Day 1

Programming:

What is programming?

Programming is the process of creating instructions that a computer can follow to perform specific tasks. It involves writing code using programming languages to solve problems, automate processes, and build applications or systems. Programmers design, test, and maintain software to make computers perform desired functions efficiently and accurately.

JavaScript:

What is JavaScript?

JavaScript is a high-level, interpreted programming language used to make web pages interactive and dynamic. It runs directly in web browsers, enabling features like animations, form validation, and real-time updates. Along with HTML and CSS, JavaScript is one of the core technologies of web development, powering both frontend and backend applications.

Here are the main features of JavaScript:

1. Lightweight and Interpreted – Executes directly in the browser without compilation.
2. Object-Oriented – Supports objects, inheritance, and encapsulation.
3. Dynamic Typing – No need to declare variable types.
4. Event-Driven – Responds to user actions like clicks or keypresses.
5. Platform Independent – Runs on any device with a browser.
6. Asynchronous and Single-Threaded – Uses callbacks, promises, and async/await for non-blocking operations.
7. Client-Side and Server-Side Support – Works in browsers and with Node.js on servers.
8. Prototype-Based – Inheritance is achieved through prototypes, not classes.

What is Node.js?

Node.js is an open-source, cross-platform runtime environment that allows developers to run JavaScript outside of a web browser. It uses the V8 JavaScript engine (from Google Chrome) to execute code efficiently. Node.js is widely used for building fast, scalable server-side applications, APIs, and real-time web services like chat apps.

Variables in JavaScript:

What is a variable?

A variable lets you store, use, and modify data in your code.

Syntax:

let name = "Aditya";

Ways to Declare Variables:

1. **var** – old method (function-scoped)
2. **let** – modern way (block-scoped, recommended)
3. **const** – for values that shouldn't change

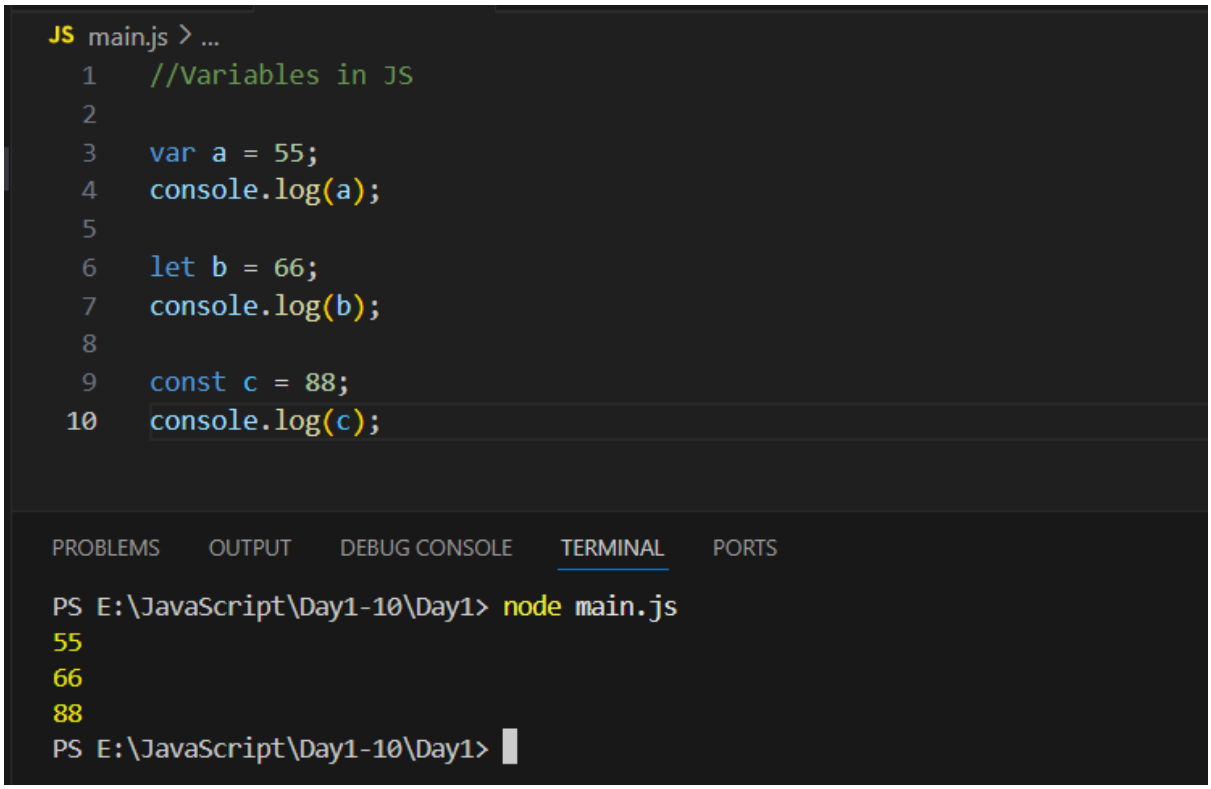
Feature / Keyword	**var**	**let**	**const**
Scope	Function-scoped	Block-scoped	Block-scoped
Redeclaration	Allowed	Not allowed in the same scope	Not allowed
Reassignment	Allowed	Allowed	**Not allowed** (value fixed)
Hoisting	Yes (initialized as undefined)	Yes (but not initialized)	Yes (but not initialized)
Default Value	undefined	undefined	Must be initialized at declaration
Use Case	Older code, or function-wide variables	Modern code, variables that change	Constants or values that never change
Example	var x = 10;	let y = 20;	const z = 30;

