DIFERENÇAS ENTRE LET, CONST E VAR
(AULA 3)

CURSO BÁSICO DE PROGRAMAÇÃO COM JAVASCRIPT

MAYARA MARQUES

mmrosatab@gmail.com

JS

SUMÁRIO

- Diferenças entre let, const e var
  - Let
  - Const
  - Let vs const
  - Var
    - Hoisting
- Mão na massa

DIFERENÇAS ENTRE LET, CONST E VAR

No JavaScript, podemos declarar variáveis usando as palavras reservadas **let**, **const** e **var**, mas afinal de contas, qual a diferença entre elas?

LET

- let

```
let nomeVariavel = valor
```

```
let x = 5;
let y = 6;
let z = x + y;
```

LET

- let
  - Variáveis declaradas com ***let*** são utilizadas quando se tem a intenção de ***modificar o valor da variável ao longo do código***.

```
let nome = "Neo";

. . .

nome = "Morpheus";
```

LET

- let
  - Uma variável ***let não pode ser redeclarada*** dentro de bloco ou escopo do código.
    - Considere um bloco como a abertura e o fechamento de chaves {}.
    - Considere escopo global o código que está fora dos blocos.

```
let nome = "Neo";

. . .

let nome = "Morpheus"; // isso resulta em erro
```

LET

- let
  - Uma variável definida como ***let*** dentro de um bloco {} não é visível fora do bloco. (escopo de bloco)

```
{
  let x = 2;
}

console.log(x) // Uncaught ReferenceError: x is not    defined at <anonymous>
```

x não pode ser acessada aqui, pois só existe dentro do bloco onde foi declarada.

erro apresentado no console do navegador

CONST

- const
  - Variáveis declaradas como ***const não podem ser redeclaradas*** *ou* ***reatribuídas*** dentro de um bloco.
  - Assim como ***let***, possuem ***escopo de bloco***.
  - ***Precisam ter um valor atribuído*** logo na declaração.

```
const nomeVariavel = valor
```

```
const PI = 3.14;
```

const

- const

```
const nomeVariavel = valor
```

```
const PI = 3.14;
PI = 3.14;      // erro
PI = PI + 10;   // erro
```

CONST

- const
  - Variáveis const ***não podem ser redeclaradas ou reatribuídas***, contudo em algumas situações o ***CONTEÚDO*** pode ser modificado.

```
Alterar os elementos da matriz/array constante

const cars = ["Saab", "Volvo", "BMW"];

// Muda o elemento 0

cars[0] = "Toyota";

// Adiciona mais um elemento no array

cars.push("Audi");

console.log(cars) // ['Toyota', 'Volvo', 'BMW', 'Audi']
```

Atenção!!

CONST

- const
  - Variáveis const ***não podem ser redeclaradas ou reatribuídas***, contudo em algumas situações o ***CONTEÚDO*** pode ser modificado.

```
Alterar as propriedades do objeto constante

const car = {type:"Fiat", model:"500", color:"white"};

// Muda a propriedade color

car.color = "red";

// Adiciona uma nova propriedade ao objeto

car.owner = "Johnson";
```

Atenção!!

LET vs CONST

Diferente do **const**, o **let** permite que você reatribua o valor da variável. Quando você faz isso, está dizendo para a variável apontar para **um novo local na memória que contém o novo valor**.

Por exemplo:

```
let numero = 42; // A variável `numero` aponta para o valor 42.

numero = 100;    // Agora `numero` aponta para o valor 100.
```

LET vs CONST

Por exemplo:

```
let numero = 42; // A variável `numero` aponta para o valor 42.

numero = 100;    // Agora `numero` aponta para o valor 100.
```

Neste caso, o valor original 42 não foi alterado. O que aconteceu foi que a variável número parou de apontar para o local de memória onde 42 estava e passou a apontar para um novo local onde 100 estará.

VAR

- var

```
var nomeVariavel = valor
```

```
var x = 5;
var y = 6;
var z = x + y;
```

VAR

- var
  - Variáveis declaradas com ***var*** podem ser ***redeclaradas***.
  - Apresentam **escopo global** se declaradas fora qualquer função ou **escopo de função** se declaradas dentro de uma função.

OBS: Na aula de **escopos** veremos o **var** com profundidade e citaremos todas as suas particularidades. Contudo, nesta aula já podemos ressaltar o porquê o uso do **var** é **desencorajado** no javascript moderno.

