

## Material Extra

### Variáveis, Tipos e Operadores

#### >\_ Índice

- >\_ O que são variáveis?
- >\_ Declaração e tipos de variáveis
- >\_ O que são tipos de dados?
  - >\_ Diferentes tipos de dados
  - >\_ Tipos de dados especiais
- >\_ O que são operadores?
- >\_ Tipos de operadores

&gt;\_

## O que são variáveis?

As **variáveis** são recursos das linguagens de programação que nos **permitem armazenar** informações. Podemos pensar em variáveis como se fossem gavetas: cada uma delas funciona como uma variável diferente, onde é possível guardar **coisas** dentro, e utilizar, quando necessário.



&gt;\_

## Declaração e tipos de variáveis

Podemos observar uma **declaração** de variável no exemplo abaixo:

```
1  
2  var nome;  
3
```

Para iniciar a declaração, **utilizamos** a palavra reservada **var**, em seguida atribuímos o identificador **nome**, que usaremos quando quisermos referenciar a variável.

Também é possível **atribuir valores**, como no exemplo abaixo:

```
1  
2  var nome = "Christopher";  
3
```

Nesta segunda declaração, **atribuímos** um valor **à variável**, diferente da primeira declaração. Com isso, a variável **nome** possui **uma informação** – um texto – que poderíamos acessar a qualquer momento.

No Javascript também há **outra forma** de declarar variáveis, **utilizando** a palavra reservada **let** ao invés de **var**. Em geral, a utilização de **let** possui mais **benefícios** em comparação ao **var**, por isso, devemos utilizar, **por padrão, let** para declarar variáveis.

```
1
2  let nome = "Christopher";
3
```

Também é possível declarar **variáveis** com valores **imutáveis** – que não mudam – utilizando **constantes**. Esse recurso é muito utilizado, e considerado uma boa prática, pois dessa forma podemos assegurar o valor da variável, **impedindo** qualquer **alteração**.

```
1
2  const nome = "Christopher";
3
```

Com a utilização de constantes, **caso** o valor seja **alterado**, o compilador **irá gerar** um **erro**, impedindo a execução do algoritmo. Com isso tornamos nosso código mais seguro.

>\_

## O que são tipos de dados?

Na programação trabalhamos com **diferentes tipos de informações**. Devido a isso, existem diferentes **tipos de dados** para lidar com cada uma delas. Usar o tipo de dado correto **para** lidar com **certa informação** é essencial para um bom funcionamento do código.

>\_

## Diferentes tipos de dados

Existem três tipos de dados **mais comuns** na programação, que são o **Number**, **String** e **Boolean**. Abaixo veremos uma descrição sobre cada um deles:

- **Number:** Tipo de dado numérico, muito utilizado para cálculos. Em algumas linguagens existem subdivisões dentro desse tipo, como **Int** indicando números inteiros, e **Float** indicando números com ponto flutuante (reais).

```
1
2  let idade = 10; // number - int
3  let preco = 23.50 // number - float
4
```

- **String:** Tipo de dado **textual**, que pode armazenar **qualquer** tipo de **caractere**, inclusive números. É importante ressaltar que por ser do tipo String, mesmo que possua números, **não** é recomendado **realizar cálculos**, pois o resultado pode acabar sendo uma concatenação;

```
1
2  let hexadecimal = "#FFFFFF"; // string
3
```

- **Boolean:** Tipo de dado lógico, sendo **possível** apenas dois valores, **true** (**verdadeiro**) ou **false** (**falso**). É utilizado para armazenar valores lógicos oriundos de comparações de quaisquer tipos de dados.

```
1
2  let verdadeiro = true; // boolean
3  let falso = false; // boolean
4
```

>\_

## Tipos de dados especiais

Também existem outros **tipos** de dados, chamados **especiais**. São eles:

- **NaN: Not a Number (Não é um número)** - Esse tipo de dado **indica** uma operação em que o valor resultante **não** pode ser passado como **um número**;

```
1
2  let inteiro = parseInt("blabla"); // conversão impossível
3  let raiz = Math.sqrt(-1) // raiz quadrada inexistente
4
```

- **Null:** Indica um tipo **vazio** – um elemento **sem** qualquer **valor** –, porém uma variável com **Null** no Javascript é considerada com valor, no caso um valor nulo;

```
1  
2 let hobbie = null; // tipo null  
3
```

- **Undefined:** Diferente do tipo **Null**, **Undefined** significa uma variável sem valor definido – com **ausência de valor** – e normalmente ocorre ao declarar uma variável sem **atribuição**.

```
1  
2 let hobbie; // tipo undefined  
3
```

>\_

## O que são operadores?

Os **operadores** são recursos que **nos permitem** manipular variáveis, **realizar** comparações e **operações** com seus valores.

>\_

## Tipos de Operadores

Existem diversos tipos de operadores, os mais comuns são os de **Atribuição**, **Comparação**, **Aritméticos**, **Lógicos** e **Concatenação**.

- **Atribuição:** Atribui valores às variáveis;

```

1
2  let x = 10 // Atribui o valor 10
3  let y = 5 // Atribui o valor 5
4  x += y // Soma o valor de Y (x = x + y)
5  x -= y // Subtrai o valor de Y (x = x - y)
6  x *= y // Multiplica o valor de Y (x = x * y)
7  x /= y // Divide o valor de Y (x = x / y)
8  x %= y // Atribui o valor do resto de Y (x = x % y)
9  x **= y // Atribui o valor da exponenciação de Y (x = x ** y)
10

```

- **Comparação:** Realizam **comparações** que retornam **true** ou **false**;

```

4
5  x == y // Verifica se é igual
6  x != y // Verifica se é diferente
7  x === y // Verifica se é estritamente igual
8  x !== y // Verifica se é estritamente diferente
9  x > y // Verifica se é maior
10 x < y // Verifica se é menor
11 x >= y // Verifica se é maior e igual
12 x <= y // Verifica se é menor e igual
13

```

- **Aritméticos:** Realizam operações como **Soma, Subtração** etc;

```

6
7  x++ // Incrementa 1 ao valor.
8  x-- // Decrementa 1 ao valor.
9  x % y // Resto da divisão de dois valores.
10 x - y // Subtrai valores.
11 x + y // Soma valores.
12 x ** y // Exponenciação do valor.
13

```

- **Lógicos:** Verificam comparações e retornam **true** ou **false**. São utilizados os sinais **&&** (Ambos valores verdadeiros), **||** (Ambos ou um valor verdadeiro) e **!** (Nega a condição);

```
5
6  true && true // True => Ambas as condições precisam ser verdadeiras
7  true || false // True => Uma ou ambas condições precisam ser verdadeiras
8  !false // True => Nega a condição atual, virando seu oposto
9
```

- **Concatenação:** Concatena textos – juntam suas partes – em apenas uma.

```
1
2  let nome = "Christopher"
3  let sobrenome = "Gonçalves"
4  let nomeCompleto = nome + " " + sobrenome // Christopher Gonçalves
5
```