Tipos de operadores

Aritméticos

Asignación

Unarios

Comparacion

Binarios

Lógicos

Operadores aritméticos

Se utilizan para realizar operaciones matemáticas

Operadores aritméticos

Los típicos

+ - * / % **

Operadores de asignación

Estos operadores nos permiten asignar información a diferentes constantes o variables a través del símbolo =, lo cuál es bastante lógico pues así lo hacemos en matemáticas.

Operadores de asignación

Nombre	Operador	Descripción
Asignación	**c = a + b**	Asigna el valor de la parte derecha
Suma y asignación	**a += b**	Es equivalente a **a = a + b**.
Resta y asignación	**a -= b**	Es equivalente a **a = a - b**.
Multiplicación y asignación	**a *= b**	Es equivalente a **a = a * b**.
División y asignación	**a /= b**	Es equivalente a **a = a / b**.
Módulo y asignación	**a %= b**	Es equivalente a **a = a % b**.
Exponenciación y asignación	**a **= b**	Es equivalente a **a = a ** b**.

Operadores unarios

Los operadores unarios se diferencian de otros operadores porque actúan sobre un solo valor o variable, en lugar de dos operandos. Esto significa que realizan operaciones utilizando únicamente un elemento almacenado en una variable.

Operadores unarios

Nombre	Operador	Descripción
Incremento	**a++**	Usa el valor de **a** y luego lo incrementa. También llamado **postincremento**.
Decremento	**a--**	Usa el valor de **a** y luego lo decrementa. También llamado **postdecremento**.
Incremento previo	**++a**	Incrementa el valor de **a** y luego lo usa. También llamado **preincremento**.
Decremento previo	**--a**	Decrementa el valor de **a** y luego lo usa. También llamado **predecremento**.

Operadores unarios

Probemos:

```
let a = 0;

while (a < 5) {
  console.log(a,a++,a);
}
```

```
let a = 0;

while (a < 5) {
  console.log(a, ++a, a);
}
```

Operadores de comparación

Los operadores de comparación se emplean en la programación, frecuentemente dentro de una estructura condicional como un 'if', aunque también pueden usarse en otros contextos, para efectuar verificaciones. Estas expresiones comparativas retornan un valor booleano, que puede ser verdadero (true) o falso (false).

Operadores de comparación

Nombre	Operador	Descripción
Operador de igualdad ==	**a == b**	Comprueba si el **valor** de **a** es igual al de **b**. **No comprueba tipo de dato**.
Operador de desigualdad !=	**a != b**	Comprueba si el **valor** de **a** no es igual al de **b**. **No comprueba tipo de dato**.
Operador mayor que >	**a > b**	Comprueba si el valor de **a** es mayor que el de **b**.
Operador mayor/igual que >=	**a >= b**	Comprueba si el valor de **a** es mayor o igual que el de **b**.
Operador menor que <	**a < b**	Comprueba si el valor de **a** es menor que el de **b**.
Operador menor/igual que <=	**a <= b**	Comprueba si el valor de **a** es menor o igual que el de **b**.
Operador de identidad ===	**a === b**	Comprueba si el **valor y el tipo de dato** de **a** es igual al de **b**.
Operador no idéntico !==	**a !== b**	Comprueba si el **valor y el tipo de dato** de **a** no es igual al de **b**.

Operadores de comparación

Probemos

```
5 == 5      // true   (ambos son iguales, coincide su valor)

"5" == 5    // true   (ambos son iguales, coincide su valor)

5 === 5     // true   (ambos son idénticos, coincide su valor y su tipo de dato)

"5" === 5   // false  (no son idénticos, coincide su valor, pero no su tipo de dato)
```

Operadores binarios

En Javascript no son muy utilizados, pero existen los operadores binarios que funcionan a nivel de bit. Es decir, con variables que solo pueden tomar valores 0 y 1.

Existen operadores para desplazar bits, y algunas lógicas.

