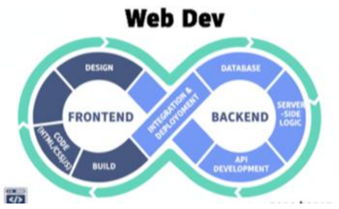
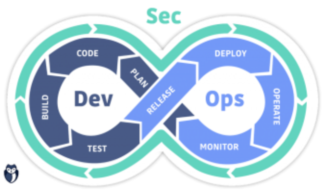




Day 30

"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:

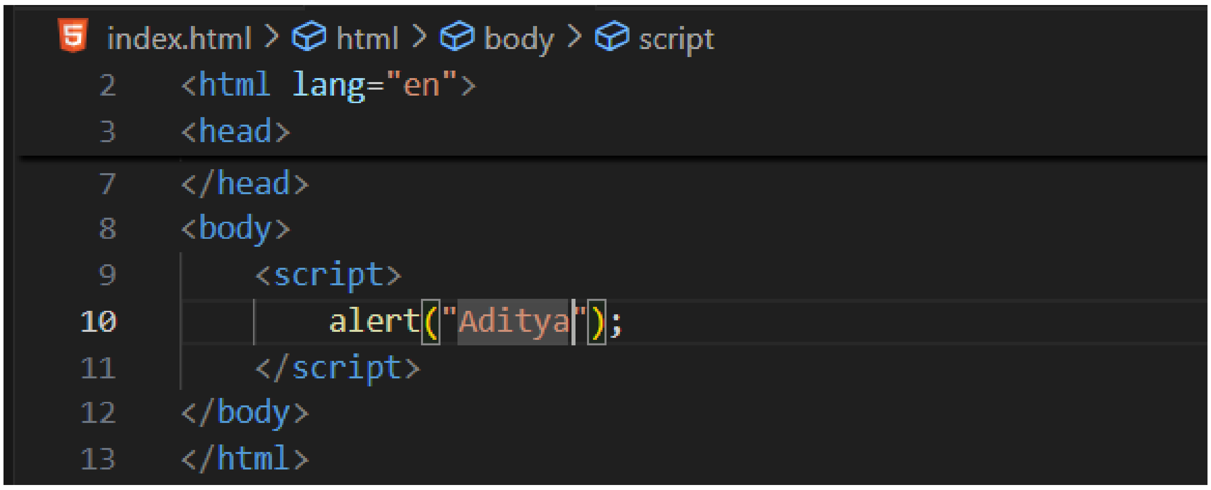
1. **Client-Side Scripting:**
    - Runs in the browser, directly on the user's device.
    - Can respond to user actions (clicks, typing, scrolling, etc.) without refreshing the page.
2. **Interpreted Language:**
    - No need to compile. The browser reads and executes JS directly.
3. **Dynamic & Flexible:**
    - You can change HTML content, CSS styles, and even add or remove elements dynamically.
4. **Versatile:**
    - Can be used for frontend (web pages), backend (Node.js), mobile apps, and even game development.
5. **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.

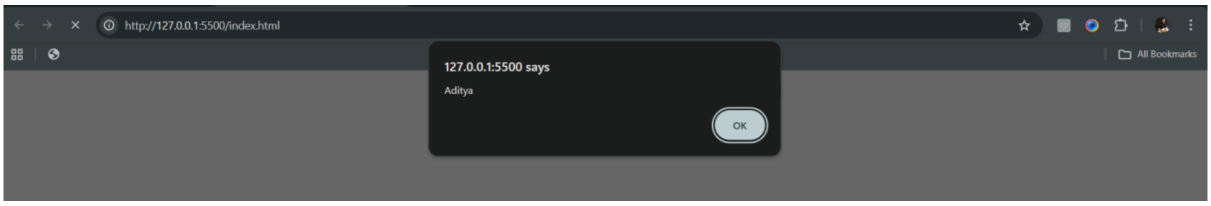
A basic example of integrating the JavaScript:

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



```
<html lang="en">
<head>
</head>
<body>
    <script>
        alert("Aditya");
    </script>
</body>
</html>
```

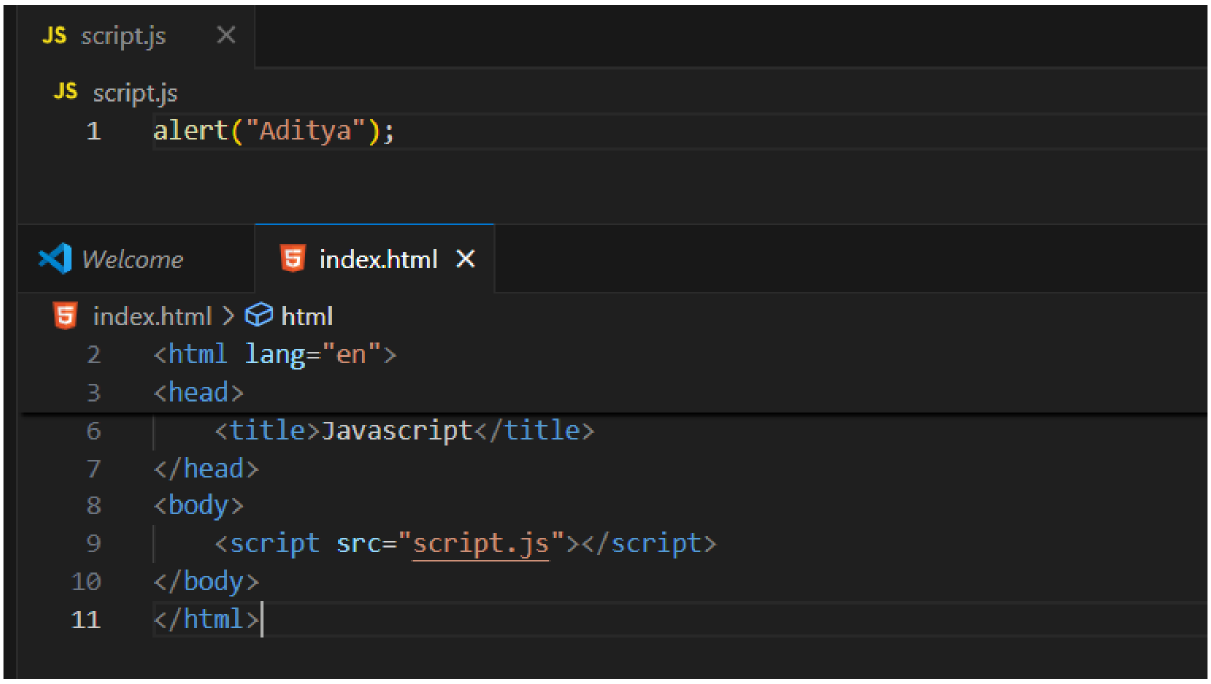
Output:



```
127.0.0.1:5500 says
Aditya
```

