Javascript

Operadores

Operadores

- Existem diferentes operadores em Javascript.

- Vamos começar pelo operador unário **-**, que troca o sinal de um número:

```
const a = 1, b = -2; //b recebe -2
const c = -a; //c recebe -1
const d = -b; //d recebe 2
```

Operadores

- Outro operador unário é o +, que não possui efeito em números, mas converte valores não numéricos para números, de forma análoga à função Number:

```
const a = +true, b = +false;
const c = +"", d = +"7", e = +"a";
const f = +"5" + +"2";
//Valores das constantes:
//   a = 1, b = 0, c = 0
//   d = 7, e = NaN, f = 7
```

Operadores

- Alguns operadores binários (2 operandos) matemáticos suportados:

Operador	Exemplo Operação	Equivalente Matemática	Resultado
+	5+2	5+2	7
-	5-2	5-2	3
*	5*2	5x2	10
/	5/2	5÷2	2.5
%	5%2	5 | 2 1 2	1
**	5**2	5^2	25

Operadores

- Operador **+** utilizado para concatenar strings:

```
const str1 = "Teste";
const str2 = "isso";
const str3 = str1 + " " + str2; //Teste Isso
```

- Quando um dos operandos do operador **+** for uma string, se o outro operando for um número, este outro será convertido para string automaticamente.

```
const str = "5" + 5; //55
```

Conversões Implícitas ao Utilizar Operadores

Operação	Resultado
`5 - 2`	3
`'5' - '2'`	3
`5 + 2`	7
`'5' + 2`	52
`5 + '2'`	52

Operação	Resultado
`5 + 2 + '1'`	71
`'5' + 2 + 1`	521
`'5' / '2'`	2.5
`'5' * '2'`	10
`'5' % '2'`	1

Operador =

- O operador = também é um operador binário, ou seja, ele possui dois operandos e, como todo operador, retorna um valor.

- Este é o operador com baixíssima precedência, ou seja, as operações que utilizam os operadores apresentados anteriormente serão sempre efetuadas antes da operação =.

```
let n1 = 5;
let n2 = (n1 = 5 + 2) * 2; //n1=7, n2=14
```

Operadores que Atualizam a Mesma Variável

Para cada operação abaixo, considere a seguinte declaração de variável:

```
let n = 5, z = 5;
```

Operação	Resultado
`n++;`	n=6
`n--;`	n=4
`n += 2;`	n=7
`n -= 2;`	n=3
`n *= 2;`	n=10

Operação	Resultado
`n /= 2;`	n=2.5
`z = n++;`	z=5, n=6
`z = ++n;`	z=6, n=6
`z = n--;`	z=5, n=4
`z = --n;`	z=4, n=4

Operadores Relacionais

Os operadores relacionais, existentes na maioria das linguagens de programação, existem em Javascript:

Operdor	Operação
&&	AND
||	OR
!	NOT

Operador ,

- Um dos operadores de menor precedência é o operador ,.
- Por ter baixíssima precedência, sempre considere a possibilidade de utilizar parênteses para definir a precedência.
- Este operador executa cada expressão separada por vírgula e retorna somente o valor da última expressão.
- Exemplo:

```
let a = 5; //a=5
let b = (a = 5 * 2, a * a); //a=10, b=100
```

Comparações

Abaixo segue uma tabela contendo operadores que nos permitem realizar comparações em Javascript:

Operador em Javascript	Operador na Matemática
>	>
>=	≥
<	<
<=	≤
==	=
! =	≠

Comparações

- Exemplos:

```
const a = 5, b = 2;
const bMaiorQueA = b > a; //false
const bMenorQueA = b < a; //true
const bMais3MaiorOuIgualA = (b + 3) >= a; //true
const bIgualA = b == a; //false
const bDiferenteDeA = b != a; //true
```

Comparações

- Comparações de strings são feitas utilizando-se a tabela Unicode. Cada caractere da string da esquerda é comparado ao caractere da string da direita para encontrar um que seja maior ou menor que o outro, caso que encerra a comparação.
- Caso os caracteres sejam iguais, o algoritmo continua até que uma das strings chegue ao fim.
- Se todos os caracteres comparados forem iguais ao chegar ao fim de uma das strings, caso ambas tenham o mesmo tamanho, as strings são iguais. Caso contrário a string que contém mais caracteres é a maior.

