**Practical-3**

**Aim:** Write a program that uses the OS module to display the current user’s username, home directory, and operating system platform.

**Source Code:**

1. Write a program that uses the OS module to display the current user’s username, home directory, and operating system platform.

const os = require('os');

const username = os.userInfo().username;

const homeDirectory = os.homedir();

const platform = os.platform();

console.log(`Username: ${username}`);

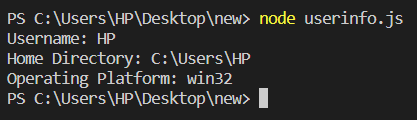
console.log(`Home Directory: ${homeDirectory}`);

console.log(`Operating Platform: ${platform}`);

**Description:**

* We begin by importing the os module, which is a built-in module in Node.js. This module provides a set of utilities related to operating system information.
* Next, we use the os.userInfo() method to retrieve information about the current user. The method returns an object containing various user-related properties, such as username, uid, gid, shell, and others.
* We extract the username property from the object obtained in the previous step. This gives us the name of the currently logged-in user on the operating system.
* Similarly, we use the os.homedir() method, which returns the home directory path for the current user.
* Lastly, we use the os.platform() method to obtain a string indicating the operating system platform (e.g., 'win32', 'Linux', 'darwin', etc.).
* We then print out the retrieved information using console.log(), providing a meaningful description for each data point. The output will display the current user's username, home directory path, and the operating system platform.

**Output:**

****

**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 displayMemoryInfo() {

const totalMemory = os.totalmem();

const freeMemory = os.freemem();

const percentageFree = (freeMemory / totalMemory) \* 100;

console.log(`Total System Memory: ${formatBytes(totalMemory)}`);

console.log(`Free Memory: ${formatBytes(freeMemory)}`);

console.log(`Percentage of Free Memory: ${percentageFree.toFixed(2)}%`);

}

// Helper function to format bytes to a human-readable string

function formatBytes(bytes) {

const units = ['B', 'KB', 'MB', 'GB', 'TB'];

let index = 0;

while (bytes >= 1024 && index < units.length - 1) {

bytes /= 1024;

index++;

}

return `${bytes.toFixed(2)} ${units[index]}`;

}

// Call the function to display memory information

displayMemoryInfo();

**Description:**

* We start by importing the os module, which is a built-in module in Node.js. This module provides a set of utilities related to operating system information, including memory-related data.
* Next, we define a function called displayMemoryInfo(), which is responsible for displaying the total system memory, free memory, and the percentage of free memory available.
* Inside the displayMemoryInfo() function:

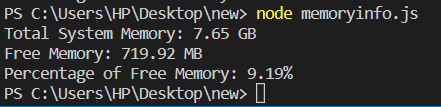
We use os.totalmem() to get the total system memory in bytes. This function returns the total amount of memory installed on the system.

We use os.freemem() to get the amount of free memory in bytes. This function returns the free memory available on the system.

We calculate the percentage of free memory available by dividing the free memory by the total memory and then multiplying by 100.

* We define a helper function called formatBytes(bytes). This function takes a number in bytes as input and converts it into a human-readable string with appropriate units (e.g., KB, MB, GB, etc.). This function makes the memory information more user-friendly.
* We call the displayMemoryInfo() function to show the memory information. When executed, the function will fetch the total system memory, free memory, and percentage of free memory, and print them using console.log().
* To run the program, you save the code in a file, for example, memoryInfo.js, and execute it with Node.js (node memoryInfo.js). The output will display the memory information in a human-readable format.

**Output:**

****

**Aim:** Experiment with chalk, upper-case any other External Modules

**Source Code:**

const chalk = require('chalk');

const upperCase = require('upper-case');

// Colorful text using chalk

console.log(chalk.bold.yellow('Experimenting with Chalk and Upper-Case Modules!'));

console.log(chalk.green('Success is ') + chalk.blue('colorful') + chalk.red('!'));

// Upper-case text using upper-case module

const text = 'hello, world!';

console.log('Original Text:', text);

console.log('Upper-case Text:', upperCase(text));

**Description:**

* We start by importing the required modules using require. In this case, we import chalk and upper-case modules.
* We use chalk to print colorful and styled text to the console. In this example, we use the chalk.bold, chalk.yellow, chalk.green, chalk.blue, and chalk.red methods to format the text in different colors and styles. The chalk.bold method makes the text bold, and the others set the text color to yellow, green, blue, and red, respectively.
* Next, we demonstrate the use of the upper-case module. We define a sample text, "hello, world!" and use the upper-case module to convert it to uppercase letters.
* We then print both the original text and the uppercase text to the console using console.log().
* When the code is executed using Node.js, it displays the output in the terminal. The colorful and styled text provided by chalk is displayed, and the original text is shown alongside its uppercase version generated using the upper-case module.

**Output:**

****

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

**Source Code:**

1. main.js

// Import the custom module we created

const calculateSquare = require('./customModule');

// Use the imported function to calculate the square of a number

const num = 5;

const result = calculateSquare(num);

console.log(`The square of ${num} is ${result}.`);

1. customModule.js

// A simple function to calculate the square of a number

function calculateSquare(number) {

return number \* number;

}

// Export the function to make it available to other modules

module.exports = calculateSquare;

**Description:**

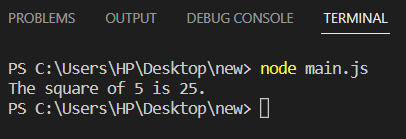
1. Custom Module (customModule.js):

* In the customModule.js file, we have defined a custom module named customModule.
* The module contains a single function called calculateSquare(number), which takes a number as input and calculates the square of that number.
* The calculateSquare function multiplies the input number by itself and returns the result, which is the square of the input number.
* To make the calculateSquare function available to other modules, we use the module.exports statement. This statement allows us to export the function from this module.

1. Main Module (main.js):

* In the main.js file, we are the main module that will use the custom module customModule.js.
* To use the functionality of the customModule, we import it using require('./customModule'). The require function allows us to load the exported functionality from the customModule.js file.
* After importing the calculateSquare function from the custom module, we use it to calculate the square of the number 5 and store the result in the result variable.
* Finally, we print the result to the console using console.log().

**Output:**

****

**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