Example: adding the JavaScript from the .js file

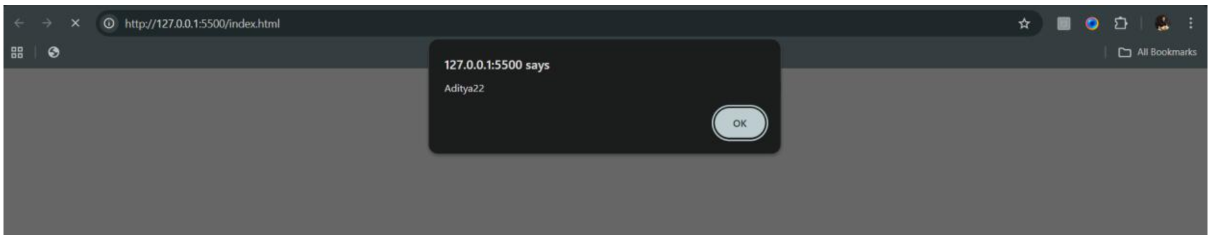
Code:



```
alert("Aditya");
```

```
<html lang="en">
<head>
    <title>Javascript</title>
</head>
<body>
    <script src="script.js"></script>
</body>
</html>
```

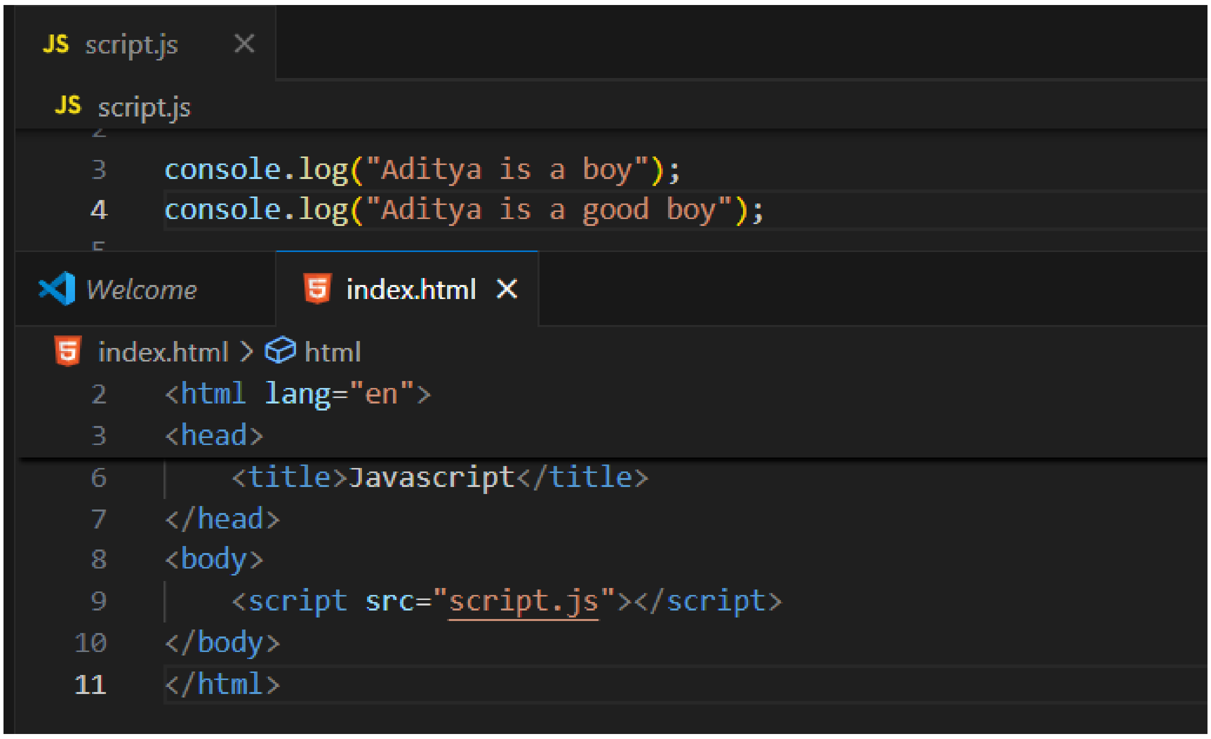
Output:



```
127.0.0.1:5500 says
Aditya22
```

Example: introducing console.log()

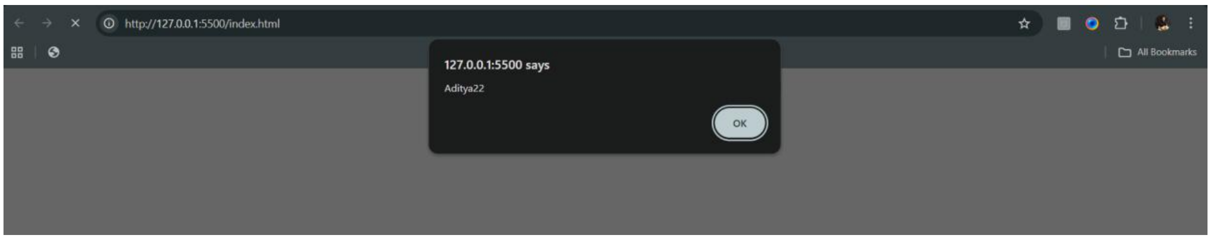
Code:



```
script.js
console.log("Aditya is a boy");
console.log("Aditya is a good boy");
```

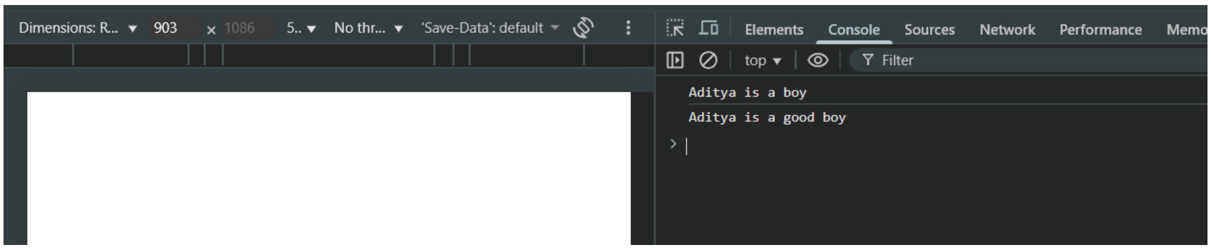
```
index.html
<html lang="en">
<head>
    <title>Javascript</title>
</head>
<body>
    <script src="script.js"></script>
</body>
</html>
```

Output:



```
127.0.0.1:5500 says
Aditya22
```

Console:



```
Aditya is a boy
Aditya is a good boy
```

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.

