

Diferenças entre =, == e === em JavaScript

Vamos ver as diferenças entre =, == e === 📌

🟡 1 = (atribuição)

👉 Usado para atribuir um valor a uma variável.
Não faz comparação, **apenas guarda um valor**.

```
let x = 10;          // x recebe 10
let nome = "Ana";    // nome recebe "Ana"
```

📖 **Leitura:**

“x recebe 10”
“nome recebe ‘Ana’”

🟢 2 == (comparação não estrita)

👉 Compara **somente os valores**,
mas **não compara o tipo** (faz conversão automática de tipo — *type coercion*).

```
5 == '5'    // true (mesmo valor, tipos diferentes)
0 == false  // true (JavaScript converte false em 0)
null == undefined // true
```

⚠️ Isso pode gerar **confusão**, porque ele tenta converter os tipos antes de comparar.

=== (comparação estrita)

👉 Compara **valor e tipo** ao mesmo tempo.
Nada de conversão automática!

```
5 === '5'    // false (número ≠ string)
```

```
5 === 5      // true
```

```
0 === false  // false (tipos diferentes)
```

```
null === undefined // false
```

Boa prática:

Sempre prefira usar === e !== (em vez de == e !=)
para evitar resultados inesperados com conversão de tipo.

Aqui vai um **exemplo prático completo** que você pode colar direto no **console do navegador** ou no VS Code para ver o resultado passo a passo 📌

```
// -----  
// Demonstração de =, == e ===  
// -----  
  
// 1 "=" -> atribuição  
let a = 10;           // Atribui o valor 10 à variável 'a'  
console.log("a =", a); // Resultado: 10  
  
// 2 "==" -> compara apenas o valor  
console.log(5 == '5'); // true → compara só o valor (5 é igual  
a '5')  
console.log(0 == false); // true → converte false em 0 antes de  
comparar  
console.log(null == undefined); // true → são tratados como  
equivalentes  
  
// 3 "===" -> compara valor e tipo  
console.log(5 === '5'); // false → número ≠ string  
console.log(0 === false); // false → número ≠ booleano  
console.log(10 === 10); // true → mesmo valor e mesmo tipo  
  
// 4 Mostrando as diferenças lado a lado  
console.log("5 == '5' →", 5 == '5'); // true  
console.log("5 === '5' →", 5 === '5'); // false  
console.log("true == 1 →", true == 1); // true  
console.log("true === 1 →", true === 1); // false
```

Vamos ver agora o **par inverso** dos operadores: `!=` e `!==`

Eles servem para **verificar se dois valores são diferentes** — mas com a mesma lógica do `==` e `===`.

Diferença entre `!=` e `!==`

Operador	Nome	Compara valor?	Compara tipo?
<code>!=</code>	Diferente solto	✓	✗
<code>!==</code>	Diferente estrito	✓	✓

Exemplo prático completo

Cole este código no **console do navegador** e veja o resultado linha por linha 

```
// -----  
// Demonstração de != e !==  
// -----  
  
// 1 "!=" -> compara apenas o valor (sem olhar o tipo)  
console.log(5 != '5');      // false → mesmo valor, JS converte  
                             // tipos  
console.log(0 != false);    // false → 0 é considerado igual a  
                             // false  
console.log(10 != 5);       // true  → valores realmente diferentes  
console.log(null != undefined); // false → são tratados como  
                             // equivalentes  
  
// 2 "!== " -> compara valor e tipo (estrito)  
console.log(5 !== '5');     // true  → número ≠ string  
console.log(0 !== false);   // true  → número ≠ booleano  
console.log(10 !== 10);     // false → mesmo valor e mesmo tipo
```

Resumo final:

- Use `==` e `!=` apenas se quiser permitir conversões automáticas (não recomendado).
- Use `===` e `!==` sempre que quiser comparar de forma **segura e previsível**.