Lenguaje de Marcas: Javascript

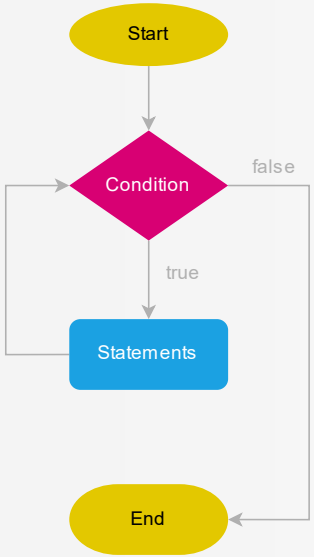
Iván Nieto Ruiz

il.nietoruiz@edu.gva.es

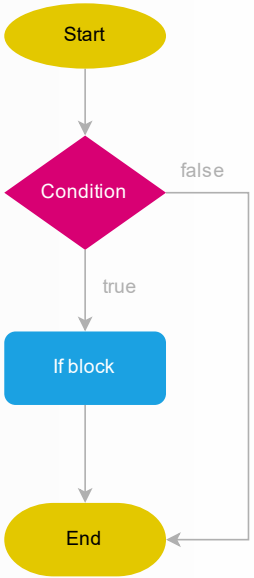
¿Qué son las estructuras de control?

•Son bloques de código que permiten tomar decisiones o repetir acciones según ciertas condiciones o reglas. En JavaScript, se dividen en:

• Estructuras de decisión (condicionales).



• Estructuras de repetición (bucles).



Estructuras de control

If:

Se usa para ejecutar código solo si una condición es verdadera.

```
let edad = 18;

if (edad >= 18) {
console.log("Eres mayor de edad.");
}
```

If...else

Se usa cuando necesitas ejecutar un bloque de código si la condición es verdadera y otro si es falsa.

```
let hora = 20;

if (hora < 12) {
console.log("Buenos días.");
} else {
console.log("Buenas tardes.");
}
```

if...else if...else

```
let nota = 85;

if (nota >= 90) {
console.log("Sobresaliente.");
} else if (nota >= 70) {
console.log("Aprobado.");
} else {
console.log("Suspendido.");
}
```

Estructuras de control

switch:

Ideal cuando hay varias condiciones posibles para una misma variable.

```
switch (expresión) {
case valor1:
// Código para el caso valor1
break;
case valor2:
// Código para el caso valor2
break;
default:
// Código si no se cumple ningún caso
}
```

```
let dia = 3;

switch (dia) {
case 1:
console.log("Lunes");
break;
case 2:
console.log("Martes");
break;
case 3:
console.log("Miércoles");
break;
default:
console.log("Día no válido.");
}
```

Estructuras de control

Estructuras de repetición (bucles)
Bucle for:
El bucle `for` es una estructura de control que se utiliza para repetir un bloque de código un número específico de veces.

- **Inicialización**: Se ejecuta una sola vez al principio del bucle y se utiliza para definir una variable de control, generalmente un contador.
- **Condición**: Se evalúa antes de cada iteración. Si la condición es verdadera, el bucle continúa. Si es falsa, el bucle se detiene.
- **Incremento/Decremento**: Se ejecuta después de cada iteración y se utiliza para modificar el valor del contador.

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

for (let i = 1; i <= 5; i++) {
console.log("Número:", i);
}
```

Estructuras de control

While:

Se usa cuando no sabes exactamente cuántas iteraciones necesitas, pero tienes una condición.

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

let contador = 1;

while (contador <= 3) {
    console.log("Contador:", contador);
    contador++;
}
```

Estructuras de control

Do...while:

Es similar a while, pero garantiza que el bloque de código se **ejecute al menos una vez.**

```
let numero = 0;

do {
    console.log("Número:", numero);
    numero++;
} while (numero < 3);
```

Estructuras de control

Bucles con for...of y for...in

Es similar a while, pero garantiza que el bloque de código se **ejecute al menos una vez.**

for...of: Itera sobre elementos de arrays u objetos iterables.

```
let colores = ["rojo", "verde", "azul"];

for (let color of colores) {
    console.log(color);
}
```

