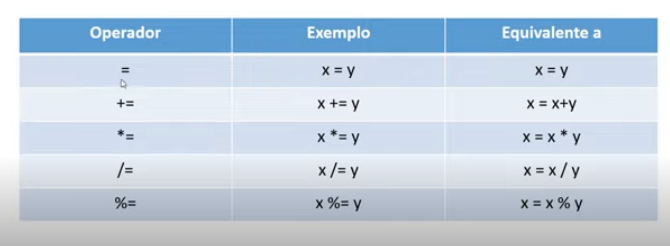
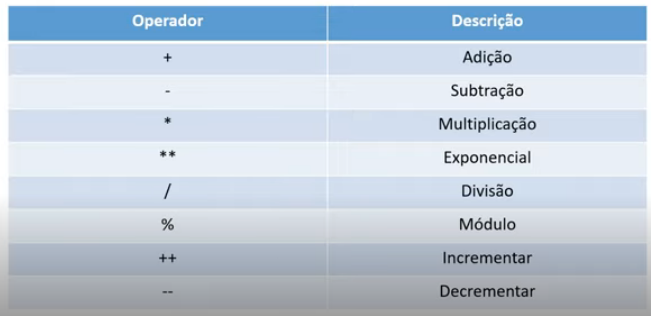
# JAVASCRIPT