Operadores lógicos

Son los que podemos ver en cualquier lenguaje de programación, aunque en Javascript, tienen sus particularidades propias

Operadores lógicos

Operador AND

Sigue la misma tabla de verdad de todos los lenguajes de programación

Operador AND		
Condición 1	Condición 2	Resultado
FALSO	FALSO	FALSO
FALSO	VERDADERO	FALSO
VERDADERO	FALSO	FALSO
VERDADERO	VERDADERO	VERDADERO

Operadores lógicos

Operador AND

Pero en Javascript tiene otra particularidad cuando lo ejecutamos entre dos variables.

Devolverá el primer valor si es false, o el segundo valor si el primero es true. Esto se puede leer de forma que «devuelve b si a y b son verdaderos, sino a».

Operadores lógicos

```
0 && undefined          // 0
undefined && 0          // undefined
55 && null              // null
null && 55              // null
44 && 20                // 20
45 && "OK"              // "OK"
false && "OK"           // false
```

Operadores lógicos

Operador OR

Si operamos con booleanos, sigue la tabla de verdad de toda la vida del operador OR.

Operador OR		
Condición 1	Condición 2	Resultado
FALSO	FALSO	FALSO
FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	VERDADERO
VERDADERO	VERDADERO	VERDADERO

Operadores lógicos

Operador OR

Si no son booleanos funcionará de la siguiente manera

Devolverá el primer valor si es true, o el segundo valor si el primero es false. Esto se puede leer de forma que «devuelve a (si es verdadero), o si no, b».

Operadores lógicos

```
0 || null         // null (se evalua como false || false, devuelve el segundo)
44 || undefined   // 44 (se evalua como true || false, devuelve el primero)
0 || 17           // 17 (se evalua como false || true, devuelve el segundo)
4 || 10           // 4 (se evalua como true || true, devuelve el primero)
```

Operadores lógicos

```
null || "Unknown name"          // "Unknown name"
false || "Unknown name"         // "Unknown name"
undefined || "Unknown name"     // "Unknown name"
0 || "Unknown name"             // "Unknown name"
```

Operadores lógicos

Operador Ternario

El operador ternario en JavaScript es un operador condicional que es una versión abreviada de la declaración if-else. Consiste en tres partes y se utiliza para asignar o retornar un valor basado en una condición

```
condición ? expresión1 : expresión2
```

Operadores lógicos

Operador Ternario

```
let edad = 20;
let mensaje = edad >= 18 ? 'Mayor de edad' : 'Menor de edad';
console.log(mensaje);
```

Operadores lógicos

Operador Ternario

```
// Sin operador ternario
let role;
if (name === "B3") {
  role = "Mago";
} else {
  role = "Novato";
}

// Con operador ternario
const role = name === "B3" ? "Mago" : "Novato";
```

Operadores lógicos

Operador Nullish coalescing ??

El operador a ?? b devuelve b sólo cuando a es undefined o null. De lo contrario devuelve a

Operadores lógicos

```
42 || 50            // 42
42 ?? 50            // 42 (ambos se comportan igual)
false || 50         // 50
false ?? 50         // false
0 || 50             // 50
0 ?? 50             // 0
null || 50          // 50
null ?? 50          // 50
undefined || 50     // 50
undefined ?? 50     // 50
```

Operadores lógicos

Asignación lógica nula ??=

Esto se usa en Javascript por ciertos tipos de operaciones

Existen ciertos casos donde, si una variable tiene valores null o undefined (valores nullish) y sólo en esos casos, queremos cambiar su valor.

Operadores lógicos

```
// Sin asignación lógica nula
if (x === null || x === undefined)
{
  x = 50;
}

// Con asignación lógica nula
x ??= 50;
```

Operadores lógicos

Operador lógico NOT

Es un operador unario

Devuelve el valor negado, es decir, lo comvierte a booleano y devuelve su negación.

Operadores lógicos

```
!true          // false
!false         // true
!!true         // true
!!false        // false
!!!true        // false
!5             // false
!0             // true
!""            // true (se evalua como !0, que es !false)
!(10 || 23)    // false (se evalua como !10, que es !true)
```