10. Objetos

Los **objetos** son una de las estructuras más versátiles y fundamentales en JavaScript. Representan entidades del mundo real o conceptos abstractos mediante una colección de **propiedades** (pares clave-valor). En este capítulo, aprenderás cómo crear, manipular y trabajar con objetos, así como explorar sus características avanzadas.

¿Qué Son los Objetos?

Un objeto es una colección de propiedades, donde cada propiedad tiene un nombre (llave) y un valor asociado. Los valores pueden ser de cualquier tipo, incluyendo números, strings, arrays, funciones u otros objetos. Los objetos permiten modelar datos complejos y organizarlos de manera lógica.

Ejemplo:

```
let persona = {
    nombre: "Ana",
    edad: 25,
    esEstudiante: true,
    saludo: function() {
        console.log(`Hola, soy ${this.nombre}`);
    }
};

console.log(persona.nombre); // Ana
persona.saludo(); // Hola, soy Ana
```

Creación de Objetos

Existen varias formas de crear objetos en JavaScript:

1. Notación Literal

Es la forma más común y recomendada para crear objetos.

```
let coche = {
    marca: "Toyota",
    modelo: "Corolla",
    año: 2020
};
```

2. Constructor `Object`

Puedes usar el constructor `Object` para crear un objeto vacío y luego agregar propiedades.

```
let coche = new Object();
coche.marca = "Toyota";
coche.modelo = "Corolla";
coche.año = 2020;
```

Acceso a Propiedades

Puedes acceder a las propiedades de un objeto utilizando dos métodos principales:

1. Notación de Punto

Se utiliza cuando conoces el nombre exacto de la propiedad.

```
let persona = { nombre: "Juan", edad: 30 };
console.log(persona.nombre); // Juan
```

2. Notación de Corchetes

Útil cuando el nombre de la propiedad es dinámico o contiene caracteres especiales.

```
let persona = { "nombre-completo": "Juan Pérez" };
console.log(persona["nombre-completo"]); // Juan Pérez
```

Modificación y Adición de Propiedades

Puedes modificar o agregar nuevas propiedades a un objeto en cualquier momento.

Modificar:

```
let persona = { nombre: "Ana", edad: 25 };
persona.edad = 26;
console.log(persona.edad); // 26
```

Agregar:

```
persona.ciudad = "Madrid";
console.log(persona.ciudad); // Madrid
```

Métodos en Objetos

Un método es una función que pertenece a un objeto. Se define como una propiedad cuyo valor es una función.

Ejemplo:

```
let calculadora = {
    sumar: function(a, b) {
        return a + b;
    },
    restar: function(a, b) {
        return a - b;
    }
};

console.log(calculadora.sumar(5, 3)); // 8
console.log(calculadora.restar(10, 4)); // 6
```

Desde ES6, puedes usar la sintaxis abreviada para definir métodos:

```
let calculadora = {
    sumar(a, b) {
        return a + b;
    },
    restar(a, b) {
        return a - b;
    }
};
```

El Objeto `this`

La palabra clave `this` se refiere al objeto actual dentro de un método. Permite acceder a las propiedades y métodos del objeto desde dentro del mismo.

Ejemplo:

```
let persona = {
    nombre: "Luis",
    saludar: function() {
        console.log(`Hola, soy ${this.nombre}`);
    }
};

persona.saludar(); // Hola, soy Luis
```

> Nota: El valor de this depende del contexto en el que se ejecute. En funciones flecha, this hereda el valor del contexto externo.

Iteración sobre Propiedades

Puedes recorrer las propiedades de un objeto usando bucles como `for...in`.

Ejemplo:

```
let persona = { nombre: "Ana", edad: 25, ciudad: "Barcelona" };

for (let clave in persona) {
    console.log(`${clave}: ${persona[clave]}`);
}
// Salida:
// nombre: Ana
// edad: 25
// ciudad: Barcelona
```

Para verificar si una propiedad pertenece directamente al objeto (y no a su prototipo), usa `hasOwnProperty`.

```
if (persona.hasOwnProperty("edad")) {
    console.log("La propiedad 'edad' existe.");
}
```

Desestructuración de Objetos

La desestructuración permite extraer propiedades de un objeto y asignarlas a variables individuales.

Ejemplo:

```
let persona = { nombre: "Carlos", edad: 30, ciudad: "Valencia" };
let { nombre, edad } = persona;

console.log(nombre); // Carlos
console.log(edad); // 30
```

Copia de Objetos

Los objetos en JavaScript son **referencias**, lo que significa que copiar un objeto directamente crea una referencia compartida, no una copia independiente.

Copia Superficial:

```
let persona1 = { nombre: "Ana", edad: 25 };
let persona2 = { ...persona1 }; // Spread operator

persona2.nombre = "Juan";
console.log(persona1.nombre); // Ana
console.log(persona2.nombre); // Juan
```

Copia Profunda:

Para copiar objetos anidados, puedes usar `JSON.parse` y `JSON.stringify`, aunque esta técnica tiene limitaciones.

```
let objeto = { a: 1, b: { c: 2 } };
let copiaProfunda = JSON.parse(JSON.stringify(objeto));
```

Métodos Útiles del Objeto Global `Object`

JavaScript proporciona métodos integrados para trabajar con objetos.

1. `Object.keys(objeto)`

Devuelve un arreglo con las claves del objeto.

```
let persona = { nombre: "Ana", edad: 25 };
console.log(Object.keys(persona)); // ["nombre", "edad"]
```

2. `Object.values(objeto)`

Devuelve un arreglo con los valores del objeto.

```
console.log(Object.values(persona)); // ["Ana", 25]
```

3. `Object.entries(objeto)`

Devuelve un arreglo de arreglos, donde cada subarreglo contiene una clave y su valor.

```
console.log(Object.entries(persona));
// [["nombre", "Ana"], ["edad", 25]]
```

4. `Object.assign(destino, fuente)`

Copia las propiedades de un objeto fuente a un objeto destino.

```
let destino = { a: 1 };
let fuente = { b: 2 };
Object.assign(destino, fuente);
console.log(destino); // { a: 1, b: 2 }
```
