

ESCOPOS E CLOSURES

(AULA 12)

CURSO BÁSICO DE PROGRAMAÇÃO COM JAVASCRIPT

MAYARA MARQUES

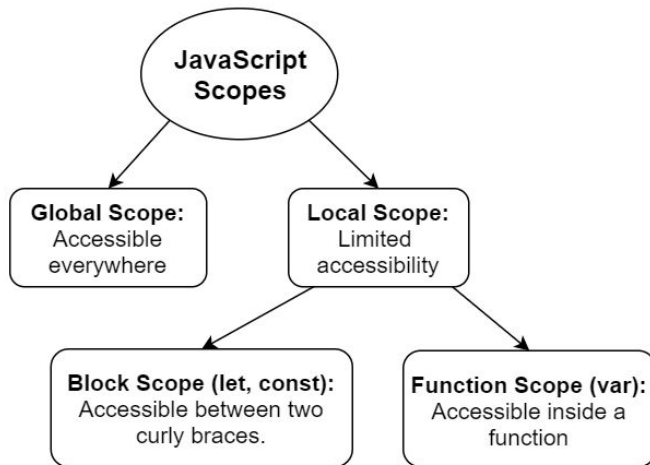
mmrosatab@gmail.com

SUMÁRIO

- Escopo e tipos de escopos
- Recordando
- let e const
- var
- Mão na massa

ESCOPO E TIPOS DE ESCOPOS

- Recordando escopo



Disponível em: <https://indepth.dev/posts/1357/getting-started-with-modern-javascript-variables-and-scope>

ESCOPO E TIPOS DE ESCOPOS

- Recordando escopo

```
function myFunction() {  
  const a = 1  
  
  if (true) {  
    const b = 2  
    console.log(a) // 1  
    console.log(b) // 2  
  }  
  
  console.log(a) // 1  
  console.log(b) // Uncaught ReferenceError: b is not defined  
}  
  
myFunction()
```

Variáveis **let** ou **const** quando declaradas dentro de um bloco estão disponíveis apenas dentro desse bloco.

ESCOPO E TIPOS DE ESCOPOS

Um escopo é a **visibilidade** e **acessibilidade** de uma variável no código.

- Variáveis que são visíveis **em qualquer parte do código** possuem **escopo global**.
- Variáveis que são visíveis **em apenas uma parte específica do código** (bloco) possuem **escopo local**.
 - Variáveis que são visíveis **somente dentro de um bloco** possuem **escopo de bloco** (if, for, while, função).
 - Variáveis que são visíveis **apenas dentro uma função** possuem **escopo de função**.

RECORDANDO

No javascript, variáveis são declaradas utilizando as palavras reservadas **var**, **let** e **const**.

Cada uma dessas formas de declarar variáveis possuem diferenças. Uma delas é o **escopo**.

CONST E LET

- Variáveis declaradas com **let** e **const**, quando **declaradas dentro de um bloco**, possuem **escopo de bloco**.

```
{  
  let x = 2;  
}
```

```
console.log(x) // x não pode ser acessada aqui
```

VAR

- Variáveis declaradas com **var** podem apresentar **escopo de função ou escopo global**.

Agora que já sabemos o que são funções e como utilizá-las, podemos abordar o funcionamento do **var** quando possui **escopo de funções**.



VAR

- Vimos em aula anterior que variáveis declaradas com **var** sofrem processo de **hoisting**, ou seja, são elevadas (**hoisted**) para o escopo mais próximo que as envolve.

Hoisting

Elevação



VAR

Recordando ...

```
nome = "Neo"  
console.log(nome) // Neo  
var nome
```



```
var nome  
nome = "Neo"  
console.log(nome) // Neo  
// undefined
```

VAR

- O **hoisting** também ocorre quando utilizamos o var em funções.

```
var filme = 'Harry Potter'
```

```
function mostrarFilme() {  
    console.log(filme)  
    var filme = 'Rocky Balboa'  
    console.log(filme)  
}
```

```
mostrarFilme()
```

O que será impresso?



