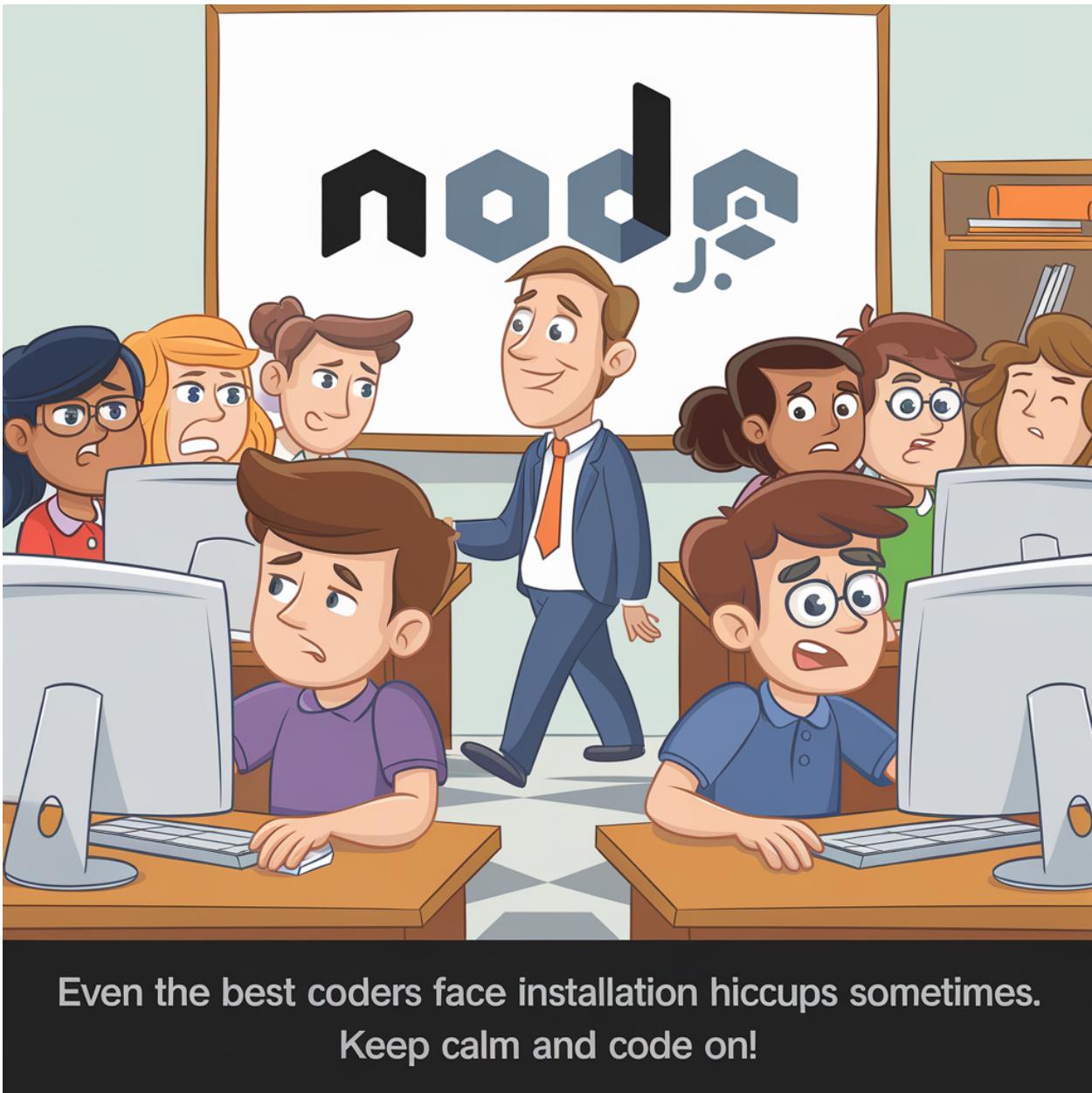


Episode 3 - writing code

Let's write some code.

