JavaScript Variables

Introduction

Variables and data types are **foundational concepts** in programming. They are the building blocks for storing, organizing, and manipulating information. In JavaScript:

- **Variables** act like containers that hold values.
- **Data types** define the kind of values those containers can store.

A strong understanding of these concepts is essential for writing code that is **clear, efficient, and bug-free**.

Variables in JavaScript

A **variable** is a named storage for data. In JavaScript, variables are declared using three keywords: **var**, **let**, and **const**.

var Keyword

- Introduced in early JavaScript.
- **Function-scoped** or **globally-scoped**.
- Can be **redeclared** and **reassigned**.

```
var n = 5;
console.log(n); // 5

var n = 20; // redeclaration allowed
console.log(n); // 20
```

Because of its scoping issues, var is less commonly used in modern JavaScript.

---

let Keyword

- Introduced in **ES6 (2015)**.
- **Block-scoped** (limited to { }).

JavaScript Variables

- Can be **reassigned**, but **not redeclared** in the same scope.

let n = 10;

n = 20; // allowed

// let n = 15; // ❌ error: cannot redeclare

console.log(n); // 20

Use let when you expect the value to change.

---

const Keyword

- Introduced in **ES6**.
- **Block-scoped**.
- Cannot be **reassigned** or **redeclared**.

const n = 100;

// n = 200; // ❌ error: assignment not allowed

console.log(n); // 100

Use const for values that should never change.

Data Types in JavaScript

JavaScript supports two broad categories of data types:

1. **Primitive Data Types** (simple, immutable values)
2. **Non-Primitive Data Types** (objects, collections, functions)

Primitive Data Types

1. **Number**
   Represents integers and floating-point numbers.

   let age = 42;

JavaScript Variables

let pi = 3.14;

2. **String**
   Represents text, enclosed in quotes.

   let greeting = "Hello, World!";

3. **Boolean**
   Represents logical values: true or false.

   let isActive = true;

4. **Undefined**
   A variable declared but not assigned a value.

   let notAssigned;

   console.log(notAssigned); // undefined

5. **Null**
   Represents intentional absence of value.

   let empty = null;

6. **Symbol**
   Represents unique, immutable values (often used as object keys).

   let sym = Symbol('unique');

7. **BigInt**
   Represents very large integers beyond Number.MAX_SAFE_INTEGER.

   let bigNumber = 1234567890123456789012345678901234567890n;

Non-Primitive Data Types

1. **Object**
   Stores key-value pairs.

   let person = { name: "Amit", age: 25 };

2. **Array**
   Stores ordered lists of values.

```
let colors = ["red", "green", "blue"];
```

3. **Function**
   A reusable block of code.

```
function greet() {
   console.log("Hello, JavaScript!");
}
```

Exploring Common Expressions in JavaScript

JavaScript sometimes behaves in surprising ways due to **type coercion** and **object references**. Let's explore:

Expression	Result	Explanation
null === undefined	false	Different types: null is intentional absence, undefined means not assigned.
5 > 3 > 2	false	Evaluated left-to-right: 5 > 3 → true → true > 2 → 1 > 2 → false.
[] === []	false	Arrays are objects; each [] is a new reference.
"10" < "9"	true	String comparison is lexicographic: "1" is less than "9".
NaN === NaN	false	NaN is never equal to itself (IEEE 754 standard). Use Number.isNaN().
true == 1	true	Loose equality (==) coerces true to 1.
undefined > 0	false	undefined becomes NaN, and comparisons with NaN are false.
"5" === 5	false	Strict equality (===) checks type and value.
[1, 2] == [1, 2]	false	Arrays are compared by reference, not content.
Infinity > 1000	true	Infinity is greater than any finite number.

Best Practices

JavaScript Variables

- Use **const** by default, and **let** when reassignment is needed.
- Avoid var unless working with legacy code.
- Use **strict equality (===)** to avoid unexpected type coercion.
- Comment tricky expressions for clarity.
- Remember: **primitive values are compared by value, objects by reference**.

Summary

- **Variables** are containers for data, declared with var, let, or const.
- **Data types** define the kind of data stored:
    - **Primitive**: Number, String, Boolean, Undefined, Null, Symbol, BigInt.
    - **Non-Primitive**: Object, Array, Function.
- JavaScript has **quirks** due to type coercion and reference comparison.
- Writing clean, predictable code requires understanding these fundamentals.