VAR

```
var filme = 'Harry Potter'
```

```
function mostrarFilme() {
```

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

```
  var filme = 'Rocky Balboa'
```

```
  console.log(filme) // Rocky Balboa
```

```
}
```

```
mostrarFilme()
```


VAR

- O que acontece quando o hoisting é aplicado?

```
var filme = 'Harry Potter'

function mostrarFilme(){
    console.log(filme) // undefined
    var filme = 'Rocky Balboa'
    console.log(filme) // Rocky Balboa
}

mostrarFilme()
```



```
var filme = 'Harry Potter'

function mostrarFilme(){
    var filme
    console.log(filme) // undefined
    filme = 'Rocky Balboa'
    console.log(filme) // Rocky Balboa
}

mostrarFilme()
```

VAR

```
var filme = 'Harry Potter'

function mostrarFilme(){
    console.log(filme) // undefined
    var filme = 'Rocky Balboa'
    console.log(filme) // Rocky Balboa
}

mostrarFilme()

console.log(filme) // ????
```

O que será impresso?



VAR

```
var filme = 'Harry Potter'

function mostrarFilme(){

    console.log(filme) // undefined

    var filme = 'Rocky Balboa'

    console.log(filme) // Rocky Balboa

}

mostrarFilme()

console.log(filme) // Harry Potter
```

VAR

```
function decidirNome() {  
  
  function falar() {  
  
    return 'bolacha'  
  
  }  
  
  return falar()  
  
  function falar() {  
  
    return 'biscoito'  
  
  }  
  
}  
  
var resultado = decidirNome()  
  
console.log(resultado)
```

O que será impresso na linha 14?



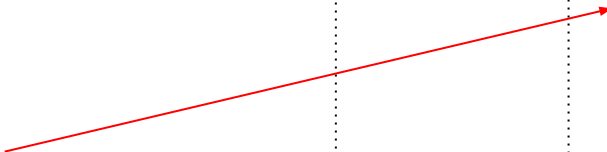
VAR

```
function decidirNome() {  
  
  function falar() {  
  
    return 'bolacha'  
  
  }  
  
  return falar()  
  
  function falar() {  
  
    return 'biscoito'  
  
  }  
  
}  
  
var resultado = decidirNome()  
  
console.log(resultado) // biscoito
```

VAR

- O que acontece no código na prática

```
function decidirNome() {  
  function falar() {  
    return 'bolacha'  
  }  
  
  return falar()  
  
  function falar() {  
    return 'biscoito'  
  }  
}  
  
var resultado = decidirNome()  
console.log(resultado)
```



```
function decidirNome() {  
  function falar() {  
    return 'bolacha'  
  }  
  
  function falar() {  
    return 'biscoito'  
  }  
  
  return falar()  
}  
  
var resultado = decidirNome()  
console.log(resultado)
```

VAR

```
function decidirNome() {  
  var falar = function() {  
    return 'bolacha'  
  }  
  
  return falar()  
  
  var falar = function() {  
    return 'biscoito'  
  }  
}  
  
var resultado = decidirNome()  
  
console.log(resultado)
```

O que será impresso na linha 14?



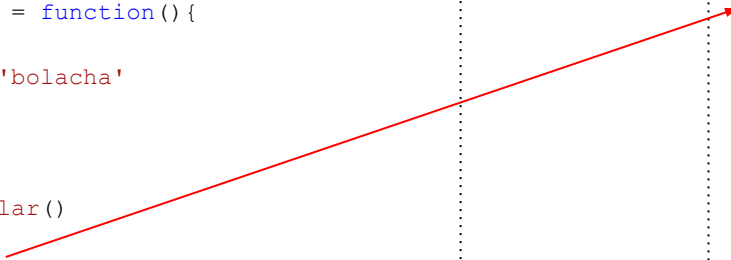
VAR

```
function decidirNome() {  
  
    var falar = function() {  
        return 'bolacha'  
    }  
  
    return falar()  
  
    var falar = function() {  
        return 'biscoito'  
    }  
  
}  
  
var resultado = decidirNome()  
  
console.log(resultado) // bolacha
```

VAR

- O que acontece quando o hoisting é aplicado?

```
function decidirNome() {  
  var falar = function() {  
    return 'bolacha'  
  }  
  
  return falar()  
  
  var falar = function() {  
    return 'biscoito'  
  }  
}  
  
var resultado = decidirNome()  
  
console.log(resultado)
```



```
function decidirNome() {  
  var falar  
  var falar  
  falar = function() {  
    return 'bolacha'  
  }  
  
  return falar()  
  
  falar = function() {  
    return 'biscoito'  
  }  
}  
  
var resultado = decidirNome()  
  
console.log(resultado)
```