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

<https://nodejs.org/en>



Even the best coders face installation hiccups sometimes.
Keep calm and code on!

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

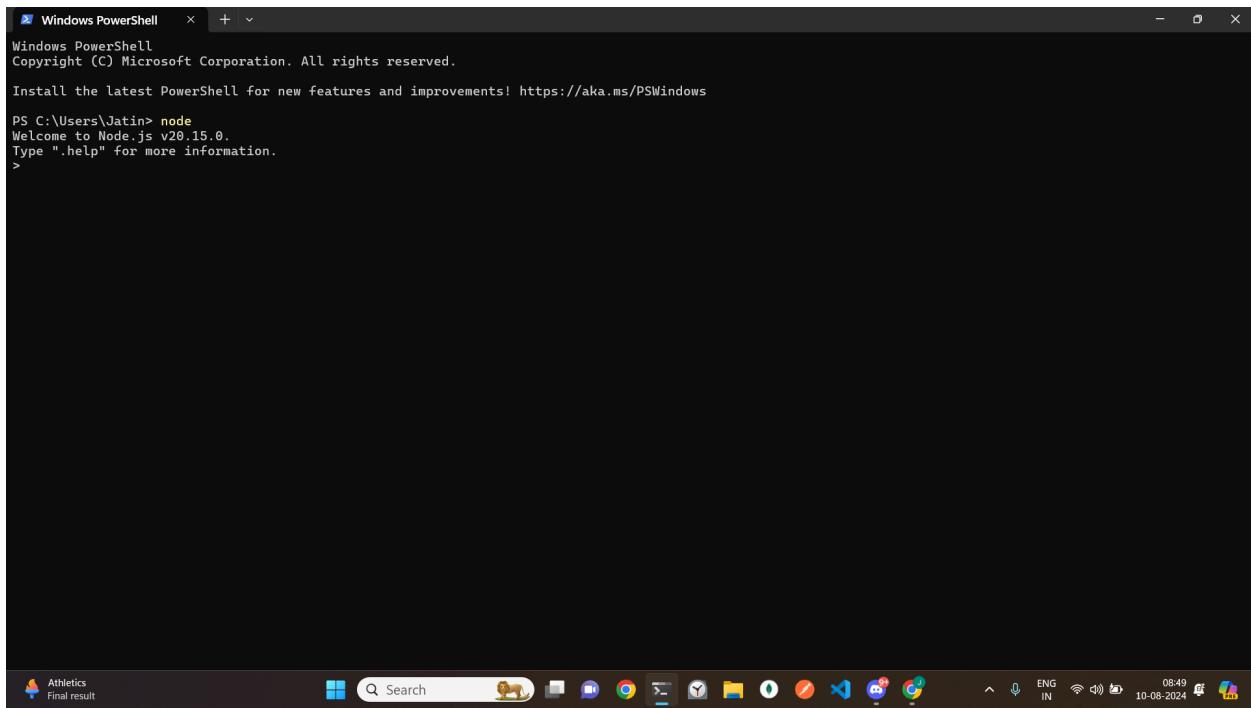
`npm ~v`

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`



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the following text:

