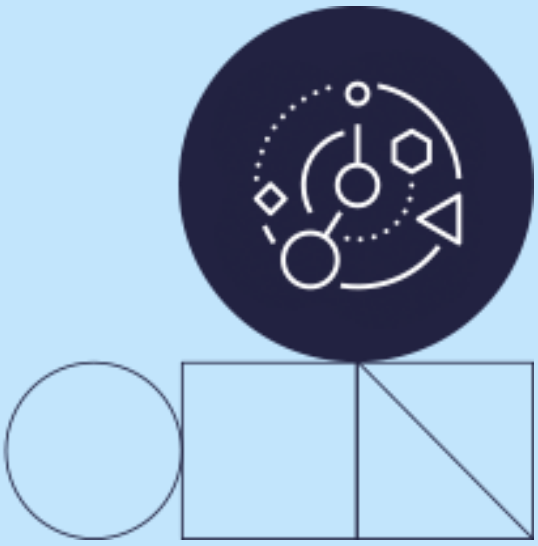
IBM SkillsBuild

IBM

Noções básicas de programação

Operadores

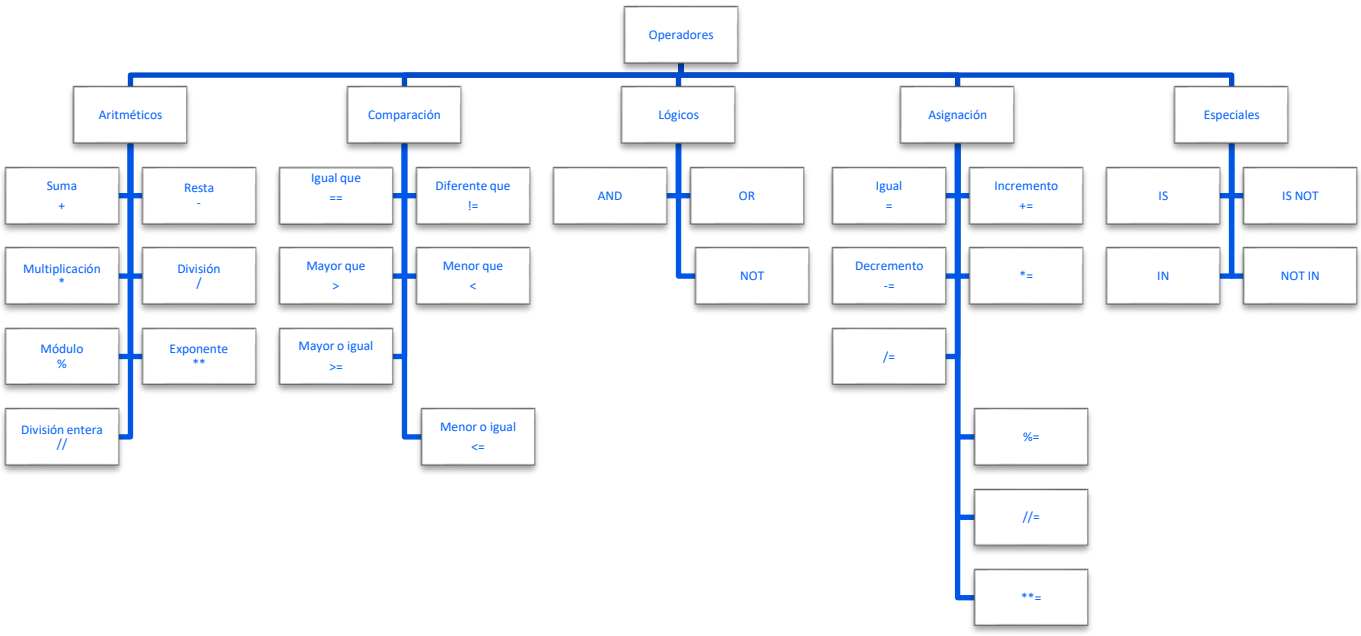


Índice

Introdução

Os operadores são fundamentais para ver as variáveis para fazer o código funcionar, pois com eles podemos comparar, atribuir, executar funções matemáticas, etc.

