MATERIAL EXTRA

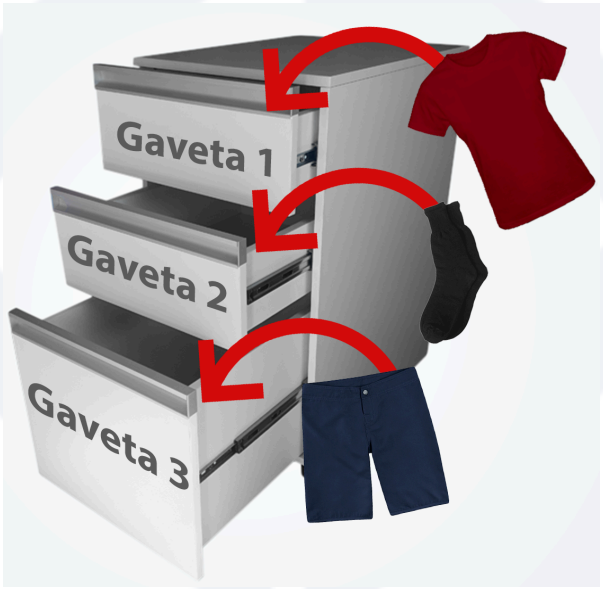
Variáveis

growdev

VARIÁVEIS

Uma variável é um espaço na memória do computador destinado a um dado que é alterado durante a execução do algoritmo.

O ato de dar um nome é chamado de declaração de variável. Para que o programa funcione corretamente, é possível que você declare logo no início todas as variáveis. Além disso, cada uma delas só mostrará um valor por vez, chamado de valor atual.



Com base nesta imagem podemos fazer uma analogia da utilização de variáveis e como o computador as gerencia.

Temos as variáveis **Gaveta1**, **Gaveta2** e **Gaveta3**, e cada variável armazena uma informação, por exemplo: **Gaveta1** armazena a informação **"Camiseta vermelha"**.

No fluxo do software definimos as variáveis, informamos o seu conteúdo, esta informação ficará armazenada em memória para posterior consulta ou substituição de seu conteúdo.

Declaração de variáveis no Javascript

var

As declarações de variáveis com a palavra reservada var são processadas antes de qualquer código seja executado. Ou seja, declarar uma variável com **var** em qualquer lugar no código é equivalente a declarar no início. Isso também significa que uma variável pode aparecer para ser usada antes dela ser declarada. Esse comportamento é chamado de "**hoisting**", a variável é movida para o início da função ou do código global.

Em resumo, a palavra chave **var**, faz com que a variável seja elevada no código com o conceito de **hoisting** e caso nenhum valor seja atribuído na declaração, será automaticamente inicializada como `undefined` (não definido).

```
bla = 2;
var bla;
//  ...

// é implicitamente entendido como:

var bla;
bla = 2;
```

Por essa razão e os problemas de **escopo de variáveis** que se encontram durante o desenvolvimento utilizando a palavra reservada **var** para declarar as variáveis JavaScript, foram implementadas outras maneiras de se declarar variáveis e que são fortemente recomendadas pelas documentações oficiais. A partir disso, a palavra reservada **var** caiu em desuso.

let

Muitas vezes em um código mais complexo e extenso, queremos declarar variáveis para serem utilizadas apenas em um pequeno trecho do código.

A partir do **ECMAScript 6** o **let** foi disponibilizado para definir o escopo de bloco para variáveis. Escopo, essencialmente, significa onde essas variáveis poderão ser utilizadas. Um bloco é uma porção de código cercado por **{}**. Um bloco vive dentro dessas chaves. Tudo o que estiver cercado por chaves é um bloco. Assim, uma variável declarada com **let** em um bloco estará disponível apenas dentro daquele bloco.