Rules for writing the variable:

Rule	**Explanation**	**Example**
1. Must start with a letter, underscore (_), or dollar sign ($)	Variable names cannot start with a number.	let name = "Aditya";, let _age = 20;, let $price = 100;, let 1name = "John";
2. Can contain letters, numbers, underscores, or dollar signs	But cannot include spaces or special characters.	let user1 = "Aman";, let user-name = "Aman";

Rule	Explanation	Example
3. Case-sensitive	age and Age are treated as two different variables.	let age = 18;, let Age = 25;
4. Cannot use JavaScript reserved keywords	Words like var, let, for, if, function etc. cannot be variable names.	let for = 10;
5. Should be meaningful	Choose names that describe the stored value clearly.	let userName = "Aditya";, let x = "Aditya";
6. Use camelCase for multi-word names	Common JavaScript naming convention.	let totalAmount = 500;

Example: first code/ declaring variables



const, let and var in JavaScript

Important point about var: var is globally scoped, var can be updated and redeclared within its scope

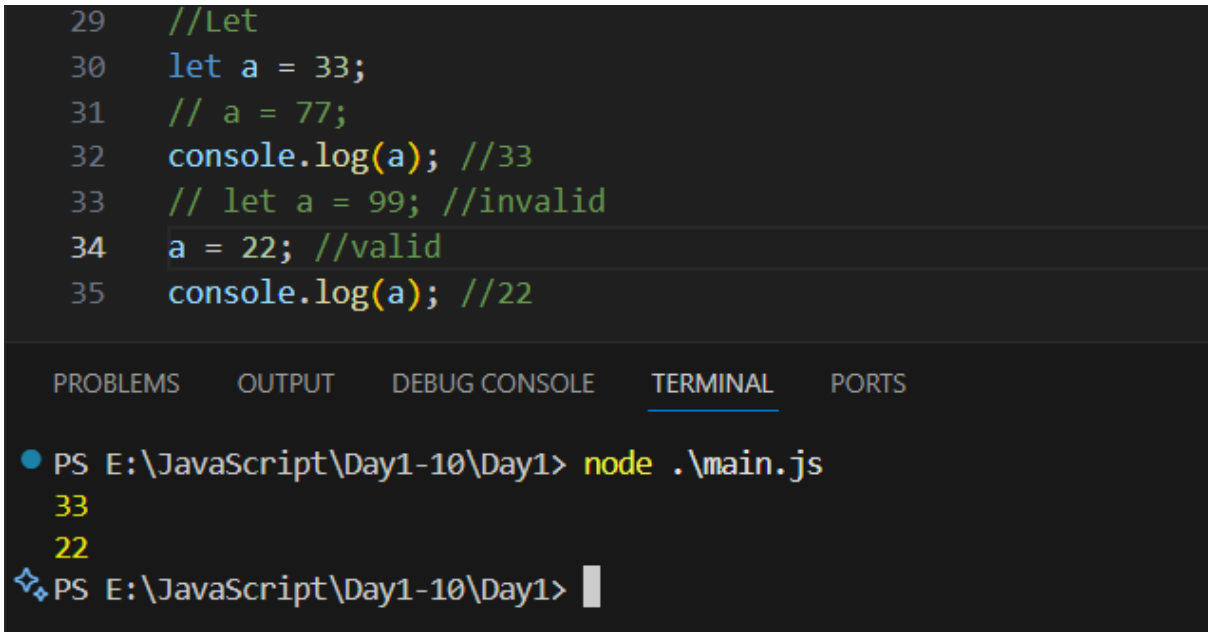
```
12  //Var vs let vs const
13
14  var a = 33;
15  console.log(a); //33
16
17  {
18      console.log(a); //33
19      var a = 22;
20      console.log(a); //22
21  }
22
23  console.log(a); //22
24
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\JavaScript\Day1-10\Day1> node .\main.js
33
33
22
22
```

