

LOS OPERADORES

DE ASIGNACIÓN

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

```
{ }
```

```
let edad = 35; // Asigno 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)

{ }

`15++` // Incremento, es igual a $15 + 1 \rightarrow 16$

`15--` // Decremento, es igual a $15 - 1 \rightarrow 14$

{ }

`15 % 5` // Módulo, el resto de dividir 15 entre 5 $\rightarrow 0$

`15 % 2` // Módulo, el resto de dividir 15 entre 2 $\rightarrow 1$

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

15		5		15		2
						
0		3		1		7

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 (<).

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**.

```
{ }
```

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



```
{ }
```

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



LÓGICOS (CONTINUACIÓN)

OR (||) al menos un valor debe evaluar como **true**.

```
{ }
```

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



```
{ }
```

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



NOT (!) niega la condición, si era true, será false y viceversa

```
{ }
```

```
!false // true
```

```
!(10 > 15) // true
```


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
```