

# Reference types vs primitive types

In JavaScript, data types are categorized into two major groups:

---

## 1. Primitive Types

## 2. Reference Types (Non-Primitive)

---

### ◆ Primitive Types

| Stored by value directly in the memory location associated with the variable.

### ✓ Characteristics:

- **Immutable**
- Stored in the **stack** memory
- Compared by **value**
- Examples:
  - `string`, `number`, `boolean`, `null`, `undefined`, `symbol`, `bigint`

### Example:

```
let a = 10;  
let b = a;  
b = 20;  
console.log(a); // 10 (a is unaffected)
```

| a and b hold independent values — changing b does not affect a.

---

### ◆ Reference Types

Stored by reference. The variable holds a reference (address) to the actual object in memory.

## ✓ Characteristics:

- **Mutable**
- Stored in the **heap**, reference stored in the **stack**
- Compared by **reference (memory address)**
- Examples:
  - `Object` , `Array` , `Function` , `Date` , `RegExp` , `Map` , `Set` , etc.

## 📦 Example:

```
let obj1 = { name: "Abhi" };  
let obj2 = obj1;  
  
obj2.name = "Gurjar";  
console.log(obj1.name); // "Gurjar"
```

Both obj1 and obj2 point to the same object in memory.

## vs Comparison Table

Feature	Primitive Types	Reference Types
Stored in	Stack	Heap (reference in stack)
Stored as	Value	Reference (address)
Mutable?	No	Yes
Compared by	Value	Reference
Memory size	Fixed	Dynamic
Examples	<code>string</code> , <code>number</code> , etc	<code>object</code> , <code>array</code> , etc

## 🔍 Example: Comparison Behavior

```
let x = 100;
let y = 100;
console.log(x === y); // true (same value)

let arr1 = [1, 2];
let arr2 = [1, 2];
console.log(arr1 === arr2); // false (different references)

let arr3 = arr1;
console.log(arr1 === arr3); // true (same reference)
```

---

## 💡 Key Takeaways

- **Primitive types are simple values** — copied and compared by value.
- **Reference types are complex** — copied and compared by reference.
- If you assign one object/array to another, both point to the **same location** in memory.

---

Let me know if you want illustrations for stack vs heap or more examples!