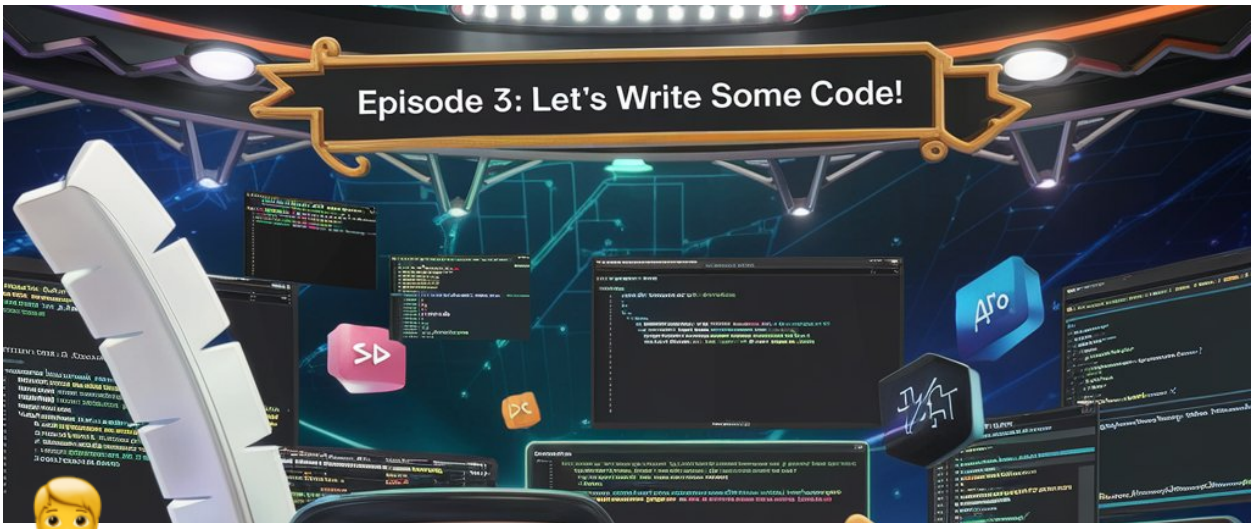
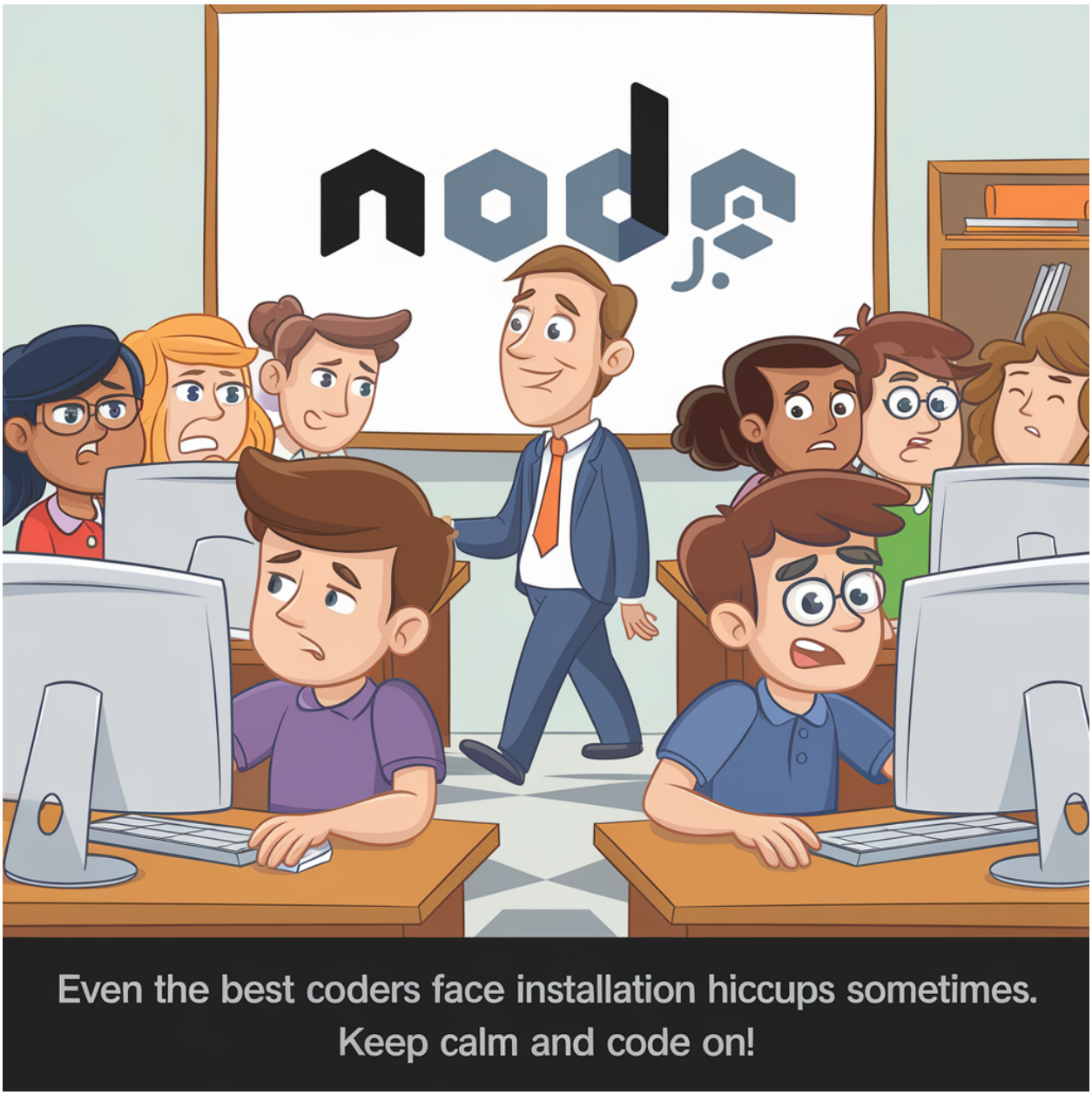