Comparações

Exemplos

Operação	Resultado
`'A'>'B'`	`false`
`'A'<'B'`	`true`
`'A'<='B'`	`true`
`'A'<='A'`	`true`
`'A'=='B'`	`false`
`'A'!='B'`	`true`

Comparações

Cuidado, na tabela Unicode o ‘A’ possui o valor 65 enquanto o ‘a’ possui o valor ‘97’

Operação	Resultado
`'A'=='a'`	`false`
`'A'>'a'`	`false`
`'A'<'a'`	`true`
`'a'<'B'`	`false`
`'a'>'B'`	`true`
`'A'!='a'`	`true`

Comparações

Exemplos de strings com mais de um caractere

Operação	Resultado
`'Abc'>'Abcd'`	`false`
`'Abc'<'Abcd'`	`true`
`'Abc'<='B'`	`true`
`'Abc'<='A'`	`false`
`'Ab'=='Ab'`	`true`
`'Ab'!='ab'`	`true`

Comparações de tipos diferentes

Quando os operandos forem de tipos diferentes, Javascript converterá os operandos para números. Exemplos:

Operação	Resultado
`'07'> 5`	`true`
`'dois'> 1`	`false, pois ao converter 'dois' para number, o resultado será NaN`
`'dois'>'1'`	`true, pois o caractere '1' possui o código 49 na tabela Unicode enquanto o caractere 'd'possui o código 100 nessa mesma tabela.`

Comparações

Cuidado:

Operação	Resultado
`Boolean(0)`	`false`
`Boolean('0')`	`true`, pois não é uma string vazia
`'0' == 0`	`true`, pois ‘0’ convertido para number será igual a 0

Comparações

Para checar a igualdade sem realizar a conversão dos operandos para number quando o tipo dos mesmos for diferente, utilize o operador ===. Para checar a diferença sem conversão, utilize o operador !==. Exemplos:

Operação	Resultado
`0 === 0`	`true`
`0 === '0'`	`false`
`'0' !== 0`	`true`

Comparações

Comparando null e undefined:

Operação	Resultado
`null === undefined`	`false`
`null == undefined`	`true`
`null > 0`	`false`, pois null se torna 0
`null >= 0`	`true`, pois null se torna 0
`undefined == 0`	`false`
`undefined > 0`	`false`
`undefined < 0`	`false`

Operadores Relacionais

Os operadores && e || também podem ser aplicados a operandos que não são booleanos. Neste caso, o operador || retorna o primeiro valor verdadeiro enquanto o operador && retorna o primeiro valor falso.
Exemplos:

```
let nome1 = "", nome2 = "João";
```

Operação	Retorno
`nome1 || 'Anônimo'`	`'Anônimo'`
`nome2 || 'Anônimo'`	`'João'`
`nome2 && null`	`null`

Uso Alternativo de Operadores Relacionais

Assim que o resultado da operação puder ser definido, como quando o operador || possui um operando à esquerda que retorna true ou quando o operador && possui um operando à esquerda que retorna false, o valor da expressão será retornado sem a avaliação dos demais operandos.

```
let logr = "Rua A", numero = "";
let endereco = logr + ", " + (numero || "Sem Número");
//endereco -> Rua A, Sem Número
```

```
let logr = "Rua A", numero = "100";
let endereco = logr + ", " + (numero || "Sem Número");
//endereco -> Rua A, 100
```

Uso Alternativo de Operadores Relacionais

Assim que o resultado da operação puder ser definido, como quando o operador || possui um operando à esquerda que retorna true ou quando o operador && possui um operando à esquerda que retorna false, o valor da expressão será retornado sem a avaliação dos demais operandos.

```
let nome = "João", sobrenome = "";
let nomeESobrenomePreenchido = Boolean(nome && sobrenome);
//nomeESobrenomePreenchido -> false
```

```
let nome = "João", sobrenome = "Silva";
let nomeESobrenomePreenchido = Boolean(nome && sobrenome);
//nomeESobrenomePreenchido -> true
```

Fim de material

Dúvidas?