

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Node.js programs.
- Explorer:** Shows a tree view of "NODEJS PROGRAMS" with files: HoistingInVariables.js, SandeepGF202322853.js, and Server.js. HoistingInVariables.js is selected.
- Code Editor:** Displays the content of HoistingInVariables.js:

```
1 console.log(a);
2 var a = 10;
3
4 try {
5   console.log(b);
6   let b = 20;
7 } catch (err) {
8   console.log("Error with let:", err.message);
9 }
10
11 try {
12   console.log(c);
13   const c = 30;
14 } catch (err) {
15   console.log("Error with const:", err.message);
16 }
```
- Terminal:** Shows the command line output:

```
PS C:\Users\rajes\OneDrive\Node.js programs> node HoistingInVariables.js
undefined
Error with let: Cannot access 'b' before initialization
Error with const: Cannot access 'c' before initialization
PS C:\Users\rajes\OneDrive\Node.js programs>
```
- Status Bar:** Lines 13, Col 18, Spaces: 4, UTF-8, CRLF, JavaScript.

Explanation of Output:

1. Using var
 - The variable a is hoisted (moved to the top in memory) but not assigned a value yet.
 - That's why printing a before its declaration gives undefined instead of an error.

2. Using let
 - let is also hoisted but it stays in a “temporal dead zone” until it is initialized.
 - Accessing it before declaration gives an error.

3. Using const
 - Works like let, but it also must be assigned a value immediately when declared.
 - Accessing it before declaration gives an error as well.

The screenshot shows the Visual Studio Code (VS Code) interface. The top bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar labeled "Node.js programs". The left sidebar has sections for Explorer, NODEJS PROGRAMS (containing FunctionDeclarationsVsExpressions.js, HoistingInVariables.js, SandeepGF202322853.js, and Server.js), and a pinned icon. The main area displays a code editor with the following content:

```
1 console.log(add(2, 3));
2
3 function add(a, b) {
4     return a + b;
5 }
6
7 console.log(add(5, 7));
8
9 try {
10     console.log(multiply(2, 3));
11 } catch (err) {
12     console.log("Error with function expression:", err.message);
13 }
14
15 const multiply = function(a, b) {
16     return a * b;
17 };
18
19 console.log(multiply(4, 5));
```

The bottom navigation bar includes PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), PORTS, and DOLPHINDB. The terminal window shows the following output:

```
PS C:\Users\rajes\OneDrive\Node.js programs> node FunctionDeclarationsVsExpressions.js
5
12
Error with function expression: Cannot access 'multiply' before initialization
20
PS C:\Users\rajes\OneDrive\Node.js programs>
```

At the bottom right, status information includes Line 11, Col 14, Spaces: 4, UTF-8, CRLF, and JavaScript.

What Works and What Fails:

1. Calling add before declaration → Works
 - Function declarations are hoisted completely (both name and body).
 - So add(2,3) runs fine even if called before the function definition.
2. Calling multiply before declaration → Fails
 - Function expressions (when stored in const/let) are not initialized until that line of code runs.
 - So calling multiply before it's defined gives an error.
3. Calling multiply after declaration → Works
 - Once assigned, multiply(2,3) runs fine and prints 6.

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer (NODEJS PROGRAMS):** Shows files like 2853.js, Server.js, HoistingInVariables.js, FunctionDeclarationsVsExpressions.js, ArrowFunctionsVsNormalFunctions.js (the active file), HigherOrderFunctions.js, HoistingInVariables.js, SandeepGF202322053.js, and Server.js.
- Code Editor:** The active file is `ArrowFunctionsVsNormalFunctions.js`. The code compares Arrow functions and Normal functions:const obj = { arrowFunc: () => { console.log("Hello from Arrow function"); }, normalFunc: function() { console.log("Hello from Normal function"); }};obj.arrowFunc();obj.normalFunc();
- Terminal:** The terminal output shows the execution of the script and its results:

```
PS C:\Users\rajes\OneDrive\Node.js programs> node ArrowFunctionsVsNormalFunctions
Hello from Arrow function
Hello from Normal function
PS C:\Users\rajes\OneDrive\Node.js programs>
```
- Status Bar:** Shows file statistics: Line 11, Col 18, Spaces: 4, UTF-8, CRLF, JavaScript.

The screenshot shows a code editor interface for Node.js development. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar labeled "Node.js programs". The left sidebar features an Explorer icon, a list of "NODEJS PROGRAMS" containing files like 2853.js, Server.js, HoistingInVariables.js, FunctionDeclarationsVsExpressions.js, ArrowFunctionsVsNormalFunctions.js, and HigherOrderFunctions.js (which is currently selected), and icons for Find, Replace, and Settings.

The main workspace displays the content of the "HigherOrderFunctions.js" file:

```
function calculate(operation, a, b) {
  return operation(a, b);
}

function add(x, y) {
  return x + y;
}

function subtract(x, y) {
  return x - y;
}

console.log(calculate(add, 10, 5));
console.log(calculate(subtract, 10, 5));
console.log(calculate((x, y) => x * y, 4, 5));
console.log(calculate((x, y) => x / y, 20, 4));
```

Below the code editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and DOLPHINDE. The TERMINAL tab is active, showing the command "node HigherOrderFunctions.js" and its output:

```
PS C:\Users\rajes\OneDrive\Node.js programs> node HigherOrderFunctions.js
15
5
20
5
PS C:\Users\rajes\OneDrive\Node.js programs>
```

The bottom status bar indicates the file is "Ln 16, Col 48" with "Spaces:4", "UTF-8", "CRLF", and "JavaScript" selected.