INSTITUTO KHIPU

Semestre IV

Sesión 06

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

COMUNITY MANAGER Y MARKETING DIGITAL

Tema

ESTRUCTURAS DE CONTROL (IF, ELSE, SWITCH) EN JAVASCRIPT

Estructuras de Control (if, else, switch) en JavaScript

1. Introducción a las estructuras de control

En la programación, las estructuras de control permiten tomar decisiones en función de condiciones específicas. En JavaScript, existen dos tipos principales de estructuras de control condicional:

- `if, else if, else`: Se usan para evaluar condiciones y ejecutar bloques de código según se cumplan o no.
- `switch`: Es útil cuando se deben evaluar múltiples condiciones relacionadas con un solo valor.

Las estructuras de control son fundamentales para la lógica de cualquier aplicación, ya que permiten modificar el flujo de ejecución de un programa dependiendo de diferentes escenarios.

---

2. Sintaxis y uso de if, else if, else

2.1 Sentencia `if`

Se usa cuando se desea ejecutar un bloque de código solo si una condición es verdadera.

📌 **Sintaxis:**

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

📌 **Ejemplo:**

```
let edad = 18;


if (edad >= 18) {
```

```
    console.log("Eres mayor de edad.");

}
```

2.2 Sentencia `if - else`

Si la condición del `if` es falsa, se ejecuta el bloque dentro de `else`.

📌 Ejemplo:

```
let temperatura = 30;


if (temperatura > 25) {
    console.log("Hace calor.");
} else {
    console.log("Hace frío.");
}
```

2.3 Sentencia `if - else if - else`

Se usa cuando hay múltiples condiciones a evaluar en secuencia.

📌 Ejemplo:

```
let nota = 85;


if (nota >= 90) {
    console.log("Tu calificación es A.");
} else if (nota >= 80) {
    console.log("Tu calificación es B.");
} else if (nota >= 70) {
    console.log("Tu calificación es C.");
```

```
} else {
    console.log("Has reprobado.");
}
```

---

3. Evaluación de condiciones (true y false)

Las condiciones dentro de `if` pueden ser expresiones que evalúan un resultado `true` o `false`. En JavaScript, los valores que se consideran `false` son:

- `0`
- `""` (cadena vacía)
- `null`
- `undefined`
- `NaN`
- `false`

📌 Ejemplo de evaluación de valores falsos:

```
if (0) {
    console.log("Esto no se ejecuta.");
} else {
    console.log("0 es considerado falso.");
}
```

---

4. Uso de operadores lógicos (`&&`, `||`, `!`) dentro de estructuras de control

Los operadores lógicos permiten combinar condiciones dentro de una estructura `if`.

📌 Operadores lógicos en JavaScript:

- `&&` (AND): Evalúa `true` solo si ambas condiciones son verdaderas.
- `||` (OR): Evalúa `true` si al menos una condición es verdadera.
- `!` (NOT): Invierte el valor de una expresión lógica.

📌 **Ejemplos:**

```
let usuario = "admin";
let clave = "12345";

if (usuario === "admin" && clave === "12345") {
    console.log("Acceso permitido.");
} else {
    console.log("Acceso denegado.");
}
let clima = "lluvioso";
let tieneParaguas = false;

if (clima === "lluvioso" || tieneParaguas) {
    console.log("Puedes salir sin mojarte.");
} else {
    console.log("Mejor lleva un paraguas.");
}
let activo = true;

if (!activo) {
    console.log("El usuario está inactivo.");
} else {
    console.log("El usuario está activo.");
}
```

5. Introducción a switch y su utilidad cuando hay múltiples opciones

Cuando hay muchas condiciones que dependen del valor de una misma variable, el uso de `switch` puede hacer el código más claro y eficiente que múltiples `if - else if`.

📌 Sintaxis básica de `switch`:

```
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 hay coincidencia
}
```

📌 Ejemplo de uso de `switch`:

```
let dia = "lunes";

switch (dia) {
    case "lunes":
        console.log("Inicio de semana.");
        break;
    case "viernes":
```

```
        console.log("Casi fin de semana.");
        break;
    case "sábado":
    case "domingo":
        console.log("Es fin de semana.");
        break;
    default:
        console.log("Día normal.");
}
```

📌 **Ventaja del `switch` sobre `if - else` en estos casos:**

- Hace el código más legible cuando se evalúan muchas opciones.
- Puede ser más eficiente en ciertos escenarios.

---

6. Ejemplo de operaciones matemáticas y comparaciones en JavaScript

Para ver el uso práctico de estructuras de control, supongamos que tenemos un programa que solicita dos números y permite elegir una operación matemática básica (`+`, `-`, `*`, `/`).

📌 **Ejemplo completo:**

```
let num1 = 10;
let num2 = 5;
let operacion = "*";

switch (operacion) {
    case "+":
        console.log(`Resultado: ${num1 + num2}`);
```

```
        break;
    case "-":
        console.log(`Resultado: ${num1 - num2}`);
        break;
    case "*":
        console.log(`Resultado: ${num1 * num2}`);
        break;
    case "/":
        if (num2 !== 0) {
            console.log(`Resultado: ${num1 / num2}`);
        } else {
            console.log("Error: No se puede dividir por cero.");
        }
        break;
    default:
        console.log("Operación no válida.");
}
```

📌 Salida esperada si `operacion` es `*`:

```
Resultado: 50
```

📌 Explicación del código:

- `switch` evalúa la operación matemática ingresada.
- Se usa `if` dentro de `case "/"` para evitar la división por cero.
- Si la operación no coincide con `+, -, *, /`, se ejecuta `default`.

Conclusión

- `if, else if, else` permiten tomar decisiones basadas en condiciones.
- Se pueden evaluar valores booleanos y operadores lógicos dentro de las estructuras de control.
- `switch` es útil cuando se deben evaluar múltiples valores de una variable.
- Las estructuras de control son esenciales para el desarrollo de lógica en JavaScript.

INSTITUTO

KHIPU