**Day 30**





**“Web Development + Security”**

**Introduction to JavaScript:**

**What is JavaScript?**

JavaScript is a high-level, interpreted programming language primarily used to make web pages interactive. Think of it as the “behavior layer” of a website: while HTML structures content and CSS styles it, JavaScript adds interactivity, logic, and dynamic behavior.

**Key Features:**

1. **Client-Side Scripting:**

* Runs in the browser, directly on the user’s device.
* Can respond to user actions (clicks, typing, scrolling, etc.) without refreshing the page.

1. **Interpreted Language:**

* No need to compile. The browser reads and executes JS directly.

1. **Dynamic & Flexible:**

* You can change HTML content, CSS styles, and even add or remove elements dynamically.

1. **Versatile:**

* Can be used for frontend (web pages), backend (Node.js), mobile apps, and even game development.

1. **Event-Driven & Asynchronous:**

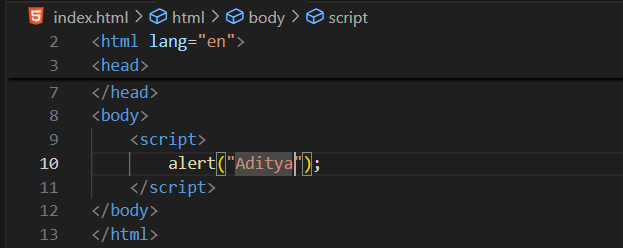
* Can handle events like clicks, API responses, timers, and more.

**Security Note:**

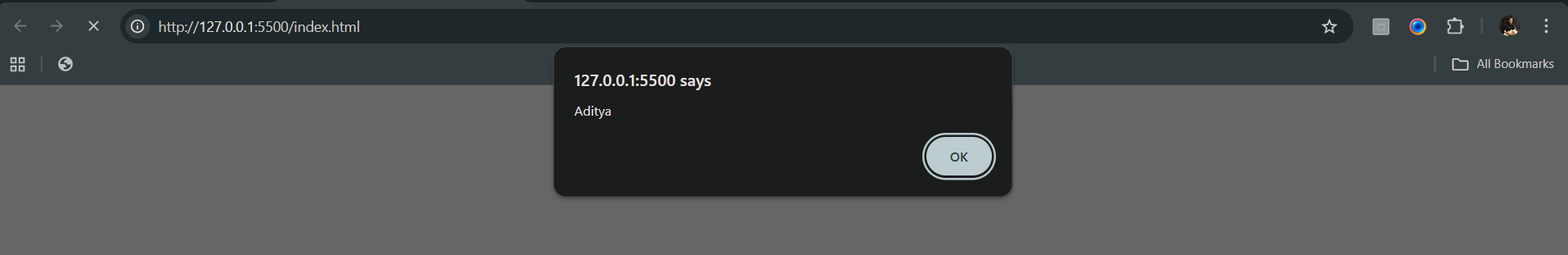
* JS runs on the client, so never trust it for critical security (like authentication or access control).
* Always validate and sanitize user input on the server side.

A basic example of integrating the JavaScript:

Code: using <script> tag in the same .html file



Output:

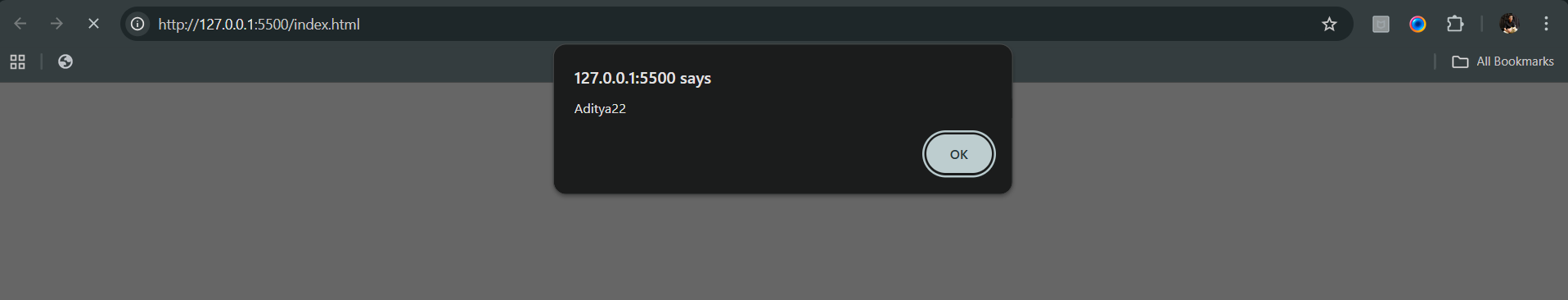


Example: adding the JavaScript from the .js file

Code:

