



Foi mal, errei!



Módulo 2 - Aula 5 - TECH



Todos os direitos reservados
©2022 Resilia Educação

RESILIA |  Senac



Review

Revisão





**Como declaramos
variáveis?**



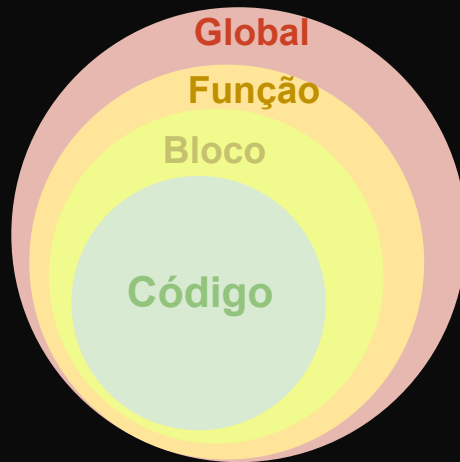


Escopo



Escopo (1)

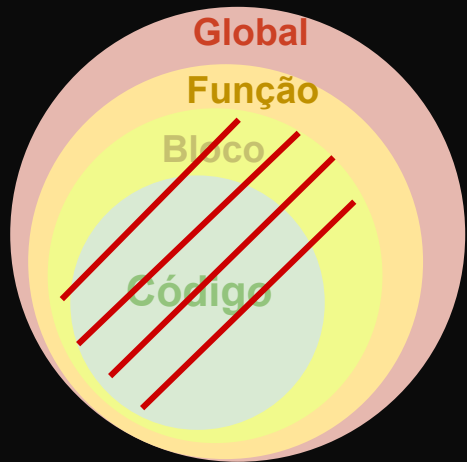
O **escopo** de um **nome** (função, variável) é a porção de **código** no qual este nome **existe** ou é **visível**.



Escopo (2)

O **escopo** de uma **variável** declarada com **var** é a **função** a qual ela pertence ou contexto **global**. Outros **escopos** mais **restritos** **não restringem** uma variável declarada com **var**.

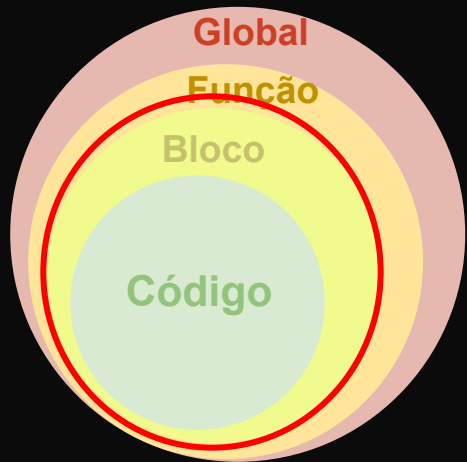
```
for (var i=0; i<3; i++) {  
    console.log(`i tem valor: ${i}`);  
}  
console.log(i) // i = 4 !!!!
```



ECMA Script - ES6: const

Variáveis declaradas com **const** possuem **escopo de bloco**.
Isso quer dizer que elas **não são visíveis/acessíveis fora do bloco** em que foram **declaradas**.

```
if (temperatura > 30) {  
    const msg = "Ei, vamos à praia?";  
    console.log(msg);  
}  
  
console.log(msg); //msg is not defined
```



ECMA Script - ES6: const

Além disso, variáveis declaradas com **const** são **imutáveis**.
Por isso, são chamadas de **constantes**.

```
const msg = "Ei, vamos à praia?";  
const msg = "Ei, vamos ao  
shopping?"; /*Uncaught TypeError:  
Assignment to constant variable.*/*
```



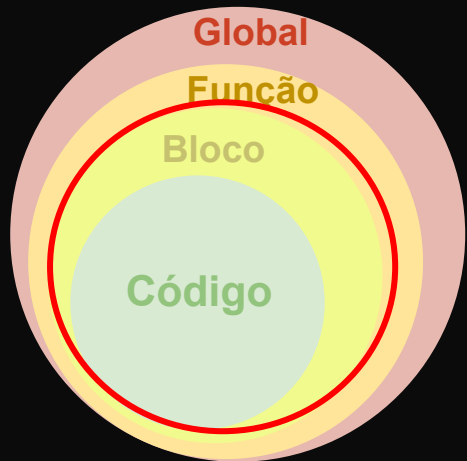
Erro: Atribuição em uma variável constante

ECMA Script - ES6: let

Variáveis declaradas com **let** possuem **escopo de bloco**.

Isso quer dizer que elas **não são visíveis/acessíveis fora do bloco** em que foram **declaradas**.

```
if (temperatura > 30) {  
    let msg = "Ei, vamos à praia?";  
    console.log(msg);  
}  
  
console.log(msg); //msg is not defined
```



const e let

O **escopo** de **const** e **let** é o **mesmo**.

O que diferencia os dois é a natureza do nome declarado.

Com **let**, o valor atribuído é **variável**. Com **const**, **não**.



Use const!!



Use let!!



NÃO USE VAR!!