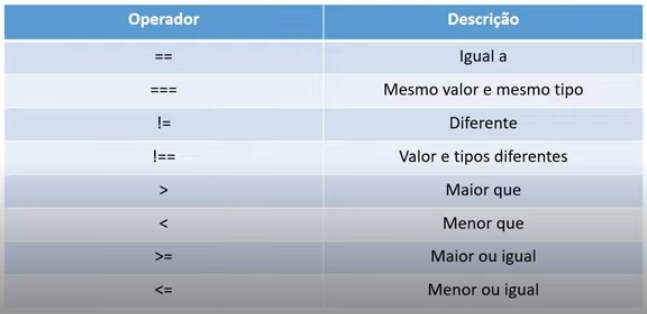
- Atribuições



- Operações aritméticas



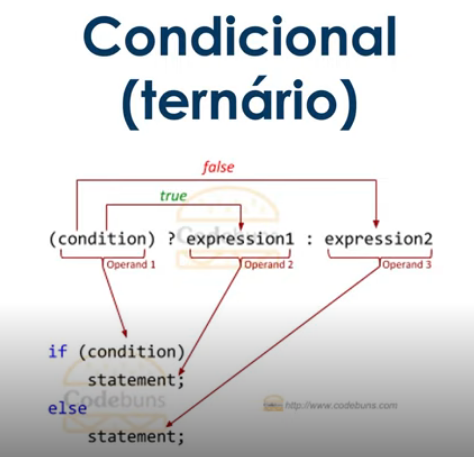
- Comparações



- Lógica



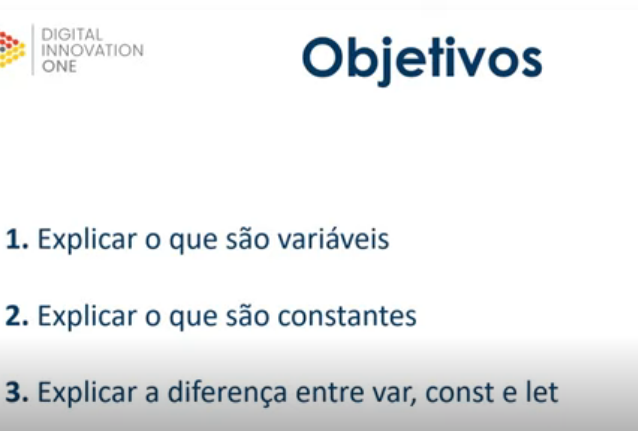
- Condicional



# Variáveis e Tipos com JS

## Variáveis





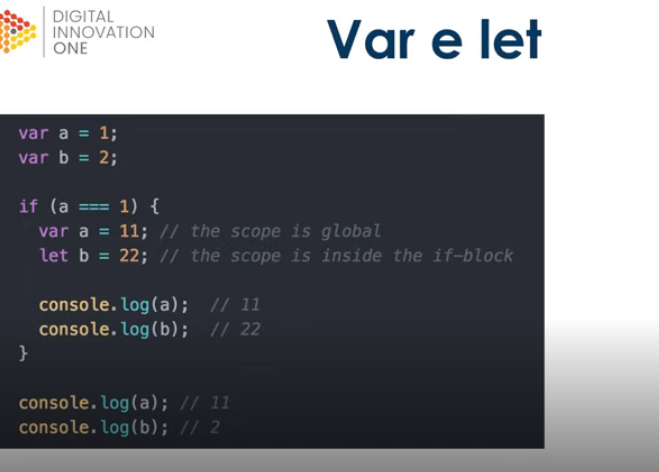


Upper Case Snake Case – usado para constantes

Camel Case – usado para variáveis e funções

Isso são boas práticas no código

## Var e Let



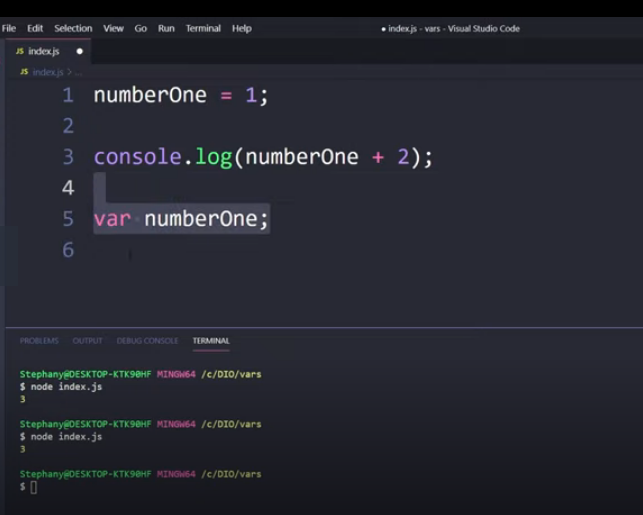
Os escopos são diferentes, o let morre no bloco da função, o var funciona fora da função.

Hoisting – consegue atribuir valores para uma variável antes de declarar ela. E só acontece com a palavra var