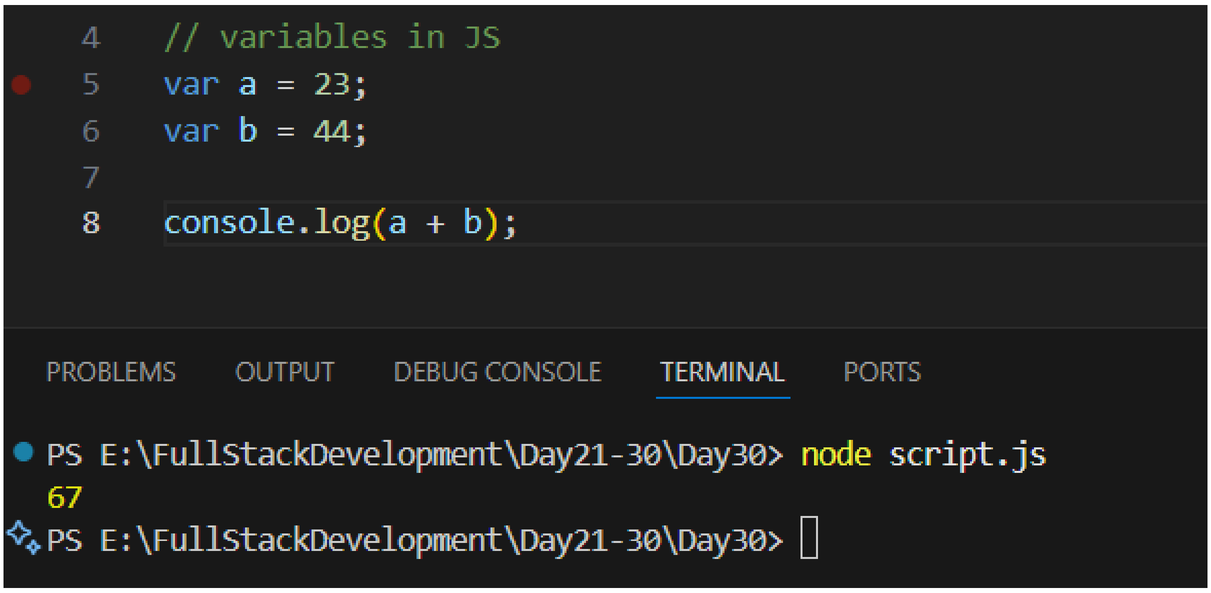
In JavaScript, we declare variables using three keywords:

- 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:

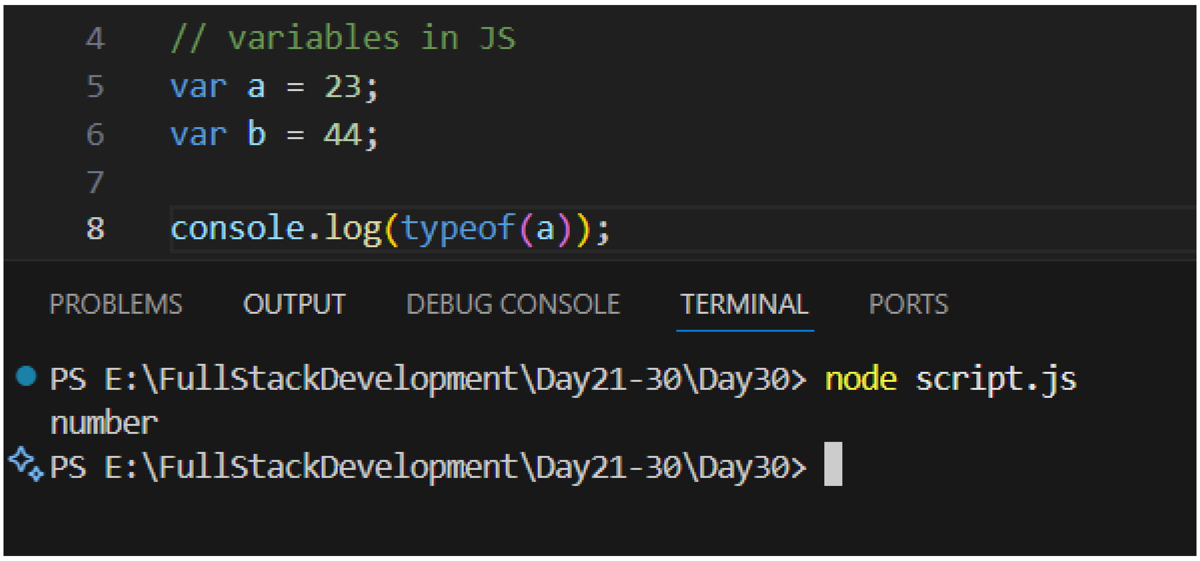


```
// variables in JS
var a = 23;
var b = 44;

console.log(a + b);
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
67
PS E:\FullStackDevelopment\Day21-30\Day30>
```

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



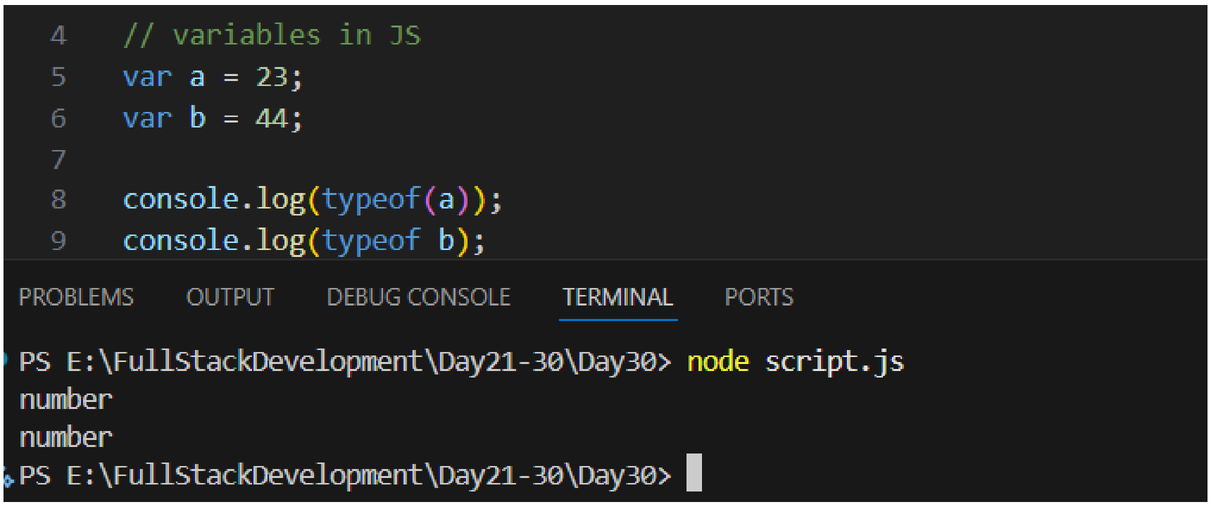
```javascript
// variables in JS
var a = 23;
var b = 44;

console.log(typeof(a));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
number
PS E:\FullStackDevelopment\Day21-30\Day30>
```

