

## 6. Booleanos

Los **booleanos** son uno de los tipos de datos más simples pero fundamentales en JavaScript. Representan valores lógicos y solo pueden tener dos posibles estados: `true` (verdadero) o `false` (falso). Aunque parezcan sencillos, los booleanos son esenciales para controlar el flujo de un programa mediante condiciones, bucles y operaciones lógicas.

En este capítulo, exploraremos qué son los booleanos, cómo se utilizan y cómo interactúan con otros tipos de datos a través de conversiones implícitas y explícitas.

### ¿Qué Son los Booleanos?

Un valor booleano representa una entidad lógica que puede ser verdadera (`true`) o falsa (`false`). Estos valores son ampliamente utilizados en estructuras de control como condicionales (`if`, `else`) y bucles (`while`, `for`), donde se evalúa si una condición es verdadera o falsa para decidir qué código ejecutar.

Ejemplo:

```
let esMayorDeEdad = true;
if (esMayorDeEdad) {
  console.log("Puede votar.");
} else {
  console.log("No puede votar.");
}
// Salida: "Puede votar."
```

### Conversiones a Booleanos

JavaScript puede convertir automáticamente otros tipos de datos en booleanos cuando se necesitan en un contexto lógico. Esto se conoce como **coerción de tipo**. Además, puedes realizar conversiones explícitas utilizando el constructor `Boolean()` o el operador `!!`.


### Valores Falsy y Truthy

En JavaScript, algunos valores se consideran **falsy** (evalúan como `false` cuando se convierten en booleanos), mientras que otros son **truthy** (evalúan como `true`). A continuación, se enumeran los valores más comunes:

Truthy - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN

En JavaScript, un valor verdadero es un valor que se considera true/verdadero cuando es evaluado en un contexto Booleano. Todos los valores son verdaderos a menos que se definan como falso (es decir, excepto false, 0, "", null, undefined, y NaN).

 <https://developer.mozilla.org/es/docs/Glossary/Truthy>

 mdn web docs

#### Valores Falsy:

- `false`
- `0` (cero)
- `NaN` (Not-a-Number)
- `""` (string vacío)
- `null`
- `undefined`

#### Valores Truthy:

- Cualquier número distinto de cero (positivo o negativo)
- Strings no vacíos (ejemplo: `"hola"`)

- Arrays y objetos (incluso si están vacíos)
- Funciones

Ejemplo:

```
console.log(Boolean(0)); // false
console.log(Boolean("hola")); // true
console.log(Boolean([])); // true
console.log(Boolean(null)); // false
```

## Conversión Explícita

Puedes usar el constructor `Boolean()` o el operador `!!` para convertir explícitamente un valor en un booleano.

Ejemplo:

```
let numero = 42;
console.log(Boolean(numero)); // true
console.log(!!numero); // true

let textoVacio = "";
console.log(Boolean(textoVacio)); // false
console.log(!!textoVacio); // false
```

## Operadores Lógicos

Los operadores lógicos permiten combinar o modificar valores booleanos. Son esenciales para construir expresiones complejas en condiciones y evaluaciones.

### Operador AND ( `&&` )

El operador `&&` devuelve `true` solo si **ambos operandos** son verdaderos. Si uno de ellos es falso, devuelve el primer valor falso encontrado.

Ejemplo:

```
console.log(true && true); // true
console.log(true && false); // false
console.log(false && true); // false
console.log(0 && "hola"); // 0 (el primer valor falso)
```

### Operador OR ( `||` )

El operador `||` devuelve `true` si **al menos uno de los operandos** es verdadero. Si ambos son falsos, devuelve el último valor falso.

Ejemplo:

```
console.log(true || false); // true
console.log(false || false); // false
console.log(0 || "hola"); // "hola" (el primer valor truthy)
```

### Operador NOT ( `!` )

El operador `!` invierte el valor booleano de un operando. Si es `true`, lo convierte en `false`, y viceversa.

Ejemplo:

```
console.log(!true); // false
console.log(!false); // true
```

```
console.log(!"hola"); // false
console.log(!""); // true
```

## Doble Negación ( !! )

El uso de `!!` convierte un valor en su equivalente booleano. Es útil para verificar si un valor es `truthy` o `falsy`.

Ejemplo:

```
console.log(!!42); // true
console.log(!!""); // false
```

## Evaluación Cortocircuitada

Los operadores `&&` y `||` usan **evaluación cortocircuitada**, lo que significa que no siempre evalúan ambos operandos.

- `&&` : Si el primer operando es `false`, no evalúa el segundo porque ya sabe que el resultado será `false`.
- `||` : Si el primer operando es `true`, no evalúa el segundo porque ya sabe que el resultado será `true`.

Ejemplo:

```
let resultado = 0 || "valor predeterminado"; // "valor predeterminado"
console.log(resultado);

resultado = 1 && "valor final"; // "valor final"
console.log(resultado);
```

## Uso en Estructuras Condicionales

Los booleanos son la base de las estructuras condicionales, que permiten controlar el flujo de un programa según ciertas condiciones.

Ejemplo con `if` :

```
let usuarioLogeado = true;

if (usuarioLogeado) {
  console.log("Bienvenido al sistema.");
} else {
  console.log("Por favor, inicia sesión.");
}

// Salida: "Bienvenido al sistema."
```

Ejemplo con `while` :

```
let contador = 3;

while (contador > 0) {
  console.log(`Quedan ${contador} intentos.`);
  contador--;
}

// Salida:
// Quedan 3 intentos.
// Quedan 2 intentos.
// Quedan 1 intentos.
```

---

## Comparaciones y Resultados Booleanos

Las comparaciones entre valores generan resultados booleanos. Los operadores de comparación incluyen:

1. **Igualdad ( `==` ):**

Evalúa si dos valores son iguales, realizando coerción de tipo si es necesario.

```
console.log(5 == "5"); // true (coerción de tipo)
```

2. **Igualdad estricta ( `===` ):**

Evalúa si dos valores son iguales sin realizar coerción de tipo.

```
console.log(5 === "5"); // false
```

3. **Desigualdad ( `!=` ):**

Evalúa si dos valores son diferentes, permitiendo coerción de tipo.

```
console.log(5 != "5"); // false
```

4. **Desigualdad estricta ( `!==` ):**

Evalúa si dos valores son diferentes sin coerción de tipo.

```
console.log(5 !== "5"); // true
```

5. **Mayor que ( `>` ), Menor que ( `<` ), Mayor o igual ( `>=` ), Menor o igual ( `<=` ):**

Comparan valores numéricos o strings.

```
console.log(10 > 5); // true  
console.log("a" < "b"); // true
```