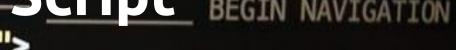


Somos un ecosistema de desarrolladores de software

## Estructuras de control Java-Script BEGIN NAVIGATION





```
">Home</a>
.html">Home Events</a>
 nu.html">Multiple Column Men
  <a href="#" class="current"
   utton-header.html">Tall But
    ogo.html">Image Logo</a></
     "tall-logo.html">Ta
        ="#">Carousels</a>
```



#### Estructuras de control

#### Propiedades

Las estructuras de control son herramientas fundamentales en programación que permiten alterar el flujo de ejecución de un programa. En JavaScript, hay tres tipos principales de estructuras de control:

- Estructuras de control condicionales: Permiten tomar decisiones basadas en condiciones. La instrucción
  if es un ejemplo común, donde un bloque de código se ejecutará si una condición es verdadera. También
  hay estructuras más complejas como else if y switch.
- Estructuras de control de bucle (ciclos): Permiten repetir un bloque de código varias veces. Los bucles más comunes son for, while, y do-while.
- Estructuras de control de salto: Permiten alterar el flujo normal de ejecución. Un ejemplo es la instrucción break para salir de un bucle, o continue para pasar a la siguiente iteración de un bucle.



#### </RIWi>

#### Condicionales

If – If else

```
let edad = 18;

if (edad ≥ 18) {
    console.log("Eres mayor de edad. Puedes votar.");
}
```

```
let hora = 14;

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

```
- \square \times
let hora = 20;
if (hora < 12) {
    console.log("Buenos días");
} else {
    if (hora < 18) {
        console.log("Buenas
tardes");
    } else {
        console.log("Buenas
noches");
```



### Condicionales

Switch

```
\square \times
switch (expresion) {
    case valor1:
        // código a ejecutar si expresion es igual a valor1
        break:
    case valor2:
        // código a ejecutar si expresion es igual a valor2
        break;
    // más casos según sea necesario
    default:
        // código a ejecutar si ninguno de los casos anteriores es verdadero
```



#### Condicionales

Switch

```
- \square \times
let diaSemana = 3;
let mensaje;
switch (diaSemana) {
    case 1:
        mensaje = "Lunes";
        break;
    case 2:
        mensaje = "Martes";
        break;
    case 3:
        mensaje = "Miércoles";
        break;
    default:
        mensaje = "Día no válido";
console.log(mensaje);
```



For convariables de control, anidados, Continue, y con etiquetas

Variables de control Este ejemplo utiliza dos variables de control (i y j) con diferentes condiciones y expresiones de incremento/decremento.

**Anidado** Un bucle for dentro de otro. Esto puede ser útil al trabajar con matrices bidimensionales o realizar operaciones en matrices anidadas.

**Continue** En este caso, se utiliza conti<mark>nue para omitir la ejecución del bloque de código</mark> restante en una iteración si se cumpl<mark>e la condición.</mark>

**Etiquetas** El uso de etiquetas permite salir de bucles específicos desde dentro de bucles anidados, en este caso, se utiliza break outerLoop para salir de ambos bucles cuando se cumple cierta condición



For convariables de control, anidados, con saltos, y con etiquetas

```
- □ ×

// Imprimir los números del 1
al 5
for (let i = 1; i ≤ 5; i++) {
   console.log(i);
}

console.log(i);
}

for (let i = 0, j = 10; i < 5; i++, j -= 2) {
   console.log(`i: ${i}, j: ${j}`);
}
```

```
- □ ×

for (let i = 1; i ≤ 3; i++) {
    console.log(`Iteración externa ${i}`);

    for (let j = 1; j ≤ 2; j++) {
        console.log(` Iteración interna ${j}`);
    }
}
```



For convariables de control, anidados, con saltos, y con etiquetas

```
for (let i = 1; i ≤ 10; i++) {
   if (i % 2 == 0) {
      continue; // Saltar la iteración si es
   un número par
   }
   console.log(i);
}
```

```
outerLoop: for (let i = 0; i < 3; i \leftrightarrow) {
    console.log(`Iteración externa ${i}`);
    innerLoop: for (let j = 0; j < 3; j \leftrightarrow ) {
        if (i \equiv 1 & j \equiv 1) {
             break outerLoop; // Salir de ambos
bucles si se cumple la condición
        console.log(` Iteración interna ${j}`);
```



For in of, for - in

```
let colors = ["red", "green", "blue"];
for (let color of colors) {
   console.log(color);
}
```

```
let person = { name: "John", age: 30,
job: "developer" };

for (let key in person) {
   console.log(key + ": " +
   person[key]);
}
```



while

```
let limit = 10;
let counter = 1;

while (counter ≤ limit) {
    console.log(counter);
    counter++;
    // Puedes modificar la condición durante la ejecución
    limit = Math.floor(Math.random() * 20) + 1;
}
```

//En este ejemplo, la condición del bucle while se verifica al final de cada iteración, y además, el límite (limit) es dinámico y se puede modificar durante la ejecución del bucle



## Bucles y ciclos while

```
let userInput;
let isValid;
do {
   userInput = prompt("Ingresa un número entre 1 y
100:");
    isValid = !isNaN(userInput) & userInput ≥ 1 &
userInput ≤ 100;
} while (!isValid);
console.log(`¡Has ingresado un número válido:
${userInput}!`);
```

# </Bea <pre>Code()