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

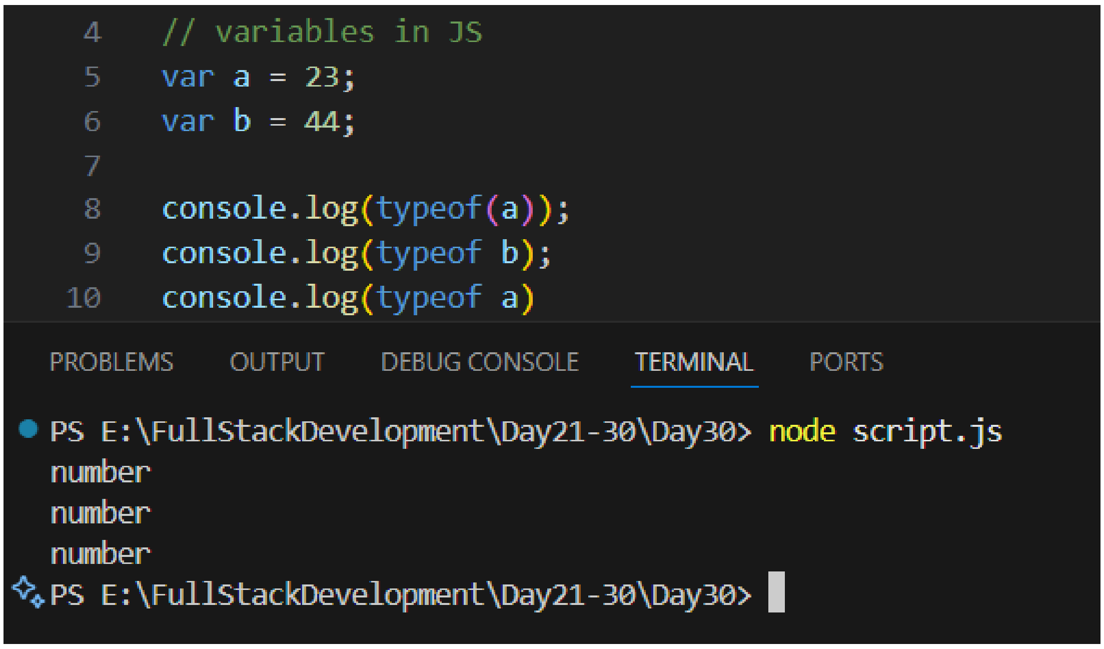


```javascript
// variables in JS
var a = 23;
var b = 44;

console.log(typeof(a));
console.log(typeof b);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
number
number
PS E:\FullStackDevelopment\Day21-30\Day30>
```



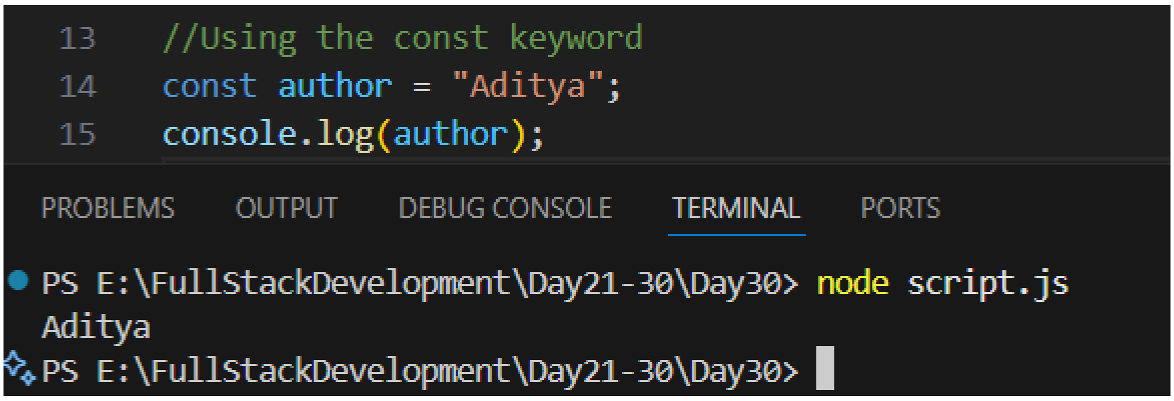
```javascript
// variables in JS
var a = 23;
var b = 44;

console.log(typeof(a));
console.log(typeof b);
console.log(typeof a)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
number
number
number
PS E:\FullStackDevelopment\Day21-30\Day30>
```

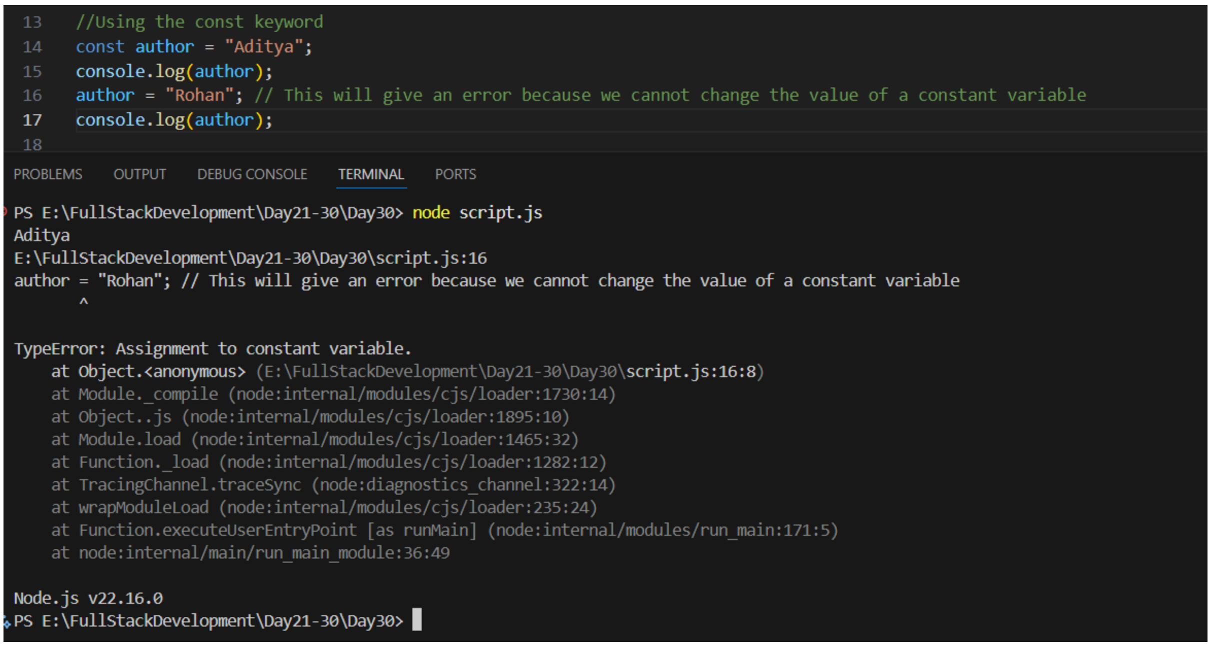
Example: use of const keyword



```javascript
//Using the const keyword
const author = "Aditya";
console.log(author);
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
Aditya
PS E:\FullStackDevelopment\Day21-30\Day30>
```

