Assignment: 2

Task 1 : Hoisting in Variables

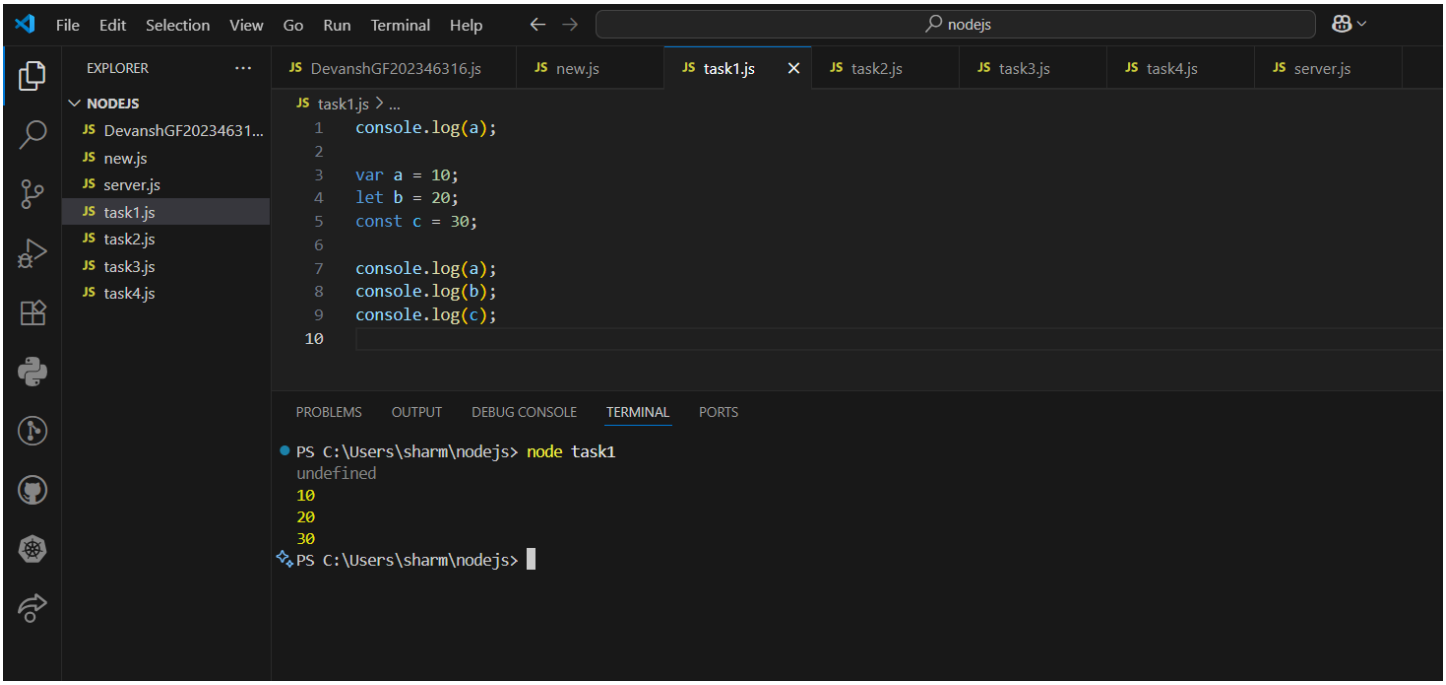
Write a Node.js program that demonstrates variable hoisting using var, let, and const.

Print a variable before it is declared

Show the difference between var, let, and const.

Explain the output.

OUTPUT:



Explanation:

- Here, **var** variables are set up early with a value of undefined, so using them before their line just gives undefined.
- Whereas, **let** and **const** are set up early too, but aren't usable until their line. If we use them before their line, it gives an error (“ReferenceError”).

