Chapter 02: JavaScript Variables

---

1. What is a Variable?

Conceptual Definition

A variable is just like a **container** in which we can store data during the execution of a program.

Real Definition

A variable is a **name of a memory location** in which we store data during the execution of the program, and we can use that data later whenever we need it.

> 👉 Variable = Name + Memory Location + Stored Value

---

2. Why Do We Need Variables?

- To store data temporarily while a program is running
- To reuse values without writing them again and again
- To perform operations on stored data

Storing Primitive Data Types in a Variable

Definition

A **primitive data type** is a data type that is **not an object** and has **no methods**. It is directly stored in a variable.

It always accupied a space in stack of a memory(RAM)

it's size is fixed so it's size is not changes during run time

---

Example

1) Number

```
let year = 2025;
console.log(year);
console.log(typeof year); // number
```

2) String

```
let statement = "I belong to ";
let city = `${statement}Delhi`;
console.log(city);
console.log(typeof city); // string
```

3) Boolean

Boolean represents only two possible values: **true** or **false**. It is used to make decisions in a program.

```
let isOnline = true;
console.log(isOnline);
console.log(typeof isOnline); // boolean
```

4) Null

null is a special value in JavaScript that represents "nothing", "empty", or "no value".

```
let nothing = null;
console.log(nothing);
console.log(typeof nothing); // object (exception)
```

5) Undefined

```
let unknown;
```

```
console.log(unknown); // undefined
```

6) Symbol

```
let sym = Symbol("a");
console.log(sym);
```

7) BigInt

```
let a = 125n;
console.log(typeof a); // bigint
```

---

5. Ways to Declare a Variable

We can declare a variable using the following ways:

- **let**
- **var**
- **const**
- Without any keyword (not recommended)

var keyword and declaring variables without any keyword are older ways to declare variables.

> In modern JavaScript, we use **let** and **const** to declare variables.

---

6. Scope of Variables

Scope defines **where a variable can be accessed or used** in a JavaScript program. In simple words, it tells us *which part of the code can see the variable*.

> 👉 Variable scope decides the **lifetime** and **visibility** of a variable.

Types of Scope in JavaScript

- Global Scope
- Block Scope
- Function Scope

---

1) Global Scope

A variable declared **outside of all functions and blocks** is called a global variable. It can be accessed from **anywhere** in the program.

> Global variables stay in memory for the entire program execution.

Case 1: All variables in Global Scope

```
// case1: all variables in global scope
let a = 9;
var b = 10;
const c = 10;

function fun() {
  // accessing from local scope
  console.log(a, b, c);

  if (true) {
    // accessing from block scope
    console.log(a, b, c);
  }
}

fun();

// accessing from global scope
console.log(a, b, c);
```

> **Conclusion:**
> Global variables (**let**, **var**, **const**) are accessible from **any scope** of a program (global, function, and block).

---

2) Block Scope

Block scope means variables declared **inside any block** `{ }` (other than function block) using **let** or **const**.

```
function fun() {

  if (true) {
    let a = 10;
    var b = 11;
    const c = 18;

    // accessing block variables from block scope
    console.log(a, b, c);
  }

  // accessing block variables from local (function) scope
  // console.log(a); // Error
  console.log(b); // Accessible (var)
  // console.log(c); // Error
}

fun();

// accessing block variables from global scope
// console.log(a); // Error
// console.log(b); // Error
// console.log(c); // Error
```

Conclusion:

- **let** and **const** are strictly **block-scoped**
- **var** is **not block-scoped**
- If **var** is inside a block:
  - If the block is inside a function → it becomes **local**

- If the block is not inside a function → it becomes **global**

---

3) Function Scope (Local Scope)

A variable declared **inside a function** is called a **function-scoped (local) variable**. Such variables are accessible only **inside that function**.

```
// example: all variables declared in local scope
function fun() {
  let a = 9;
  var b = 10;
  const c = 11;

  // accessing local variables from local scope
  console.log(a, b, c);

  if (true) {
    // accessing local variables from block scope
    console.log(a, b, c);
  }
}

fun();

// accessing local variables from global scope
// console.log(a, b, c); // Error
```

Conclusion:
Local variables (**let**, **var**, **const**) are accessible **only inside the function** in which they are declared.
