Primitive data types (string, number, boolean, null, undefined, symbol, bigint)

In JavaScript, **primitive data types** are the most basic building blocks of data. They are **immutable**, meaning their values cannot be changed once created. They are **stored directly in memory** and are compared by value.

There are **7 primitive data types** in JavaScript:

🔷 1. String

Represents a sequence of characters, used for text.

```
let name = "Abhi";
let greeting = 'Hello';
let template = `Hi, ${name}`;
```

- Can be declared using **single quotes**, **double quotes**, or **template literals** (backticks).
- Template literals allow **string interpolation** and multi-line strings.

🔷 2. Number

Represents both **integers** and **floating-point** numbers.

```
let age = 25;
let price = 99.99;
let negative = -100;
```

- JavaScript has only one number type (`IEEE 754` double precision).
- Special numeric values:

- `Infinity`, `Infinity`
- `NaN` (Not a Number)

```
console.log(10 / 0);      // Infinity
console.log("abc" - 1);   // NaN
```

🔷 3. Boolean

Represents a logical entity and can have only two values:

```
let isLoggedIn = true;
let isAdmin = false;
```

- Often used in conditions:

```
if (isLoggedIn) {
  console.log("Welcome!");
}
```

🔷 4. Null

Represents the **intentional absence** of any value.

```
let user = null;
```

- It's a primitive type but **typeof null === "object"** (this is a well-known JavaScript bug).

🔷 5. Undefined

A variable that has been **declared but not assigned** a value is `undefined`.

```
let score;
console.log(score); // undefined
```

- It’s also the default return value of functions that don't return anything.

🔷 6. Symbol (ES6)

Used to create **unique identifiers** for object properties.

```
let sym1 = Symbol("id");
let sym2 = Symbol("id");
console.log(sym1 === sym2); // false
```

- Even with the same description, symbols are always unique.
- Useful in situations like creating private object keys.

🔷 7. BigInt (ES11)

Used to represent **integers larger than** `2^53 - 1`.

```
let big = 12345678901234567890123456789012345678901234567890n;
```

- Note the `n` at the end — it’s mandatory.
- You can perform arithmetic with BigInts:

```
let a = 10n;
let b = 20n;
console.log(a + b); // 30n
```

🧠 Summary Table

Type	Example	`typeof` Result
String	`"Hello"`	`string`
Number	`42` , `3.14` , `NaN`	`number`
Boolean	`true` , `false`	`boolean`
Null	`null`	`object` (bug)
Undefined	`undefined`	`undefined`

Symbol	`Symbol("id")`	`symbol`
BigInt	`12345678901234567890n`	`bigint`

✅ Key Characteristics of Primitive Types

- Immutable: You cannot change the value — you can only **reassign**.
- Stored by **value**, not by reference.
- Compared using `===` or `==` by their actual **value**.

Let me know if you want visual diagrams or want to see how these differ from reference types!