Operadores Relacionais em JavaScript

1. OPERADORES RELACIONAIS: A BASE DA COMPARAÇÃO LÓGICA

Fala moçada, beleza? Na aula de hoje, vamos aprender sobre os **operadores relacionais** em JavaScript! Se você está começando sua jornada na programação, este é um dos conceitos mais importantes que você vai dominar. Operadores relacionais são ferramentas que usamos para realizar **comparações** entre dois valores. O resultado de qualquer comparação feita com eles será sempre um valor booleano, ou seja, `true` (verdadeiro) ou `false` (falso).

Esses operadores são a espinha dorsal da tomada de decisões em nossos programas. É através deles que podemos criar lógicas como "se um número for maior que outro, faça isso" ou "enquanto esta condição for verdadeira, continue executando aquilo". Dominá-los é fundamental para controlar o fluxo de execução do seu código. Em JavaScript, temos seis operadores relacionais principais:

- **`>`** - Maior
- **`>=`** - Maior ou igual
- **`<`** - Menor
- **`<=`** - Menor ou igual
- **`==`** - Igual
- **`!=`** - Diferente

Agora que você já conhece a lista, vamos ver como cada um deles funciona na prática. Show de bola?

1.1. Configuração do Ambiente e Demonstrações Práticas

Para testar nossos operadores relacionais, primeiro precisamos de alguns valores para comparar. A maneira mais simples de fazer isso é declarando algumas variáveis. Ao longo desta apostila, usaremos um cenário com três variáveis numéricas para ilustrar cada operação. Vamos inicializar nossas variáveis com os seguintes valores, que nos permitirão explorar todos os tipos de comparação:

```
let num1 = 10;
let num2 = 5;
let num3 = 10;
```

Aqui, a variável **`num um`** recebe o valor **10**, a variável **`num dois`** recebe **5**, e a **`num três`** também recebe o valor **10**. Essa configuração nos permite comparar valores diferentes (**num um** e **num dois**) e valores idênticos (**num um** e **num três**). Para visualizar o resultado de cada operação, usaremos a função **`console.log()`**, que exibe a saída (`true` ou `false`) diretamente no console do navegador ou do ambiente de desenvolvimento.
Vamos começar analisando os operadores de grandeza, que verificam se um valor é maior ou menor que outro.

1.2. Analisando Operadores de Grandeza: Maior e Menor

Os operadores de grandeza mais básicos são o **`>`** (Maior) e o **`<`** (Menor). Eles são usados para determinar, de forma direta, se um valor é numericamente superior ou inferior a outro.

1.2.1. Operador `>` (Maior)

Vamos usar este operador para comparar se o valor de **`num um`** é maior que o de **`num dois`**. A expressão lógica é `num1 > num2`.

```
console.log(num1 > num2);
```

A análise aqui é simples: estamos perguntando ao JavaScript "10 é maior que 5?". Como a afirmação é verdadeira, o console exibirá o resultado **`true`**.

1.2.2. Operador `<` (Menor)

Agora, vamos inverter a lógica e verificar se **`num um`** é menor que **`num dois`**, utilizando a expressão `num1 < num2`.

```
console.log(num1 < num2);
```

Neste caso, a pergunta é "10 é menor que 5?". Obviamente, essa afirmação é incorreta, então o resultado exibido no console será **`false`**. Esses operadores são muito úteis, mas e se quisermos incluir a possibilidade de os números serem iguais na nossa comparação? Para isso, temos os próximos operadores.

1.3. Explorando a Inclusão da Igualdade: >= e <=

Os operadores **`>=`** (Maior ou igual) e **`<=`** (Menor ou igual) adicionam uma camada extra à nossa lógica. Eles retornam `true` não apenas se a condição de maior/menor for atendida, mas também se os valores comparados forem exatamente iguais.

1.3.1. Operador >= (Maior ou igual)

Vamos comparar **`num um`** com **`num três`**. Sabemos que ambos têm o valor 10. Veja o que acontece:

```
console.log(num1 >= num3);
```

O que essa operação retorna pra gente, verdadeiro ou falso? Antes de continuar, coloca aí nos comentários, quero saber quem acertou! Pense um pouco antes de ver a resposta. O resultado desta operação é **`true`**. Por quê? A expressão é verdadeira porque o operador verifica se o valor é "Maior **OU** igual". Embora a parte 'Maior' seja falsa (**num um** não é maior que **num três**), a parte '**OU igual**' é verdadeira. Como uma das duas condições é atendida, a expressão inteira se torna verdadeira. Pegou a liga aí?