VAR

- var
  - Nesta aula trataremos de situações com **var** dentro de blocos que **não** sejam funções, como estruturas de controle, loops for ou blocos if.

```
nome = "Neo"

console.log(nome) // Neo

var nome
```

HOISTING

- Variáveis declaradas com **var** dentro de blocos que **não** sejam funções (como estruturas de controle, como loops for ou blocos if) são elevadas (**hoisted**) para o escopo mais próximo que as envolve, que geralmente é o escopo da função mais próxima ou o escopo global, dependendo de onde o bloco está localizado. No entanto, **o valor da variável não é definido até o ponto em que a declaração é encontrada** durante a execução (atribuição).

Hoisting

Elevação

HOISTING

- O que na verdade aconteceu no código anterior?

```
// undefined
var nome

nome = "Neo"

console.log(nome) // Neo

var nome
```

HOISTING

```
var nome

console.log(nome) // undefined

nome = "Neo"

console.log(nome) // Neo
```

HOISTING

```
console.log(nome) // erro nome não está definido

nome = "Neo"

console.log(nome)
```

```
console.log(nome) // undefined

var nome = "Neo"

console.log(nome) // Neo
```

HOISTING

- O Javascript também permite usar variáveis sem declará-las, contudo, isso é considerado ***uma má prática***. Sempre declare variáveis!

```
nome = "Maria"

console.log(nome) // Maria
```

Erros sobre atribuições de variáveis não declaradas ocorrem apenas em código de **modo estrito**. Em código **não-estrito**, eles são silenciosamente ignorados.

HOISTING

- Quando utilizamos **const** ou **let**, temos um código mais previsível.

```
var nome = "Neo"
console.log(nome) // Neo
if(true){
    var nome = "Morpheus"
    console.log(nome) // Morpheus
}
console.log(nome) // Morpheus
```

```
const nome = "Neo"
console.log(nome) // Neo
if(true){
    const nome = "Morpheus"
    console.log(nome) // Morpheus
}
console.log(nome) // Neo
```

HOISTING

- O Javascript também permite usar variáveis sem declará-las, contudo, isso é considerado ***uma má prática***. Sempre declare variáveis.

```
“use strict”

nome = “Maria”

console.log(nome) // erro
```

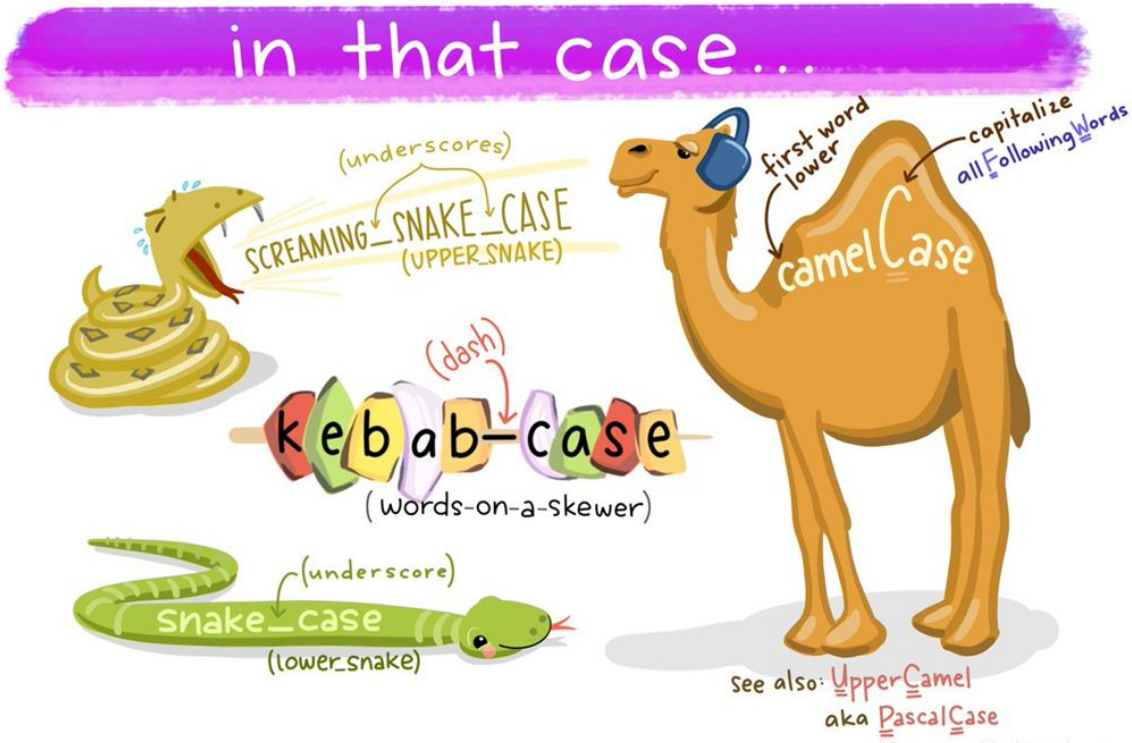
Erros sobre atribuições de variáveis não declaradas ocorrem apenas em código de modo estrito. Em código não-estrito, eles são silenciosamente ignorados.

