This is a blocking (synchronous) file read.
  // The script will pause here until the entire file has been read.
  const data = fs.readFileSync(filePath, 'utf8');
  console.log('File content (Blocking):');
  console.log(data);
} catch (err) {
  // Handle any errors that occur during the file read.
  console.error('An error occurred:', err.message);
}

// This message will only be logged after the file read is complete.
console.log('...Finished the blocking file read.');
```

2.Non-Blocking File System:

Practical 4 > JS Blocking_fileread.js > ...

```js
// Import the file system module.
const fs = require('fs');

// Path to the file we will read.
const filePath = 'temp.txt';

// Log a message to show the script has started.
console.log('Starting the blocking file read...');

try {
  // This is a blocking (synchronous) file read.
  // The script will pause here until the entire file has been read.
  const data = fs.readFileSync(filePath, 'utf8');
  console.log('File content (Blocking):');
  console.log(data);
} catch (err) {
  // Handle any errors that occur during the file read.
  console.error('An error occurred:', err.message);
}

// This message will only be logged after the file read is complete.
console.log('...Finished the blocking file read.');
```

Output:

```
Starting the blocking file read...
File content (Blocking):
Hello, this is a temporary file.
```

```
Starting the non-blocking file read...
...This message appears before the file
File content (Non-Blocking):
Hello, this is a temporary file.
```

Part 3: Asynchronous Programming

Task:

Write a function that fetches user data (simulate with setTimeout) and logs "Data received".

Code:

Practical 5 > JS Asynchronous_Data)_Fetch.js > ...

```
// A function to simulate fetching user data from an API.
// It is an asynchronous operation.
function fetchUserData() {
    console.log('Fetching user data...');

    // Use setTimeout to simulate a network request that takes time.
    // This is a non-blocking operation. The program will continue to execute
    // the lines below this function call while the timer is running.
    setTimeout(() => {
        console.log('Data received.');
        // In a real application, you would handle the fetched data here.
    }, 2000); // Simulates a 2-second delay
}

// Call the function to start the asynchronous operation.
fetchUserData();

// This message will be logged immediately, showing that the script does not wait
// for the fetchUserData function to complete.
console.log('Continuing with other tasks...');
```

Output:

```
Fetching user data...
Continuing with other tasks...
Data received.
```