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



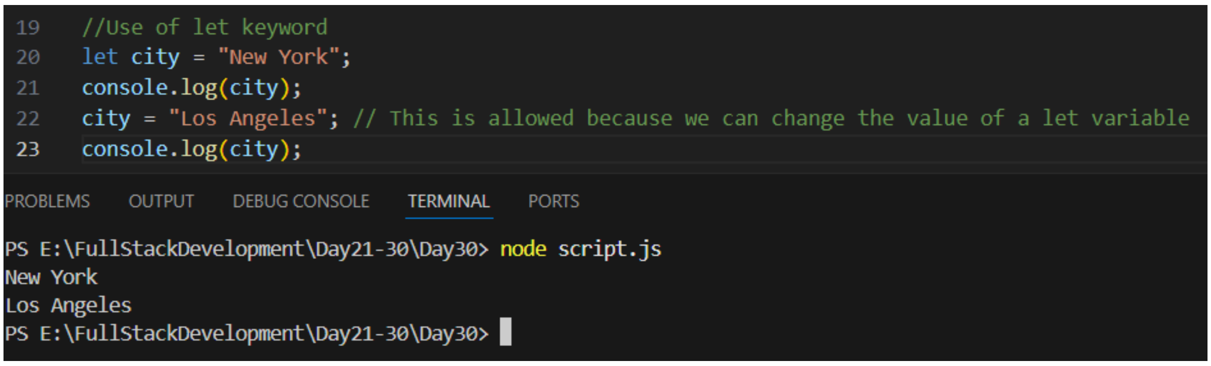
```javascript
13  //Using the const keyword
14  const author = "Aditya";
15  console.log(author);
16  author = "Rohan"; // This will give an error because we cannot change the value of a constant variable
17  console.log(author);
18
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
Aditya
E:\FullStackDevelopment\Day21-30\Day30\script.js:16
author = "Rohan"; // This will give an error because we cannot change the value of a constant variable
       ^

TypeError: Assignment to constant variable.
    at Object.<anonymous> (E:\FullStackDevelopment\Day21-30\Day30\script.js:16:8)
    at Module._compile (node:internal/modules/cjs/loader:1730:14)
    at Object..js (node:internal/modules/cjs/loader:1895:10)
    at Module.load (node:internal/modules/cjs/loader:1465:32)
    at Function._load (node:internal/modules/cjs/loader:1282:12)
    at TracingChannel.traceSync (node:diagnostics_channel:322:14)
    at wrapModuleLoad (node:internal/modules/cjs/loader:235:24)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:171:5)
    at node:internal/main/run_main_module:36:49

Node.js v22.16.0
PS E:\FullStackDevelopment\Day21-30\Day30>
```

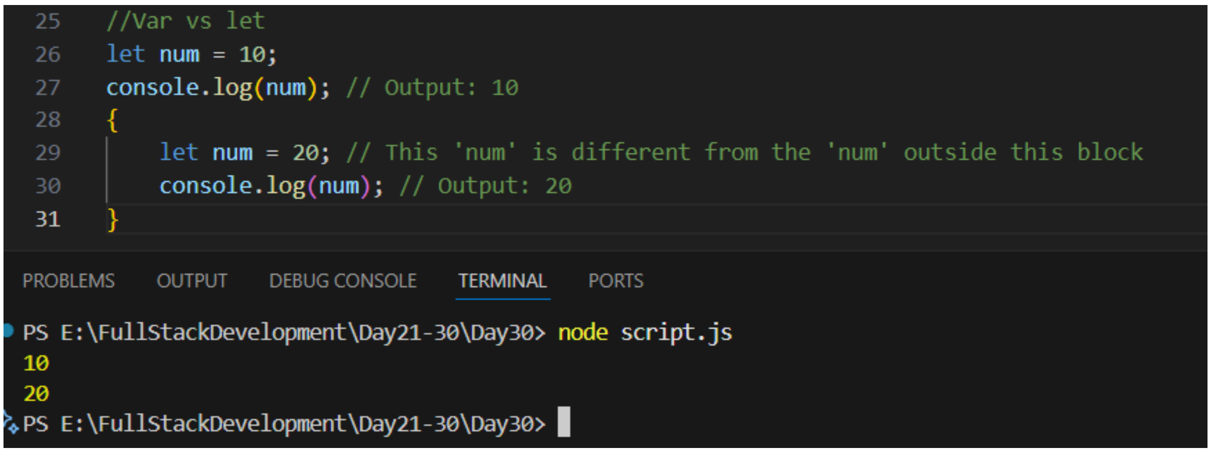
Example: using the let keyword



```javascript
//Use of let keyword
let city = "New York";
console.log(city);
city = "Los Angeles"; // This is allowed because we can change the value of a let variable
console.log(city);
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
New York
Los Angeles
PS E:\FullStackDevelopment\Day21-30\Day30>
```

