Condiciones en JavaScript 

Objetivos

Comprender que son las condiciones

Sintaxis de una condición

Operadores lógicos y de comparación

¿Qué son las condiciones?

En JavaScript, las "condiciones" se refieren a expresiones que se evalúan como verdaderas o falsas. Estas condiciones se utilizan para controlar el flujo del programa y tomar decisiones en base a ciertas situaciones.

Las condiciones en JavaScript generalmente involucran operadores de comparación, operadores lógicos y valores booleanos. Los valores booleanos pueden ser "true" (verdadero) o "false" (falso).

Existen varias formas de evaluar si un valor es falso o verdadero si cumple o no una condición instrucciones como if, else, else if, operadores ternarios y switch-case nos ayudan a realizar estas operaciones.

Instrucción if

La instrucción "if" es una de las estructuras de control más fundamentales en cualquier lenguaje de programación, incluido JavaScript. Permite tomar decisiones y ejecutar diferentes bloques de código en función de una condición que se evalúa como verdadera o falsa.

La sintaxis básica de la instrucción "if" en JavaScript es la siguiente:

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

Instrucción "else"

La cláusula "else" es una parte de la instrucción "if" en JavaScript que se utiliza para proporcionar un bloque de código alternativo que se ejecutará cuando la condición del "if" sea falsa. En otras palabras, cuando la expresión evaluada en el "if" resulte en un valor falso, se ejecutará el bloque de código dentro de la cláusula "else".

La sintaxis básica de la instrucción "if" con la cláusula "else" es la siguiente:

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

Instrucción "else if"

La instrucción "else if" es una extensión de la instrucción "if" en JavaScript que se utiliza para agregar múltiples condiciones a una serie de decisiones. Permite evaluar varias condiciones en orden y ejecutar diferentes bloques de código según la primera condición que sea verdadera.

La sintaxis básica de la instrucción "if" con "else if" es la siguiente:

```
if (condicion1) {
  ejecutar si la condición1 es verdadera
} else if (condicion2) {
  ejecutar si condicion1 es falsa y condicion2 es verdadera
} else if (condicion3) {
  ejecutar si condicion1 y condicion2 son falsas y condicion3 es true
} else {
  ejecutar si ninguna de las condiciones anteriores es verdadera
}
```

Operador ternario

Los operadores ternarios, también conocidos como operadores condicionales, son una forma abreviada de la instrucción "if" en JavaScript. Permiten tomar decisiones basadas en una condición y asignar un valor a una variable en función de si la condición es verdadera o falsa.

La sintaxis del operador ternario es la siguiente:

```
condicion ? valorSiVerdadero : valorSiFalso;
```

Ejemplo con código:

```
let edad = 20;
let mensaje = edad >= 18 ? "Eres mayor de edad" : "Eres menor de edad";
Imprimirá "Eres mayor de edad"
console.log(mensaje);
```

Los operadores ternarios son útiles cuando deseas asignar un valor a una variable de manera concisa, en función de una condición simple. Sin embargo, es importante usarlos con moderación, ya que pueden dificultar la legibilidad del código si se utilizan en exceso o en situaciones complejas.

switch-case

La estructura "switch-case" es otra forma de control de flujo en JavaScript que se utiliza para tomar decisiones basadas en el valor de una expresión. Es una alternativa a la instrucción "if" y se utiliza cuando se tienen múltiples casos posibles y se desea evitar el uso de múltiples instrucciones "if" o "else if".

Ejemplo con código:

```
switch (expresion) {
  case valor1:
  Código a ejecutar si la expresión es igual a valor1
    break;
  case valor2:
  Código a ejecutar si la expresión es igual a valor2
    break;
    Puedes tener tantos casos como necesites...
  default:
  ejecutar si la expresión no coincide con ninguno de los valores
}
```

Es importante mencionar que, después de ejecutar el bloque de código de un "case", se utiliza la palabra clave "break" para salir del switch y evitar que se ejecuten otros casos. Si se omite el "break", JavaScript continuará ejecutando los bloques de código de los casos siguientes, incluso si no coinciden con la expresión.

