MATERIAL EXTRA

Operadores JavaScript

growdev

OPERADORES JAVASCRIPT

Os operadores no JavaScript são **símbolos** ou **palavras-chave** que indicam uma operação a ser realizada em um ou mais valores (ou operandos). Eles desempenham um papel crucial na manipulação de dados e execução de instruções dentro de um algoritmo, permitindo a criação de expressões que processam informações e tomam decisões lógicas. Cada operador tem uma função específica e, ao utilizá-los, você consegue realizar cálculos matemáticos, atribuir valores a variáveis, comparar dados e tomar decisões lógicas em um programa.

Para que servem os operadores?

Os operadores são ferramentas que permitem manipular e transformar dados de acordo com a necessidade do algoritmo. Eles servem para realizar cálculos matemáticos, comparações entre valores, operações lógicas, entre outras tarefas essenciais em um programa. Sem eles, seria impossível processar informações e controlar o fluxo de execução do código de maneira eficiente e dinâmica.

Por que são tão importantes em um algoritmo?

Os operadores são fundamentais para a construção de algoritmos porque possibilitam a execução de operações que permitem que o programa tome decisões e manipule dados. Em essência, eles permitem que o código seja dinâmico e responsivo às condições e entradas fornecidas pelo usuário ou por outros componentes do sistema. Além disso, os operadores facilitam a organização e clareza do código, tornando-o mais legível e funcional.

Tipos básicos de operadores em JavaScript

Os operadores em JavaScript podem ser categorizados em diferentes tipos, cada um com uma função específica. Vamos explorar os principais tipos:

Operadores Aritméticos

Esses operadores são usados para realizar operações matemáticas básicas como adição, subtração, multiplicação e divisão. Eles são essenciais para cálculos e manipulação de valores numéricos.

Operador	Nome	Propósito	Exemplo
+	Adição	Soma dois valores	5 + 7 resultado: 12
-	Subtração	Subtrai o segundo valor do primeiro	10 - 2 resultado: 8
*	Multiplicação	Multiplica dois valores	9 * 3 resultado: 27
/	Divisão	Divide o primeiro valor pelo segundo	10 / 2 resultado: 5
%	Módulo (resto da divisão)	Retorna o resto da divisão, em números inteiros, do número da esquerda pelo número da direita.	8 % 3 resultado: 2
**	Exponenciação	Eleva o valor da esquerda à potência da direita	2 ** 3 resultado: 8

Operadores de Atribuição

Os operadores de atribuição são usados para atribuir valores a variáveis. O operador mais básico é o =, mas existem variações que combinam atribuição com operações aritméticas.

Operador	Nome	Propósito	Exemplo
=	Atribuição	Atribui um valor a uma variável	let x = 10
+=	Atribuição de soma	Executa a soma e atribui o resultado ao operando esquerdo	x += 5 equivalente à x = x + 5
-=	Atribuição de subtração	Executa a subtração e atribui o resultado ao operando esquerdo	x -= 5 equivalente à x = x - 5
***=**	Atribuição de multiplicação	Executa a multiplicação e atribui o resultado ao operando esquerdo	x *= 5 equivalente à x = x * 5
/=	Atribuição de divisão	Executa a divisão e atribui o resultado ao operando esquerdo	x /= 5 equivalente à x = x / 5
%=	Atribuição de restante	Executa o restante e atribui o resultado ao operando esquerdo	x %= 5 equivalente à x = x % 5

Operadores Relacionais

Os operadores relacionais são usados para comparar dois valores e retornam um valor booleano (`true` ou `false`). Eles são essenciais para a criação de condições e estruturas de controle no JavaScript.

Operador	Nome	Propósito	Exemplo
==	Igualdade	Retorna verdadeiro caso os operando sejam iguais	3 == '3'
===	Igualdade estrita	Retorna verdadeiro caso os operandos sejam iguais e do mesmo tipo.	3 === 3
!=	Desigualdade	Retorna verdadeiro caso os operandos não sejam iguais.	2 != "3"
!==	Desigualdade estrita	Retorna verdadeiro caso os operandos não sejam iguais e/ou não sejam do mesmo tipo.	3 !== "3"
>	Maior	Retorna verdadeiro caso o operando da esquerda seja maior que o da direita.	12 > 2
<	Menor	Retorna verdadeiro caso o operando da esquerda seja menor que o da direita.	2 < 4
>=	Maior ou igual	Retorna verdadeiro caso o operando da esquerda seja maior ou igual ao da direita.	10 >= 9
<=	Menor ou igual	Retorna verdadeiro caso o operando da esquerda seja menor ou igual ao da direita.	5 >= 4

Operadores Lógicos

Os operadores lógicos são utilizados para combinar expressões booleanas e são essenciais para a criação de condições complexas dentro do algoritmo.

