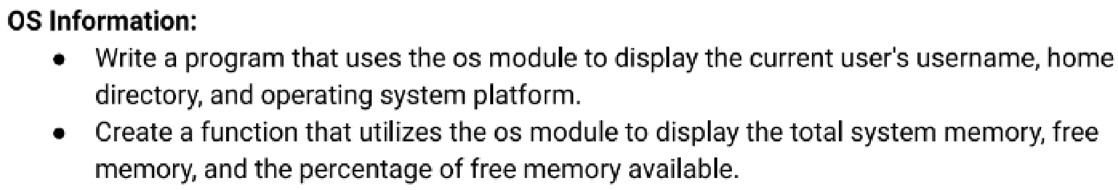
Task-1

Aim:



OS Information:

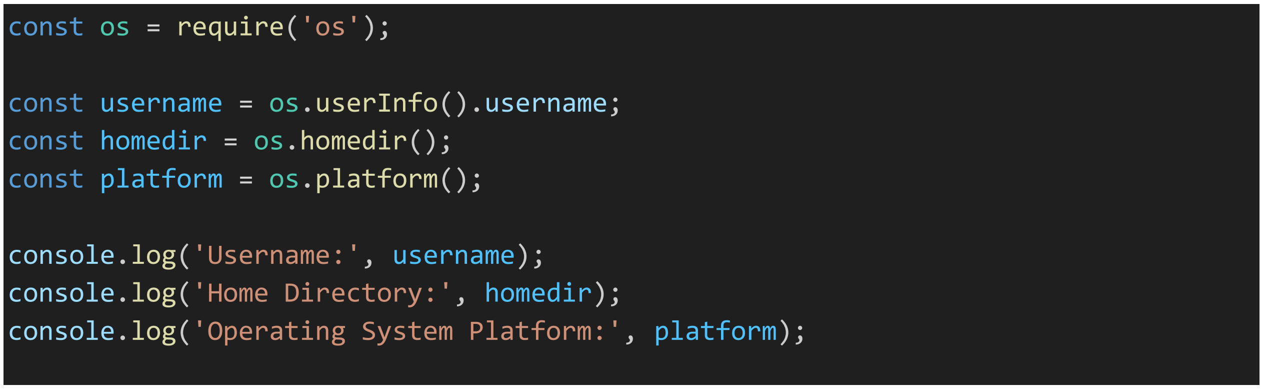
- Write a program that uses the os module to display the current user's username, home directory, and operating system platform.
- Create a function that utilizes the os module to display the total system memory, free memory, and the percentage of free memory available.

Description:

To write a program that displays the current user's username, home directory, and operating system platform using the os module, you can follow Import the os module,Retrieve the current user's username etc.

● The OS module in Node.js provides a set of utility methods for interacting with the operating system. It allows you to retrieve information about the system, such as network interfaces, CPU architecture, memory usage, and more. Here are some commonly used functionalities of the os module:

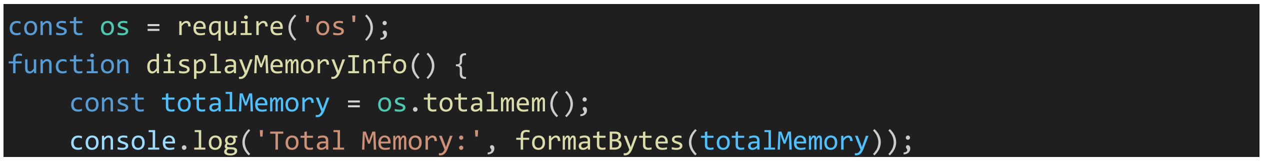
Source Code:



