Los operadores

DigitalHouse >
Coding School

Los **operadores** nos permiten **manipular el valor** de las variables, realizar **operaciones** y **comparar** sus valores

De asignación

Asignan el valor de la derecha en la variable de la izquierda.

```
{}  let edad = 35; // Asigna el número 35 a edad
```

Aritméticos

Nos permiten hacer operaciones matemáticas, devuelven el resultado de la operación.

```
{}  10 + 15 // Suma → 25
    10 - 15 // Resta → -5
    10 * 15 // Multiplicación → 150
    15 / 10 // División → 1.5
```

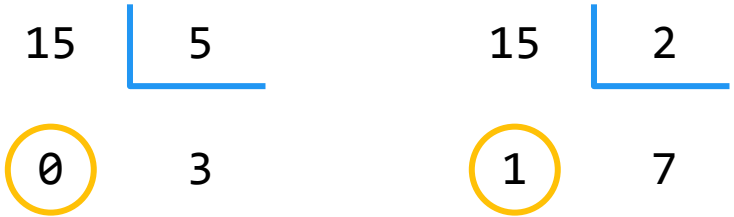
Aritméticos (continuación)

Nos permiten hacer operaciones matemáticas, devuelven el resultado de la operación.

```
15++ // Incremento, es igual a 15 + 1 → 16
15-- // Decremento, es igual a 15 - 1 → 14
```

```
15 % 5 // Módulo, el resto de dividir 15 entre 5 → 0
15 % 2 // Módulo, el resto de dividir 15 entre 2 → 1
```

El operador de módulo **%** nos devuelve el resto de una división.



Los **operadores** aritméticos siempre **devolverán** el **resultado numérico** de la **operación** que se esté realizando.

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
```


DigitalHouse>
Coding School