Erros



JavaScript: Erros

Erros são problemas que **ocorrem** ao **executar** determinadas operações ou **comandos**. Como JavaScript é uma **linguagem interpretada**, os **erros** todos ocorrem em **tempo de execução**.

Ex:

- ⇒ Tentativa de acessar ou modificar atributo de undefined
- ⇒ Erros de sintaxe
- ⇒ Erro de referência (variável inexistente)


Erros: fontes externas

A partir do momento que uma aplicação passa a depender de outras, como **API's**, **erros** podem ocorrer tanto na **comunicação** quanto na **outra aplicação**.

Erros nunca devem passar de forma silenciosa!

Portanto, podemos utilizar **tratadores de erros** para que a **aplicação web** possa se “recuperar”.



A black and white photograph of a person standing on a paved road next to a fallen bicycle. The person is wearing dark clothing and sneakers, and is looking down at the bicycle. The bicycle is lying on its side, and a helmet is on the ground next to it. The background shows a grassy area and a curb.

Caiu?
Levanta e tenta
de novo!



A close-up photograph of a person's arm and hand. The person is applying a light-colored, rectangular adhesive bandage to their forearm. Their fingers are visible, holding the bandage in place. The background is a soft, out-of-focus light color.

Tratamento de Erros



Erros: tratamento - try

O tratamento de erros deve ser realizado utilizando o par **try/catch**.

Com o **try** (tentar) criamos um **bloco** em que uma **operação** que pode resultar em **erro** ocorre.



Erros: tratamento - catch

O **bloco** de código associado ao **catch** (pegar) é responsável pelo **tratamento** do **erro**. Este bloco recebe como **parâmetro** o **erro lançado** no bloco try caso ele ocorra.

Erros: tratamento - exemplo

```
try {  
    mochila.guardar("estojo");  
} catch (erro) {  
    console.log("Não foi possível guardar na mochila");  
    console.log(erro)  
}  
console.log("Esse código é executado mesmo quando erros  
ocorrem");
```


Lançando erros



Todos os direitos reservados
© 2022 Rosilva Educação

Erros: tratamento - exemplo

Programas podem **detectar problemas em tempo de execução** e, caso necessário, lançar **erros** para que outras camadas da aplicação os tratem.

A **palavra** utilizada para o lançamento de erros é **throw**, que significa “jogar” em inglês.

```
throw new Error("Erro de conexão: offline");
```

A young child with blonde hair, wearing a white floral shirt and denim shorts, is watering a plant in a garden. The child is holding a silver watering can and is bent over, pouring water onto the soil. The background is a lush garden with various plants and trees. The text "Como tratar os erros?" is overlaid in yellow.

Como tratar os erros?





try/catch!



Erros: lançamento - exemplo

Em **JavaScript** um número diferente de zero, quando dividido por zero resulta em **infinito**. Em termos matemáticos isso não está muito correto.

Vamos implementar um **mecanismo de divisão de balas por crianças** que, quando o número fornecido for **zero ou negativo**, **lança e trata erros**.



**ATIVIDADE
tá errado!**



Atividade: tá errado!

Vamos criar um código que verifica se o que a pessoa usuária passa para nós está correto:

Objetivo:

- ⇒ Criar uma função que irá verificar se o número passado pela pessoa usuária é par ou ímpar, porém, caso a pessoa passe algo que não seja um número, devemos lançar o erro "Valor inválido! Digite um número";

Requisitos:

- ⇒ Criar uma página que recebe um valor, via input e retorna se ele é par ou ímpar;
- ⇒ Dentro de um bloco try/catch, criar uma função que verifica se o valor recebido é um número
 - Se for um número, dizer se é par ou ímpar;
 - Senão, lançar um erro com a mensagem "Valor inválido! Digite um número".
- ⇒ Exemplo:
 - input 4 -> "É par";
 - input 5 -> "É ímpar";
 - input oi -> Erro: "Valor inválido! Digite um número".
- ⇒ Exibir o resultado na tela via manipulação do DOM;
- ⇒ Utilizar jQuery na manipulação do DOM;
- ⇒ Utilizar const e let na declaração de variáveis.



Estrutura de Dados



Estrutura de dados

Estruturas de dados são um tópico muito importante em computação. A forma como dados são armazenados num programa está diretamente relacionada com seu desempenho e facilidade de implementação/abstrações.

Algumas das principais são:

- ⇒ Arrays
- ⇒ Hashmap
- ⇒ Listas
- ⇒ Árvores
- ⇒ Filas
- ⇒ Pilhas



Pilha



Pilha

A **pilha** é uma estrutura de dados **clássica**!

Nela, temos uma estrutura responsável por armazenar elementos/dados.

Caso desejemos **adicionar um elemento**, ele deve ser armazenado sempre no **topo da pilha**. Caso seja necessário **remover um elemento**, o único que pode ser removido é o **elemento do topo**.



Requisitos de uma Pilha

Requisitos de uma Pilha

- ⇒ O que é necessário para criarmos uma classe Pilha?
- ⇒ Quais métodos precisam existir?
- ⇒ O que teria no constructor?



MOMENTO ATIVIDADE EM PARES

