//so they behave diffrently in comparison and arithmetic operation

// NOTE

// 1.null > 0 → false

// JavaScript converts null to 0 when using relational operators like >, <, >=, <=.

// So it becomes: 0 > 0 → false

// 2. null == 0 → false

// Here, JavaScript uses loose equality (==), but null is only equal to:

// undefined

// So: null == 0 → false (no coercion to number here)

// 3. null >= 0 → true

// Again, null is coerced to 0, so:

// 0 >= 0 → true

What is Type Coercion?

Type coercion is JavaScript's automatic **conversion of one data type to another** when different types are compared or operated on.

🧠 **Two Types of Coercion:**

Type	When It Happens	Example
Implicit	JS automatically converts types	"5" * 2 → 10
Explicit	You manually convert types	Number("5") → 5

🔁 **Examples of Implicit Coercion:**

➤ **In Comparisons:**

js

CopyEdit

"5" == 5 // true (string converted to number)

null == 0 // false (no coercion here!)

undefined == null // true (they are only loosely equal to each other)

➤ **In Arithmetic:**

js

CopyEdit

"5" - 1 // 4 (string converted to number)

"5" + 1 // "51" (number converted to string)

true + 1 // 2 (true → 1)

false * 5 // 0 (false → 0)

⚠️ **Why it's Confusing**

- Operators like +, ==, <, >= all behave **differently** depending on the **types involved**.
- The same value (null, for example) behaves differently with == vs >=.

💡 **Tip to Avoid Confusion**

✅ Use:

- === and !== (no coercion)
- Explicit conversions: Number(), String(), Boolean()
