🎖️

Operadores Matemáticos

> Em diversos momentos precisaremos utilizar os operadores matemáticos que já conhecemos, por isso precisamos vê-los em uso.

Iremos utilizá-los em várias aplicações e páginas (quase sem exceções), mesmo cujo escopo não seja matemático/financeiro/exato.

Alguns exemplos de uso são: somas simples, interpolações, mensagens renderizadas ao nosso usuário e etc...

Assim como na matemática, o JavaScript possui uma ordem de leitura em cálculos no geral. A lógica é a mesma que já sabemos, Divisão e Multiplicação são resolvidas antes e depois soma e subtração, da esquerda para a direita. Um exemplo é a seguinte expressão:

```
10 + 20 / 5 + 5
```

O resultado da expressão acima é 19, pois:

1º Divisão: 20/5 = 4

2º Soma: 10 + 4 = 14

3º Soma 14 + 5 = 19

O JavaScript realizará essa conta exatamente assim, como a matemática.

Caso pense que o resultado é 11, lembre das regras citadas acima. Para o resultado 11 se tornar correto, teríamos que inserir parênteses:

```
(10 + 20) / 5 + 5
```

Assim, seria 11, pois:

1º Parênteses: (10 + 20) = 30

2º Divisão: 30 / 5 = 6

3º Soma: 6 + 5 = 11

+ →

O símbolo de **adição** retorna a soma de valores quaisquer, não somente númericos, mas também strings, booleans e suas misturas.

Exemplo de uso:

```
//String
const nome = 'Clóvis';
//Number
const idade = 20;

console.log('A idade do ' + nome + ' é de ' + idade + ' anos.');
//Retorna: A idade do Clóvis é de 20 anos.
```

O intuito do exemplo acima é mostrar a capacidade de renderizarmos dados de diferentes tipos no mesmo contexto.

- →

O símbolo de subtração retorna a subtração de valores, cujo valor é numérico ou booleano.

Exemplos de uso:

Number - number:

```
//Number
const ano = 2021;
//Number
const idade = 20;

console.log(ano - idade);
//Resultado = 2001
```

Number - boolean:

```
//Number
const num1 = 1;
//Booleano (true = 1 e false = 0)
const boolean = true;

console.log(num1 - boolean);
//Resultado = 0
```

`*` →

O símbolo de multiplicação retorna a multiplicação de valores, cujo valor é numérico ou booleano.

Exemplos de uso:

Number * number

```
//Number
const num1 = 2;
//Number
const num2 = 5;

console.log(num1 * num2);
//Resultado = 10;
```

Number * boolean

```
//Number
const num1 = 7;
//Booleano (true = 1 e false = 0)
const boolean = false;

console.log(num1 * boolean);
//Resultado = 0;
```

`/` →

O símbolo de divisão retorna a divisão de valores, cujo valor é numérico ou booleano.

Exemplos de uso:

Number / number

```
//Number
const num1 = 10;
//Number
const num2 = 5;

console.log(num1 / num2);
//Resultado = 2;
```

Number / boolean

```
//Number
const num1 = 3;
//Booleano (true = 1 e false = 0)
const boolean = true;
```

```
console.log(num1 / boolean);
//Resultado = 3;
```

% →

O símbolo de porcentagem nos retorna o resto de uma divisão (0 ou 1).

Exemplos de uso:

```
const num1 = 5;

const num2 = 2;

console.log(num1 % num2);
//Retorna: 1
```

```
const num1 = 5;

const num2 = 5;

console.log(num1 % num2);
//Retorna: 0
```

** →

Usamos esses dois asteriscos juntos para elevar um número a um expoente. Posteriormente veremos alguns métodos como o Math.pow() que serão mais completos.

Exemplo de uso:

```
let num1 = 10;
let num2 = 3;

console.log(num1 ** num2); // 1000 => 10 * 10 * 10
```

```
let num1 = 10;
let boolean = true;

console.log(num1 ** boolean); // 10
```

Ps.: Em cálculos cujos valores são do tipo number e do tipo string (com exceção da soma) o resultado é **NaN**.

Incrementação e Decrementação

Quando precisamos adicionar ou subtrair repetidamente 1 de algum valor, usamos a incrementação (++) e/ou decrementação (- -). São muito usadas em laços de repetição em geral.

++ →

Esses símbolos formam a chamada **'incrementação',** ela adiciona 1 ao número em questão (funciona apenas com variáveis do tipo number).

Exemplo de uso:

```
//Variável chamada 'num1'
let num1 = 7;

//Incrementação:
num1++;

console.log(num1);
//Resultado = 8
```

- - →

Esses símbolos formam a chamada **'decrementação',** ela subtrai 1 ao número em questão (funciona apenas com variáveis do tipo number).

Exemplo de uso:

```
//Variável chamada 'num1'
let num1 = 7;

//Decrementação:
num1--;

console.log(num1);
//Resultado = 6
```

Conversão

Em algumas vezes recebemos valores númericos como string ou em outros tipos de dados, por um erro ou detalhe no formulário preenchido pelo usuário. Porém, é possível transformarmos esse valor em um valor númerico para que seja possível realizar cálculos e expressões.

- Sintaxe:

```
Number(variavel)
```

Exemplo de uso:

```
let number = '1';
console.log(number + 4);
// Retorna: 14

let number = '1';
console.log(Number(number) + 4);
// Retorna: 5
```

Formatação de valores

Para formatarmos as casas decimais de um número qualquer utilizamos o método toFixed(). Nele passamos por parâmetro a quantidade de casas decimais desejadas. Ele nos retorna a quantidade de casa determinada e arredonda o último algarismo. Por exemplo:

```
let number = 4.92394023021

number.toFixed(3);  // Retorna: 4.923
number.toFixed(5);  // Retorna: 4.92394
number.toFixed(9);  // Retorna: 4.923940230
```

Atribuição composta

Operadores	Exemplo	Equivalente a	O que faz
=	a = b	a = b	Atribuição, como visto anteriormente.
+=	a += b	a = a + b	Soma 'b' com 'a'.
-=	a -= b	a = a - b	Subtrai o 'b' de 'a'.
*=	a *= b	a = a * b	Multiplica 'a' com 'b'.
/=	a /= b	a = a / b	Divide 'a' por 'b'.
%=	a %= b	a = a % b	Retorna o resto de 'a' por 'b'.
**=	a **= b	a = a ** b	Retorna 'a' elevado a 'b'.
Untitled			

Bibliografias:

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Math;
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Expressions_and_operators;