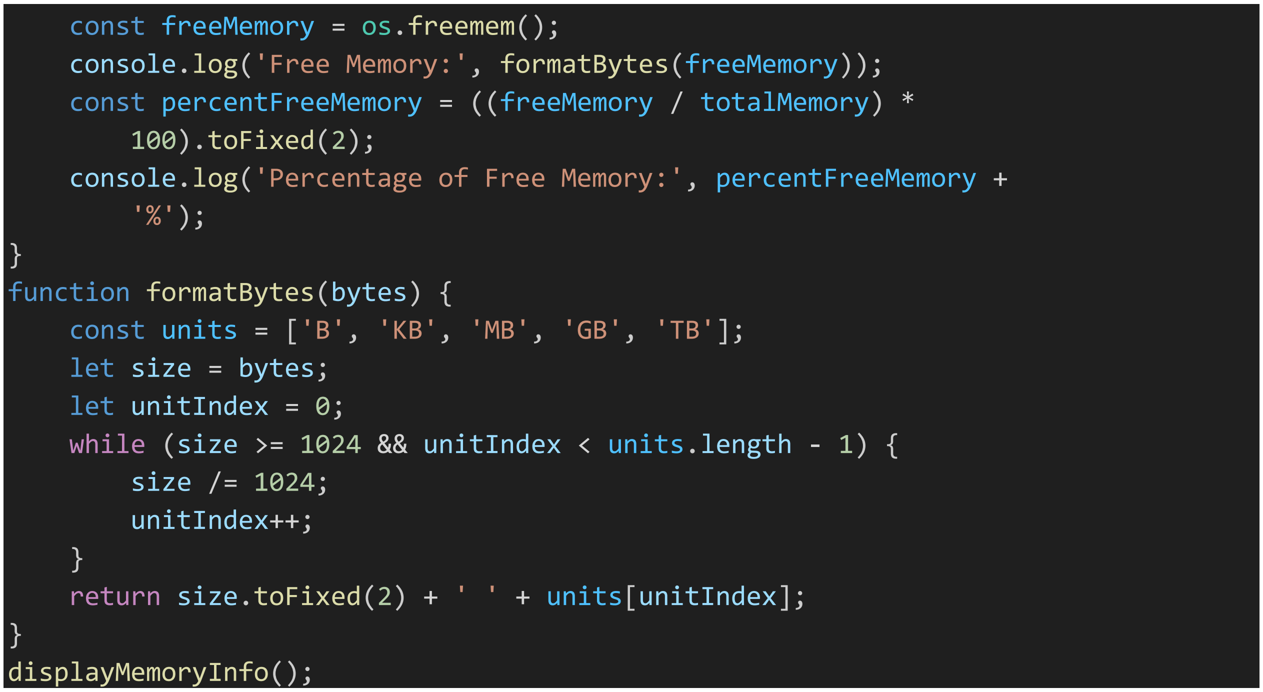
```
const os = require('os');

const username = os.userInfo().username;
const homedir = os.homedir();
const platform = os.platform();

console.log('Username:', username);
console.log('Home Directory:', homedir);
console.log('Operating System Platform:', platform);
```

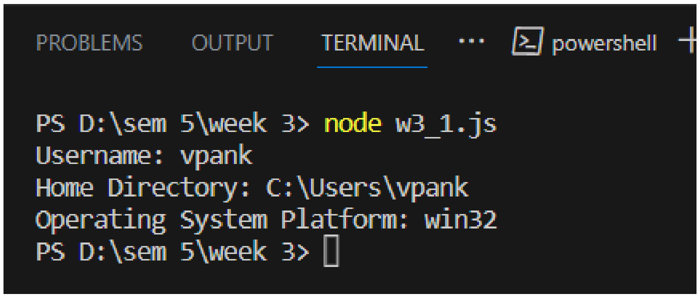


```
const os = require('os');
function displayMemoryInfo() {
    const totalMemory = os.totalmem();
    console.log('Total Memory:', formatBytes(totalMemory));
```

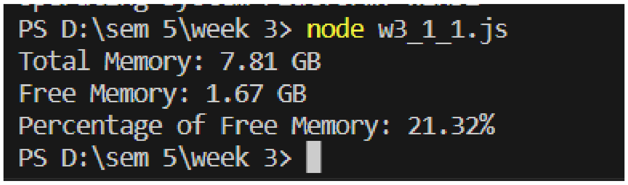


```
    const freeMemory = os.freemem();
    console.log('Free Memory:', formatBytes(freeMemory));
    const percentFreeMemory = ((freeMemory / totalMemory) *
        100).toFixed(2);
    console.log('Percentage of Free Memory:', percentFreeMemory +
        '%');
}
function formatBytes(bytes) {
    const units = ['B', 'KB', 'MB', 'GB', 'TB'];
    let size = bytes;
    let unitIndex = 0;
    while (size >= 1024 && unitIndex < units.length - 1) {
        size /= 1024;
        unitIndex++;
    }
    return size.toFixed(2) + ' ' + units[unitIndex];
}
displayMemoryInfo();
```

Output:



```
PS D:\sem 5\week 3> node w3_1.js
Username: vpank
Home Directory: C:\Users\vpank
Operating System Platform: win32
PS D:\sem 5\week 3> 
```



```
PS D:\sem 5\week 3> node w3_1_1.js
Total Memory: 7.81 GB
Free Memory: 1.67 GB
Percentage of Free Memory: 21.32%
PS D:\sem 5\week 3> 
```

Task-2

Aim:

Experiment with chalk,upper-case any other External Modules

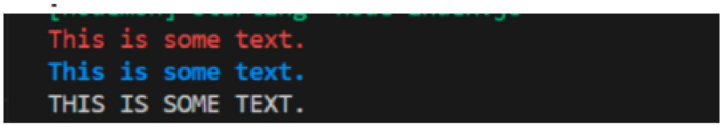
Description:

- Use chalk to style text: chalk is a module that allows you to add colors and styles to text in the console. You can apply various styles like bold, underline, and color to your text using chalk functions.