Ejemplo con código:

```
let diaSemana = "miércoles";

switch (diaSemana) {
  case "lunes":
    console.log("Hoy es lunes");
    break;
  case "martes":
    console.log("Hoy es martes");
    break;
  case "miércoles":
    console.log("Hoy es miércoles");
    break;
  case "jueves":
    console.log("Hoy es jueves");
    break;
  case "viernes":
    console.log("Hoy es viernes");
    break;
  default:
    console.log("Es fin de semana");
}
```

Operadores de comparación

En JavaScript, los operadores de comparación se utilizan para comparar valores y expresiones y evaluar si son iguales, diferentes, mayores, menores o similares. Estos operadores devuelven un valor booleano (true o false) que indica si la comparación es verdadera o falsa.

Los operadores de comparación son los siguientes:

Igualdad (==):
Comprueba si dos valores son iguales, después de realizar una conversión de tipo si es necesario.

Desigualdad (!=):
Comprueba si dos valores no son iguales, después de realizar una conversión de tipo si es necesario.

Igualdad estricta (===):
Comprueba si dos valores son iguales y tienen el mismo tipo de dato. No realiza conversión de tipo.

Desigualdad estricta (!==):
Comprueba si dos valores no son iguales o no tienen el mismo tipo de dato. No realiza conversión de tipo.

Mayor que (>):
Comprueba si el valor de la izquierda es mayor que el valor de la derecha.

Menor que (<):
Comprueba si el valor de la izquierda es menor que el valor de la derecha.

Mayor o igual que (>=):
Comprueba si el valor de la izquierda es mayor o igual que el valor de la derecha.

Menor o igual que (<=):
Comprueba si el valor de la izquierda es menor o igual que el valor de la derecha.

Por ejemplo:

```
let a = 5;
let b = 10;

console.log(a == b); // false
console.log(a != b); // true
console.log(a === b); // false
console.log(a !== b); // true
console.log(a > b); // false
console.log(a < b); // true
console.log(a >= b); // false
console.log(a <= b); // true
```

Es importante tener en cuenta que el operador de igualdad simple (==) realiza una conversión de tipo implícita, lo que significa que intentará convertir los operandos a un tipo común antes de realizar la comparación. Por lo tanto, es recomendable usar el operador de igualdad estricta (===) cuando se desea evitar conversiones de tipo no deseadas y garantizar que los valores y los tipos sean idénticos.

Operadores de lógicos

En JavaScript, los operadores lógicos se utilizan para combinar o negar expresiones booleanas y producir nuevos valores booleanos (true o false). Los operadores lógicos permiten crear condiciones más complejas al trabajar con múltiples expresiones.

Los operadores lógicos en JavaScript son los siguientes:

AND lógico (&&):
Devuelve true si ambas expresiones que se evalúan son verdaderas. Si al menos una de las expresiones es falsa, devuelve false.

OR lógico (||):
Devuelve true si al menos una de las expresiones que se evalúan es verdadera. Si ambas expresiones son falsas, devuelve false.

NOT lógico (!):
Niega una expresión booleana. Si la expresión es true, el operador ! la convierte en false, y si la expresión es false, el operador ! la convierte en true.

Ejemplos de operadores lógicos

```
let x = 5;
let y = 10;
Operador AND (&&)
console.log(x < 10 && y > 5); // true, ambas expresiones son verdaderas
console.log(x > 10 && y > 5); // false, la primera expresión es falsa

Operador OR (||)
console.log(x < 10 || y > 5);true, cuando uno de los valores verdadero
console.log(x > 10 || y > 15);false, ambas expresiones son falsas

Operador NOT (!)
let isTrue = true;
let isFalse = false;

console.log(!isTrue);false, se niega el valor true
console.log(!isFalse);true, se niega el valor false
```

Ahora que ya sabemos lo esencial en cuanto a validaciones, vamos a realizar algunos ejercicios, mucha suerte...!!