

## ✓ Symbol

- Unique, immutable primitive useful for object keys

```
const sym = Symbol('id');
```

## What is a Symbol?

A **Symbol** is a **primitive data type** introduced in **ES6**. Every **Symbol** is **unique and immutable**, even if they have the same description.

Used mostly as **unique property keys** to avoid name conflicts.

---

## ✓ Create a Symbol

```
const sym1 = Symbol("id");
const sym2 = Symbol("id");

console.log(sym1 === sym2); // ✖ false (each symbol is unique)
```

---

## ✓ Using Symbol as a Unique Object Key

```
const user = {
  name: "Alice",
  [Symbol("id")]: 101 // hidden-like unique key
};

console.log(user); // { name: "Alice", [Symbol(id)]: 101 }
```

🔒 Symbol keys don't show up in:

- `for...in`

- `Object.keys()`
  - `JSON.stringify()`
- 

## ✓ Accessing Symbol Key

```
const id = Symbol("id");
const user = {
  name: "Bob",
  [id]: 123
};

console.log(user[id]); // ♦ 123
```

---

## 🧠 Why Use **Symbol**?

- Avoid property name clashes in objects
- Create hidden-like private properties (not truly private, but protected)
- Often used in libraries/frameworks (e.g., `Symbol.iterator`, `Symbol.toStringTag`)



## ✓ Built-in Symbols (Advanced Usage)

```
const obj = {
  [Symbol.toPrimitive]() {
    return 42;
  }
};

console.log(+obj); // ♦ 42
```

---

## Summary

Feature	Symbol
Type	Primitive
Unique	 Always unique
Use case	Hidden keys, safe extensions
Shown in loops	 Hidden