9. Condicionales

Las **condicionales** son estructuras de control que permiten ejecutar bloques de código específicos basados en una condición o conjunto de condiciones. En este capítulo, exploraremos las principales formas de implementar condicionales en JavaScript: `if`, `else`, `else if`, el **operador ternario**, y la estructura `switch-case`.

---

1. Estructura `if`

La declaración `if` evalúa una condición y ejecuta un bloque de código si la condición es verdadera (`true`).

Sintaxis:

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

Ejemplo:

```
let edad = 18;

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

---

2. Estructura `if...else`

La declaración `if...else` permite ejecutar un bloque de código si la condición es verdadera y otro bloque si es falsa.

Sintaxis:

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

Ejemplo:

```
let hora = 14;

if (hora < 12) {
    console.log("Buenos días");
```

```
} else {
    console.log("Buenas tardes");
}
// Salida: "Buenas tardes"
```

3. Estructura `if...else if...else`

Cuando hay múltiples condiciones a evaluar, puedes usar `else if`. Esta estructura permite verificar varias condiciones en secuencia hasta que una sea verdadera.

Sintaxis:

```
if (condición1) {
    // Código si condición1 es verdadera
} else if (condición2) {
    // Código si condición2 es verdadera
} else {
    // Código si ninguna condición es verdadera
}
```

Ejemplo:

```
let calificacion = 85;

if (calificacion >= 90) {
    console.log("Excelente");
} else if (calificación >= 70) {
    console.log("Aprobado");
} else {
    console.log("Suspendido");
}
// Salida: "Aprobado"
```

4. Operador Ternario

El **operador ternario** es una forma compacta de escribir una condicional simple. Evalúa una condición y devuelve un valor si es verdadero o otro si es falso.

Sintaxis:

```
condición ? valorSiVerdadero : valorSiFalso;
```

Ejemplo:

```
let edad = 20;
let mensaje = edad >= 18 ? "Mayor de edad" : "Menor de edad";
```

```
console.log(mensaje); // "Mayor de edad"
```

5. Estructura `switch-case`

La declaración `switch-case` es útil cuando necesitas comparar un valor con múltiples casos posibles. Es más legible que usar múltiples `else if` cuando se trata de comparaciones directas.

Sintaxis:

```
switch (valor) {
  case valor1:
    // Código para valor1
    break;
  case valor2:
    // Código para valor2
    break;
  default:
    // Código si ningún caso coincide
}
```

- `break`: Finaliza el bloque `case` actual y evita que se ejecuten los siguientes.
- `default`: Se ejecuta si ningún caso coincide con el valor.

Ejemplo:

```
let dia = "lunes";

switch (dia) {
  case "lunes":
    console.log("Hoy es lunes.");
    break;
  case "martes":
    console.log("Hoy es martes.");
    break;
  default:
    console.log("No es lunes ni martes.");
}
// Salida: "Hoy es lunes."
```

6. Uso de `break` y `default`

- `break`: Es crucial incluir `break` al final de cada caso para evitar que el programa siga ejecutando los casos siguientes (caída libre).
- `default`: Proporciona una opción por defecto si ningún caso coincide con el valor evaluado.

Ejemplo sin `break`:

```
let numero = 2;

switch (numero) {
   case 1:
      console.log("Uno");
   case 2:
      console.log("Dos");
   case 3:
      console.log("Tres");
}
// Salida:
// "Dos"
// "Tres"
```

En este ejemplo, omitir `break` hace que todos los casos después del caso coincidente se ejecuten.