EM RESUMO

- Em vez de **var**, é recomendável usar **let** ou **const**, introduzidos no ECMAScript 6 (ES6).
- Use **let** para variáveis que precisam ser reatribuídas.
- Use **const** para variáveis cujo valor não deve ser reatribuído (constantes), lembrando que há situações em que o conteúdo pode ser modificado com em **arrays e objetos**.
- Evite o uso de **var**, a menos que você esteja trabalhando com **código legado** ou tenha uma razão específica para usá-lo.

Entender como o **var** funciona e seu problema será útil em ocasiões de manutenção de código legado.

O uso de **let** e **const** oferece escopo de bloco, evita problemas de **hoisting** e **torna o código mais previsível e seguro**.



Disponível em: http://visualdicas.blogspot.com/2021/05/quais-as-formas-mais-populares-para.html

Bónus

Leitura recomendada Convenções de nomenclatura: Camel, Pascal, Kebab e Snake case | Alura

MÃO NA MASSA

MÃO NA MASSA

Para solução do exercícios abaixo, utilize **let** ou **const** e **descarte** o uso do var.

1. Crie um programa que recebe um texto digitado pelo usuário e imprima esse texto.

2. Cria um programa que dado um número inteiro digitado pelo usuário exiba o número antecessor e o sucessor a este número.

3. Crie um programa que receba três notas de prova de um aluno digitadas pelo usuário, calcule a média aritmética dessas notas e imprima a média.

4. Crie um programa que recebe dois números inteiros digitados pelo usuário e exiba o resultado da multiplicação desses números.

MÃO NA MASSA

5. Crie um programa que recebe dois números inteiros digitados pelo usuário e exiba a soma desses dois números.

6. Crie um programa que receba um valor decimal e imprima este valor aumentado em 10%.

7. Crie um programa que calcule o resto da divisão de dois números inteiros digitados pelo usuário no console.

8.Faça um algoritmo que leia a idade de uma pessoa expressa em anos, meses e dias e mostre-a expressa em dias. Leve em consideração o ano com 365 dias e o mês com 30 dias. (Ex: 3 anos, 2 meses e 15 dias = 1170 dias.)

MÃO NA MASSA

Explique com suas palavras (use um arquivo .txt para colocar as respostas)

9. Quais são as formas de declarar variáveis no JavaScript e quais são as diferenças entre cada uma?

10. O que é hoisting e quando ele acontece?

11. Cite exceções em que variáveis declaradas com o **const** podem ter seu conteúdo modificado.

MÃO NA MASSA

12. Considere o código abaixo para responder as perguntas:

```
const value1 = 10
const value2 = 5
let sum = value1 + value2
console.log(sum++)
```

a. Qual valor será impresso ?
b. O que acontece se a variável ***sum*** for declarada com ***const*** no lugar ***let***?

MÃO NA MASSA

13. Considere o código abaixo para responder a pergunta:

```
const value1 = 10
const value2 = 5
const sum = value1 + value2
console.log(sum)
```

a. Qual valor será impresso ?

MÃO NA MASSA

14. Considere o código abaixo para responder a pergunta:

```
const value1 = 11
const value2 = 5
let result = value1 % value2
console.log(--result)
```

a. Qual valor será impresso ?
b. O que acontece se a variável ***result*** for declarada com ***const*** no lugar ***let***?

MÃO NA MASSA

15. Considere o código abaixo para responder a pergunta:

```
console.log(value)
var value = 11
console.log(value)
```

a. O código funciona sem erro?
b. O será exibido no console do navegador?

MÃO NA MASSA

16. Considere o código abaixo para responder a pergunta:

```
console.log(value)
let value = 11
console.log(value)
```

a. O código funciona sem erro?
b. O será exibido no console do navegador?

REFERÊNCIAS

- Apostila Caelum Estruturação de páginas usando HTML e CSS
- https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript
- https://developer.mozilla.org/pt-BR/docs/Web/API/window/prompt
- https://www.w3schools.com/js/default.asp