Task2 : Function Declarations vs Expressions

Create two functions in Node.js:

A function declaration (function add(a,b) {})

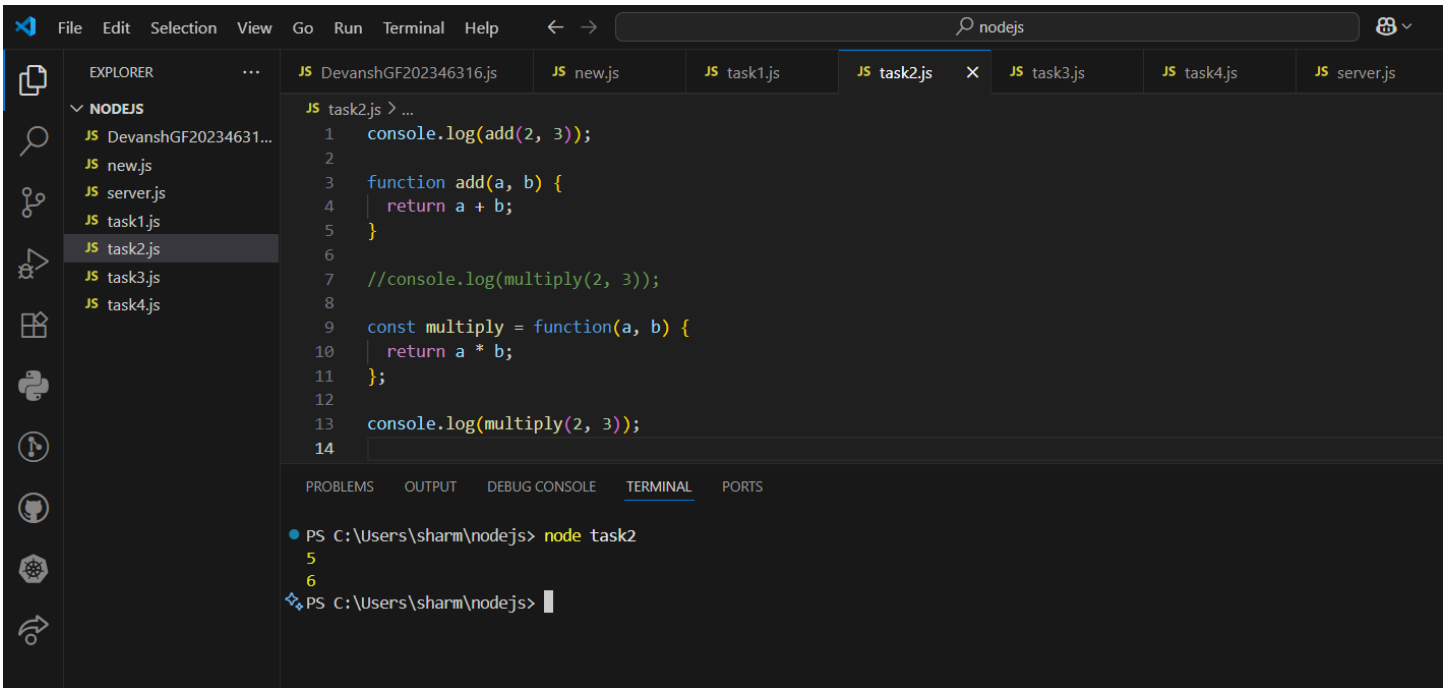
A function expression (const multiply = function(a,b) ())

Call both functions before and after their definitions.

Record what works and what fails.

Explain why.

OUTPUT :



- Calling add(2,3) before its declaration works, and prints 5.
- Calling multiply(2,3) before its assignment (if uncommented) fails, throwing a (“ReferenceError”).
- Calling multiply(2,3) after initialization works, printing 6.

<u>Explanation</u>:

- The function declaration add is hoisted along with its implementation, so it can be called before its definition in the code.
- The function expression multiply using a const variable is hoisted only as a variable declaration without initialization (in the temporal dead zone), so calling it before assignment results in a ReferenceError.