Source Code:

```
const chalk = require('chalk');
const text = 'This is some text.';
console.log(chalk.red(text));
console.log(chalk.blue.bold(text));
const upperCaseText = text.toUpperCase();
console.log(upperCaseText);
```

Output:



```
This is some text.
This is some text.
THIS IS SOME TEXT.
```

Learning Outcome:

Overall, this program helps you understand how to utilize the os module to access and display relevant user information and operating system details, expanding your knowledge of working with node js and interacting with the underlying system.

Task-3

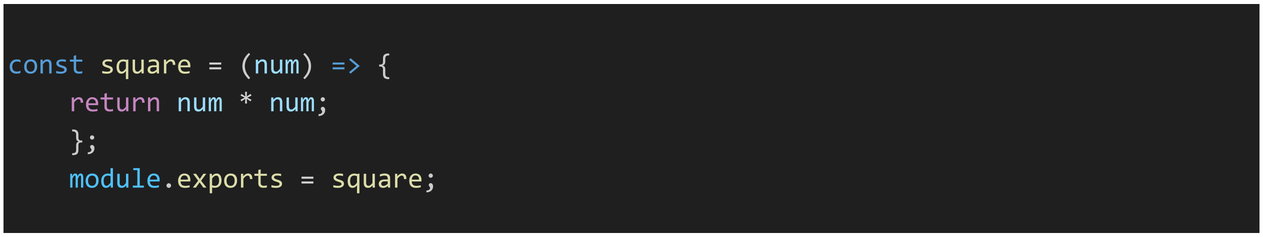
Aim:

Create your own custom module and import/export it to the main module.

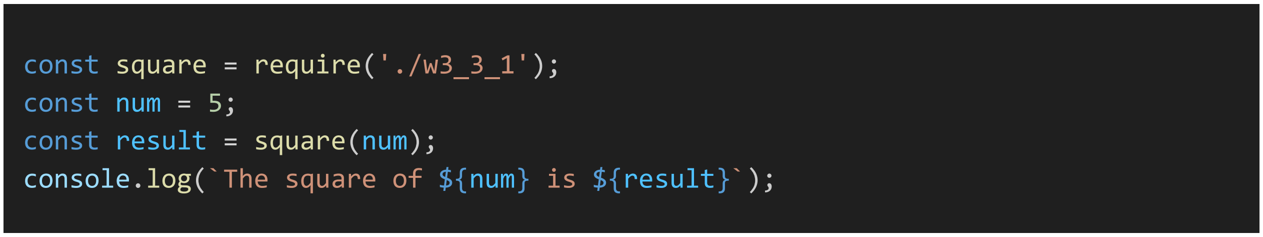
Description:

- Create a new JavaScript file Start by creating a new JavaScript file that will serve as your custom module. For example, create a file named myModule.js.
- Define functionality in the custom module , Export the functionality Use the imported functionality and Once imported, you can access the exported functionality from your custom module using the variable or object you assigned it to during the import.

Source Code:

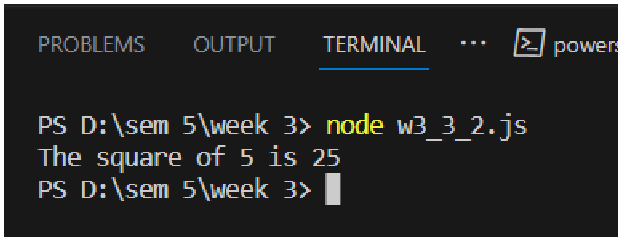


```
const square = (num) => {
    return num * num;
    };
    module.exports = square;
```



```
const square = require('./w3_3_1');
const num = 5;
const result = square(num);
console.log(`The square of ${num} is ${result}`);
```

Output:



```
PS D:\sem 5\week 3> node w3_3_2.js
The square of 5 is 25
PS D:\sem 5\week 3> █
```

Learning Outcome:

By following these steps, you can create your own custom module in Node.js and import/export it to the main module or any other module as needed, enabling modular code organization and reuse.