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

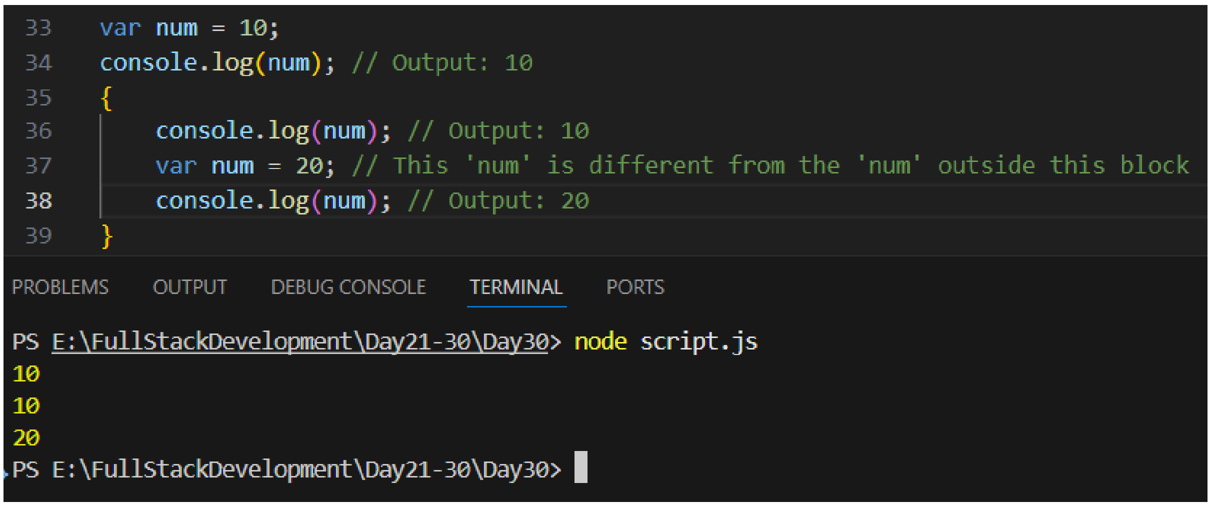


```
//Var vs let
let num = 10;
console.log(num); // Output: 10
{
    let num = 20; // This 'num' is different from the 'num' outside this block
    console.log(num); // Output: 20
}
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
10
20
PS E:\FullStackDevelopment\Day21-30\Day30>
```

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

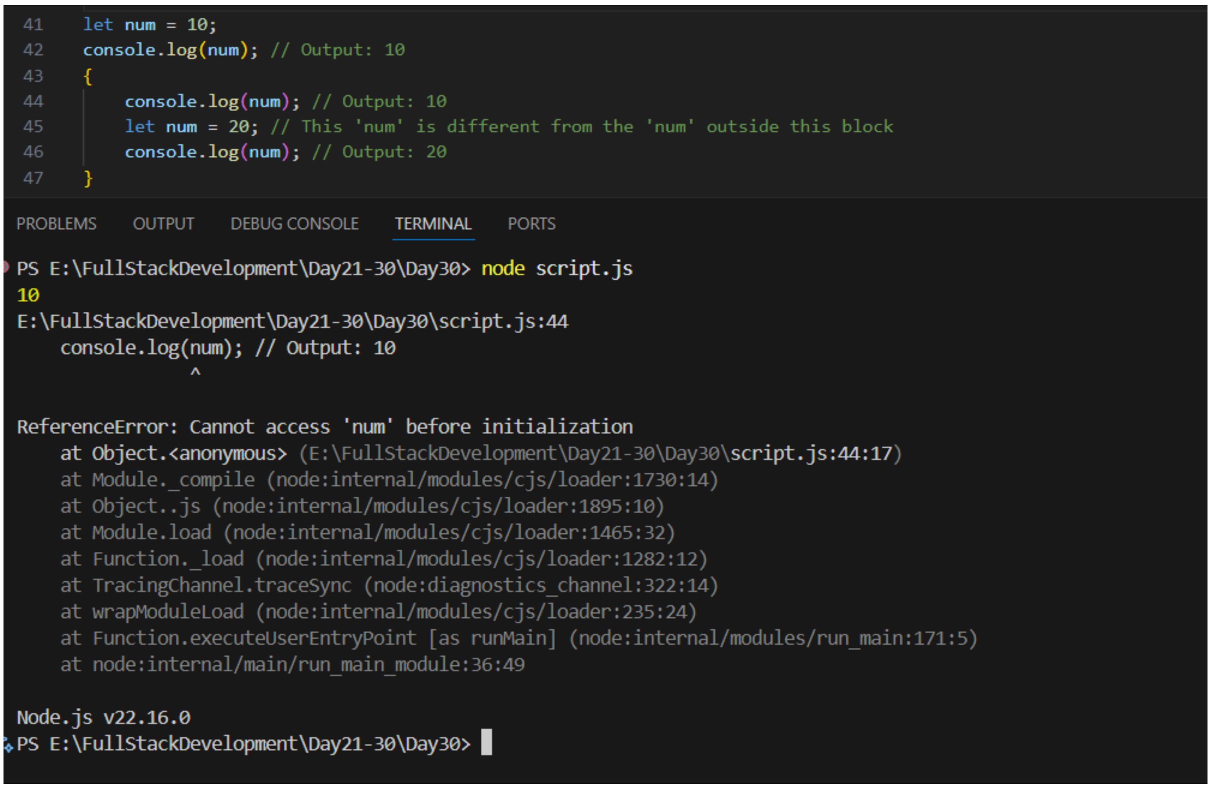
While in below example when we used var:



```
var num = 10;
console.log(num); // Output: 10
{
    console.log(num); // Output: 10
    var num = 20; // This 'num' is different from the 'num' outside this block
    console.log(num); // Output: 20
}
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
10
10
20
PS E:\FullStackDevelopment\Day21-30\Day30>
```

Also,



```
let num = 10;
console.log(num); // Output: 10
{
    console.log(num); // Output: 10
    let num = 20; // This 'num' is different from the 'num' outside this block
    console.log(num); // Output: 20
}
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
10
E:\FullStackDevelopment\Day21-30\Day30\script.js:44
    console.log(num); // Output: 10
            ^

ReferenceError: Cannot access 'num' before initialization
    at Object.<anonymous> (E:\FullStackDevelopment\Day21-30\Day30\script.js:44:17)
    at Module._compile (node:internal/modules/cjs/loader:1730:14)
    at Object..js (node:internal/modules/cjs/loader:1895:10)
    at Module.load (node:internal/modules/cjs/loader:1465:32)
    at Function._load (node:internal/modules/cjs/loader:1282:12)
    at TracingChannel.traceSync (node:diagnostics_channel:322:14)
    at wrapModuleLoad (node:internal/modules/cjs/loader:235:24)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:171:5)
    at node:internal/main/run_main_module:36:49

Node.js v22.16.0
PS E:\FullStackDevelopment\Day21-30\Day30>
```

