Operadores, condicionales y bucles Javascript

Aprende a utilizar los operadores, condicionales y bucles en Javascript

Empezar

Descripción general

En este curso aprenderás a utilizar los operadores, condicionales y bucles en Javascript para controlar el flujo de ejecución de tus programas. Conocerás las diferentes formas de realizar comparaciones y tomar decisiones, así como la utilización de bucles para repetir tareas. ¡Mejora tus habilidades de programación con Javascript!

01 Aritméticos

01 | Operadores aritméticos

En JavaScript, los operadores aritméticos se utilizan para realizar operaciones matemáticas en diferentes valores. Estos operadores nos permiten realizar acciones tales como sumar, restar, multiplicar y dividir. También nos permiten obtener el residuo de una división y realizar operaciones de incremento y decremento.

Suma

El operador de suma (`+`) se utiliza para sumar dos valores y devolver el resultado. Por ejemplo:

```
 let x = 5;
let y = 3;
let suma = x + y; // resultado: 8
```

En este ejemplo, declaramos dos variables `x` e `y` y las sumamos utilizando el operador de suma (`+`), almacenando el resultado en la variable `suma`.

Resta

El operador de resta (`-`) se utiliza para restar un valor de otro y devolver el resultado. Por ejemplo:

```
 let x = 5;
let y = 3;
let resta = x - y; // resultado: 2
```

En este ejemplo, restamos el valor de `y` a `x` utilizando el operador de resta (`-`), almacenando el resultado en la variable `resta`.

Multiplicación

El operador de multiplicación (`*`) se utiliza para multiplicar dos valores y devolver el resultado. Por ejemplo:

```
 let x = 5;
let y = 3;
let multiplicacion = x * y; // resultado: 15
```

En este ejemplo, multiplicamos los valores de `x` e `y` utilizando el operador de multiplicación (`*`), almacenando el resultado en la variable `multiplicacion`.

División

El operador de división (`/`) se utiliza para dividir un valor entre otro y devolver el resultado. Por ejemplo:

```
 let x = 10;
let y = 2;
let division = x / y; // resultado: 5
```

En este ejemplo, dividimos el valor de `x` entre `y` utilizando el operador de división (`/`), almacenando el resultado en la variable `division`.

Residuo de división

El operador de residuo de división (`%`) se utiliza para obtener el resto de una división entre dos valores. Por ejemplo:

```
 let x = 10;
let y = 3;
let resto = x % y; // resultado: 1
```

En este ejemplo, obtenemos el resto de la división de `x` entre `y` utilizando el operador de residuo de división (`%`), almacenando el resultado en la variable `resto`.

Incremento y decremento

Los operadores de incremento (`++`) y decremento (`--`) se utilizan respectivamente para aumentar o disminuir en 1 el valor de una variable. Estos operadores pueden utilizarse antes (`++x`, `--x`) o después (`x++`, `x--`) del nombre de la variable, con diferencias sutiles en su efecto. Por ejemplo:

```
 let x = 5;
x++; // x ahora es igual a 6
let y = 10;
--y; // y ahora es igual a 9
```

En este ejemplo, utilizamos el operador de incremento (`++`) para aumentar en 1 el valor de `x`, y el operador de decremento (`--`) para disminuir en 1 el valor de `y`.

Los operadores aritméticos en JavaScript nos permiten realizar una variedad de operaciones matemáticas en nuestros programas. Al comprender y utilizar correctamente estos operadores, podemos realizar cálculos y manipulaciones numéricas de manera eficiente y precisa.

Conclusión - Operadores aritméticos

En resumen, los operadores aritméticos en Javascript nos permiten realizar diferentes operaciones matemáticas, como sumar, restar, multiplicar y dividir. Con estos operadores, podemos realizar cálculos y obtener resultados numéricos precisos. Es importante tener en cuenta la precedencia de los operadores y utilizar paréntesis para controlar el orden de las operaciones. Además, los operadores de comparación en Javascript nos permiten comparar valores y evaluar si una expresión es verdadera o falsa. Podemos utilizar operadores como igualdad, desigualdad, mayor que, menor que, entre otros. Estos operadores nos resultan útiles en la creación de condiciones y toma de decisiones en nuestros programas. Por último, los condicionales if-else nos permiten controlar el flujo de ejecución de nuestro código según una condición determinada. Podemos ejecutar diferentes bloques de código dependiendo de si la condición es verdadera o falsa. Los condicionales if-else nos brindan flexibilidad y nos permiten crear aplicaciones más interactivas y personalizadas. En conclusión, el dominio de los operadores, condicionales y bucles en Javascript es fundamental para poder crear programas eficientes y robustos. Estas herramientas nos permiten controlar el flujo de ejecución y tomar decisiones en nuestras aplicaciones. A través de la práctica y la experimentación, podremos utilizar estos

conceptos de manera efectiva y potenciar nuestras habilidades como desarrolladores de Javascript.