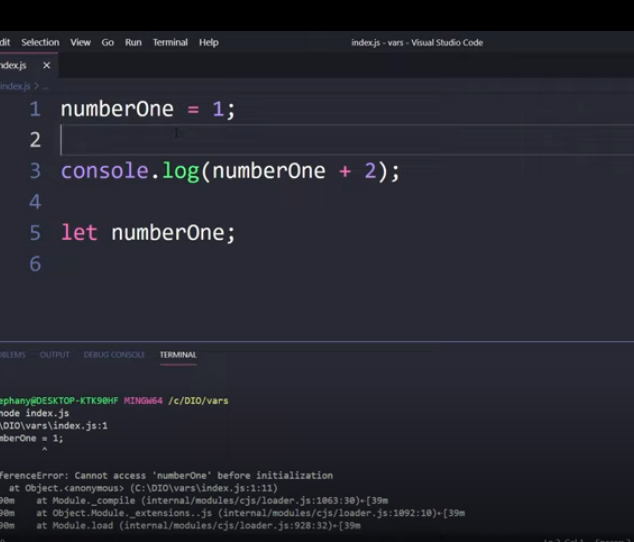
Redeclarar – só funciona com o var

Reatribuição –

Sempre utilizar camelcase

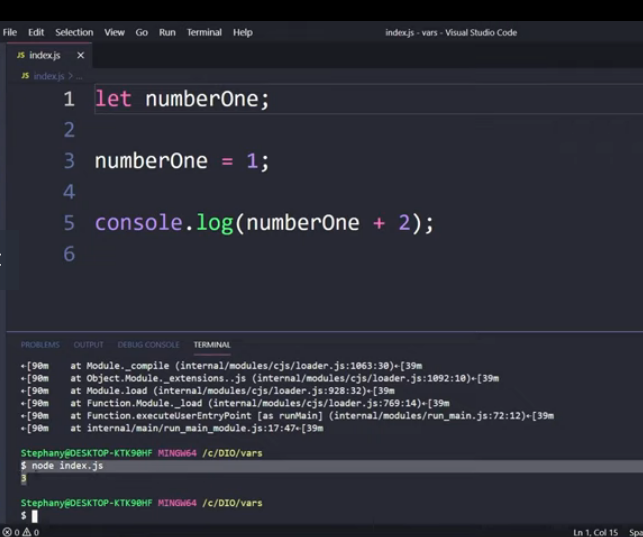


Exemplo de atribuição e hoisting do var

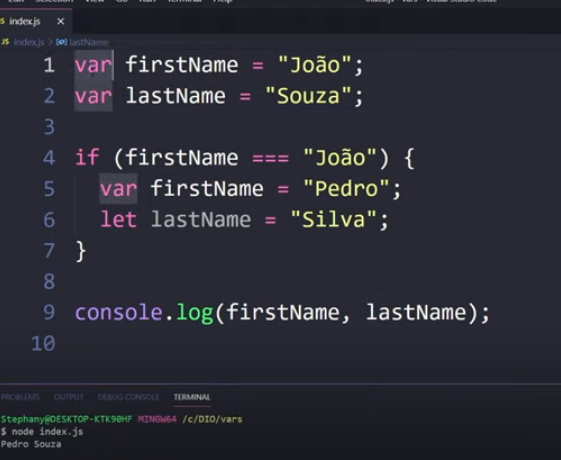


Quando você tenta fazer o hoisting com o let, vai dar erro e pedir para colocar no início.

O certo é colocar da seguinte forma:



Exemplo do let e var numa função if



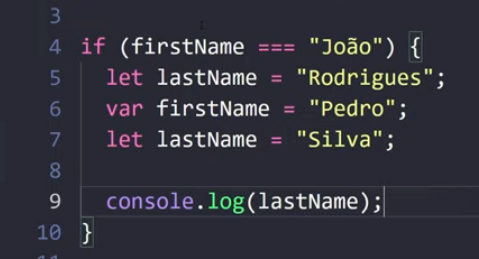
O let está dentro do if, e não está sendo chamado em mais lugar nenhum no código, nota-se que até a cor fica diferente pois há esse erro no código. Portanto o resultado dessa função será: Pedro Souza.

Var = global, funciona no código todo

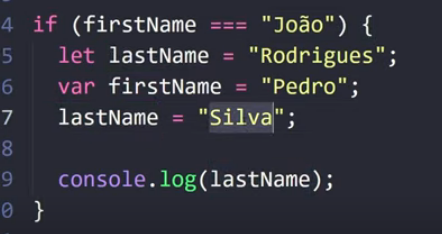
Let = escopo de bloco, só funciona onde foi chamado

Como boas praticas sempre utilizar dentro do bloco e não de forma global, pois em códigos muito grandes pode dar erro, ou atribuir o valor de forma errada e como forma de segurança.

Dar preferencia em chamar o LET e colocar o console.log dentro da função



E esse seria o modelo de uma reatribuição



O let é muito mais utilizado do que o var.