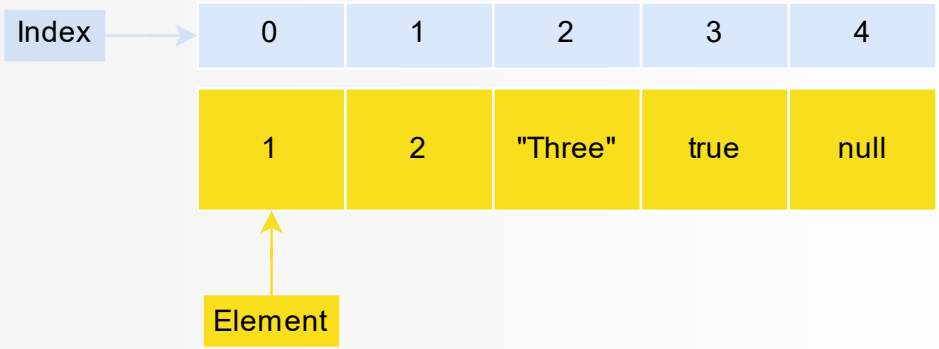
`for...in`: Itera sobre las propiedades de un objeto.

```
let persona = { nombre: "Juan", edad: 30 };

for (let propiedad in persona) {
    console.log(propiedad, ":",         persona[propiedad]);
}
```

Arrays en JS

- Un **array** es una colección ordenada de elementos.
- Los arrays pueden almacenar cualquier tipo de dato: **números**, **cadenas**, **objetos**, incluso otros arrays.




```
let frutas = ['Manzana', 'Banana', 'Naranja'];
console.log(frutas); // ["Manzana", "Banana", "Naranja"]
```

Arrays

Array literal (más común):

```
let frutas = ['Manzana', 'Banana', 'Naranja'];
console.log(frutas); // ["Manzana", "Banana", "Naranja"]
```

Constructor `Array`:

```
let numeros = new Array(1, 2, 3, 4);
console.log(numeros); // [1, 2, 3, 4]
```

Array vacío y luego agregar elementos:

```
let vacio = [];
vacio.push('Elemento1');
console.log(vacio); // ["Elemento1"]
```

Arrays

Arrays normales y asociativos

```
let numeros = [10, 20, 30];
console.log(numeros[0]); // 10
```

Array asociativo

```
const persona = {
nombre: "Juan",
edad: 30,
profesion: "Desarrollador",
pais: "España"
};

// Acceder a los valores
console.log(persona.nombre); // Output: Juan
console.log(persona.edad); // Output: 30
```

Arrays

Acceso y modificación de elementos

```
let frutas = ['Manzana', 'Banana', 'Naranja'];
console.log(frutas[0]); // "Manzana"

// Modificar un elemento
frutas[1] = 'Fresa';
console.log(frutas); // ["Manzana", "Fresa", "Naranja"]
```

Propiedad importante: `length`

```
console.log(frutas.length); // 3
```

Arrays

Recorrido de arrays :

Bucle for:
Se utiliza generalmente con **arrays** y otras estructuras de datos que tienen índices numéricos.

```
const numeros = [10, 20, 30, 40, 50];

for (let i = 0; i < numeros.length; i++) {
console.log(numeros[i]);
}

// Output:
// 10
// 20
// 30
// 40
// 50
```

Arrays

Recorrido de arrays :

forEach:

```
numeros.forEach((numero) => {
        console.log(numero);
});
```

Arrays - Métodos básicos de arrays

Agregar elementos:

`push` (al final):

```
let numeros = [1, 2, 3, 4, 5];
numeros.push(6);
console.log(numeros); // [1, 2, 3, 4, 5, 6]
```

`unshift` (al inicio):

```
let numeros = [1, 2, 3, 4, 5,6];
numeros.unshift(0);
console.log(numeros); // [0, 1, 2, 3, 4, 5, 6]
```

Eliminar elementos:

`pop` (elimina el último):

```
numeros.pop();
console.log(numeros); // [0, 1, 2, 3, 4, 5]
```

`shift` (elimina el primero):

```
numeros.shift();
console.log(numeros); // [1, 2, 3, 4, 5]
```