# Episode 3 - writing code

Let's write some code.

## 1.lets start By installing Node.js on your system:

**https://nodejs.org/en**




## 2. Verify installed correctly or node by writing command on terminal

`node ~v`

The command `node -v` is used to check the installed version of Node.js. If Node.js is not installed, this command will produce an error indicating that the `node` command is not

recognized or not found

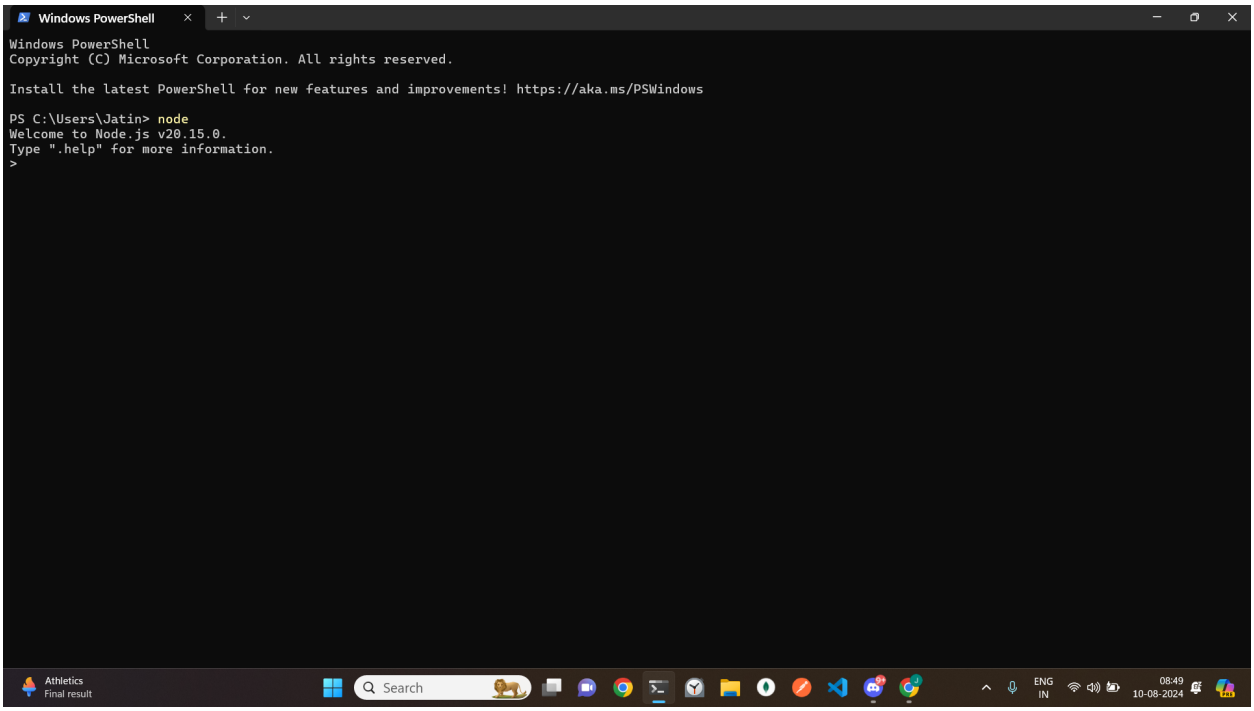
`npm ~v`

The command `npm -v` displays the version of **NPM** (Node Package Manager) that is currently installed on your system.