Ao tentarmos acessar uma variável declarada com **let** fora do seu escopo o resultado será um erro.

```
let mensagemForaDoEscopo = "Growdev";

if (true) {
  let mensagemDentroDoEscopo = "FullStack";
}

console.log(mensagemForaDoEscopo);
console.log(mensagemDentroDoEscopo);
```

Saída:

```
  Growdev
  ReferenceError: mensagemDentroDoEscopo is not defined
```

Uma variável declarada com let pode ter seu valor atualizado, mas não declarado novamente. Isso significa que, não poderemos ter duas variáveis de mesmo nome declaradas com let dentro de um mesmo escopo.

```
let formacao = "Full Stack";
let formacao = "Mobile"; // erro: identificador 'formacao' já foi declarado
```

const

Constantes tem um comportamento similar ao **let** com algumas particularidades importantes, ao definirmos uma constante com **const**, devemos obrigatoriamente atribuir um valor de inicialização. E como o próprio nome nos sugere, uma constante não pode ter seu valor alterado. Devendo então ser apenas utilizado para armazenar valores que não serão alterados durante a execução do sistema.

```
const pi = 3.14;
console.log(pi);
```

Caso seja efetuada durante a execução do código uma tentativa de alterar o valor de um constante, uma exceção (erro) será gerada.

```
const pi = 3.14;
console.log(pi);
pi = 2.444;
```

Saída:

```
3.14
TypeError: Assignment to constant variable.
```

Nomes de variáveis no JavaScript

Você pode chamar uma variável praticamente qualquer nome que queira, mas há limitações. Geralmente você deve se limitar a utilizar somente caracteres latinos (0-9, a-z, A-Z) e o caractere underline (_).

- Você não deve usar outros caracteres porque eles podem causar erros ou ser difíceis de entender por uma audiência internacional. Isso inclui evitar acentuações.
- Não use underline no início do nome de variáveis — isso é utilizado em certos construtores JavaScript para significar coisas específicas, então pode deixar as coisas confusas.
- Não use número no início do nome de variáveis. Isso não é permitido e irá causar um erro.
- Uma convenção segura é se ater à chamada "lower camel case", onde você junta várias palavras, usando minúscula para a primeira palavra inteira e, em seguida, maiúscula a primeira letra das palavras subsequentes. Temos utilizado esse procedimento para os nomes das nossas variáveis até aqui.
- Faça nomes de variáveis intuitivos, para que descrevam o dado que ela contém. Não use letras ou números únicos, ou frases muito longas.
- As variáveis diferenciam letras maiúsculas e minúsculas — então **minhaidade** é uma variável diferente de **minhaIdade**.
- Evite utilizar palavras reservadas pelo JavaScript (palavras que fazem parte da sintaxe do JavaScript) como nome para suas variáveis. Então você não pode usar palavras como var, function, let e for como nome de variáveis. Os navegadores vão reconhecê-las como itens de código diferentes e, portanto, você terá erros.

Exemplos de bons nomes:

```
idade
minhaIdade
inicio
corInicial
valorFinalDeSaida
audio1
audio2
```

Exemplos ruins de nomes:

```
1
a
_12
minhaidade
MINHAIDADE
var
Document
skjfndskjfnbdskjfb
esseeumnomedevariavelbemlongo
```

Trabalhando com textos

```
let texto = 'Desenvolvimento Web Full Stack';
```

Aspas simples ou aspas duplas

Em JavaScript, você pode escolher aspas simples ou duplas para envolver suas strings. Ambas linhas abaixo funcionarão bem:

```
let html = 'Páginas web';
let css = "Folhas de estilo";
```

Concatenando strings

Concatenar é uma palavra chique da programação que significa "colocar junto". Para colocar strings juntas em JavaScript, usamos o operador (+), o mesmo usamos para adicionar números, mas neste contexto é algo diferente.

```
const linguagem = "Programa";
const formacao = "Full Stack Web Developer";
console.log(linguagem + " " + formacao);
// JavaScript Full Stack Web Developer
```

Template strings

Template Strings são strings que permitem expressões embutidas. Você pode utilizar string multi-linhas e interpolação de string com elas. Basicamente é uma nova forma de criar strings e tornar o seu código um pouco mais legível.

Template strings são envolvidas por (acentos graves) (``) em vez de aspas simples ou duplas. Template strings podem possuir placeholders. Estes são indicados por um cifrão seguido de chaves `${expression}`.

As expressões nos placeholders, bem como o texto em volta delas são passados a uma função.

```
const linguagem = "JavaScript";
const formacao = "Full Stack Web Developer";
console.log(`${linguagem} ${formacao}`);
// JavaScript Full Stack Web Developer
```

REFERÊNCIAS

1. ECMA Internacional
2. Versões ECMAScript
3. var, let e const - Qual é a diferença?
4. var - MDN
5. let - MDN
6. const - MDN

BONS ESTUDOS

growdev