JavaScript

Estructuras de control

CertiDevs

Índice de contenidos

Las estructuras de control en JavaScript permiten ejecutar diferentes bloques de código dependiendo de condiciones específicas o repetir la ejecución de un bloque de código un número determinado de veces. Entre las estructuras de control más comunes se encuentran:

1. Condicionales

Las estructuras condicionales ejecutan bloques de código específicos según si una condición es verdadera o falsa.

1.1. if

La declaración `if` ejecuta un bloque de código si la condición especificada es verdadera.

```
const numero = 10;

if (numero > 5) {
    console.log('El número es mayor que 5.');
}
```

1.2. else

La declaración `else` se usa junto con `if` para ejecutar un bloque de código si la condición especificada es falsa.

```
const numero = 4;

if (numero > 5) {
    console.log('El número es mayor que 5.');
} else {
    console.log('El número es menor o igual que 5.');
}
```

1.3. else if

- La declaración `else if` se utiliza para agregar condiciones adicionales después de una declaración `if`.
    - Si la condición if es falsa, se evalúa la siguiente condición else if.
    - Si ninguna condición es verdadera, se ejecuta el bloque de código en la declaración else (si está presente).

```
const numero = 10;
```

```
if (numero > 20) {
    console.log('El número es mayor que 20.');
} else if (numero > 10) {
    console.log('El número es mayor que 10 y menor o igual que 20.');
} else {
    console.log('El número es menor o igual que 10.');
}
```

1.4. Operador Ternario

El operador **ternario** (o operador condicional) es una forma abreviada de la estructura if-else.

Utiliza el símbolo `?` para separar la condición y la expresión que se evalúa si la condición es verdadera, y el símbolo `:` para separar las expresiones que se evalúan si la condición es verdadera o falsa.

```
const numero = 10;

const resultado = numero > 5 ? 'El número es mayor que 5.' : 'El número es menor o
igual que 5.';

console.log(resultado); // 'El número es mayor que 5.'
```

2. Bucles

Los **bucles** permiten ejecutar un bloque de código repetidamente hasta que se cumple una condición específica.

2.1. for

El bucle `for` consta de tres partes: inicialización, condición y actualización, separadas por puntos y comas.

El bloque de código se ejecuta repetidamente mientras la condición sea verdadera.

```
for (let i = 0; i < 5; i++) {
    console.log(i); // 0, 1, 2, 3, 4
}
```

2.2. while

El bucle `while` ejecuta un bloque de código mientras la condición especificada sea verdadera.

```
let i = 0;

while (i < 5) {
    console.log(i); // 0, 1, 2, 3, 4
    i++;
}
```

2.3. do-while

El bucle `do while` ejecuta un bloque de código al menos una vez y continúa ejecutándolo mientras la condición especificada sea verdadera. La condición se evalúa después de cada iteración del bucle.

```
let i = 0;

do {
    console.log(i); // 0, 1, 2, 3, 4
    i++;
} while (i < 5);
```

3. Control de Flujo en Bucles

Puedes utilizar las declaraciones `break` y `continue` para controlar el flujo de ejecución dentro de los bucles.

3.1. break

La declaración `break` se utiliza para salir de un bucle antes de que se complete la condición de terminación.

Cuando se encuentra una declaración break, el bucle se interrumpe y el flujo de ejecución continúa después del bucle.

```
for (let i = 0; i < 10; i++) {
    if (i === 5) {
        break;
    }
    console.log(i); // 0, 1, 2, 3, 4
}
```

3.2. continue

La declaración `continue` se utiliza para omitir la ejecución del resto de un bloque de código en una iteración específica del bucle y continuar con la siguiente iteración.

Cuando se encuentra una declaración continue, el flujo de ejecución salta al final del bloque de código y luego continúa con la siguiente iteración del bucle.

```
for (let i = 0; i < 5; i++) {
    if (i === 2) {
        continue;
    }
    console.log(i); // 0, 1, 3, 4
}
```

Las estructuras de control en JavaScript permiten ejecutar bloques de código en función de condiciones específicas y repetir la ejecución de bloques de código.

Al comprender las diferencias entre las estructuras condicionales (if, else, else if y el operador ternario) y los bucles (for, while, do-while), así como las declaraciones break y continue, se puede escribir código más flexible y eficiente en aplicaciones de JavaScript.