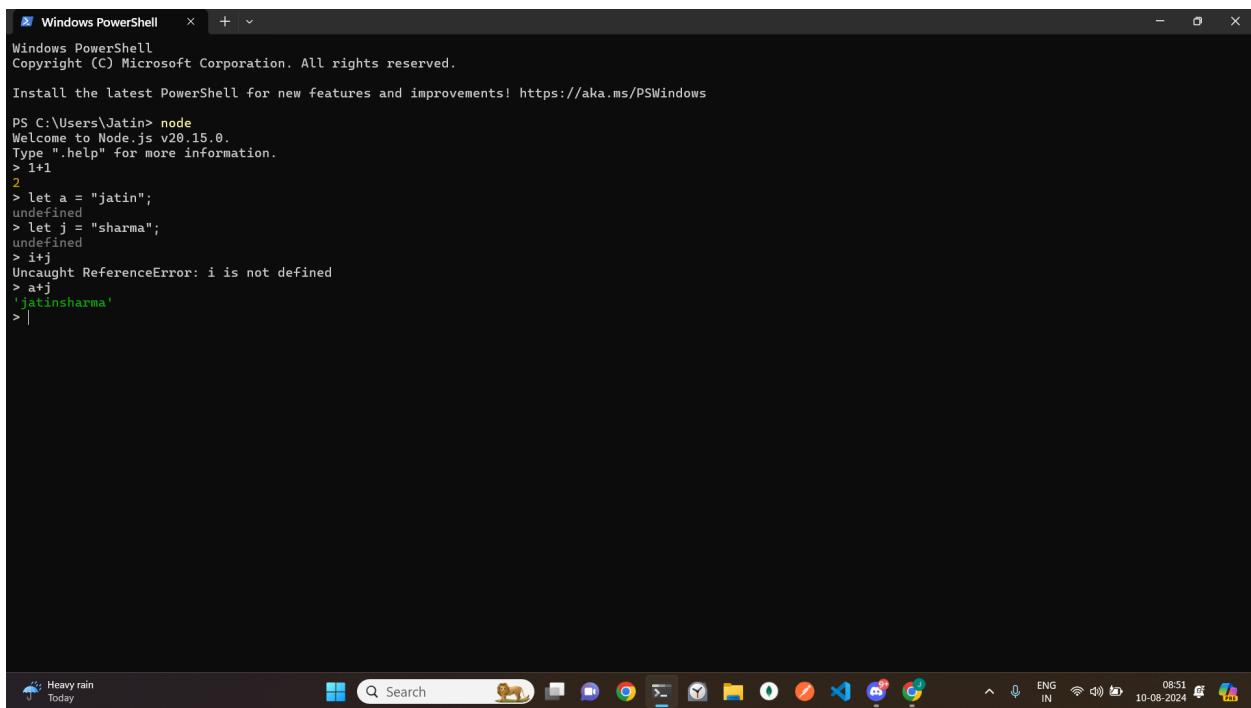
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Jatin> node
Welcome to Node.js v20.15.0.
Type ".help" for more information.
>
```

The window is running on a Windows 10 desktop, as indicated by the taskbar icons at the bottom. The taskbar includes icons for Athletics, Final result, Search, File Explorer, Task View, Task Manager, File History, OneDrive, Edge, Google Chrome, Notepad, Visual