Curso de JavaScript avanzado

Estructuras de datos

Expresiones y operadores

ÍNDICE

Operadores y expresiones en JavaScript

- Llamamos **expresión** a cualquier acción u operación **que nos arroja un resultado** (C)
- Una operación se realiza **entre uno o más operandos** (A, B) **y un operador**
- Los operadores se dicen **unarios** si actúan sobre **un único operando** y **binarios si actúan entre dos**

Operadores y expresiones en JavaScript

```
// operando1 operador operando2
5 * 6 // -> 30
73 - 38 // -> 35
```

Operaciones con
operadores binarios

```
// operando1 operador
10++ // -> 11

// operador operando1
!10 // -> false
```

Operaciones con
operadores unarios

Operadores de comparación

==	“1” == 1	true
!=	“2” != 1	true
===	“1” === 1	false
!==	“1” !== 1	true
>=,<=	2 <= 2	true
>,<	2 > 3	false

- **El operador de comparación** compara sus operandos y devuelve un booleano
- Los operadores == y != **realizan coerción de los operandos si es necesario**
- Los operadores === y !== **comparan tipo y valor evitando coerción**

Operadores de asignación

x += y	x = x + y
x %= y	x = x % y
x **= y	x = x ** y
x &&= y	x && (x = y)
x ||= y	x || (x = y)
>,<	2 > 3

- Las asignaciones se evalúan de **derecha a izquierda**
- Para asignaciones más complejas podemos **usar el destructuring**

```
let variable = [2, 4];
let uno = variable[0];
let dos = variable[1];

// Con destructuring
let [uno, dos] = variable;
```

Operadores aritméticos

%	12 % 5	2
++	3++	4
--	3--	2
-	- true	-1
+	+true	1
**	2 ** 3	8

- Hay operaciones **que pueden resultar ambiguas y JS lanza un error (Como -4**2)**
- Operaciones como X / 0 devuelven un objeto **Infinity**

Otros operadores importantes

```
// condicion ? valor1 : valor2;
x = 5 > 2 ? 'sí!' : 'no :(';
// x vale 'sí!'

x = 5 > 2 ? (2 < 1 ? 'es menor' : 'es mayor') : 'no :(';
// x vale 'es mayor'
```

- El operador ternario nos sirve para asignar un valor entre dos posibles **evaluando una condición**
- **Es posible anidar ternarios dentro de los ternarios** si fuera necesario

Otros operadores importantes

```
let a = null
let b = 3

let res = a || b // -> 3

a = 5
b = 4

let res = a || b // -> 5

a = undefined
b = 4

let res = a || b // -> 4
```

- Los operadores binarios && y || permiten asignar valores **dependiendo de los valores falsy**
- El más usado es || sobre todo para hacer asignaciones **default** en caso de que un parámetro sea **falsy**
- El operador && **tiene el efecto opuesto**

PARA RESUMIR

- Una expresión es **cualquier acción que nos arroja un resultado**, y en el caso de las operaciones se realizan entre **uno o más operandos y un operador**

- Los operadores **pueden ser binarios si se realizan entre dos operandos o unarios solo con uno**

- La coerción de tipos es un **factor importante a tener en cuenta a la hora de usar operadores**, y es gracias a ella que muchos operadores tienen un uso extra en ciertas circunstancias