

Name of Student: Pushkar Sane		
Roll Number: 45		Lab Assignment Number: 2
Title of Lab Assignment: Create an application to demonstrate Node.js modules.		
DOP: 12-09-2023		DOS: 13-09-2023
CO Mapped: CO1	PO Mapped: PO3, PO5, PSO1, PSO2	Faculty Signature:

Practical No. 2

Aim:

Create an application to demonstrate the Node.js modules.

1) Built-in modules

1. Write a program to print information about the computer's operating system using the OS module (Use any 5 methods).
2. Print "Hello" every 500 milliseconds using the timer module. The message should be printed exactly 10 times. Use setInterval, clearInterval and setTimeout methods.

2) Custom Modules

- a) Create a Calculator Node.js Module with functions add, subtract and multiply, Divide. And use the Calculator module in another Node.js file.
- b) Create a circle module with functions to find the area and perimeter of a circle and use it.

Theory:

1. Built-in Module:

In Node.js, built-in modules are pre-existing libraries and modules that are included with the Node.js runtime environment. These modules provide a wide range of functionalities to help developers perform common tasks and interact with various aspects of the system. Below are some of the examples of built in modules.

- a) **fs (File System)**: Used for reading and writing files, as well as manipulating directories.
- b) **http and https**: Modules for creating HTTP and HTTPS servers and making HTTP requests.
- c) **os (Operating System)**: Provides information about the operating system.
- d) **path**: Helps in working with file and directory paths.
- e) **events**: Allows you to create and handle custom events.
- f) **crypto**: Offers cryptographic functionality for hashing, encryption, and decryption.
- g) **util**: Provides utility functions for debugging and formatting.

2. Custom Module:

Custom modules in Node.js are user-defined JavaScript files that encapsulate specific functionality, making code more organized, modular, and reusable.

- a) Organize code into smaller, manageable units.
- b) Promote code reusability across your application.
- c) Improve code readability and maintainability.
- d) Extend Node.js capabilities beyond built-in modules.

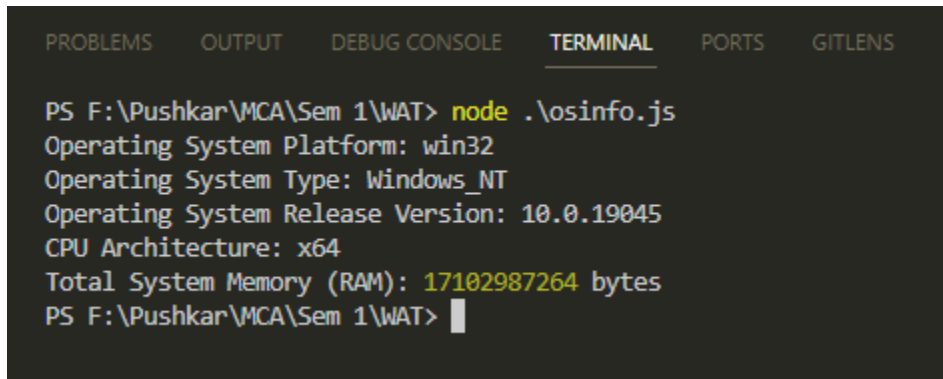
Custom modules are essential for building scalable and maintainable Node.js applications by facilitating code modularization and promoting best practices in software development.

- 1) Write a program to print information about the computer's operating system using the OS module (use any 5 methods).

Code:

```
const os = require('os');  
console.log('Operating System Platform:', os.platform());  
console.log('Operating System Type:', os.type());  
console.log('Operating System Release Version:', os.release());  
console.log('CPU Architecture:', os.arch());  
console.log('Total System Memory (RAM):', os.totalmem(), 'bytes');
```

Output:



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), 'PORTS', and 'GITLENS'. The terminal content shows a command prompt at 'PS F:\Pushkar\MCA\Sem 1\WAT>' followed by the command 'node .\osinfo.js'. The output of the script is displayed line by line: 'Operating System Platform: win32', 'Operating System Type: Windows_NT', 'Operating System Release Version: 10.0.19045', 'CPU Architecture: x64', and 'Total System Memory (RAM): 17102987264 bytes'. The prompt returns to 'PS F:\Pushkar\MCA\Sem 1\WAT>' with a cursor.

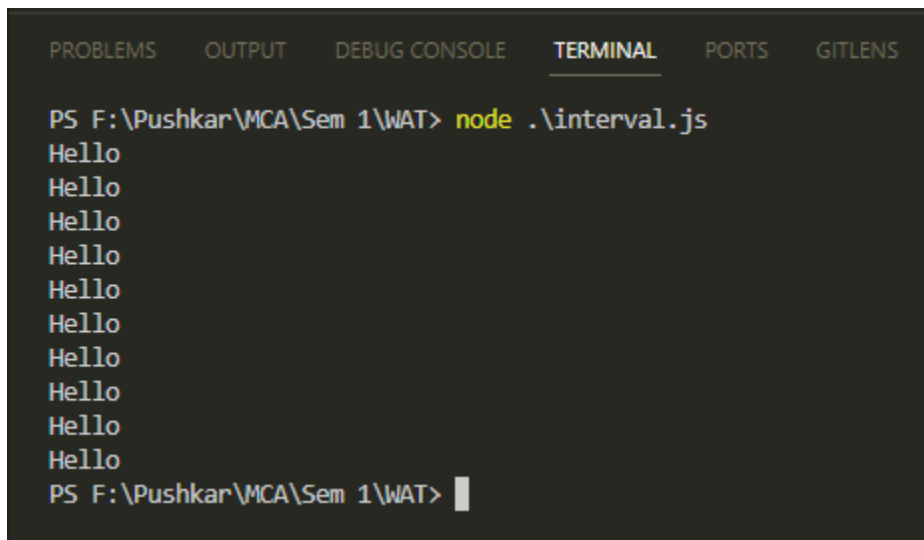
```
PS F:\Pushkar\MCA\Sem 1\WAT> node .\osinfo.js  
Operating System Platform: win32  
Operating System Type: Windows_NT  
Operating System Release Version: 10.0.19045  
CPU Architecture: x64  
Total System Memory (RAM): 17102987264 bytes  
PS F:\Pushkar\MCA\Sem 1\WAT> █
```

