Estructuras de control: Bifurcaciones, comparaciones y bucles

Las estructuras de control permiten dirigir el flujo de ejecución del código basándose en condiciones.

Bifurcaciones: if, else if, else

La estructura if permite ejecutar un bloque de código si una condición es verdadera.

La estructura else permite ejecutar un bloque de código si la condición del if es falsa.

Se pueden encadenar múltiples condiciones con else if.

```
if (condición) {
// Código a ejecutar si la condición es verdadera
} else if (otraCondición) {
// Código a ejecutar si otraCondición es verdadera
} else {
// Código a ejecutar si ninguna condición es verdadera
}
```

Bifurcaciones: switch

La sentencia switch permite ejecutar diferentes bloques de código dependiendo del valor de una expresión.

Cada caso (case) representa un valor posible de la expresión.

La sentencia break se utiliza para salir del switch después de ejecutar un caso.

El caso default se ejecuta si ninguno de los casos coincide con el valor de la expresión.

```
switch (expresión) {
case valor1:
    // Código a ejecutar si expresión === valor1
    break;
case valor2:
    // Código a ejecutar si expresión === valor2
    break;
default:
    // Código a ejecutar si no coincide ningún caso
}
```

Comparaciones

Los operadores de comparación se utilizan para comparar valores:

```
== :  Igualdad débil (solo compara el valor).
=== :  Igualdad estricta (compara el valor y el tipo).
!= :  Desigualdad débil.
!== :  Desigualdad estricta.
> :  Mayor que.
* :  Menor que.
>= :  Mayor o igual que.
<= :  Menor o igual que.
```

Bucles

Los bucles permiten repetir un bloque de código varias veces .

for:

Se utiliza cuando se conoce el número de iteraciones .

```
for (inicialización; condición; actualización) {
// Código a ejecutar en cada iteración
}
```

inicialización: Se ejecuta al principio del bucle (ej., declarar una variable contador).
condición: Se evalúa antes de cada iteración. El bucle se ejecuta mientras la condición sea verdadera.
actualización: Se ejecuta al final de cada iteración (ej., incrementar el contador).

while:

Se utiliza cuando no se conoce el número de iteraciones y se quiere repetir el bloque de código mientras una condición sea verdadera.

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

do...while:

Similar al while, pero el bloque de código se ejecuta al menos una vez, incluso si la condición es falsa desde el principio.

```
do {
// Código a ejecutar al menos una vez
} while (condición);
```

Ámbito de los bucles:

Es importante entender cómo las variables declaradas dentro de un bucle (con var, let o const) son accesibles fuera del bucle.

let y const tienen un ámbito de bloque, mientras que var tiene un ámbito de función.