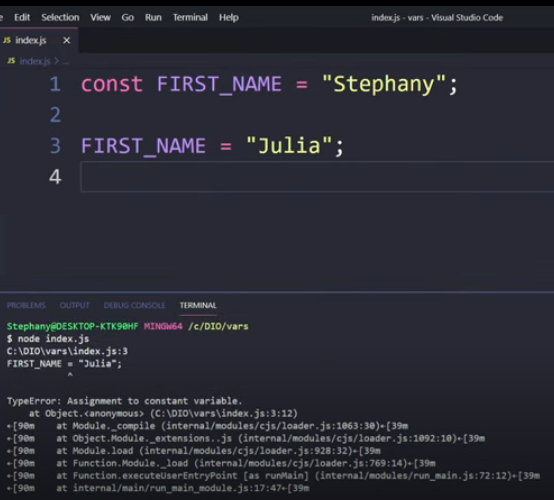
## Constantes

Declaração de número

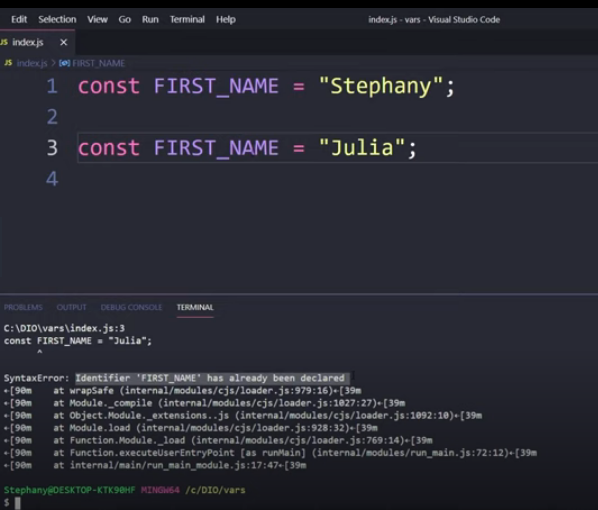


Declaração de string

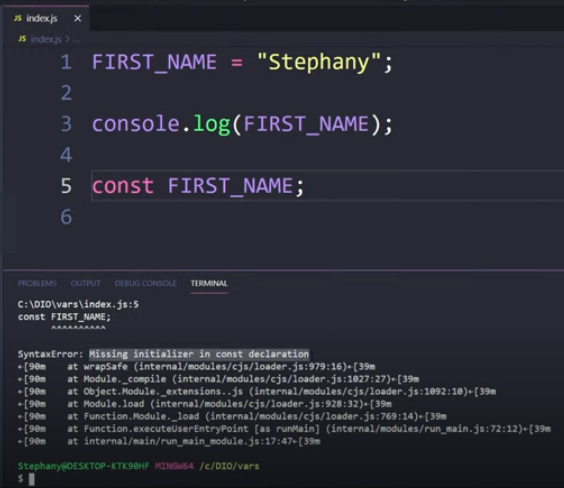
Constante não pode ser reatribuida, caso tente dará o seguinte erro:



Uma constante não pode ser redeclarado, pois dará o seguinte erro:



Não funciona hoisting, dará o seguinte erro:



O jeito de colocar a constante e rodar, ficaria assim:



## Resumo de var, let e const

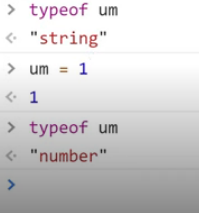


# TIPOS

## Estruturas de Dados

- JS é uma linguagem de tipagem dinâmica, ou seja, antes de declarar um valor, você não especifica o tipo dele.

Para descobrir o tipo usar typeof ‘nome da variável’:

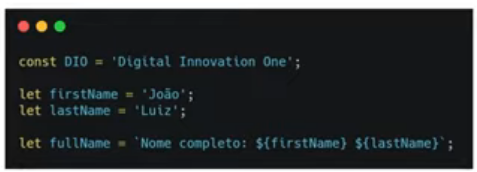




Tipos primitivos não tem métodos dentro deles e são escritos em letra minúscula.

Tipos não primitivos são objetos e arrays.

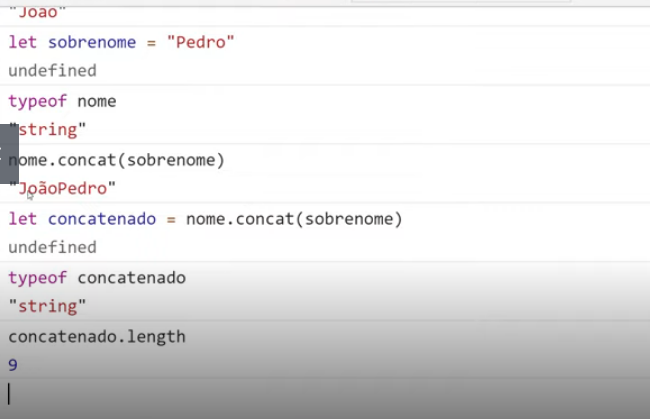
## Strings



String chamando variáveis

- Textos ou frases que você quer escrever no programa;

- Geralmente são escritas dentro de aspas ou crases;

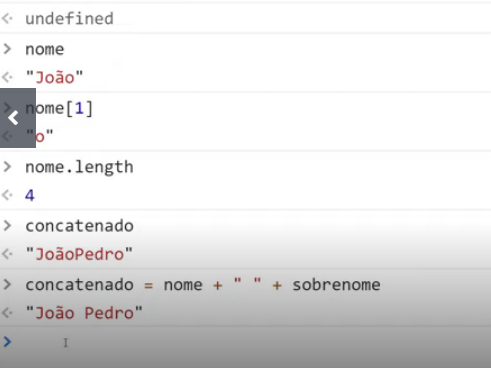


Exemplo de concatenar

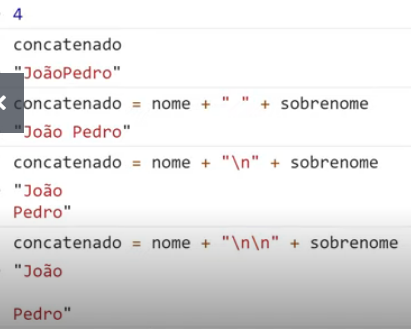
- índice começa do 0;