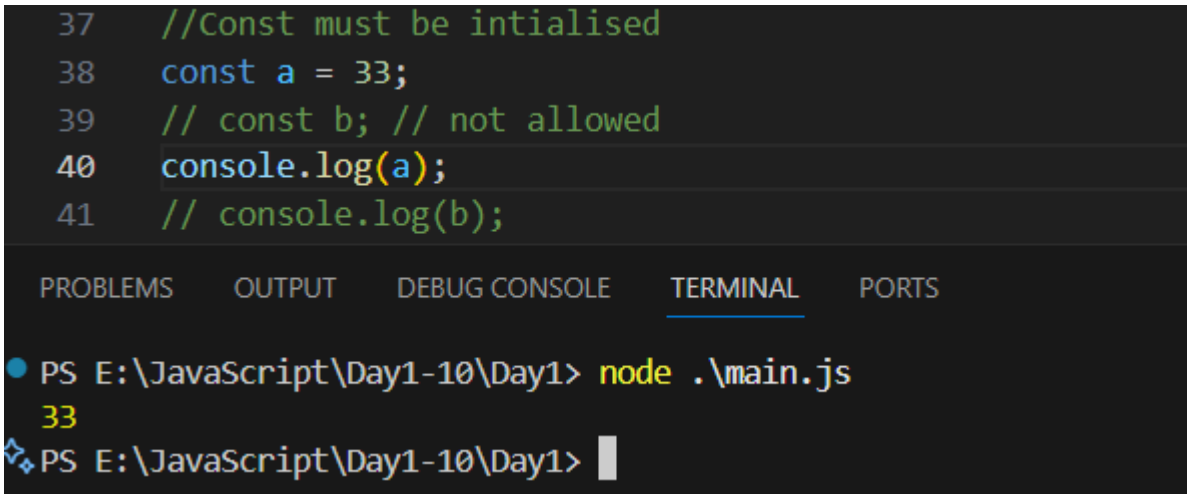
```
14  var a = 33;
15  console.log(a); //33
16
17  {
18      console.log(a); //33
19      var a = 22;
20      console.log(a); //22
21      var a = 1;
22      console.log(a); //1
23  }
24
25  console.log(a); //22
26
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\JavaScript\Day1-10\Day1> node .\main.js
33
33
22
1
1
PS E:\JavaScript\Day1-10\Day1>
```

Important point about let: let can be updated but not redeclared. Let has block scope.



Also, const must be initialised:



Also, var and let can be left uninitialized

```
43   // Let and var can be left uninitialised
44   let a = 33; //as usual
45   let b; //can be left like so
46   var c = 66; //as usual
47   var d; //can be left like so
48   console.log(a);
49   console.log(b);
50   console.log(c);
51   console.log(d);
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\JavaScript\Day1-10\Day1> node .\main.js
33
undefined
66
undefined
PS E:\JavaScript\Day1-10\Day1>
```

Primitives and Objects in JavaScript:

What are Data Types in JavaScript?

JavaScript data types are mainly divided into two categories:

Category	Types Included
Primitive Types	Number, String, Boolean, Undefined, Null, Symbol, BigInt
Non-Primitive (Object) Types	Object, Array, Function, Date, etc.

