ESCOPO
CONST·LET·VAR

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA | M1S03-A3

- **Escopo**
  - Var
  - Const & Let
  - Hoisting (içamento)
  - "use strict"

ESCOPO

ESCOPO

- Escopo é o que responde a pergunta
  “Onde essas variáveis estão disponíveis?”

- Pensando no JavaScript, temos 3 grandes regiões onde a variável pode estar disponível:

    - Escopo global;
    - Escopo da função;
    - Escopo do bloco;

ESCOPO

- Escopo global: Disponível para toda a aplicação;
- Escopo da função: Disponível apenas dentro da função na qual foi criada;
- Escopo do bloco: Disponível dentro dos blocos;

Obs.: Quando vemos chaves* abrindo e fechando no código, ali há um bloco
*não se tratando de objeto

ESCOPO | Var

Onde as variáveis estão disponíveis?

```
function definirLargura() {
  var largura = 100;
  console.log(largura);
}

definirLargura();

console.log(largura);
```

ESCOPO | Var

Onde as variáveis estão disponíveis?

```
var altura = 100;

if (altura > 90) {
  var largura = 100;
  console.log(largura);
}

console.log(largura);
```

ESCOPO | Var

- Quando dentro de uma função, o escopo é de função.
- Quando fora de uma função, o escopo é global;

ESCOPO | Var

O que acontecerá neste caso?

```
// Como pode confundir?
var idade = 31;

if (idade > 12) {
  var idadeEmAnosCaninos = idade * 7;
  console.log(`Você tem ${idadeEmAnosCaninos} anos em idade canina!`);
}

// Continua acessível fora do bloco onde foi criada!
console.log(idadeEmAnosCaninos);
```

ESCOPO | Let & Const

- Escopo de bloco (acessíveis apenas dentro do bloco).

ESCOPO | Let

O que acontecerá neste caso?

```
let pontos = 50;
let venceu = false;

if (points > 40) {
  console.log('Passei pelo if!');
  let venceu = true;
}

console.log('Venceu:', venceu);
// Venceu: false
```

ESCOPO | Const

- Diferente de **“`let`”** e **“`var`”**, **“`const`”** não pode ter sua referência alterada depois do momento da sua criação (reatribuição);

- Como a referência não pode ser atribuída fora do momento de sua criação, também não pode ser declarada sem que se insira imediatamente o seu valor;

- Não é que seus valores são imutáveis, mas ela não pode ter sua referência alterada.
  Se o valor dentro da referência for alterado, sem problemas

ESCOPO | Const

O que acontecerá neste caso?

```
const pessoa = {
  nome: 'Vitória',
  idade: 31
};

// O que vai acontecer nas ocasiões abaixo?
pessoa = { nome: 'Chinforínfola'};

pessoa.idade = 40;

// Como impedir que propriedades sejam alteradas?
const vitoria = Object.freeze(pessoa);
```

ESCOPO | Qual usar?

- **`const`**: Caso não for necessário reatribuição (primeira alternativa)
- **`let`**: Caso for necessário reatribuição (quando **const** não for suficiente)
- **`var`**: Caso precisar que algo esteja disponível de forma global
  e você estiver consciente, disposto a lidar com os riscos

  (também é possível disponibilizar algo no escopo global criando uma nova entrada no objeto raiz global **window**)

ESCOPO

Escopo	Const	Let	Var
Função	yes	yes	yes
Bloco	yes	yes	no
Reatribuido	no	yes	yes

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40

ESCOPO | Hosting

Var

O JavaScript apenas eleva (hoists) as declarações, não as inicializações. Se uma variável for declarada e inicializada após usá-la, o valor será undefined. Por exemplo:

```
console.log(num); // undefined
var num;
num = 6;
```

```
num = 6;
console.log(num); // 6
var num;
```

Var

Se você declarar a variável depois que ela for usada, mas inicializá-la antecipadamente, ela retornará o valor

ESCOPO | Hosting

- Declarações de variável **“`var`“** e **“`function`”** são ***hoisted*** (içadas);
- Declarações de **“`let`”** e **“`const`”** não;

```
// Não lança exceção:
console.log(pizza);
var pizza = 'Hmm... 🍕🍕';

// Lança exceção:
console.log(pastel);
const pastel = 'Delícia... 🥟🥟';
```

ESCOPO | Use strict

```
"use strict";
```

- Introduzido no ES5, **`"use strict"`**, quando presente no começo do documento, nos impede de usar variáveis que não foram propriamente declaradas com **`const`**, **`let`** ou **`var`**
- Ajuda a prevenir equívocos como utilizar nomes errados de variáveis e interferir no escopo global acidentalmente
- Automaticamente aplicado em módulos JavaScript

ATIVIDADE

- Crie uma função que receba como parâmetro um objeto contendo 2 atributos, texto e termo, como o exemplo e teste se o conteúdo de texto contém o conteúdo do termo.

```
procuraTermo({
  texto: "O rato roeu a roupa do rei de Roma.",
  termo: "rato"
}); // Deve retornar: true

procuraTermo({
  texto: "O rato roeu a roupa do rei de Roma.",
  termo: "batata"
}); // Deve retornar: false
```

MATERIAL COMPLEMENTAR

Var, Let, Const - Tudo o que você precisa saber | https://youtu.be/ZOx7iTnBqFQ

Como funciona o var, let e const? #01 | https://youtu.be/EFoEqHIwxqY

Differences Between Var, Let, and Const | https://youtu.be/9WIJQDvt4Us

O que é Hoisting? | https://youtu.be/M1Qvz7cRsqQ

Strict Mode — "use strict" | https://youtu.be/uqUYNqZx0qY

MATERIAL COMPLEMENTAR

let - JavaScript | https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let

const - JavaScript | https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const

var - JavaScript | https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/var

Hoisting - Glossário | https://developer.mozilla.org/pt-BR/docs/Glossary/Hoisting

Strict mode - JavaScript | https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>