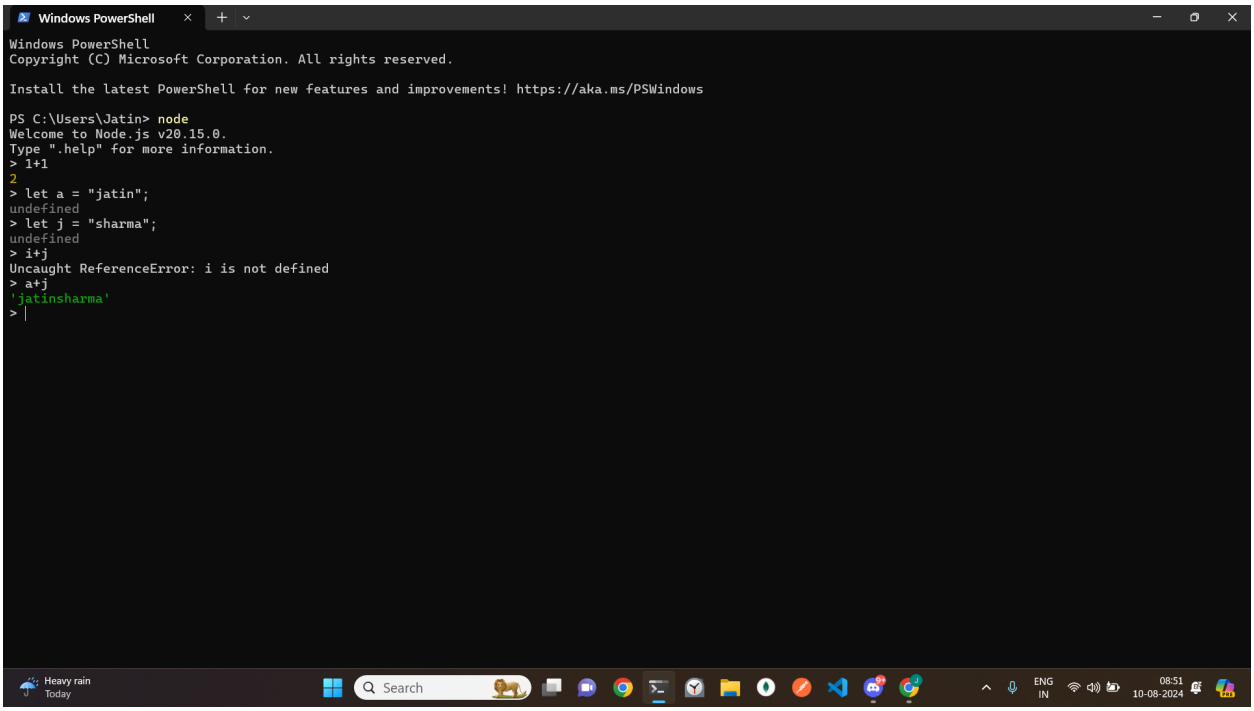
## 3.Writing code

- **Node REPL [Read, Evaluate, Print, Loop]**

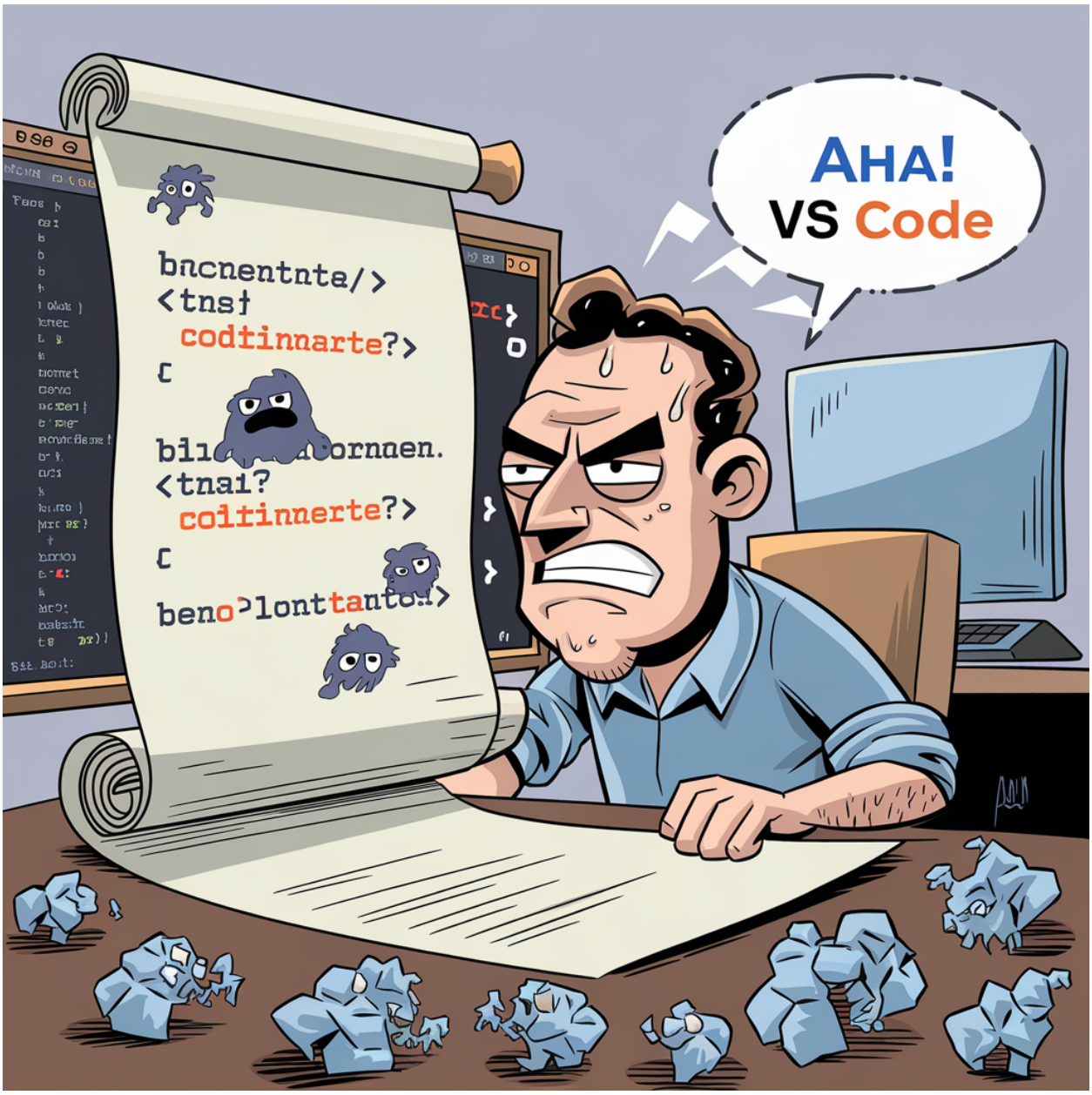
  → write command on terminal `node`




→ You can now write and execute any code on **REPL** (Read-Evaluate-Print Loop).




- This is all running in a Node runtime environment.
- NodeJS is a JS runtime environment, and behind the scenes, it is using the v8 engine.
- its similar to a browser console.
- We cannot write code like this for a long time; it's frustrating. Let's write code on Vscode or any compiler you like.



