11. Ciclos (Loops)

Los **ciclos** o **bucles** son estructuras de control que permiten ejecutar un bloque de código repetidamente mientras se cumpla una condición específica. Son esenciales para automatizar tareas repetitivas, como iterar sobre elementos de un arreglo o procesar datos en bucle. En este capítulo, exploraremos los tipos principales de ciclos en JavaScript: `while`, `do...while`, `for`, `for...in` y `for...of`.

---

1. Ciclo `while`

El ciclo `while` ejecuta un bloque de código mientras una condición sea verdadera (`true`). Si la condición es falsa desde el inicio, el bloque no se ejecutará.

Sintaxis:

```
while (condición) {
  // Código a ejecutar mientras la condición sea verdadera
}
```

Ejemplo:

```
let contador = 0;

while (contador < 5) {
  console.log(`Iteración ${contador}`);
  contador++;
}
// Salida:
// Iteración 0
// Iteración 1
// Iteración 2
// Iteración 3
// Iteración 4
```

> Precaución: Asegúrate de que la condición eventualmente se vuelva falsa; de lo contrario, el ciclo podría volverse infinito.

---

2. Ciclo `do...while`

El ciclo `do...while` es similar al ciclo `while`, pero garantiza que el bloque de código se ejecute al menos una vez, incluso si la condición inicial es falsa.

Sintaxis:

```
do {
   // Código a ejecutar
} while (condición);
```

Ejemplo:

```
let contador = 5;

do {
   console.log(`Iteración ${contador}`);
   contador--;
} while (contador > 0);
// Salida:
// Iteración 5
// Iteración 4
// Iteración 3
// Iteración 2
// Iteración 1
```

---

3. Ciclo `for`

El ciclo `for` es uno de los más utilizados en JavaScript. Proporciona una forma compacta de iterar un número específico de veces, utilizando tres componentes: inicialización, condición y expresión final.

Sintaxis:

```
for (inicialización; condición; expresiónFinal) {
   // Código a ejecutar
}
```

- **Inicialización**: Define una variable de control (por ejemplo, `i = 0`).
- **Condición**: Evalúa si el ciclo debe continuar.
- **Expresión Final**: Actualiza la variable de control después de cada iteración.

Ejemplo:

```
for (let i = 0; i < 5; i++) {
   console.log(`Iteración ${i}`);
}
// Salida:
// Iteración 0
// Iteración 1
// Iteración 2
```

```
// Iteración 3
// Iteración 4
```

4. Ciclo `for...in`

El ciclo `for...in` se utiliza para iterar sobre las propiedades enumerables de un objeto. Es especialmente útil cuando necesitas recorrer las claves de un objeto.

Sintaxis:

```
for (let clave in objeto) {
  // Código a ejecutar
}
```

Ejemplo:

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

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

> Nota: Este ciclo también funciona con arreglos, pero no es recomendable usarlo para iterar sobre ellos porque puede incluir propiedades adicionales no numéricas.

5. Ciclo `for...of`

El ciclo `for...of` se utiliza para iterar sobre valores iterables, como arreglos, strings, mapas y conjuntos. Es ideal para acceder directamente a los valores de estos objetos.

Sintaxis:

```
for (let valor of iterable) {
  // Código a ejecutar
}
```

Ejemplo con Arreglo:

```
let frutas = ["manzana", "plátano", "uva"];
```

```
for (let fruta of frutas) {
  console.log(fruta);
}
// Salida:
// manzana
// plátano
// uva
```

Ejemplo con String:

```
let mensaje = "Hola";

for (let letra of mensaje) {
  console.log(letra);
}
// Salida:
// H
// o
// l
// a
```

Comparación entre `for...in` y `for...of`

Característica	`for...in`	`for...of`
Objetivo	Itera sobre **claves** de un objeto	Itera sobre **valores** de un iterable
Útil para	Objetos	Arreglos, strings, mapas, conjuntos
Funciona con arreglos?	Sí, pero no es recomendable	Sí, es la opción preferida

---

Cada tipo de ciclo tiene su propósito específico:

- Usa `while` o `do...while` cuando no sabes cuántas iteraciones necesitarás.
- Usa `for` para iterar un número fijo de veces.
- Usa `for...in` para recorrer las propiedades de un objeto.
- Usa `for...of` para iterar sobre valores de arreglos, strings u otros iterables.