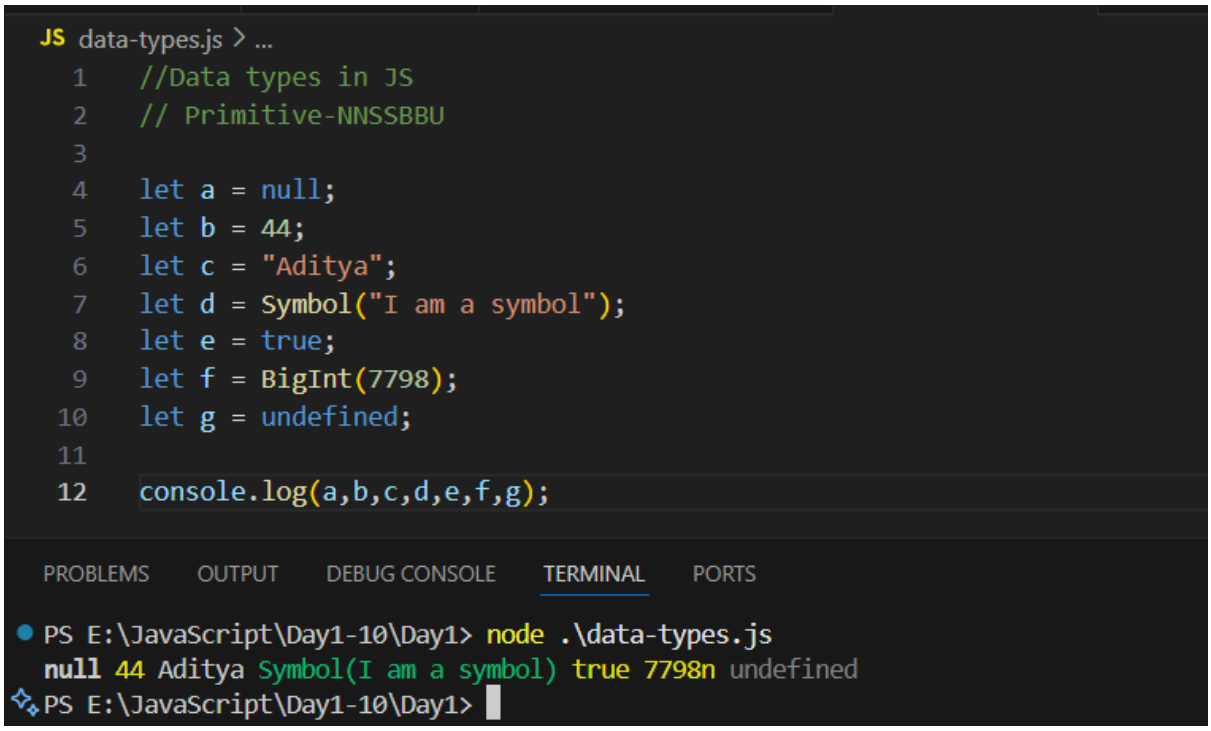
Primitive Data Types:

Primitive types are basic, single-value data types. They are immutable (cannot be changed directly) and copied by value.

Type	Example	Description
Number	let age = 25;	Represents numeric values (integers or decimals).
String	let name = "Aditya";	Sequence of characters inside quotes.
Boolean	let isActive = true;	Represents true or false.
Undefined	let x;	Variable declared but not assigned a value.

Type	Example	Description
Null	let emptyValue = null;	Represents intentional absence of value.
Symbol	let id = Symbol("123");	Represents unique and immutable identifiers.
BigInt	let bigNum = 123456789012345678901234567890n;	Used for very large integers beyond Number limit.

Example: primitive data type



Example: use of typeof

Objects in JavaScript

Objects are collections of key–value pairs. They are non-primitive and copied by reference (not by value).

A very basic object:

```
25  // Object in JS
26  let obj = {
27      "name": "Aditya",  // put in quotes because it's a string
28      "class": "UKG",    // put in quotes too
29      "age": 6
30  };
31
32  // Accessing values
33  console.log(obj.name);     // Way 1 - dot notation
34  console.log(obj["name"]);  // Way 2 - bracket notation
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\JavaScript\Day1-10\Day1> node .\data-types.js
Aditya
Aditya
PS E:\JavaScript\Day1-10\Day1>
```

Example: printing something which is not in the object will return undefined

```
25  // Object in JS
26  let obj = {
27      "name": "Aditya",  // put in quotes because it's a string
28      "class": "UKG",    // put in quotes too
29      "age": 6
30  };
31
32  // Accessing values
33  console.log(obj.finance); //will print undefined
34
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\JavaScript\Day1-10\Day1> node .\data-types.js
undefined
PS E:\JavaScript\Day1-10\Day1>
```

--The End--