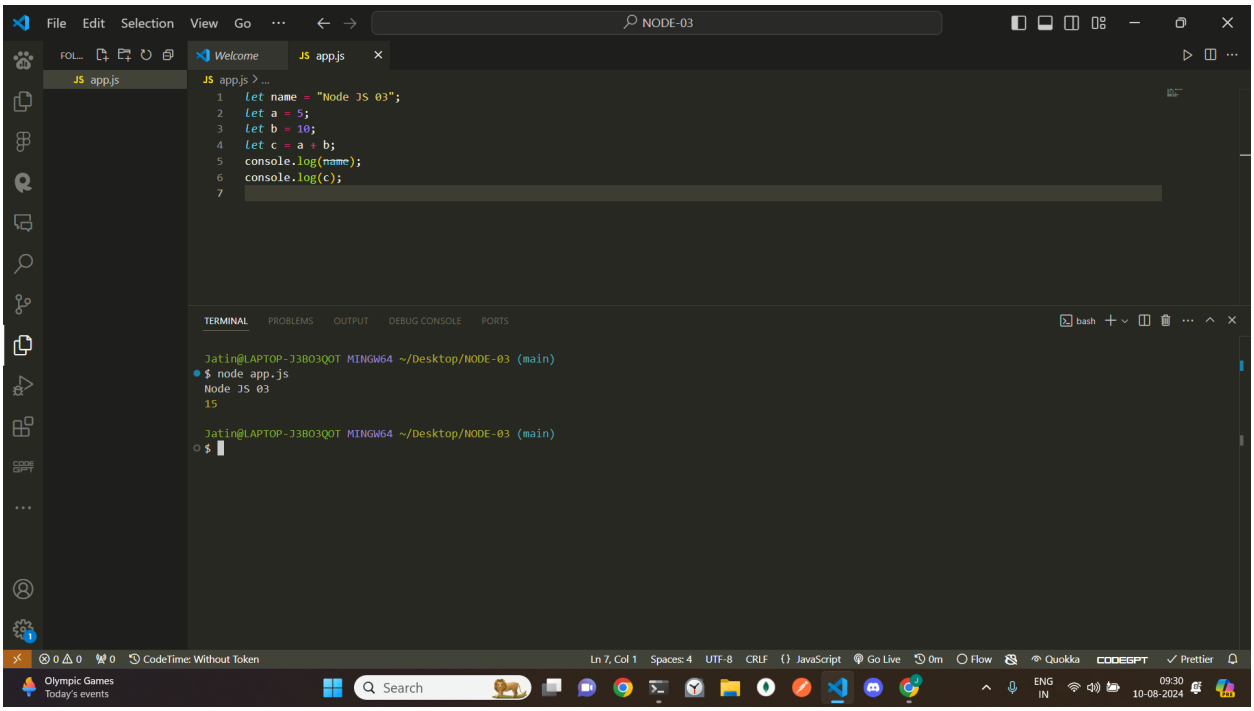

## 4. Code on VScode.

- **Create a Folder**: Start by creating a new folder on your computer. You can name it anything you like (e.g., `my-nodejs-project`).
- **Open the Folder in VSCode**: Open **Visual Studio Code (VSCode)** and go to `File > Open Folder`. Select the folder you just created.