O diagrama a seguir lista os operadores comuns a quase todas as linguagens de programação de alto nível.



Operadores básicos

A fim de explicar os operadores, nos basearemos naqueles contidos na linguagem de programação Javascript.

No JavaScript temos os operadores aritméticos usuais em linguagens de programação como adição, subtração, multiplicação, divisão e o operador que retorna o restante de uma divisão entre inteiros (em outras linguagens chamadas mod operator ou modulus de uma divisão).

Embora em outras línguas exista um operador de exponenciação para calcular poderes, este não é o caso em JavaScript.

As operações com operadores seguem uma **ordem de precedência ou de precedência** que determina a ordem em que são realizadas. Com operadores matemáticos, a multiplicação e a divisão têm precedência sobre a adição e a subtração. Se houver expressões com vários operadores no mesmo nível, a operação é executada da esquerda para a direita. Para evitar resultados indesejados, nos casos em que possa haver dúvidas, recomenda-se o uso de parênteses para deixar claro em que ordem as operações devem ser executadas.

Operadores aritméticos

Operador	Função	Sintaxe
+	Soma	2+3
-	Subtrair	3-2
*****	Multiplicação	3*4
/	Divisão	10/5
%	O resto da divisão	11%5 valeria: 1

Nota: O operador "Restante de uma divisão" entre números inteiros em outros idiomas é chamado mod.

Operadores incremento y decréscimo

Operador	Função	Sintaxe
+=	X+=Y	X=X+Y
-=	X-=Y	X=X-Y
***=**	X*=Y	X=X*Y
/=	X/=Y	X=X/Y
++	Incremento	X++ Como dizer: X=X+1
--	Decréscimo	X-- Como dizer: X=X-1

++ e -- são válidos apenas para variáveis numéricas e servem para aumentar o valor da variável em uma unidade. Dependendo de onde são colocados (antes ou depois da variável) o resultado do cálculo pode diferir devido ao tempo da adição da unidade.

Os operadores **+=, -= e *=** são formas abreviadas de escrever operações comuns.

Note que **++, --, +=, -=, -=, -= e *=** são expressões que sempre se aplicam às variáveis. Por exemplo, não é válido escrever 2++ porque 2 não é uma variável. Todas estas operações podem ser substituídas por um equivalente mais óbvio. Muitos programadores preferem não usar esses operadores porque eles tornam o código menos legível. Outros programadores gostam de usá-los porque poupam a digitação.

Importante: Em relação ao operador de incremento.

variable++ não é o mesmo que ++variable.

Se colocarmos a variable++, primeiro consideramos o valor da variável e depois adicionamos 1 a ela.

Se colocarmos ++variable primeiro adicionamos 1 e depois consideramos o valor da variável.

No exemplo a seguir temos uma instrução que deve imprimir o valor de uma variável e depois adicionar 1 ao valor dessa variável. Não se preocupe se você não entender alguma da sintaxe, estas são palavras de linguagem que veremos mais tarde:

```
<script>
  var numero = 5;
  document.write(numero++);
  document.write("<br>");
  document.write(numero);
</script>
```

O resultado será que imprimirá 5 e depois 6. Ou seja, primeiro imprime o valor da variável, neste caso 5, depois adiciona 1 e na impressão seguinte a variável é 6, portanto imprime 6.

Mas se colocarmos:

```
<script>
  var numero = 5;
  document.write(++numero);
  document.write("<br>");
  document.write(numero);
</script>
```

Imprimirá 6 e depois 6. Como, primeiro faz a soma da variável+1, com o que a variável vale 6, ela a imprime e já permanece com este valor. Na próxima impressão, imprime novamente 6.

Operadores relacionais

Eles devolvem um booleano com o resultado da operação relacionada com os operandos.