****

Output:

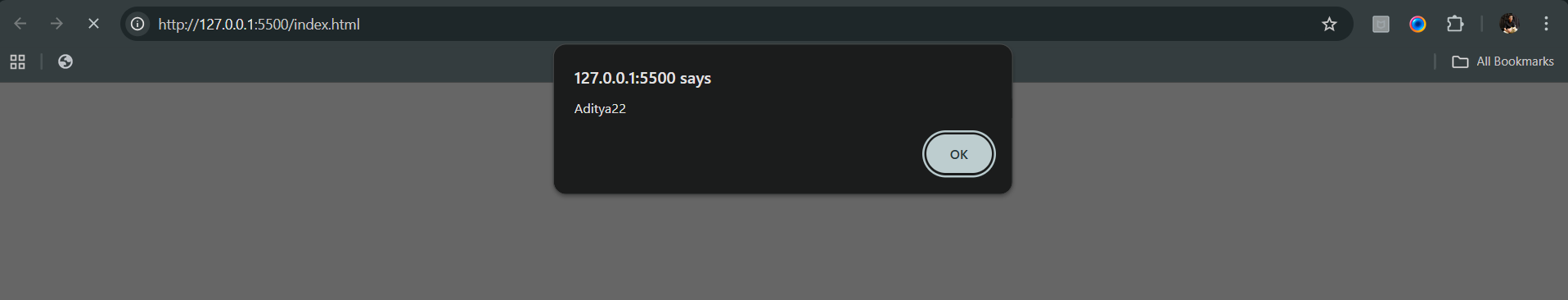


Example: introducing console.log()

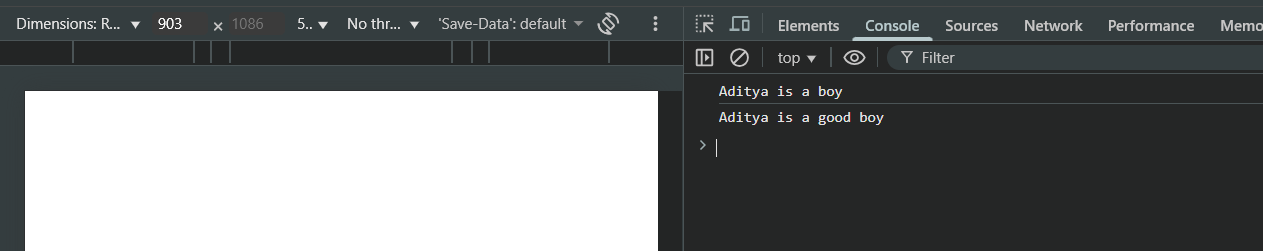
Code:



Output:



Console:



**Variables in JavaScript:**

**What is a Variable?**

A variable is like a container that stores data. You can store values like numbers, text, or objects, and use them later in your program.

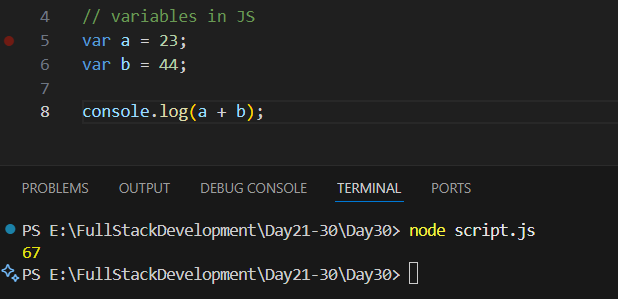
In JavaScript, we declare variables using three keywords:

* var // old way (avoid)
* let // modern and recommended
* const // for constants (unchangeable)

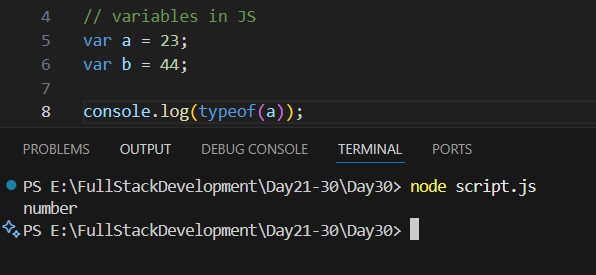
**Difference between var, let and const:**

| **Feature** | **var** | **let** | **const** |
| --- | --- | --- | --- |
| **Scope** | Function-scoped | Block-scoped | Block-scoped |
| **Re-declare** | Allowed | Not allowed | Not allowed |
| **Re-assign** | Allowed | Allowed | Not allowed |
| **Hoisting behavior** | Hoisted (initialized as undefined) | Hoisted but not initialized | Hoisted but not initialized |