- **Create a File Named** `app.js`: Inside your opened folder in VSCode, create a new file named `app.js`.

- **Write Code**: You can now start writing  your code⬇️ inside the `app.js` file.

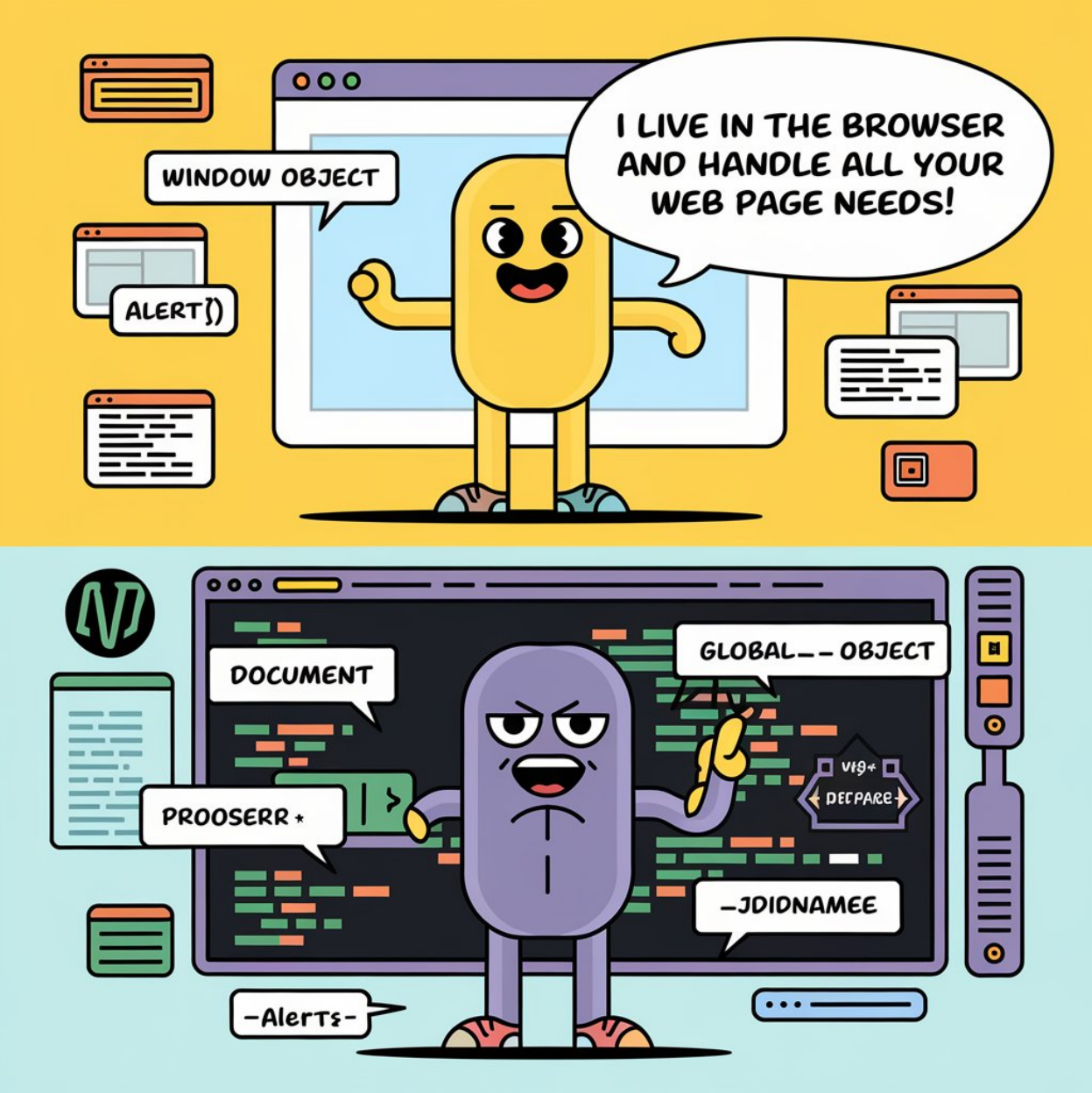
```
let name = "Node JS 03";
let a = 5;
let b = 10;
let c = a + b;
console.log(name);
console.log(c);
```