<u>Data Types in JavaScript:</u>

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

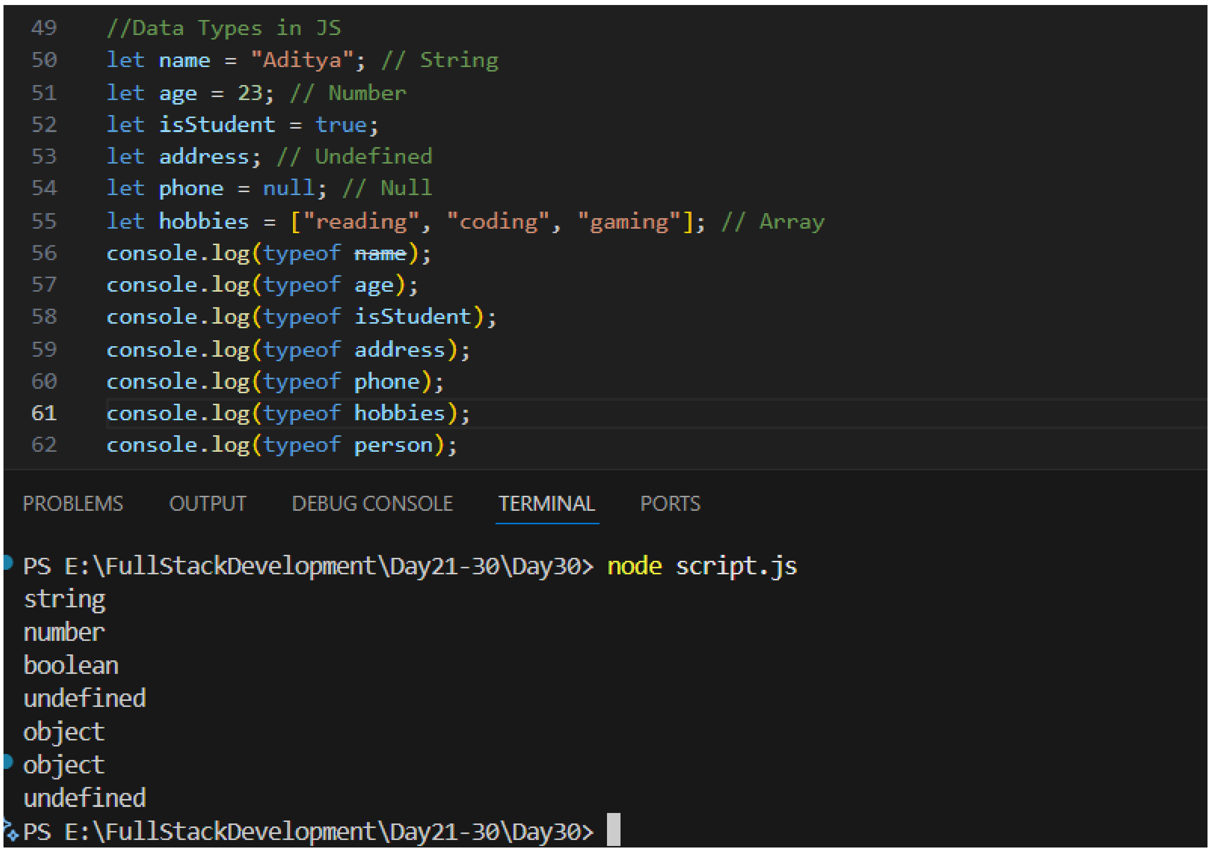
Data Type	Example	Description
Null	let y = null;	Empty or unknown value
Symbol	Symbol("id")	Unique identifiers (ES6)
BigInt	12345678901234567890n	For very large integers (ES2020)

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:

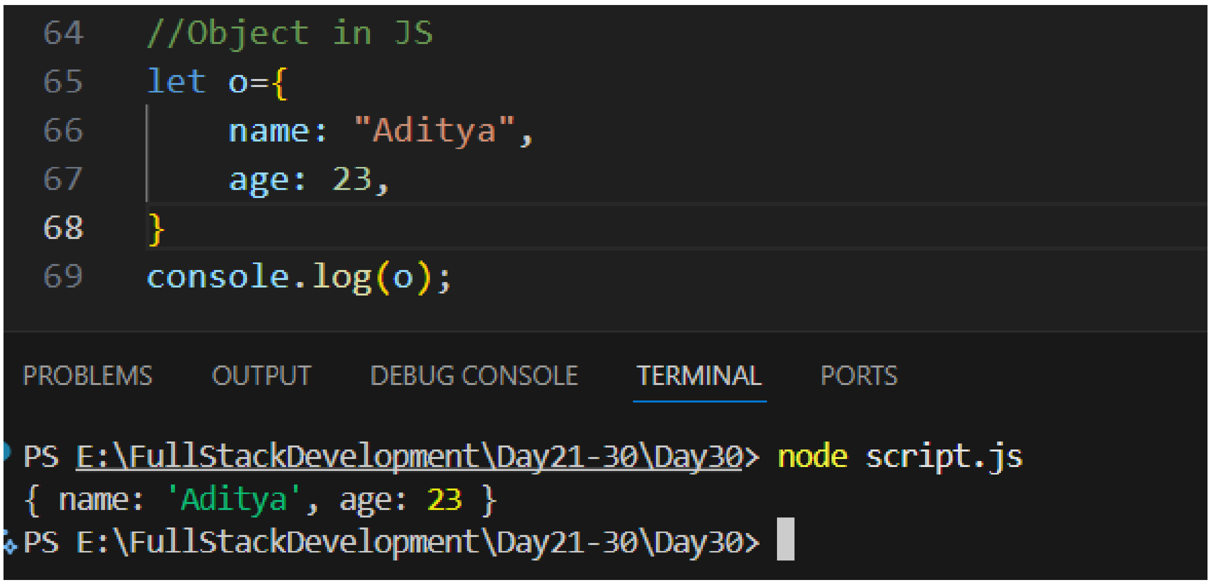


```
//Data Types in JS
let name = "Aditya"; // String
let age = 23; // Number
let isStudent = true;
let address; // Undefined
let phone = null; // Null
let hobbies = ["reading", "coding", "gaming"]; // Array
console.log(typeof ~~name~~);
console.log(typeof age);
console.log(typeof isStudent);
console.log(typeof address);
console.log(typeof phone);
console.log(typeof hobbies);
console.log(typeof person);
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
string
number
boolean
undefined
object
object
undefined
PS E:\FullStackDevelopment\Day21-30\Day30>
```