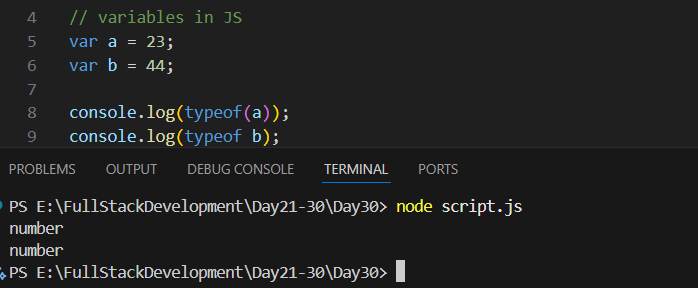
A basic code showing the declaration of variable in JS, and also adding them:

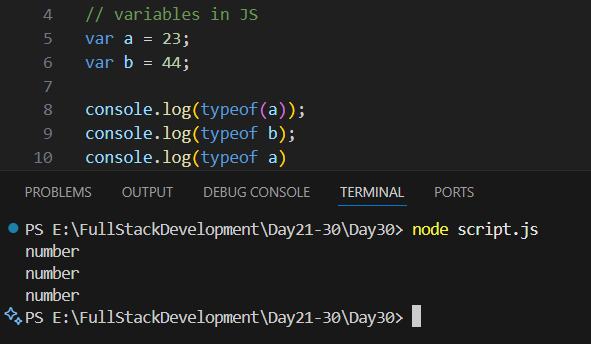


Example: use of typeof() to know the variable type

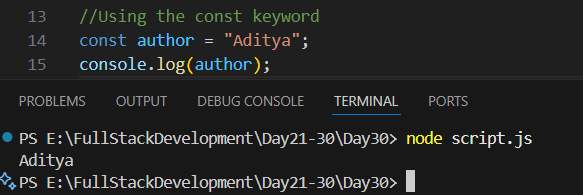


We can write like this as well: (typeof b); and (typeof b) without semicolon

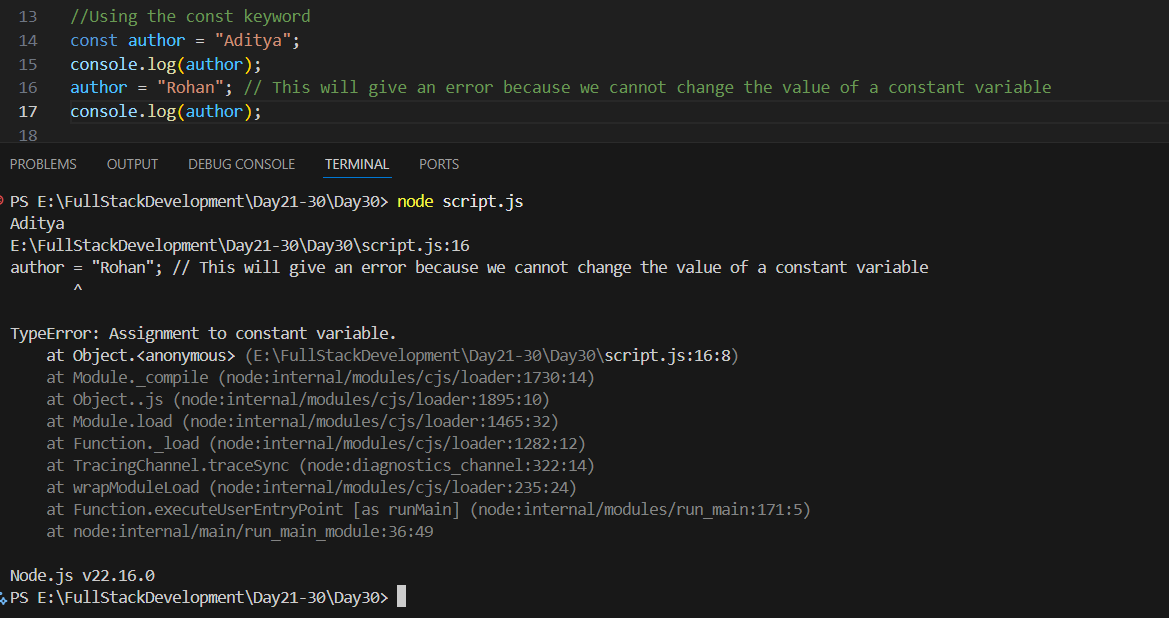




