Lenguaje de Marcas: Javascript

Iván Nieto Ruiz

il.nietoruiz@edu.gva.es

Referencia y copia en JavaScript

Tipos primitivos (copia por valor):

- Incluyen: `number`, `string`, `boolean`, `null`, `undefined`, `symbol`, `bigint`.
- Cuando se asigna una variable con un valor primitivo a otra, se hace una **copia del valor**.

Tipos de referencia (copia por referencia):

- Incluyen: `object`, `array`, `function`.
- Cuando se asigna una variable con un objeto a otra, **ambas apuntan al mismo objeto en memoria**.

Ejemplo de **Copia por Valor** (Primitivos)

```
let a = 10;
let b = a; // Se copia el valor de 'a' en 'b'

b = 20; // Modificamos 'b', pero 'a' no cambia

console.log(a); // 10
console.log(b); // 20
```

Referencia y copia en JavaScript

Ejemplo de **Copia por Referencia** (Objetos y Arrays) - **comparten la misma referencia** en memoria

```
let obj1 = { name: "Juan" };
let obj2 = obj1; // Ambas variables apuntan al mismo objeto en memoria

obj2.name = "Carlos"; // Modificamos 'obj2'

console.log(obj1.name); // "Carlos" (también cambió en 'obj1')
console.log(obj2.name); // "Carlos"
```

Copiar objetos sin modificar el original

Copia superficial (shallow copy)

```
let obj1 = { name: "Juan", age: 30 };

// Usando spread operator (...):
let obj2 = { ...obj1 };
obj2.name = "Carlos";

console.log(obj1.name); // "Juan" (no se modifica)
console.log(obj2.name); // "Carlos"

// Usando Object.assign():
let obj2 = Object.assign({}, obj1);
```

Referencia y copia en JavaScript

Problema: Si el objeto tiene **objetos anidados**, estos seguirán compartiendo la referencia.

```
let obj1 = { name: "Juan", address: { city: "Madrid" } };
let obj2 = { ...obj1 };

obj2.address.city = "Barcelona";

console.log(obj1.address.city); // "Barcelona" CUIDADO (¡Se modificó el original!)
```

Si hay objetos anidados, usa **structuredClone()** o **JSON.parse(JSON.stringify(obj))**.

```
let obj1 = { name: "Juan", address: { city: "Madrid" } };
let obj2 = structuredClone(obj1); // ¡Ahora sí es independiente!

obj2.address.city = "Barcelona";

console.log(obj1.address.city); // "Madrid" - OK
console.log(obj2.address.city); // "Barcelona"
```

structuredClone() es nativo en JS moderno, mientras que JSON.parse(JSON.stringify(obj)) funciona pero **pierde funciones y símbolos**.

Referencia y copia en JavaScript

Lo mismo ocurre con los **arrays**

```
let arr1 = [1, 2, 3];
let arr2 = arr1; // Copia por referencia

arr2.push(4);

console.log(arr1); // [1, 2, 3, 4] ❌ (¡Se modificó el original!)
console.log(arr2); // [1, 2, 3, 4]
```

Forma correcta de copiar un array (shallow copy):

```
let arr2 = [...arr1]; // Operador spread
// o
let arr2 = arr1.slice(); // Método slice()
```

Copia profunda de arrays anidados:

```
let arr1 = [[1, 2], [3, 4]];
let arr2 = structuredClone(arr1);

arr2[0][0] = 99;

console.log(arr1[0][0]); // 1 --> OK
console.log(arr2[0][0]); // 99
```

Ejemplos

Ejercicio 1:

```
let a = 5;
let b = a;

b = 10;

console.log(a); // ¿?
console.log(b); // ¿?
```

¿Por qué **a** no cambia cuando modificamos b?

Ejemplos

Ejercicio 1:

```
let persona1 = {
        nombre: "Lucas",
        direccion: { ciudad: "Madrid", pais: "España" }
};

let persona2 = { ...persona1 };

persona2.direccion.ciudad = "Barcelona";

console.log(persona1.direccion.ciudad); // ¿?
console.log(persona2.direccion.ciudad); // ¿?
```

Ejemplos

Ejercicio 3:

```
function deepCopy(obj) {
        return structuredClone(obj);
}

let user1 = { nombre: "Sofía", datos: { edad: 28, ciudad: "Valencia" } };
let user2 = deepCopy(user1);

user2.datos.ciudad = "Sevilla";

console.log(user1.datos.ciudad); // ¿?
console.log(user2.datos.ciudad); // ¿?
```

¿Preguntas?