Operadores de comparación

02 | Operadores de comparación

Los operadores de comparación son herramientas fundamentales en JavaScript que nos permiten comparar valores y determinar si dos valores son iguales, diferentes, mayores o menores. Estos operadores devuelven un valor booleano (verdadero o falso) dependiendo del resultado de la comparación.

En JavaScript, tenemos los siguientes operadores de comparación:

Operador de igualdad (==)

El operador de igualdad compara dos valores y devuelve `true` si son iguales, o `false` si son diferentes. Aunque este operador hace una comparación débil, es decir, no tiene en cuenta el tipo de dato, es importante tener en cuenta que puede haber algunas peculiaridades a la hora de realizar comparaciones.

Por ejemplo:

```
 console.log(5 == 5); // true
console.log('5' == 5); // true
console.log(5 == '5'); // true
console.log(5 == 10); // false
```

Operador de desigualdad (!=)

El operador de desigualdad compara dos valores y devuelve `true` si son diferentes, o `false` si son iguales. Al igual que el operador de igualdad, este operador también realiza una comparación débil.

Por ejemplo:

```
 console.log(5 != 10); // true
console.log('Hello' != 'hello'); // true
console.log(5 != '5'); // false
```

Operador de igualdad estricta (===)

El operador de igualdad estricta compara dos valores y devuelve `true` si son iguales y tienen el mismo tipo de dato, o `false` si son diferentes en algún aspecto. A diferencia del operador de igualdad, este operador realiza una comparación fuerte, teniendo en cuenta el tipo de dato.

Por ejemplo:

```
 console.log(5 === 5); // true
console.log('5' === 5); // false
console.log(5 === '5'); // false
```

Operador de desigualdad estricta (!==)

El operador de desigualdad estricta compara dos valores y devuelve `true` si son diferentes o tienen diferentes tipos de dato, o `false` si son iguales en ambos aspectos. Al igual que el operador de igualdad estricta, este operador realiza una comparación fuerte.

Por ejemplo:

```
 console.log(5 !== '5'); // true
console.log(5 !== 5); // false
console.log('Hello' !== 'hello'); // true
```

Otros operadores de comparación

Además de los operadores de igualdad y desigualdad, también tenemos los siguientes operadores de comparación:

- Operador mayor que (>) - devuelve `true` si el valor de la izquierda es mayor que el de la derecha.
- Operador menor que (<) - devuelve `true` si el valor de la izquierda es menor que el de la derecha.
- Operador mayor o igual que (>=) - devuelve `true` si el valor de la izquierda es mayor o igual que el de la derecha.
- Operador menor o igual que (<=) - devuelve `true` si el valor de la izquierda es menor o igual que el de la derecha.

Por ejemplo:

```
 console.log(5 > 2); // true
console.log(10 < 5); // false
console.log(5 >= 2); // true
console.log(5 <= 2); // false
```

Es importante entender cómo funcionan estos operadores de comparación, ya que son fundamentales para la toma de decisiones en JavaScript. Con ellos, podemos construir condicionales y bucles que nos permiten controlar el flujo de nuestro programa.

Condicionales if-else

03 | Condicionales if-else

¿Qué son los condicionales if-else?

Los condicionales son estructuras de control que permiten realizar diferentes acciones o decisiones basadas en una condición específica. En JavaScript, uno de los condicionales más comunes es el if-else.

La estructura básica del condicional if-else es la siguiente:

```
 if (condición) {
  // código que se ejecuta si la condición es verdadera
} else {
  // código que se ejecuta si la condición es falsa
}
```

El condicional if-else evalúa la condición proporcionada entre paréntesis y, si es verdadera, ejecuta el código dentro del bloque de código del if. Si la condición es falsa, se ejecuta el código dentro del bloque de código del else.

Es importante destacar que el bloque de código del else es opcional. Si solo se necesita ejecutar código cuando la condición sea verdadera, el bloque de código del else puede ser omitido.

Ejemplos de condicionales if-else

A continuación, se muestran algunos ejemplos prácticos de condicionales if-else en JavaScript:

Ejemplo 1: Verificar si un número es positivo o negativo

```
 let numero = 10;

if (numero > 0) {
  console.log("El número es positivo");
} else {
  console.log("El número es negativo");
}
```

En este ejemplo, se utiliza un condicional if-else para verificar si el número es positivo o negativo. Si el número es mayor que cero, se muestra el mensaje "El número es positivo". Si el número es menor o igual a cero, se muestra el mensaje "El número es negativo".

Ejemplo 2: Verificar si un número es par o impar

```
 let numero = 7;

if (numero % 2 === 0) {
  console.log("El número es par");
} else {
  console.log("El número es impar");
}
```

En este ejemplo, se utiliza un condicional if-else para verificar si el número es par o impar. La condición `numero % 2 === 0` verifica si el resto de la división del número entre 2 es igual a cero. Si esta condición es verdadera, se muestra el mensaje "El número es par". Si la condición es falsa, se muestra el mensaje "El número es impar".