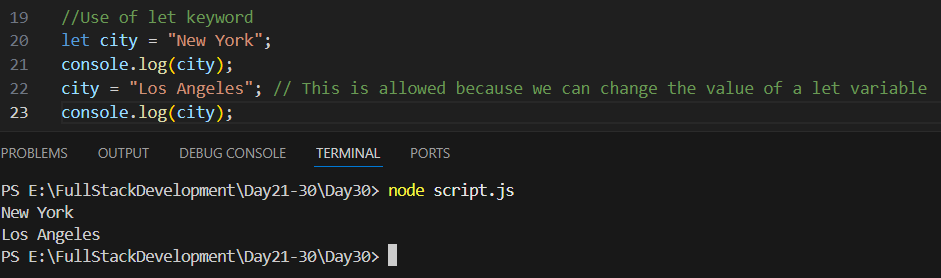
Example: use of const keyword



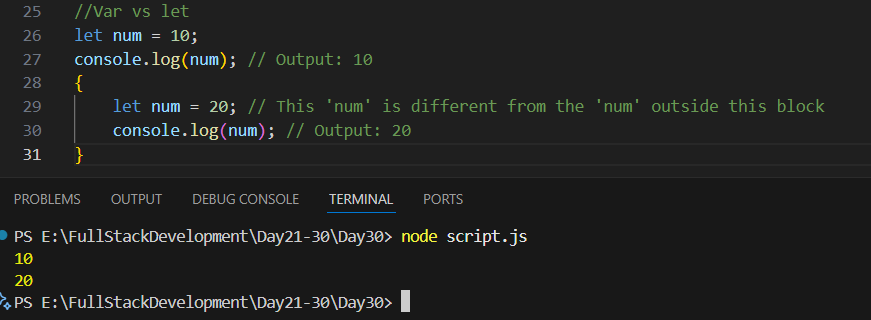
We can’t edit the value we had given in the const variable:



Example: using the let keyword

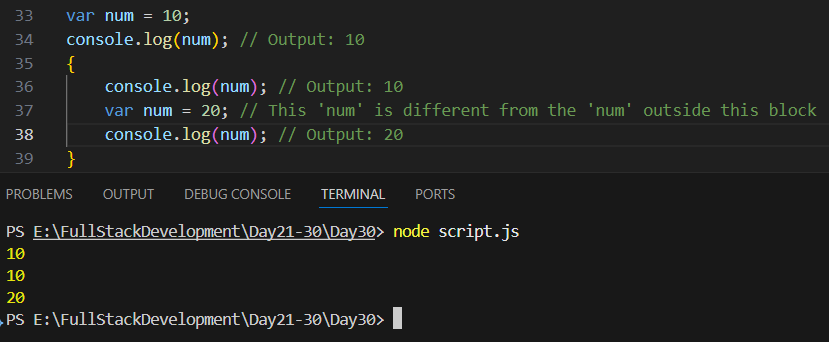


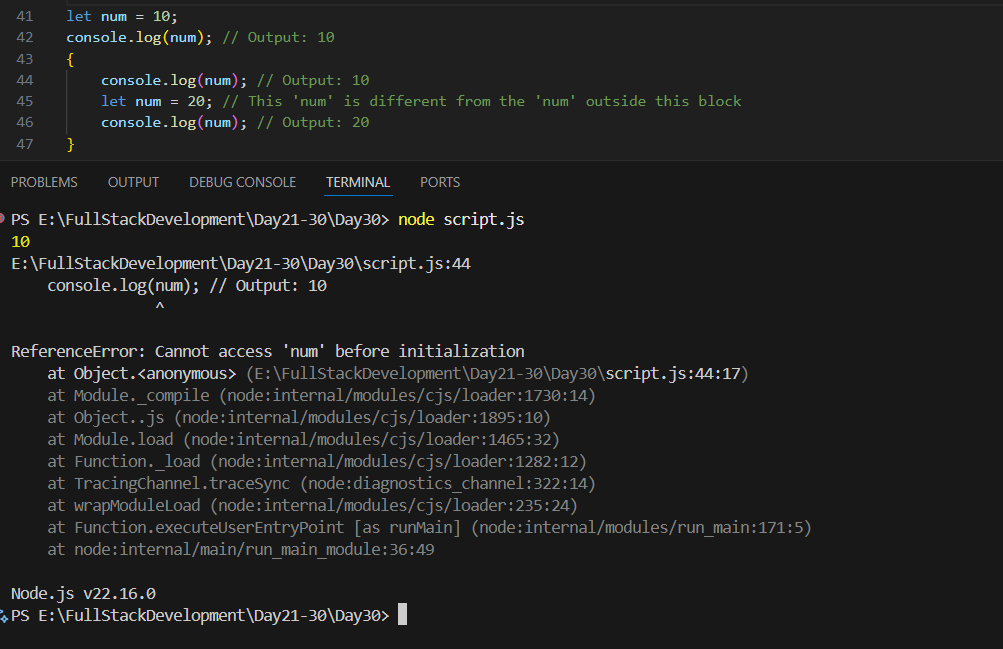
So, what’s the issue with var that we needed let? Var scope is global while let scope is local