- tamanho começa do 1;

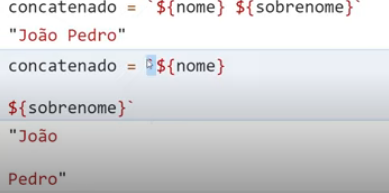
Como colocar o valor concatenado tudo junto com espaço entre variáveis diferentes.



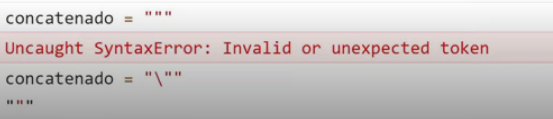
Como fazer com códigos de quebra de linhas e espaços



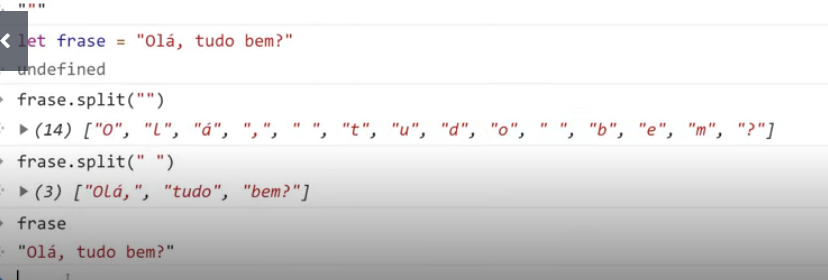
Como fazer sem colocar um monte de caracteres

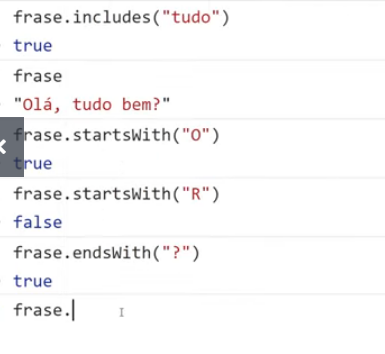


Para imprimir aspas no texto precisa colocar a “\””



Se estiver usando crase nem precisa colocar a \





## Numbers

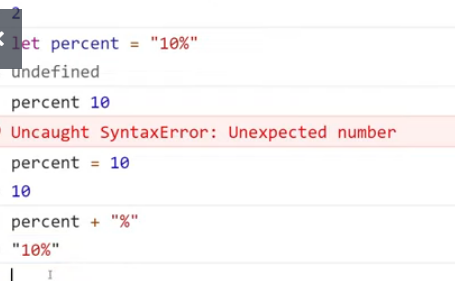
- inteiros ou decimais;

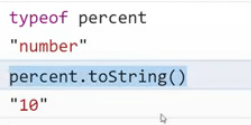
- Os números são declarados sem aspas ou crase;

- podem ser feitos todas as operações aritméticas básica com eles;

- somente fazendo as operações, quando você chama novamente o número, ele não é modificado;

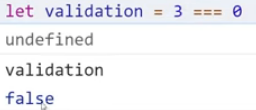
- para aparecer o valor do resultado das operações, você precisa chamar novamente essa variável;