Arrays - Métodos básicos de arrays

Concatenar arrays:

```
let numeros = [1, 2, 3, 4, 5];
let masNumeros = [6, 7, 8];
let combinados = numeros.concat(masNumeros);
console.log(combinados); // [1, 2, 3, 4, 5, 6, 7, 8]
```

Verificar elementos:

```
let numeros = [1, 2, 3, 4, 5,6];
console.log(numeros.includes(3)); // true
console.log(numeros.includes(10)); // false
```

Arrays - Métodos avanzados

`map`: Crear un nuevo array aplicando una función

```
let numeros = [1, 2, 3, 4, 5];
let cuadrados = numeros.map((numero) => numero * numero);
console.log(cuadrados); // [1, 4, 9, 16, 25]
```

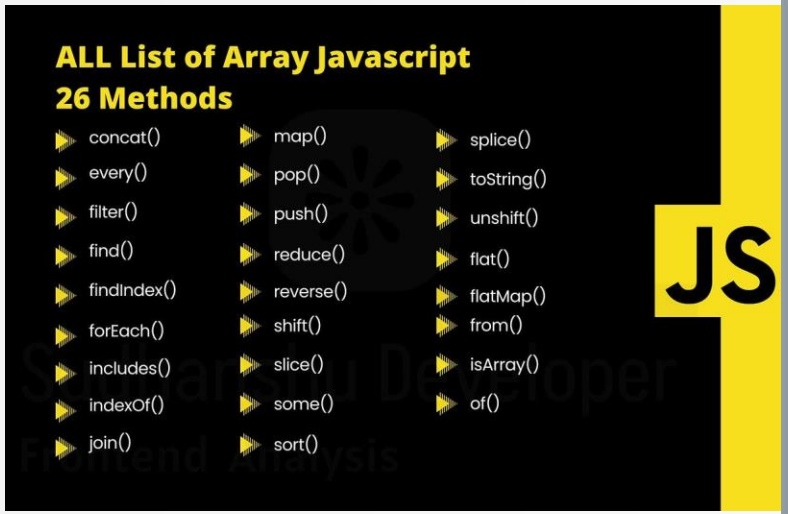
`filter`: Filtrar elementos según una condición

```
let numeros = [1, 2, 3, 4, 5,6];
let mayoresA3 = numeros.filter((numero) => numero > 3);
console.log(mayoresA3); // [4, 5]
```

`find`: Encuentra el primer elemento que cumple una condición

```
let encontrado = numeros.find((numero) => numero > 3);
console.log(encontrado); // 4
```

Arrays - Métodos avanzados




```
/  JavaScript Array Methods

[3, 4, 5, 6].at(1); // 4
[3, 4, 5, 6].pop(); // [3, 4, 5]
[3, 4, 5, 6].push(7); // [3, 4, 5, 6, 7]
[3, 4, 5, 6].fill(1); // [1, 1, 1, 1]
[3, 4, 5, 6].join("-"); // '3-4-5-6'
[3, 4, 5, 6].shift(); // [4, 5, 6]
[3, 4, 5, 6].reverse(); // [6, 5, 4, 3]
[3, 4, 5, 6].unshift(1); // [1, 3, 4, 5, 6]
[3, 4, 5, 6].includes(5); // true
[3, 4, 5, 6].map((num) => num + 6); // [9, 10, 11, 12]
[3, 4, 5, 6].find((num) => num > 4); // 5
[3, 4, 5, 6].filter((num) => num > 4); // [5, 6]
[3, 4, 5, 6].every((num) => num > 5); // false
[3, 4, 5, 6].findIndex((num) => num > 4); // 2
[3, 4, 5, 6].reduce((acc, num) => acc + num, 0); // 18
```

Arrays - Métodos avanzados

Arrays multidimensionales
Un array dentro de otro array.

```
let matriz = [
[1, 2, 3],
[4, 5, 6],
[7, 8, 9]
];
console.log(matriz[1][2]); // 6
```

Recorrido de una matriz:

```
matriz.forEach((fila) => {
    fila.forEach((columna) => {
        console.log(columna);
    });
});
```

¿Preguntas?