See, in above example, we can clearly see that num = 20, be in block and hence be 20.

While in below example when we used var:



Also, 

**Data Types in JavaScript:**

**What are Data Types?**

Data types define the kind of value a variable can hold — like numbers, text, true/false, objects, etc.  
JavaScript is a dynamically typed language → you don’t need to declare the type; JS figures it out automatically.

Example:

* let age = 20; // number
* let name = "Aditya"; // string
* let isOnline = true; // Boolean

**Two main Categories of data types:  
1. Primitive data types:**

These store single immutable values (copied by value).

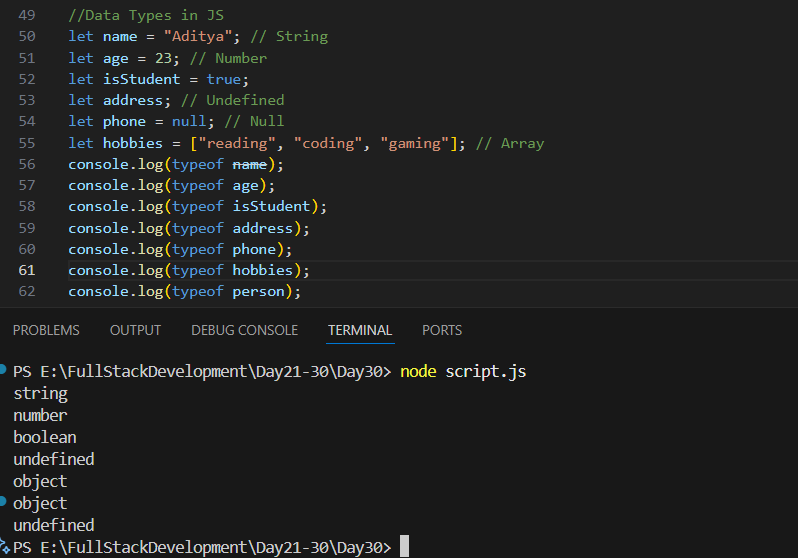
| **Data Type** | **Example** | **Description** |
| --- | --- | --- |
| **String** | "Hello" | Text data, enclosed in quotes |
| **Number** | 42, 3.14 | Both integers and floats |
| **Boolean** | true, false | Logical values |
| **Undefined** | let x; | Variable declared but not assigned |
| **Null** | let y = null; | Empty or unknown value |
| **Symbol** | Symbol("id") | Unique identifiers (ES6) |
| **BigInt** | 12345678901234567890n | For very large integers (ES2020) |

2. **Non-Primitive (Reference) Data Types:**

These store collections or complex structures (copied by reference).

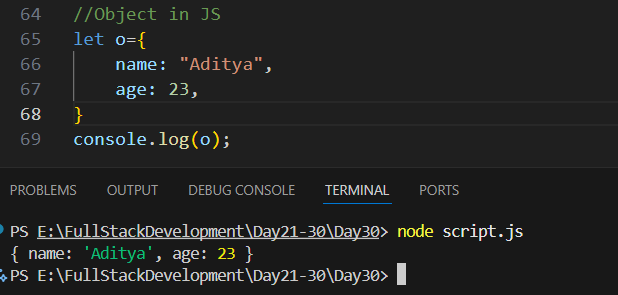
| **Data Type** | **Example** | **Description** |
| --- | --- | --- |
| **Object** | { name: "Aditya", age: 20 } | Key-value pairs |
| **Array** | [10, 20, 30] | Ordered list of values |
| **Function** | function greet() {} | Reusable block of code |