VAR

- O código a seguir funciona?

```
console.log(falar())
```

```
function falar() {  
  return 'bolacha'  
}
```

VAR

- O código a seguir funciona?

```
console.log(falar()) // bolacha
```

```
function falar() {  
  return 'bolacha'  
}
```

SIM



VAR

- O que acontece na prática

```
console.log(falar())
```

```
function falar() {
```

```
  return 'bolacha'
```

```
}
```

Hoisting

```
function falar() {
```

```
  return 'bolacha'
```

```
}
```

```
console.log(falar())
```


CLOSURE

Uma closure é uma função que **lembra o ambiente onde foi criada**. Isso significa que uma closure é uma função que **consegue acessar valores definidos em um escopo externo onde foi declarada**.

CLOSURE

Exemplo

```
function counter(){
  let count = 0

  return {
    increment: function(){
      count++
      return count
    },
    decrement: function(){
      count--
      return count
    }
  }
}

let mycounter = counter()
console.log(mycounter.increment()) // 1
console.log(mycounter.increment()) // 2
console.log(mycounter.decrement()) // 1
console.log(mycounter.decrement()) // 0
```

CLOSURE

É importante entender que uma closure é criada quando uma **função interna é definida dentro de uma função externa**, e a **função interna faz referência a variáveis da função externa**.



MÃO NA MASSA

MÃO NA MASSA



1. O que será impresso no console na execução do código abaixo?

```
var x = 'Recuso-me a dizer o meu valor'
```

```
function imprimir() {
```

```
    console.log(x)
```

```
    var x = 'Acredito que meu valor é demonstrado através das minhas atitudes'
```

```
    console.log(x)
```

```
}
```

```
imprimir()
```

```
console.log(x)
```

MÃO NA MASSA



2. O código abaixo é executado com sucesso? Sim ou não? Justifique!

```
let x = 'Recuso-me a dizer o meu valor'

function imprimir() {

  console.log(x)

  let x = 'Acredito que meu valor é demonstrado através das minhas atitudes'

  console.log(x)

}

imprimir()

console.log(x)
```

MÃO NA MASSA



3. O código abaixo é executado com sucesso? Sim ou não? Justifique!

```
let x = 'Recuso-me a dizer o meu valor'

function imprimir() {

  console.log(x)

  // let x = 'Acredito que meu valor é demonstrado através das minhas atitudes'
  // console.log(x)
}

imprimir()

console.log(x)
```

MÃO NA MASSA



4. O que será impresso no console na execução do código abaixo?

```
var x = 'Recuso-me a dizer o meu valor'
```

```
if(10 > 5){
```

```
    console.log(x)
```

```
    var x = 'Acredito que meu valor é demonstrado através das minhas atitudes'
```

```
    console.log(x)
```

```
}
```

```
console.log(x)
```


MÃO NA MASSA



4. O que será impresso no console na execução do código abaixo?

```
var x = 10

function func() {

  if (true) {

    var x = 20

    console.log(x)

  }

  console.log(x)

}

func()

console.log(x)
```

MÃO NA MASSA



5. O que será impresso no console na execução do código abaixo?

```
let num = 30

function func2() {

  if (true) {

    let num = 40

    console.log(num)

  }

  console.log(num)

}

func2()

console.log(num)
```

MÃO NA MASSA



6. O que será impresso no console na execução do código abaixo?

```
const z = 50

function func3() {

  if (true) {

    const z = 60

    console.log(z)

  }

  console.log(z)

}

func3()

console.log(z)
```

MÃO NA MASSA



7. O que será impresso no console na execução do código abaixo?

```
var a = 5

let b = 10

const c = 15


if (true) {

  var a = 20

  let b = 25

  const c = 30

  console.log(a, b, c)

}

console.log(a, b, c)
```

MÃO NA MASSA



8. Crie uma função contador que retorne outra função. Cada vez que a função retornada for chamada, ela deve incrementar e exibir um contador.

Exemplo de uso

```
const contar = contador()  
contar() // 1  
contar() // 2
```

REFERÊNCIAS

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