Operador	Descrição
x==y	Fraca igualdade
x===y	Igualdade estrita (sem conversão de tipo)
Object.is	Método que determina se dois valores são o mesmo
x!=y	Desigualdade fraca
x!==y	Estrita desigualdade (sem conversão de tipo)
x>y	x é maior que y?
x<y	x é menor do que y?
x>=y	x é maior ou igual a y?
x<=y	x é menor ou igual a y?
in	Determina se um objeto tem um determinado bem
instance of	Determina se um objeto é uma instância de outro objeto

No JavaScript temos os operadores habituais em linguagens de programação como "é igual", "é diferente", menor, menor ou igual, maior, maior ou igual a, e (e), ou (ou) e não (não). A sintaxe é baseada em símbolos como veremos abaixo e é importante ter cuidado para não confundir == com = porque eles implicam em coisas diferentes.

Além da alocação tradicional baseada em =

O = operador é o operador de atribuição e deve ficar claro que ele não é usado para fazer comparações. Para fazer comparações, você deve usar == (é igual a) ou ==== (é estritamente igual a). A atribuição **a = b** diz: "atribui ao conteúdo de b". Se b for uma operação ou expressão lógica, a armazenará o valor numérico resultante da operação ou o valor booleano resultante da avaliação da expressão lógica.

Por exemplo, **a = 3 > 5** implicará que a é falso porque **3 > 5** é falso.

O operador ==== é interpretado como "é estritamente igual" e **!**== é interpretado como "não é estritamente igual". Estes operadores são um pouco mais complexos de entender, portanto, voltaremos a eles mais tarde. Por enquanto, tenha em mente que se uma variável contém **texto1= "1"** e nós fazemos a comparação:

texto1 ==== 1

Ficaremos falsos, ou seja, não é igual (porque um texto não é igual a um número). Entretanto, uma comparação como **texto == 1** retornará verdadeiro porque esta comparação não é rigorosa e tenta realizar conversões automáticas para verificar se uma equivalência pode ser estabelecida entre os dois valores. Neste caso, o equivalente numérico do texto é pesquisado e então a comparação é feita, e é por isso que a verdade é devolvida.

Operadores lógicos

Operador	Descrição
&&	AND Lógico
||	OR Lógico
!	NOT Lógico

Expressões usando operadores lógicos e relacionais retornam um valor booleano, ou seja, verdadeiro (*true*) ou falso (*false*).

Por exemplo, se **a = 7 e b = 5**, a expressão **a < b** retorna falsa. Se **a = true** e **b = false**, a expressão **a && b** retorna falso (é falso porque a e b não são verdadeiros). Se **a = true** e **b = false** a expressão **a || b** retorna verdadeiro porque um dos dois operandos é verdadeiro. Se **a = true** a expressão !a retorna *false* (o oposto).

O **||** operador é obtido na maioria dos teclados pressionando **ALT GR + 1**, ou seja, a tecla ALT GR e o número 1 simultaneamente.

Os **&& e ||** operadores são chamados operadores em curto-circuito porque se a condição de um termo não for cumprida, o resto da operação não é avaliada. Por exemplo:

(a == b && c != d && h >= k)

Tem três avaliações: a primeira verifica se a variável a é igual a b. Se esta condição não for cumprida, o resultado da expressão é falso e as outras duas condições subseqüentes não são avaliadas.

Em um caso como (a < b ||| c != d ||| h <= k) é avaliado se **a** é menor do que **b**. Se esta condição for cumprida, o resultado da expressão é verdadeiro e as outras duas condições pós-condição não são avaliadas.

O operador **!** não é recomendado até que você tenha alguma habilidade de programação. Uma expressão como **(!esVisible)** retorna *false* se **(esVisible == true)**, ou *true* se **(esVisible == false)**.

Em geral, existem expressões equivalentes que lhe permitem evitar o uso deste operador quando você quiser.