A very basic example:

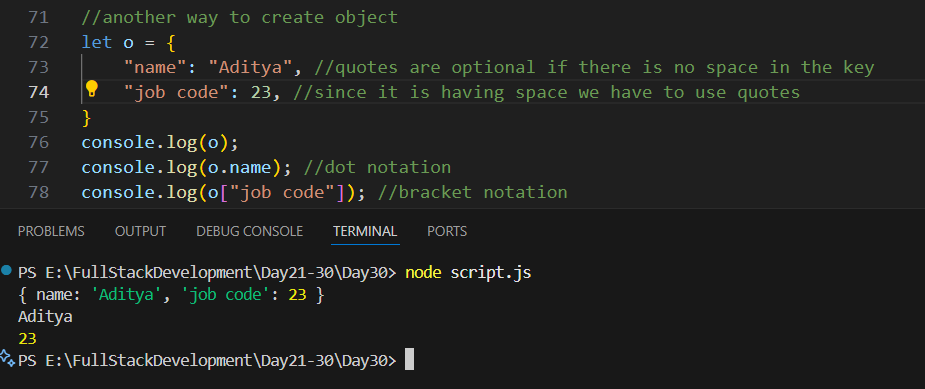


**Note**: data type of null is object.

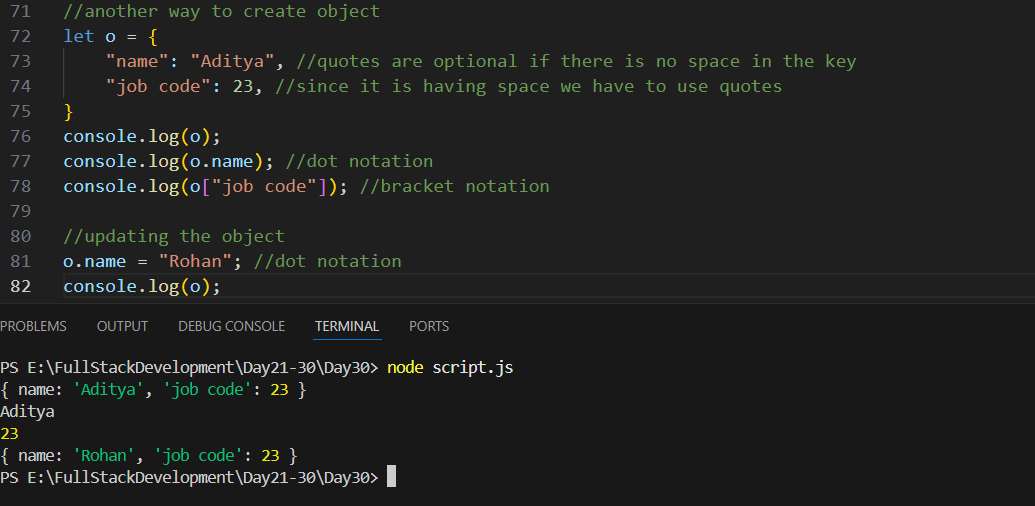
Example: object in JS



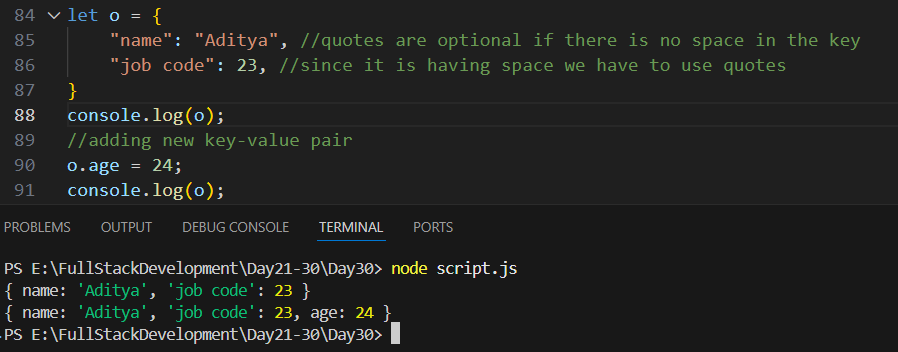
Example: another way to create object, also accessing the stored value.



Example: another way to write.



Example: adding a new key-value pair



--The End--