Week-3(Node.js)

Task-1

1. **Aim :** Write a program that uses the os modules to display the current user's username, home directory, and operating system platform.

Source Code:

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

const username = os.userInfo().username;

const homeDirectory = os.homedir();

const platform = os.platform();

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

Output:

```
PS C:\STUDY\Sem 5\FSWD\week3> node task1.js
Username: rd
Home Directory: C:\Users\rd
Platform: win32
PS C:\STUDY\Sem 5\FSWD\week3>
```

2. **Aim :** Create a function that utilizes the os module to display the total system memory, free memory, and the percentage of free memory available

Source Code:

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

function getSystemMemoryInfo() {
  const totalMemoryBytes = os.totalmem();
  const freeMemoryBytes = os.freemem();
  const percentageFree = (freeMemoryBytes / totalMemoryBytes) * 100;

  const totalMemoryGB = (totalMemoryBytes / 1024 / 1024 /
1024).toFixed(2);
  const freeMemoryGB = (freeMemoryBytes / 1024 / 1024 / 1024).toFixed(2);
  const percentageFreeFormatted = percentageFree.toFixed(2);

  console.log('Total System Memory:', totalMemoryGB, 'GB');
  console.log('Free Memory:', freeMemoryGB, 'GB');
  console.log('Percentage of Free Memory:', percentageFreeFormatted, '%');
}

// Call the function to display system memory information
getSystemMemoryInfo();
```

Output:

```
PS C:\STUDY\Sem 5\FSWD\week3> node task1.1.js
Total System Memory: 7.36 GB
Free Memory: 0.76 GB
Percentage of Free Memory: 10.31 %
PS C:\STUDY\Sem 5\FSWD\week3>
```

Theoretical Background:

Since OS modules in Node.js include functions and attributes to access data about the operating system that your Node.js application is executing on, we use them. It enables you to acquire system data including memory, CPU, and network specifics. Additionally, it aids in managing resources, enhancing performance, and making choices depending on the resources of the system at hand. The os module eliminates operating system variations so that your code runs uniformly on all systems. It also offers user-specific data and features for engaging with the operating system. Overall, the os module is helpful for Node.js applications that need to access operating system-specific data, programme at the system level, or create cross-platform.

Task-2

Aim :
Experiment with chalk,upper-case and any other External Module

Source Code:

```
// Importing external modules
const chalk = require('chalk');
const upperCase = require('upper-case');
const moment = require('moment');

// Chalk usage
console.log(chalk.green('Hello, this text is in green!'));
console.log(chalk.blue.bold('This text is in bold blue!'));
console.log(chalk.red.inverse('This text has a red background!'));

// upper-case usage
const text = 'uppercase text';
console.log('Uppercase:', upperCase(text));

// moment usage
const now = moment();
console.log('Current Date and Time:', now.format('YYYY-MM-DD HH:mm:ss'));
```

Output:

```
Hello, this text is in green!
This text is in bold blue!
This text has a red background!
Uppercase: UPPERCASE TEXT
Current Date and Time: 2023-07-17 14:30:45
```

Theoretical Background:

1 - Importing external modules:

- The code starts by importing three external modules using the require function:

2 - chalk:

- A popular module that enables terminal text styling with colors and text formatting options.

3 - upper-case:

- A module that provides a utility function to convert a given text to uppercase.

4- moment:

- A library for parsing, validating, manipulating, and formatting dates and times.

5 - Chalk usage:

- The code showcases how to use chalk to style terminal output with different colors and text formatting options:
- chalk.green: Changes the color of the text to green.
- chalk.blue.bold: Makes the text bold and changes the color to blue.
- chalk.red.inverse: Sets a red background for the text.

6 - upper-case usage:

- The code demonstrates how to use the upper-case module to convert a given text to uppercase. It takes the input string "uppercase text" and converts it to "UPPERCASE TEXT".

7 - moment usage:

- The code uses the moment library to work with dates and times:

8 - moment():

- Creates a new moment object representing the current date and time.

now.format('YYYY-MM-DD HH:mm:ss'): Formats the moment object to a specific date and time format (year-month-day hour:minute:second).

Task-3

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

Source Code:

mathOperations.js

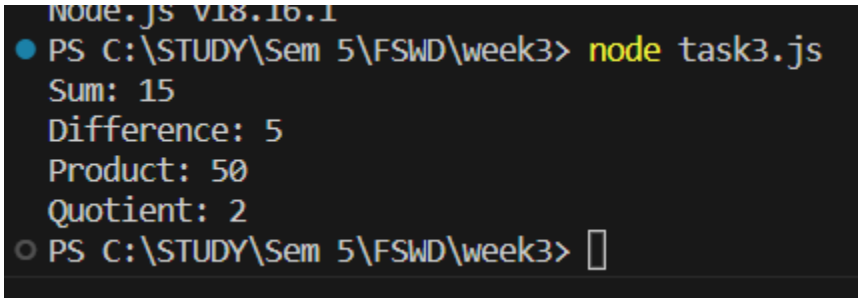
```
function add(a, b) {
    return a + b;
  }

  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,
  };
```

Now we will import mathOperations.js in task3.js

task3.js

```
// app.js

// Import the custom mathOperations module
const mathOperations = require('./mathOperations');

// Use the functions from the custom module
const num1 = 10;
const num2 = 5;

console.log(`Sum: ${mathOperations.add(num1, num2)}`);
console.log(`Difference: ${mathOperations.subtract(num1, num2)}`);
console.log(`Product: ${mathOperations.multiply(num1, num2)}`);

try {
  console.log(`Quotient: ${mathOperations.divide(num1, num2)}`);
} catch (error) {
  console.log(`Error: ${error.message}`);
}
```

Output :



Theoretical Background :

1 - Custom Module (mathOperations.js):

- The code defines a custom module named mathOperations. This module contains four functions for basic arithmetic operations:

- add(a, b): Returns the sum of two numbers a and b.
- subtract(a, b): Returns the difference between two numbers a and b.
- multiply(a, b): Returns the product of two numbers a and b.
- divide(a, b): Returns the quotient of dividing a by b, but throws an error if the divisor (b) is zero.

2 - Main Module (app.js):

- The code imports the custom mathOperations module into the main module using the require function.

- const mathOperations = require('./mathOperations'): Imports the custom module from the file path ./mathOperations.

3 - Using the Custom Module:

- The main module (app.js) demonstrates how to use the functions from the custom module:
- It defines two variables num1 and num2 with values 10 and 5, respectively.
- It then calls the various functions from the custom mathOperations module with num1 and num2 as arguments and prints the results.

Learning Outcome :

CO1 : Understand various technologies and trends impacting single page web applications.
CO4 : Demonstrate the use of JavaScript to fulfill the essentials of front-end development To back-end development