Studio Code, and others. The system tray shows the date (10-08-2024), time (08:49), battery level, and network status.

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

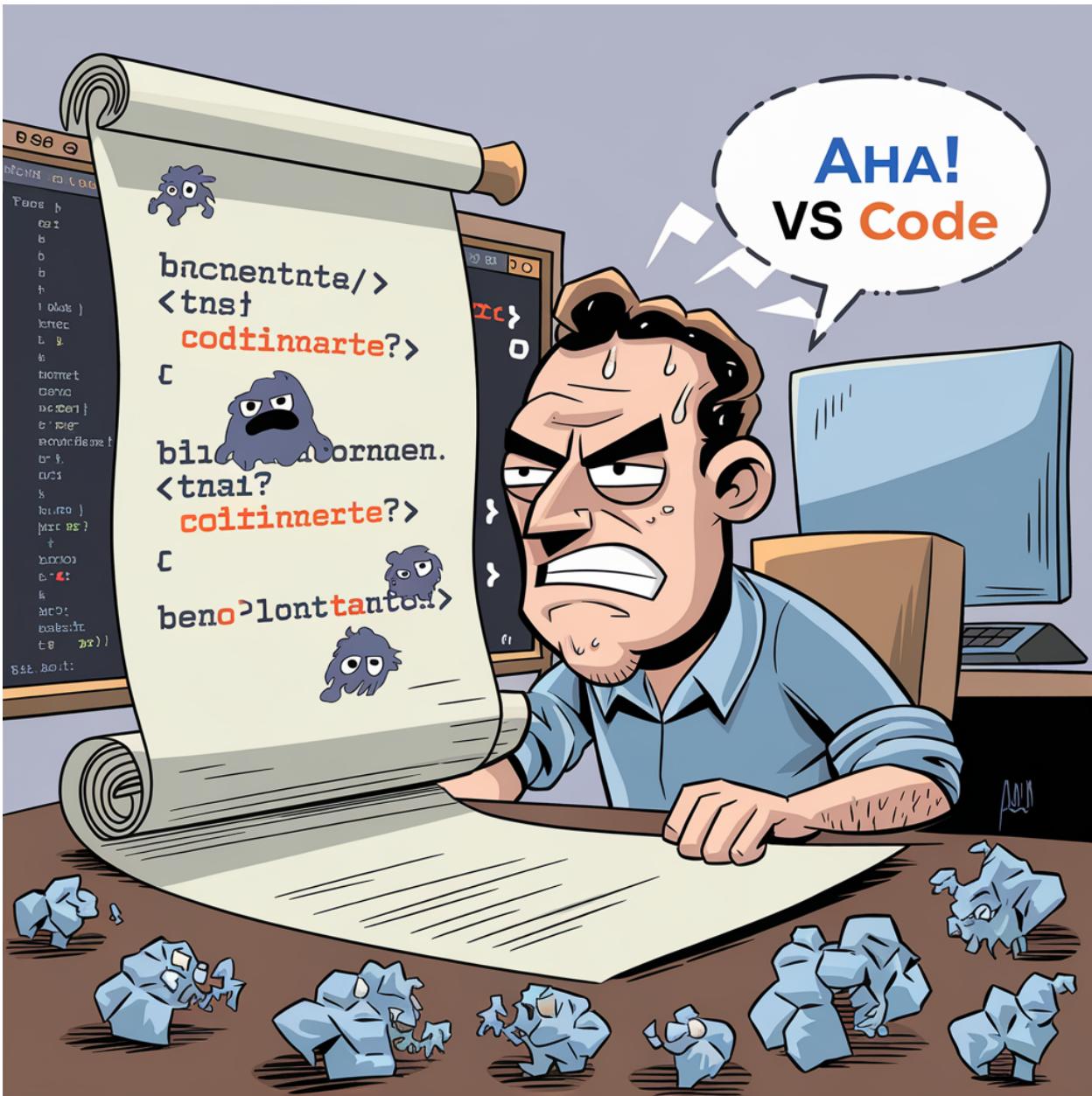


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Jatin> node
Welcome to Node.js v20.15.0.
Type ".help" for more information.
> i+j
Uncaught ReferenceError: i is not defined
> a+j
'jatinsharma'
> |
```

- 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.



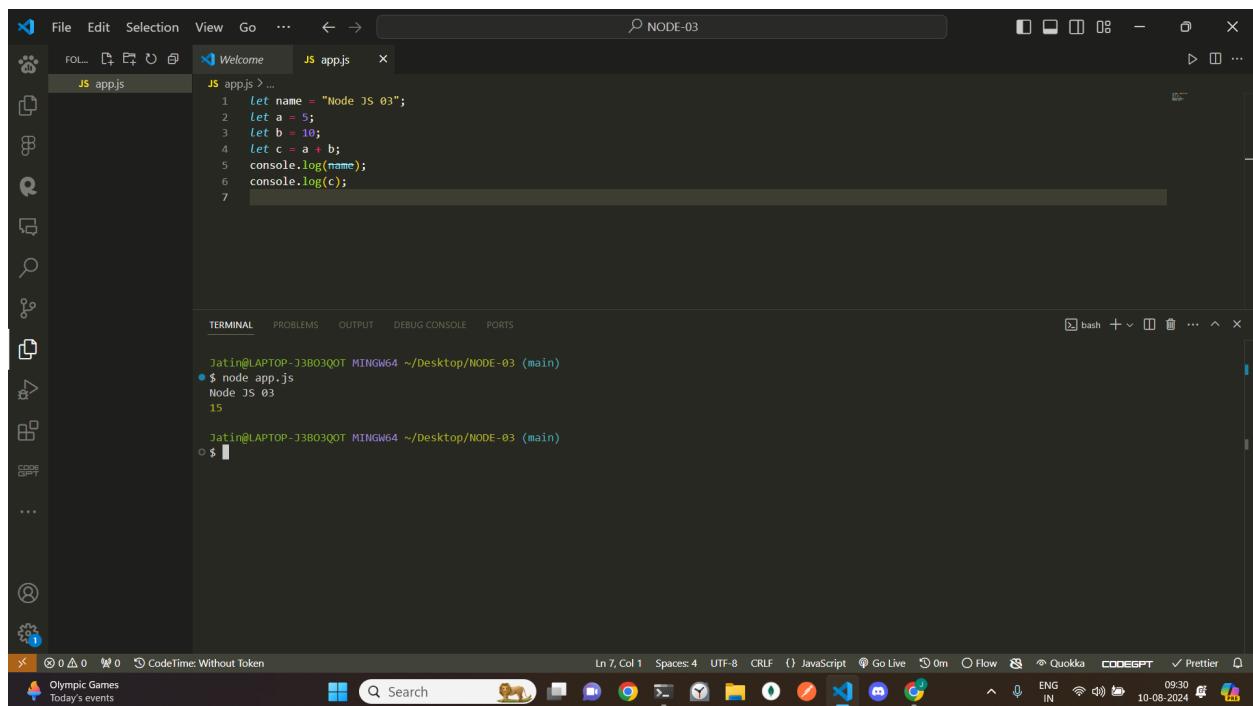
4. Code on VScode.

- **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



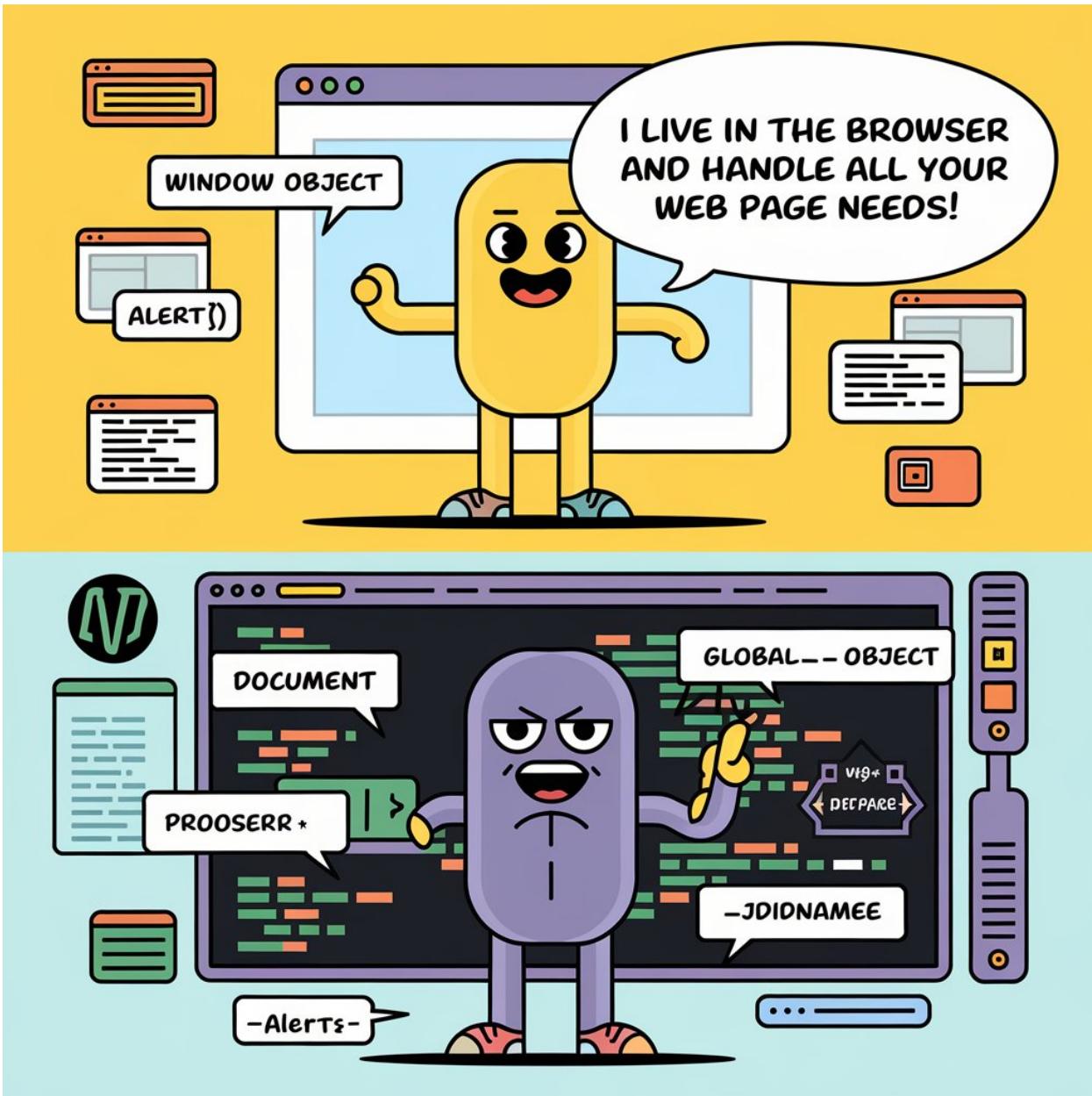
The screenshot shows a dark-themed instance of Visual Studio Code. On the left is the sidebar with icons for files, folders, and other extensions like CodeGPT. The main area has two tabs: 'Welcome' and 'app.js'. The 'app.js' tab contains the following code:

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

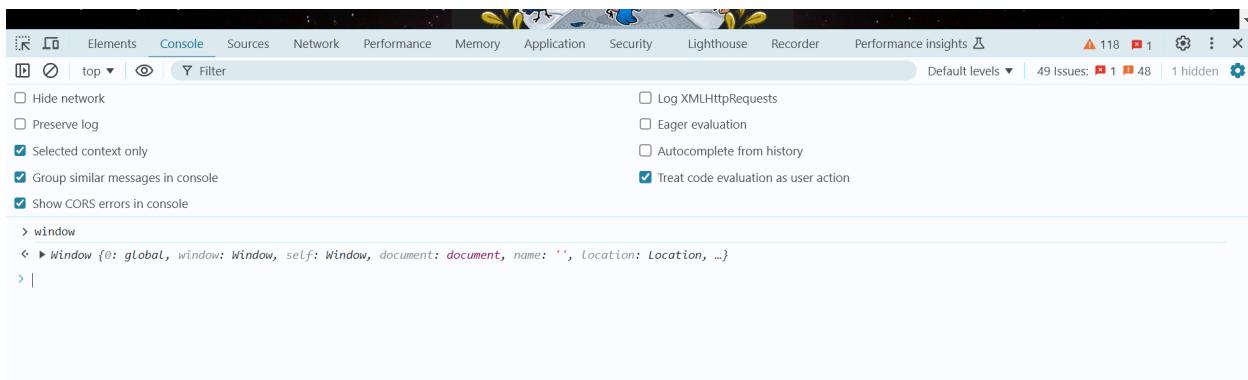
Below the editor is the terminal window, which displays the command \$ node app.js followed by the output Node JS 03. The status bar at the bottom shows the current date and time as 10-08-2024.

5. Let's talk about global objects in NodeJS.

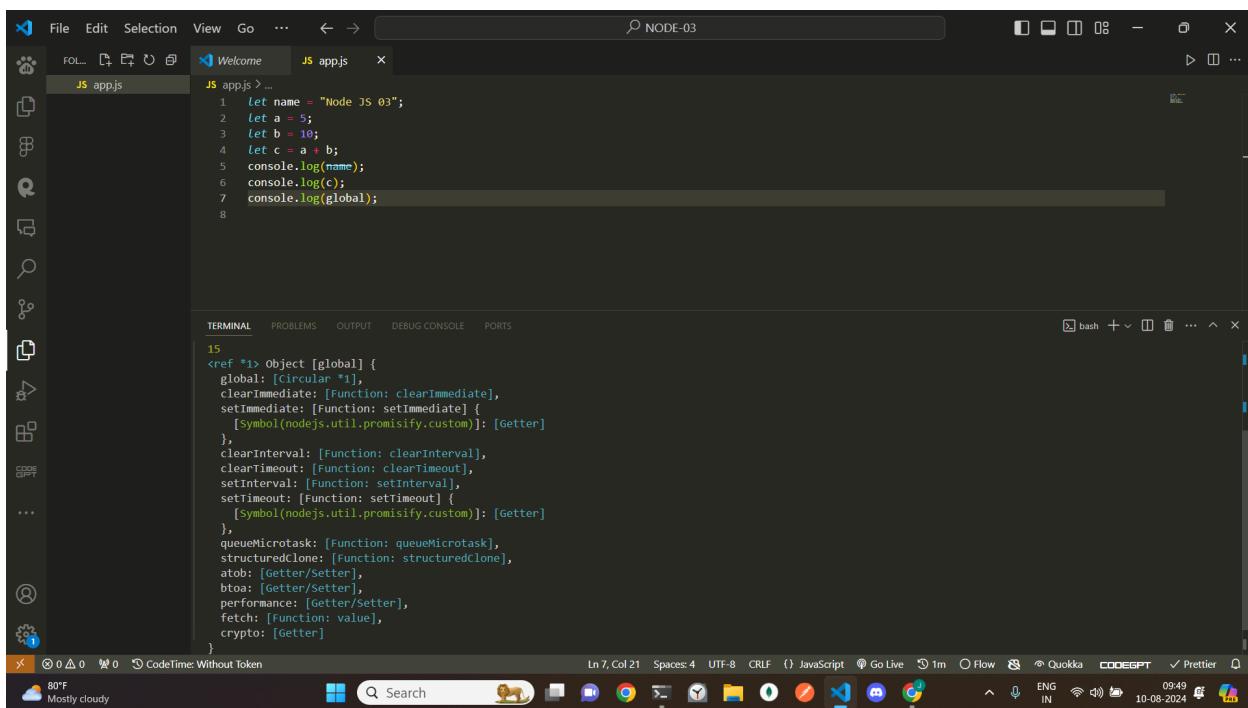
https://developer.mozilla.org/en-US/docs/Glossary/Global_object



- 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.js`, it will log an empty object, indicating that `this` does not refer to the global object `in this` context.

The screenshot shows a terminal window with the following content:

```
8   console.log(this); //empty object
9

TERMINAL      PROBLEMS      OUTPUT      DEBUG CONSOLE      PORTS

Jatin@LAPTOP-J3B03Q0T MINGW64 ~/Desktop/NODE-03 (main)
● $ node app.js
{}
```

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.

```
8 // console.log(this); //empty object
9 console.log(globalThis);
10

TERMINAL    PROBLEMS    OUTPUT    DEBUG CONSOLE    PORTS

Jatin@LAPTOP-J3B03QOT 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]
}
```

