

# Los operadores lógicos

## De comparación simple

Comparan dos valores, devuelven verdadero o falso.

```
{ } 10 == 15 // Igualdad → false  
10 != 15 // Desigualdad → true
```

## De comparación estricta

Comparan el valor y el tipo de dato también.

```
{ } 10 === "10" // Igualdad estricta → false  
10 !== 15 // Desigualdad estricta → true
```

En el primer caso el valor es 10 en ambos casos, pero los tipos de datos son number y string. Como estamos comparando que ambos (valor y tipo de dato) sean iguales, el resultado es false.

## De comparación (continuación)

Comparan dos valores, devuelven verdadero o falso.

```
{}  
15 > 15 // Mayor que → false  
15 >= 15 // Mayor o igual que → true  
10 < 15 // Menor que → true  
10 <= 15 // Menor o igual que → true
```



Siempre debemos escribir el símbolo mayor (>) o menor (<) antes que el igual (>= o <=). Si lo hacemos al revés (=> o =<) JavaScript lee primero el operador de asignación = y luego no sabe qué hacer con el mayor (>) o el menor (<).

“

Los **operadores** de **comparación** siempre **devolverán** un booleano, es decir **true** o **false**, como resultado.



”

## Lógicos

Permiten combinar valores booleanos, el resultado también devuelve un booleano.

Existen tres operadores **y** (and), **o** (or), **negación** (not).

**AND** (&&) → **todos** los valores deben evaluar como **true** para que el resultado sea true.

```
{ } (10 > 15) && (10 != 20) // false
```



```
{ } (12 % 4 == 0) && (12 != 24) // true
```



**OR ( || )** → **al menos un** valor debe evaluar como **true** para que el resultado sea true.

```
{ } (10 > 15) || (10 != 20) // true
```



```
{ } (12 % 5 == 0) || (12 != 12) // false
```



**NOT ( ! )** → **niega la condición**. Si era true, será false y viceversa.

```
{ } !false // true  
    !(20 > 15) // false
```

“

Los **operadores lógicos** siempre **devolverán** un booleano, es decir **true** o **false**, como resultado.



”

## De concatenación

Sirve para unir cadenas de texto. Devuelve otra cadena de texto.

```
let nombre = 'Teodoro';  
{ } let apellido = 'García';  
let nombreCompleto = nombre + ' ' + apellido;
```

Si mezclamos otros tipos de datos, estos se convierten a cadenas de texto.

```
let fila = 'M';  
{ } let asiento = 7;  
let ubicacion = fila + asiento; // 'M7' como string
```