1. open terminal using the shortcut **Ctrl + `**.
2. now write command `node app.js` to run the code

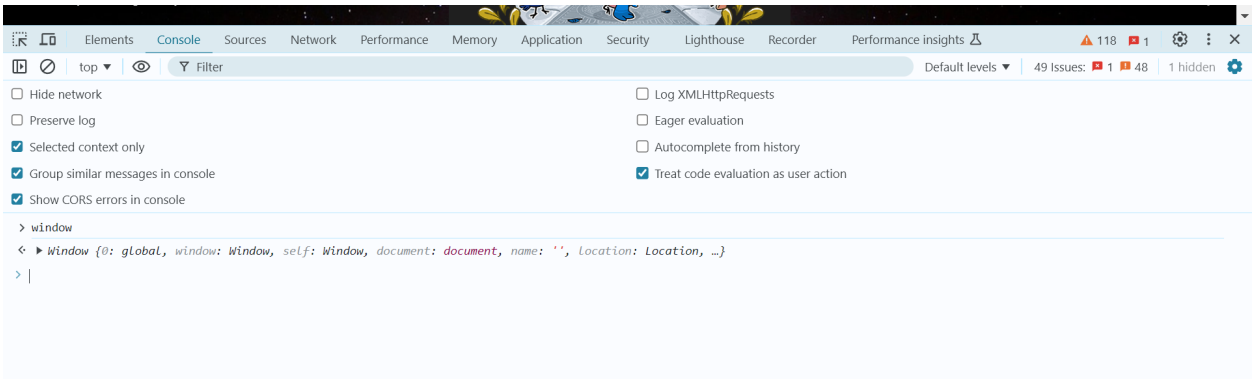



## 5. Let’s talk about global objects in NodeJS.

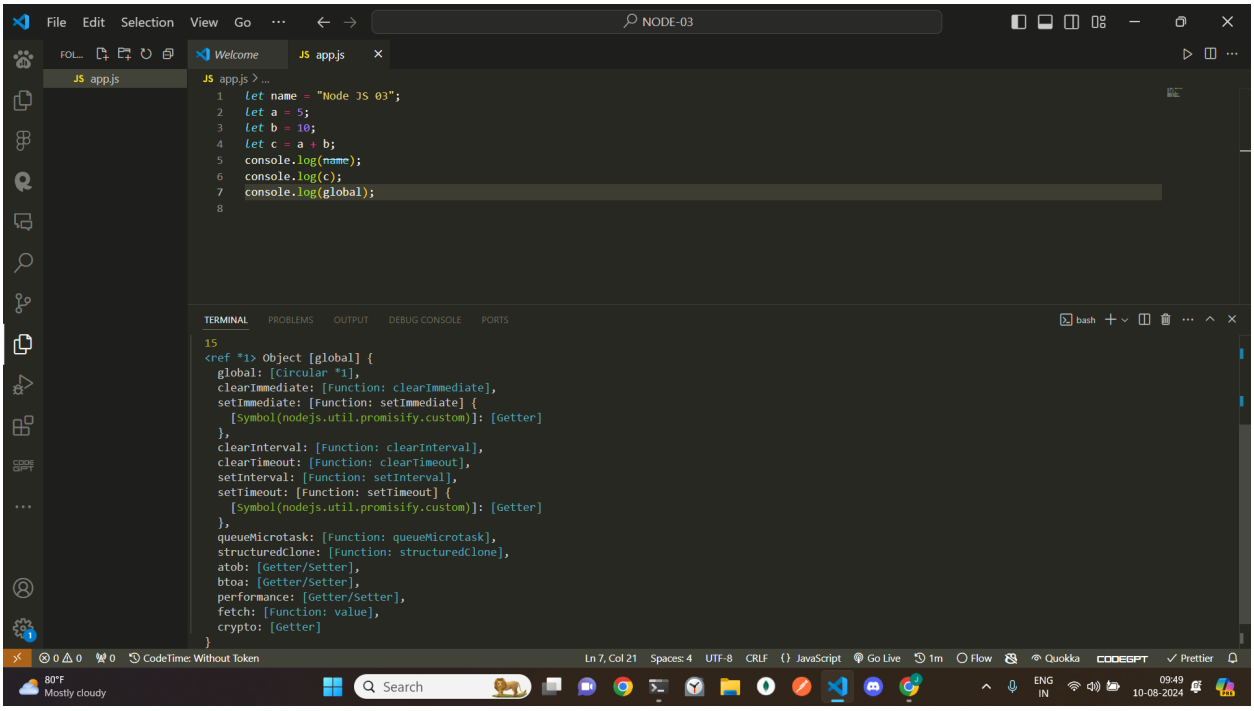
https://developer.mozilla.org/en-US/docs/Glossary/Global_object




- The `window` object is a global object provided by the **browser**, not by the **V8 engine**.




- Now ,
  In
  **Node.js**, the global object is known as `global`, which is equivalent to the `window` object in the browser.




- A global object is not a part of the **V8 engine**; instead, it's a feature provided by **Node.js**.
- This **global object** offers various functionalities, such as `setTimeout()`, `setInterval()`, and more.

## Important Note:

```
console.log(this); // Outputs: {}

When you use console.log(this); at the global level in Node.j
s, it will log an empty object, indicating that this does not
refer to the global object in this context.
```


## Global this

is always a global object, regardless of where it is accessed. It was introduced in ECMAScript 2020 to provide a standardized way to refer to the global object in any environment (browsers, Node.js, etc.).

- In browsers, `global` is equivalent to `window`.
- In Node.js, `globalThis` is equivalent to `global`.
- It provides a consistent way to access the global object without worrying about the environment.

```
// console.log(this); //empty object
console.log(globalThis);

```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE PORTS

```
Jatin@LAPTOP-J3BO3QOT MINGW64 ~/Desktop/NODE-03 (main)
$ node app.js
<ref *1> Object [global] {
  global: [Circular *1],
  clearImmediate: [Function: clearImmediate],
  setImmediate: [Function: setImmediate] {
    [Symbol(nodejs.util.promisify.custom)]: [Getter]
  },
  clearInterval: [Function: clearInterval],
  clearTimeout: [Function: clearTimeout],
  setInterval: [Function: setInterval],
  setTimeout: [Function: setTimeout] {
    [Symbol(nodejs.util.promisify.custom)]: [Getter]
  },
  queueMicrotask: [Function: queueMicrotask],
  structuredClone: [Function: structuredClone],
  atob: [Getter/Setter],
  btoa: [Getter/Setter],
  performance: [Getter/Setter],
  fetch: [Function: value],
  crypto: [Getter]
}
```


We're so glad you were here!
Until next time, Keep coding
and stay curious!
V8
Javesount Engine