Variables and Data Types in JavaScript

What is a Variable?

A **variable** is a named container for storing data. In JavaScript, we can declare variables using:

```
let name = "Harry";
const age = 25;
var city = "Delhi";
```

`let` vs `const` vs `var`

Keyword	Reassignable?	Block Scoped?	Hoisted?
`let`	Yes	Yes	Yes
`const`	No	Yes	Yes
`var`	Yes	No	Yes (but undefined)

Use `let` when:

- You plan to **change the value** later.

```
let score = 0;
score = 10;
```

Use `const` when:

- The value **should not change**.

```
const pi = 3.14159;
// pi = 3.14; Error
```

Avoid `var`

- It behaves inconsistently due to hoisting and lack of block scoping.

JavaScript Data Types

JavaScript has two categories of data types:

---

1. Primitive (Value) Data Types

These are **immutable** and **stored directly in memory**.

Data Type	Example
`string`	`"Hello World"`
`number`	`42` , `3.14` , `-100`
`boolean`	`true` , `false`
`null`	`null`
`undefined`	`undefined`
`bigint`	`1234567890123456789On`
`symbol`	`Symbol("id")`

```
let name = "Harry";          // string
let age = 25;                // number
let isCool = true;           // boolean
let noValue = null;          // null
let notDefined;              // undefined
```

Note: `typeof null` is `"object"` due to a long-standing bug in JS.

---

2. Non-Primitive (Reference) Data Types

These hold **references** to memory, not actual values.

Type	Example
Object	`{ name: "Harry" }`
Array	`[1, 2, 3]`
Function	`function() {}`
Date, RegExp, etc.	Built-in Objects

```
let person = { name: "Harry", age: 25 }; // Object
let colors = ["red", "blue", "green"];   // Array
let greet = function() { console.log("Hi") }; // Function
```

Differences Between Primitive and Reference Types

Feature	Primitive	Reference
Stored as	Value	Memory address (reference)
Mutable?	Immutable	Mutable
Copied as	Value	Reference

```
let a = 10;
let b = a; // Copy by value
b = 20;
console.log(a); // 10 (unchanged)

let obj1 = { x: 1 };
```

```javascript
let obj2 = obj1; // Copy by reference
obj2.x = 2;
console.log(obj1.x); // 2 (both point to same object)
```

`typeof` Operator

Use `typeof` to check the type of a variable:

```javascript
console.log(typeof "Hello");      // string
console.log(typeof 100);          // number
console.log(typeof true);         // boolean
console.log(typeof undefined);    // undefined
console.log(typeof null);         // object (quirk!)
console.log(typeof {});           // object
console.log(typeof []);           // object (array is also object)
console.log(typeof function(){});// function
```

Summary

- Use `let` for changeable values, `const` for constants.
- Understand the difference between **primitive** (copied by value) and **reference** types (copied by reference).
- `typeof` is useful but not perfect (e.g., `typeof null === "object"`).