1.3.2. Operador <= (Menor ou igual)

O mesmo princípio se aplica ao operador de menor ou igual. Vamos testá-lo com as mesmas variáveis.

```
console.log(num1 <= num3);
```

Novamente, o resultado é **`true`**. A variável **`num um`** não é menor que **`num três`**, mas como ela é igual, a condição "menor **OU** igual" é validada. Compreender essa nuance é crucial. Agora, vamos abordar um ponto que causa muita confusão para iniciantes: a diferença entre comparar valores e atribuir um valor.

1.4. Distinção Crucial: Comparação (==) vs. Atribuição (=)

Um dos erros mais comuns ao começar a programar em JavaScript (e em muitas outras linguagens) é confundir o operador de atribuição (**`=`**) com o operador de comparação de igualdade (**`==`**). Dominar essa diferença é fundamental para evitar bugs que podem ser difíceis de rastrear.

1.4.1. Operador de Atribuição (=)

Um único sinal de igual (**`=`**) é usado exclusivamente para **atribuir um valor a uma variável**. Por exemplo, na linha `let num1 = 10;`, estamos dizendo ao programa para armazenar o valor `10` dentro da variável **`num um`**. Esta operação não compara nada e, portanto, não retorna `true` ou `false`.

1.4.2. Operador de Comparação (==)

Dois sinais de igual (**`==`**) são usados para **comparar se dois valores são iguais**. Esta operação, sim, resulta em `true` ou `false`. Vamos comparar **`num um`** e **`num três`** para ver isso em ação.

```
console.log(num1 == num3);
```

Como **`num um`** e **`num três`** ambos contêm o valor `10`, a comparação `10 == 10` resulta em **`true`**. Agora que sabemos como verificar se dois valores são iguais, vamos aprender a fazer o oposto: verificar se eles são diferentes.

1.5. Verificando a Diferença e a Negação Lógica: `!=` e `!`

O operador **`!=`** (Diferente) nos permite verificar se dois valores **não** são iguais. Ele é, em essência, um operador dedicado para a **negação da igualdade**, funcionando como o oposto lógico do operador `==`. É importante também não confundi-lo com o operador de negação lógica **`!`** (Not), que tem um papel diferente, mas relacionado.

1.5.1. Operador `!=` (Diferente)

Vamos usar este operador para verificar se **`num um`** é diferente de **`num três`**.

```
console.log(num1 != num3);
```

O resultado desta operação é **`false`**. A expressão está afirmando que "10 é diferente de 10", o que é uma declaração falsa. Portanto, o retorno é `false`.

1.5.2. Diferenciando do Operador `!` (Negação/Not)

O operador de negação **`!`** (conhecido como "Not") funciona de uma maneira diferente: ele **inverte um valor booleano**. Se uma expressão resulta em `true`, o operador `!` a transforma em `false`. Se resulta em `false`, ele a transforma em `true`. Veja como podemos obter o mesmo resultado do exemplo anterior usando a negação:

```
console.log(!(num1 == num3));
```

Vamos analisar essa expressão em partes:

1. Primeiro, o JavaScript resolve o que está dentro dos parênteses: `(num1 == num3)`. Como vimos, isso resulta em **`true`**.
2. Em seguida, o operador **`!`** é aplicado a esse resultado. Ele inverte `true` para **`false`**.

Para ver a inversão no outro sentido, vamos aplicar o `!` à nossa verificação de diferença:

```
console.log(!(num1 != num3));
```

Aqui a lógica é:

1. A expressão `(num1 != num3)` resulta em **`false`**.
2. O operador **`!`** inverte esse resultado para **`true`**.

Portanto, `!(num1 == num3)` é logicamente equivalente a `num1 != num3`, e `!(num1 != num3)` é logicamente equivalente a `num1 == num3`.

1.6. Tabela de Referência Rápida: Resumo dos Operadores Relacionais

Ao longo desta aula, vimos como os operadores relacionais são os blocos de construção fundamentais para criar programas com lógica e capacidade de decisão. Eles nos permitem comparar valores e direcionar o fluxo do nosso código com base nos resultados `true` ou `false`. Para consolidar seu aprendizado, aqui está uma tabela de resumo com todos os operadores que discutimos:

Operador	Nome	Descrição
`>`	Maior	Retorna `true` se o valor da esquerda for maior que o da direita.
`<`	Menor	Retorna `true` se o valor da esquerda for menor que o da direita.
`>=`	Maior ou igual	Retorna `true` se o valor da esquerda for maior ou igual ao da direita.
`<=`	Menor ou igual	Retorna `true` se o valor da esquerda for menor ou igual ao da direita.
`==`	Igual	Retorna `true` se os dois valores forem iguais.
`!=`	Diferente	Retorna `true` se os dois valores não forem iguais.

Continue praticando com esses operadores! Crie suas próprias variáveis e teste diferentes combinações. Essa base sólida será essencial para os próximos passos em sua jornada com JavaScript. Espero você na próxima aula! Um forte abraço e continue programando!