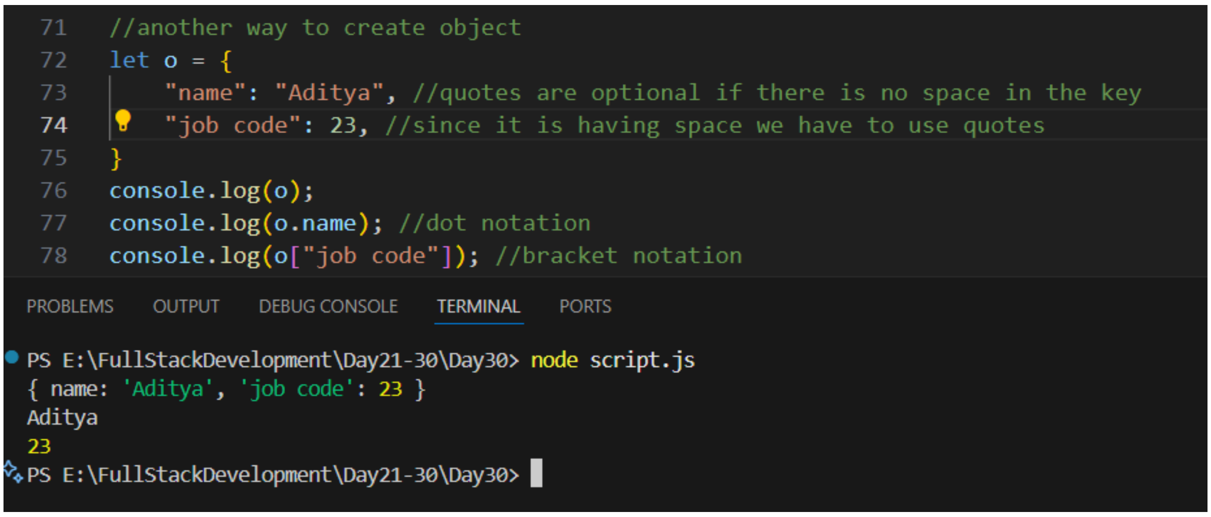
Note: data type of null is object.

Example: object in JS


```
//Object in JS
let o={
    name: "Aditya",
    age: 23,
}
console.log(o);
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

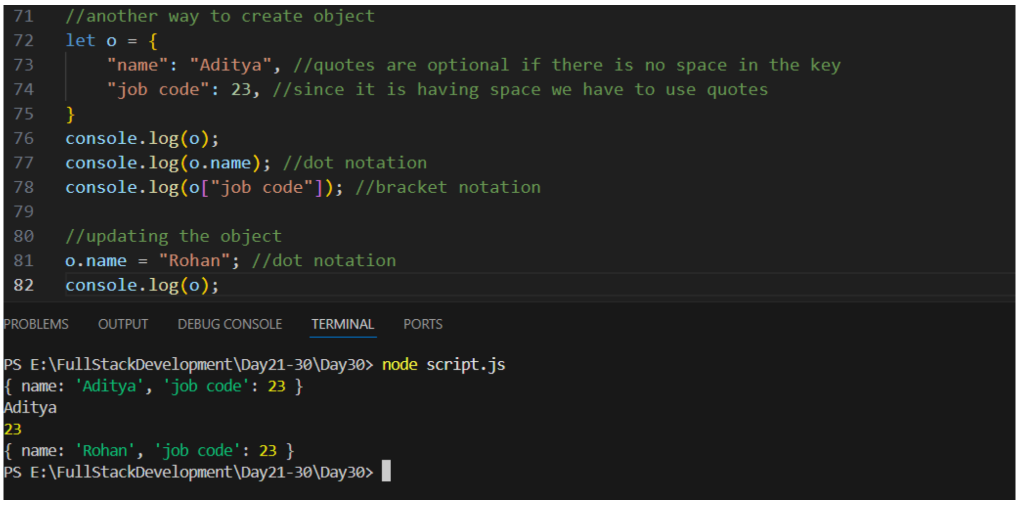
```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
{ name: 'Aditya', age: 23 }
PS E:\FullStackDevelopment\Day21-30\Day30>
```

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


```
//another way to create object
let o = {
    "name": "Aditya", //quotes are optional if there is no space in the key
    "job code": 23, //since it is having space we have to use quotes
}
console.log(o);
console.log(o.name); //dot notation
console.log(o["job code"]); //bracket notation
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
{ name: 'Aditya', 'job code': 23 }
Aditya
23
PS E:\FullStackDevelopment\Day21-30\Day30>
```

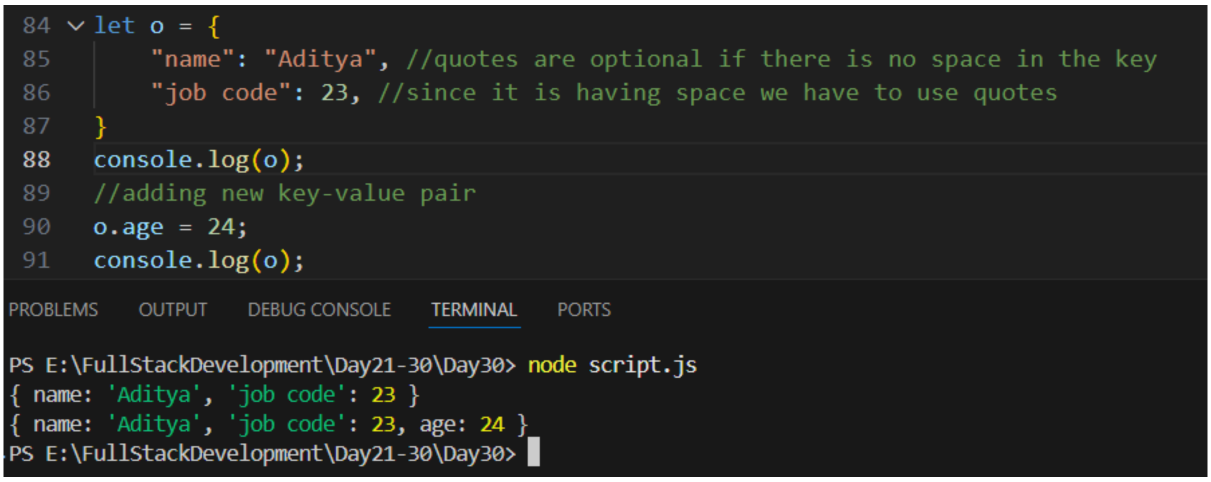
Example: another way to write.


```
//another way to create object
let o = {
    "name": "Aditya", //quotes are optional if there is no space in the key
    "job code": 23, //since it is having space we have to use quotes
}
console.log(o);
console.log(o.name); //dot notation
console.log(o["job code"]); //bracket notation

//updating the object
o.name = "Rohan"; //dot notation
console.log(o);
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
{ name: 'Aditya', 'job code': 23 }
Aditya
23
{ name: 'Rohan', 'job code': 23 }
PS E:\FullStackDevelopment\Day21-30\Day30>
```

Example: adding a new key-value pair



```
let o = {
    "name": "Aditya", //quotes are optional if there is no space in the key
    "job code": 23, //since it is having space we have to use quotes
}
console.log(o);
//adding new key-value pair
o.age = 24;
console.log(o);
```

```
PS E:\FullStackDevelopment\Day21-30\Day30> node script.js
{ name: 'Aditya', 'job code': 23 }
{ name: 'Aditya', 'job code': 23, age: 24 }
PS E:\FullStackDevelopment\Day21-30\Day30>
```

--The End--