Task 3: Arrow Functions vs Normal Functions

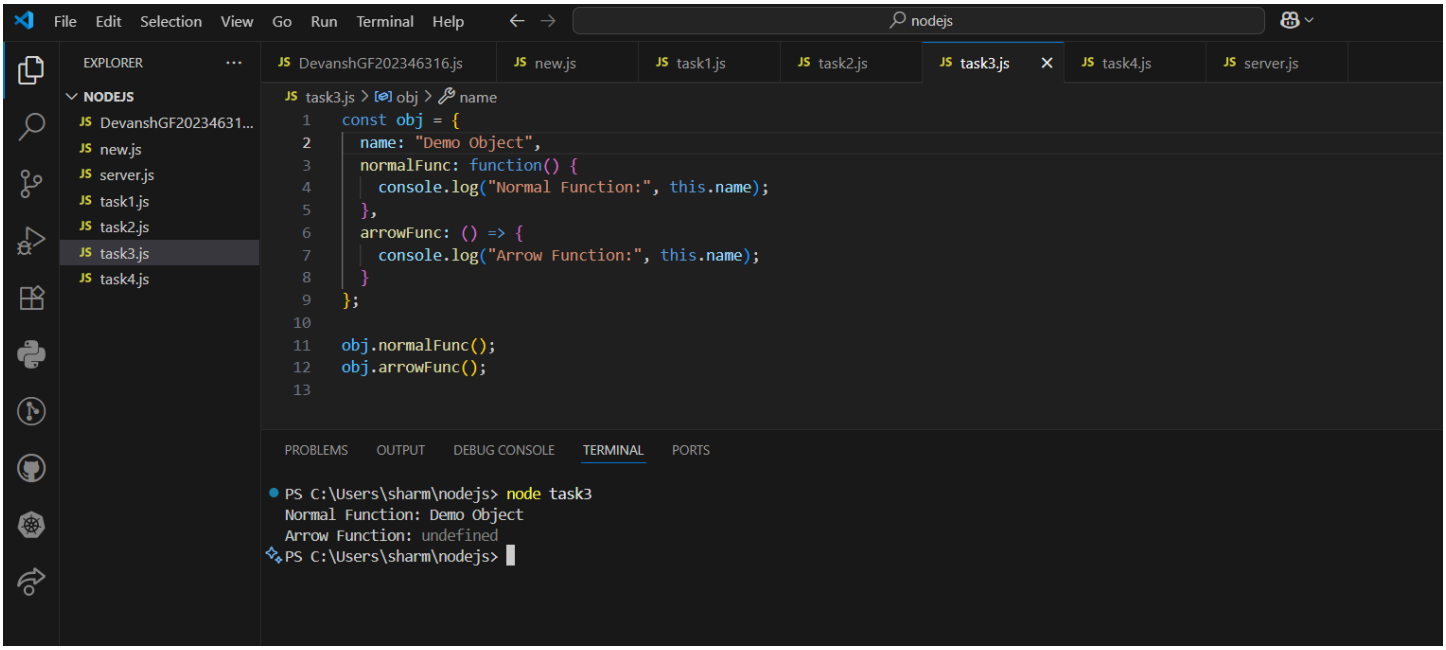
Create two functions inside an object

- One arrow function
- One normal function

Both should print this.

Compare their outputs when called as methods of the object.

OUTPUT:



<u>Comparison:</u>

Here,

1. obj.normalFunc(); prints: (Normal Function: Demo Object)

2. obj.arrowFunc(); prints: (Arrow Function: undefined)

<u>Explanation:</u>

- **Normal function** has its own this context, which is dynamically set to the object that called it (obj). Therefore, this.name correctly refers to "Demo Object".
- **Arrow function** does not have its own this context. Since this is the global or module scope (not the object), this.name is undefined.

Task 4: Higher Order Functions

Write a Node js function calculate(operation, a, b) where operation is another function (like add, subtract).

Pass different functions to calculate and print results

Example calculate((x,y) => x*y, 4, 5) should return 20

OUTPUT: