🟩 Windows vs Global in Node.js

🔹 Introduction

JavaScript **behaves differently in the browser and Node.js** because of the environment it runs in:

Feature	Browser	Node.js
Global object	`window`	`global`
DOM access	Yes (`document`, `window`)	No
Runs in	Browser	Server / Command Line
Console	`console.log` is available in both	`console.log` is available in both

Tip: Node.js runs **outside the browser**, so it cannot manipulate HTML or CSS.

🔹 Node.js Context – `global`

- Node.js **does not have a `window` or `document`** because it runs **outside the browser**.
- Node.js provides a **`global` object**, similar to `window` in the browser, for **server-side global variables and functions**.

```
global.myVar = "Hello Node.js";
console.log(myVar); // Hello Node.js
```

- **Common properties in `global`:**
  - `process` → Information about Node.js process
  - `Buffer` → Handle binary data
  - `setTimeout`, `setInterval` → Timers
  - `__dirname` and `__filename` → Current file/directory info

◆ `globalThis`

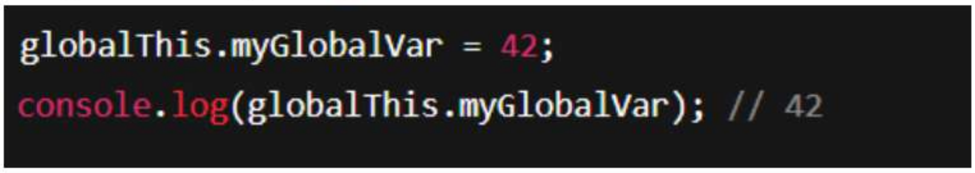
- Introduced in **ECMAScript 2020 (ES11)**.
- Provides a **standard way to access the global object** in **any JavaScript environment**.

Why `globalThis` is useful:

- Accessing the global object used to be environment-specific:

Environment	**Global Object**
Browser	`window`
Node.js	`global`
Web Workers	`self`
Other JS environments	Could vary

- Now, you can always use **`globalThis`**:



```
globalThis.myGlobalVar = 42;
console.log(globalThis.myGlobalVar); // 42
```

- Works in **both Node.js and browser environments**.

◆ Key Differences: `window` vs `global` vs `globalThis`

Feature	**`window`**	**`global`**	**`globalThis`**
Environment	Browser	Node.js	Any JS environment
Access	Global variables, DOM APIs	Global variables, Node APIs	Global variables consistently
Use case	Frontend apps	Backend/server apps	Universal JS code
Example	`window.alert("Hi")`	`global.myVar = 1`	`globalThis.myVar = 1`

🔹 Important Notes

1. **Node.js global object is not truly global** like the `window`. Variables declared with `var` in Node.js **do not automatically become global**. You must attach them explicitly to `global` or `globalThis`.

2. **Avoid polluting the global scope** in Node.js — it can lead to **hard-to-debug issues**.

3. **Use modules** (`exports`/`require`) for sharing data instead of globals whenever possible.

🔹 Summary

- Node.js **doesn't have `window` or `document`**.
- **`global`** is the server-side equivalent of `window`.
- **`globalThis`** standardizes access across all environments.
- Always prefer **modules over global variables** for maintainability.