Ejemplo 3: Verificar si un año es bisiesto

```
 let año = 2020;

if ((año % 4 === 0 && año % 100 !== 0) || año % 400 === 0) {
  console.log("El año es bisiesto");
} else {
  console.log("El año no es bisiesto");
}
```

En este ejemplo, se utiliza un condicional if-else para verificar si un año es bisiesto. La expresión `(año % 4 === 0 && año % 100 !== 0) || año % 400 === 0` evalúa dos condiciones: si el año es divisible por 4 y no es divisible por 100, o si el año es divisible por 400. Si esta expresión es verdadera, se muestra el mensaje "El año es bisiesto". Si la expresión es falsa, se muestra el mensaje "El año no es bisiesto".

Conclusiones

Los condicionales if-else son una herramienta fundamental en JavaScript para tomar decisiones y ejecutar diferentes acciones basadas en condiciones específicas. Con estos condicionales, es posible controlar el flujo de ejecución de un programa y adaptarlo a diferentes escenarios. El conocimiento y la comprensión de los condicionales if-else son fundamentales para el desarrollo de aplicaciones en JavaScript.

Ejercicios Practicos

Pongamos en práctica tus conocimientos

04 | Ejercicios Practicos

En esta lección, pondremos la teoría en práctica a través de actividades prácticas. Haga clic en los elementos a continuación para verificar cada ejercicio y desarrollar habilidades prácticas que lo ayudarán a tener éxito en el tema.

Suma y resta

Escribe un programa en Javascript que solicite al usuario dos números y muestre por consola la suma y resta de los mismos.

Mayor o menor

Escribe un programa en Javascript que solicite al usuario dos números y muestre por consola si el primero es mayor, menor o igual que el segundo.

Calificación de un examen

Escribe un programa en Javascript que solicite al usuario una calificación numérica (entre 0 y 100) y muestre por consola su equivalente en letras (A, B, C, D o F), de acuerdo con la siguiente escala: 90-100: A, 80-89: B, 70-79: C, 60-69: D, 0-59: F.

Resumen

Repasemos lo que acabamos de ver hasta ahora

05 | Resumen

- ✓ En resumen, los operadores aritméticos en Javascript nos permiten realizar diferentes operaciones matemáticas, como sumar, restar, multiplicar y dividir. Con estos operadores, podemos realizar cálculos y obtener resultados numéricos precisos. Es importante tener en cuenta la precedencia de los operadores y utilizar paréntesis para controlar el orden de las operaciones. Además, los operadores de comparación en Javascript nos permiten comparar valores y evaluar si una expresión es verdadera o falsa. Podemos utilizar operadores como igualdad, desigualdad, mayor que, menor que, entre otros. Estos operadores nos resultan útiles en la creación de condiciones y toma de decisiones en nuestros programas. Por último, los condicionales if-else nos permiten controlar el flujo de ejecución de nuestro código según una condición determinada. Podemos ejecutar diferentes bloques de código dependiendo de si la condición es verdadera o falsa. Los condicionales if-else nos brindan

flexibilidad y nos permiten crear aplicaciones más interactivas y personalizadas. En conclusión, el dominio de los operadores, condicionales y bucles en Javascript es fundamental para poder crear programas eficientes y robustos. Estas herramientas nos permiten controlar el flujo de ejecución y tomar decisiones en nuestras aplicaciones. A través de la práctica y la experimentación, podremos utilizar estos conceptos de manera efectiva y potenciar nuestras habilidades como desarrolladores de Javascript.

Prueba

Comprueba tus conocimientos respondiendo unas preguntas

06 | Prueba

Pregunta 1/6

¿Cuál es el resultado de la expresión 5 + 3?

- ◯ 8

- ◯ 7
- ◯ 9

Pregunta 2/6

¿Cuál es el resultado de la expresión 10 / 2?

- ◯ 5
- ◯ 2
- ◯ 8

Pregunta 3/6

¿Cuál es el resultado de la expresión 10 % 3?

- ◯ 1
- ◯ 2
- ◯ 3

Pregunta 4/6

¿Cuál es el resultado de la comparación 5 > 3?

- ◯ true
- ◯ false
- ◯ undefined

Pregunta 5/6

¿Cuál es el resultado de la comparación 10 === '10'?

- false
- true
- undefined

Pregunta 6/6

¿Cuál es el resultado de la expresión if (5 > 3) { console.log('Verdadero') } else { console.log('Falso') }?

- Verdadero
- Falso
- Ambos

Entregar

Conclusión

Felicidades!

¡Felicitaciones por completar este curso! Has dado un paso importante para desbloquear todo tu potencial. Completar este curso no se trata solo de adquirir conocimientos; se trata de poner ese conocimiento en práctica y tener un impacto positivo en el mundo que te rodea.

Comparte este curso