- 2) Print "Hello" every 500 milliseconds using the Timer Module. The message should be printed exactly 10 times. Use `setInterval`, `clearInterval` and `setTimeout` methods.

Code:

```
let count = 0;
const intervalId = setInterval(() => {
  console.log("Hello");
  count++;
  if(count == 10){
    clearInterval(intervalId);
  }
}, 500);
setTimeout(() => {
  clearInterval(intervalId);
}, 5500);
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS F:\Pushkar\MCA\Sem 1\WAT> node .\interval.js
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
PS F:\Pushkar\MCA\Sem 1\WAT> 
```

- 3) **Create a Calculator Node.js Module with functions add, subtract and multiply, Divide. And use the Calculator module in another Node.js file.**

Code:

Calculator.js

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

function subtract(a, b){
    return(a - b);
}

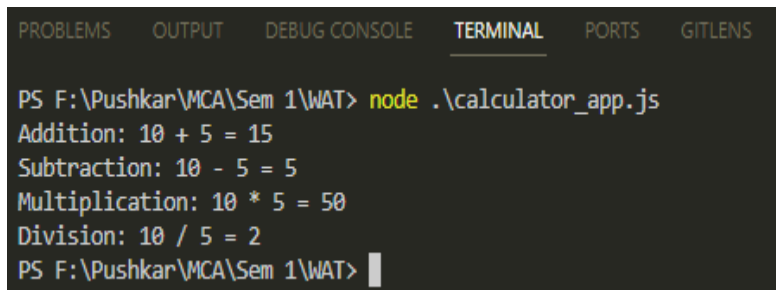
function multiply(a, b){
    return(a * b);
}

function divide(a, b){
    if(b === 0){
        throw new Error("Divide by zero error!")
    }
    return(a / b);
}

module.exports = {
    add,
    subtract,
    multiply,
    divide,
};
```

Calculator_App.js

```
const calculator = require('./calculator');  
const num1 = 10;  
const num2 = 5;  
console.log(`Addition: ${num1} + ${num2} = ${calculator.add(num1, num2)}`);  
console.log(`Subtraction: ${num1} - ${num2} = ${calculator.subtract(num1, num2)}`);  
console.log(`Multiplication: ${num1} * ${num2} = ${calculator.multiply(num1, num2)}`);  
console.log(`Division: ${num1} / ${num2} = ${calculator.divide(num1, num2)}`);
```

Output:

The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), 'PORTS', and 'GITLENS'. The terminal content shows the command 'node .\calculator_app.js' being executed in a PowerShell prompt. The output consists of four lines: 'Addition: 10 + 5 = 15', 'Subtraction: 10 - 5 = 5', 'Multiplication: 10 * 5 = 50', and 'Division: 10 / 5 = 2'. The prompt returns to 'PS F:\Pushkar\MCA\Sem 1\WAT>' with a cursor.

```
PS F:\Pushkar\MCA\Sem 1\WAT> node .\calculator_app.js  
Addition: 10 + 5 = 15  
Subtraction: 10 - 5 = 5  
Multiplication: 10 * 5 = 50  
Division: 10 / 5 = 2  
PS F:\Pushkar\MCA\Sem 1\WAT>
```

- 4) Create a circle module with functions to find the area and perimeter of a circle and use it.

Code:

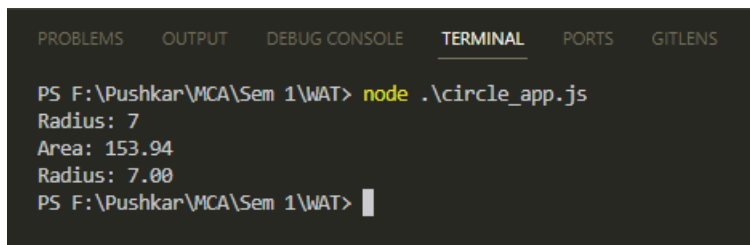
Circle.js

```
function CalculateArea(radius){
    return Math.PI * Math.pow(radius, 2);
}
function calculatePerimeter(radius){
    return 2 * Math.PI * radius;
}
module.exports = {
    CalculateArea,
    calculatePerimeter,
};
```

Circle_App.js

```
const circle = require('./circle');
const radius = 7;
const area = circle.CalculateArea(radius);
const perimeter = circle.calculatePerimeter(radius);
console.log(`Radius: ${radius}`);
console.log(`Area: ${area.toFixed(2)}`);
console.log(`Radius: ${radius.toFixed(2)}`);
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS F:\Pushkar\MCA\Sem 1\WAT> node .\circle_app.js
Radius: 7
Area: 153.94
Radius: 7.00
PS F:\Pushkar\MCA\Sem 1\WAT> 
```


Conclusion:

Created an application by using Node.js modules such as,

- Built-in Module
- Custom Module