ORIGAMID

JAVASCRIPT PARA INICIANTES

Escopo

Escopo de Função

Variáveis declaradas dentro de funções não são acessadas fora das mesmas.

```
function mostrarCarro() {
  var carro = 'Fusca';
  console.log(carro);
}

mostrarCarro(); // Fusca no console
console.log(carro); // Erro, carro is not defined
```

> *Escopo evita o conflito entre nomes.*

Variável Global (Erro)

Declarar variáveis sem a palavra chave `var`, `const` ou `let`, cria uma variável que pode ser acessar em qualquer escopo (global). Isso é um erro.

```
function mostrarCarro() {
  carro = 'Fusca';
  console.log(carro);
}

mostrarCarro(); // Fusca
console.log(carro); // Fusca
```

> `'use strict'` impede isso.

Escopo de Função (Pai)

Variáveis declaradas no escopo pai da função, conseguem ser acessadas pelas funções.

```
var carro = 'Fusca';

function mostrarCarro() {
  var frase = `Meu carro é um ${carro}`;
  console.log(frase);
}

mostrarCarro(); // Meu carro é um Fusca
console.log(carro);  // Fusca
```

Escopo de Bloco

Variáveis criadas com `var`, vazam o bloco. Por isso com a introdução do ES6 a melhor forma de declarmos uma variável é utilizando `const` e `let`, pois estas respeitam o escopo de bloco.

```
if(true) {
  var carro = 'Fusca';
  console.log(carro);
}
console.log(carro); // Carro
```

Var Vaza o Bloco

Mesmo com a condição falsa, a variável ainda será declarada utilizando hoisting e o valor ficará como undefined.

COPIAR

```
if(false) {
  var carro = 'Fusca';
  console.log(carro);
}
console.log(carro); // undefined
```

Const e Let no lugar de Var

A partir de agora vamos utilizar apenas `const` e `let` para declarmos variáveis.

```
if(true) {
  const carro = 'Fusca';
  console.log(carro);
}
console.log(carro); // erro, carro is not defined
```

{} cria um bloco

Chaves `{}` criam um escopo de bloco, não confundir com a criação de objetos `= {}`

```
{
  var carro = 'Fusca';
  const ano = 2018;
}
console.log(carro); // Carro
console.log(ano); // erro ano is not defined
```

For Loop

Ao utilizar `var` dentro de um `for` loop, que é um bloco, o valor do variável utilizada irá `vazar` e existir fora do loop.

```
for(var i = 0; i < 10; i++) {
  console.log(`Número ${i}`);
}
console.log(i); // 10
```

For Loop com Let

Com o `let` evitamos que o número vaze.

```
for(let i = 0; i < 10; i++) {
  console.log(`Número ${i}`);
}
console.log(i); // i is not defined
```

Const

Mantém o escopo no bloco, impede a redeclaração e impede a modificação do valor da variável, evitando bugs no código.

```
const mes = 'Dezembro';
mes = 'Janeiro'; // erro, tentou modificar o valor
const semana; // erro, declarou sem valor

const data = {
  dia: 28,
  mes: 'Dezembro',
  ano: 2018,
}

data.dia = 29; // Funciona
data = 'Janeiro'; // erro
```

> *Variáveis com valores constantes devem utilizar o* `const`.

Let

Mantém o escopo no bloco, impede a redeclaração, mas permite a modificação do valor da variável.

```
let ano;
ano = 2018;
ano++;
console.log(ano); // 2019

let ano = 2020; // erro, redeclarou a variável
```

> *Geralmente vamos utilizar o* `const`.

Exercício

```
// Por qual motivo o código abaixo retorna com erros?
{
  var cor = 'preto';
  const marca = 'Fiat';
  let portas = 4;
}
console.log(var, marca, portas);

// Como corrigir o erro abaixo?
function somarDois(x) {
  const dois = 2;
  return x + dois;
}
function dividirDois(x) {
  return x + dois;
}
somarDois(4);
dividirDois(6);
```

```
var numero = 50;

for(var numero = 0; numero < 10; numero++) {
  console.log(numero);
}

const total = 10 * numero;
console.log(total);
```