Operador	Nome	Propósito	Exemplo
&&	AND	Retorna `true` se ambas as expressões forem verdadeiras	expressao1 && expressao2
||	OR	Retorna `true` se pelo menos uma das expressões for verdadeira	expressao1 || expressao2
!	NOT	Inverte o valor lógico de uma expressão booleana	!expressao

Tabela verdade

Tabelas da verdade são usadas para representar o resultado de todas as possíveis combinações de entradas em uma expressão lógica, onde A representa o valor booleano ao lado esquerdo da expressão, e B representa o valor booleano ao lado direito da expressão. Tabelas verdade podem ser úteis para visualização dos diferentes resultados de uma expressão lógica.

E (AND)			OU (OR)		
A	B	A && B	A	B	A || B
true	true	true	true	true	true
true	false	false	true	false	true
false	true	false	false	true	true
false	false	false	false	false	false

Precedência de Operadores

Além de entender o que são operadores e como usá-los, é importante compreender a precedência de operadores, que define a ordem em que as operações são realizadas quando há múltiplos operadores em uma expressão. A precedência determina quais operações são executadas primeiro, influenciando o resultado final da expressão.

No JavaScript, cada operador tem um nível de precedência associado. Quando há vários operadores em uma mesma expressão, aqueles com precedência mais alta são executados primeiro. Por exemplo, a multiplicação e a divisão têm precedência maior do que a adição e a subtração. Isso significa que, em uma expressão como `3 + 5 * 2`, a multiplicação será executada antes da adição, resultando em `3 + (5 * 2) = 13`, e não em `(3 + 5) * 2 = 16`.

Consideremos a expressão:

```
let resultado = 10 + 5 * 2 - 8 / 4;
```

Aqui, a ordem de execução é a seguinte:

- A multiplicação (5 * 2), resultando em 10.
- A divisão (8 / 4), resultando em 2.
- A adição (10 + 10), resultando em 20.
- A subtração (20 - 2), resultando em 18.
- 18 será o valor da variável resultado.

Se quisermos alterar a ordem natural das operações, podemos usar parênteses, que têm a precedência mais alta e forçam a execução de operações antes das demais. Por exemplo:

```
let resultado = ((10 + 5) * (8 - 2)) / 3;
```

Neste caso, as operações dentro dos parênteses são executadas primeiro, seguidas pela multiplicação e pela divisão. Neste exemplo, a ordem de execução é a seguinte:

- A soma (10 + 5), resultando em 15.
- A subtração (8 - 2), resultando em 6.
- A multiplicação (15 * 6), resultando em 90.
- divisão (90 / 3), resultando em 30.
- 30 será o valor da variável resultado.

A ordem de precedência para os operadores aritméticos, relacionais, de atribuição e lógicos no JavaScript, se dá conforme numeração abaixo, do mais alto (com maior grau de precedência) ao mais baixo (com menor grau de precedência).

1. **Parênteses**
    a. Uma expressão entre parênteses sempre terá maior precedência.
    b. As expressões são resolvidas de dentro para fora, bem como na matemática, caso haja vários níveis de expressões dentro de parênteses.

2. **Operadores Aritméticos**
    a. Exponenciação (**)
    b. Multiplicação (*), Divisão (/) e Módulo (%)
    c. Adição (+) e Subtração (-)

3. **Operadores Relacionais**
    a. Comparação (<, <=, > e >=)
    b. Igualdade e Desigualdade (==, !=, === e !==)

4. **Operadores Lógicos**
    a. AND Lógico (&&)
    b. OR Lógico (||)

5. **Operadores de Atribuição (associativo da direita para a esquerda)**
    a. Atribuição Simples e Compostos (=, +=, -=, *=, /= e %=)

Esta ordem define como expressões combinando esses operadores serão avaliadas, permitindo estruturar corretamente operações matemáticas, comparações e atribuições em seu código. **Usar parênteses é uma boa prática para garantir a ordem de execução desejada**.

BONS ESTUDOS

growdev