Booleanos

Los booleanos son un tipo de dato primitivo comúnmente usado en lenguajes de programación. Por definición, un booleano tiene 2 posibles valores: true (verdadero) o false (falso).

En JavaScript, frecuentemente se genera de manera implícita la coerción de tipos a booleano. Si, por ejemplo, tienes una sentencia *if* que evalúa cierta expresión, esa expresión genera coerción a booleano:

```
const a = 'una cadena';
if (a) {
  console.log(a); // arroja 'una cadena'
}
```

En el ejemplo anterior, la sentencia *if* evalúa a la constante a. Dado que el valor de a no es de tipo booleano, JavaScript genera coerción de tipo para poder evaluar la sentencia y concluir si se trata de un valor *Truthy* (verdadero) o *Falsy* (falso). Obtenemos como resultado 'una cadena' debido a que las cadenas de texto en JavaScript son un valor verdadero.

Solo hay unos cuantos valores que generan coerción a falso:

- false (en realidad no genera coerción porque ya es falso)
- null
- undefined
- NaN
- 0
- -0
- 0n, -0n (BigInt)
- " ", ' ', ` ` (cadena de texto vacía)
- document.all

Cualquier otro valor diferente de los anteriores generan coerción a verdadero.

Una manera en que se usa la coerción de tipos es con el uso del operador ***o*** (||) y del operador ***y*** (&&):

```
const a = 'palabra';
const b = false;
const c = true;
const d = 0
const e = 1
const f = 2
const g = null
```

```
console.log(a || b); // 'palabra'
console.log(c || a); // true
console.log(b || a); // 'palabra'
console.log(e || f); // 1
console.log(f || e); // 2
console.log(d || g); // null
console.log(g || d); // 0
console.log(a && c); // true
console.log(c && a); // 'palabra'
```

En el ejemplo anterior, el operador ***o*** (||) verifica el primer operando. Si dicho operando es el valor booleano verdadero (true) o pertenece a lista de los evaluados como verdaderos, JavaScript arroja el resultado inmediatamente sin necesidad de verificar el segundo operando. A esto se le llama una **evaluación cortocircuito**. Por tal razón, obtenemos 'palabra' en el primer caso y true en el segundo caso. Sin embargo, cuando el primer operando **no es** verdadero, devuelve el segundo operando. Entonces obtenemos 'palabra' en el tercer caso.

El operador ***y*** (&&) funciona de manera similar. Sin embargo, para que la operación sea verdadera, ambos operandos deben ser verdaderos. Devolverá siempre el segundo operando si ambos son verdaderos. De lo contrario, devolverá falso. Por tal razón, obtenemos true en el penúltimo caso y 'palabra' en el último caso.

El objeto Boolean

También existe un objeto nativo de JavaScript. El valor pasado como primer parámetro es convertido a un valor booleano si es necesario. Si el valor es omitido, 0, -0, **0n, -0n (BigInt)**, null, false, NaN, undefined, document.all, o una cadena vacía (" ", ' '. ` `), el objeto tiene valor inicial de falso. Cualquier otro valor, incluidos cualquier objeto o la cadena de texto "false", crean un objeto de valor inicial verdadero.

No confundas los valores booleanos primitivos verdadero y falso con los valores verdaderos y falsos del objeto Boolean.

Más detalles

Cualquier objeto que no tenga como valor undefined o null, incluyendo cualquier objeto Boolean que tenga como valor falso, será evaluado como verdadero cuando se pase por una sentencia condicional. Al ser verdadero, ejecutará la función. Por ejemplo, la condición en la siguiente sentencia *if* es evaluada como verdadera:

```
const x = new Boolean(false);
```

```
if (x) {
  // este código se ejecuta
}
```

Este comportamiento no aplica a los valores booleanos primitivos. Por ejemplo, la condición en la siguiente sentencia *if* es evaluada como falsa:

```
const x = false;
if (x) {
  // este código no se ejecuta
}
```

No uses un objeto Boolean para convertir un valor no booleano a uno booleano. En cambio, usa Boolean como función para realizar esta tarea:

```
const x = Boolean(expresión);     // preferido
const x = new Boolean(expresión); // no lo uses
```