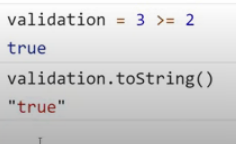


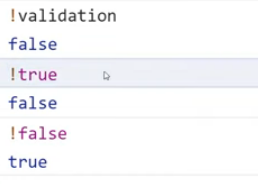


## Booleans

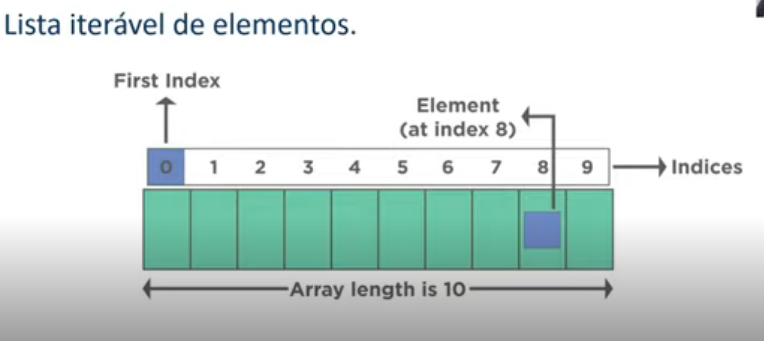
- sempre retorna com true ou false





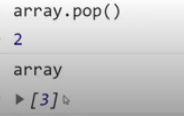


## Arrays

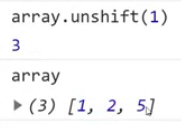


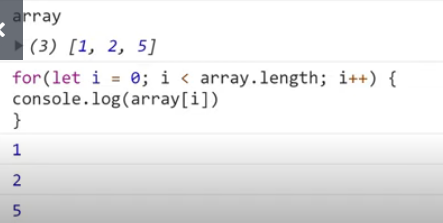
Arrays podem ser chamados de vetores no campo estudantil.



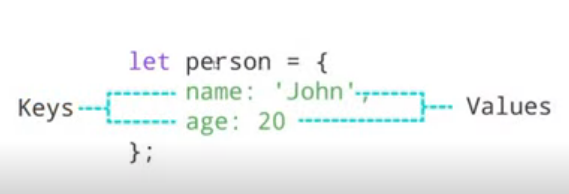






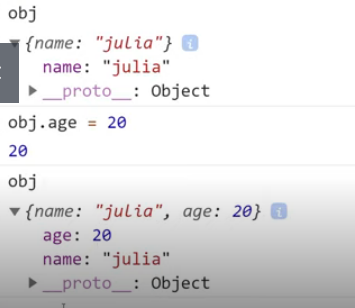


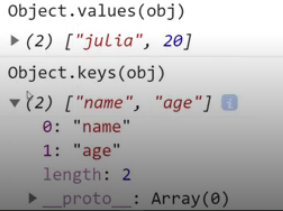
## Objetos

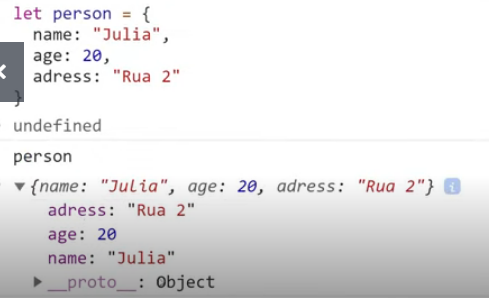


- estrutura tipo chave e valor;









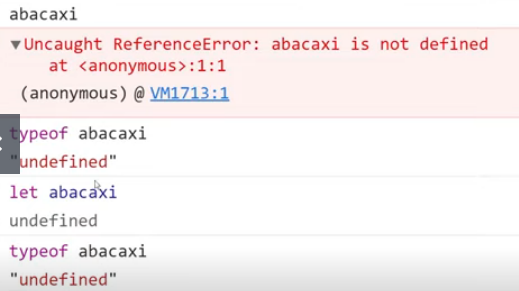




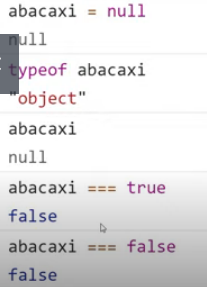
## Empty, Undefined e Null



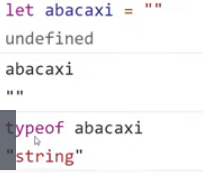
- Exemplo de undefined



- exemplo de null



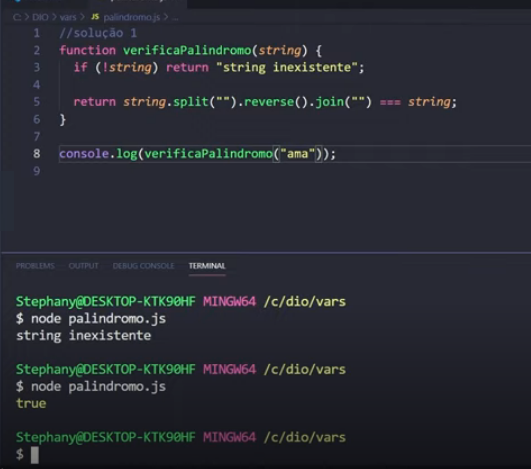
- exemplo de empty

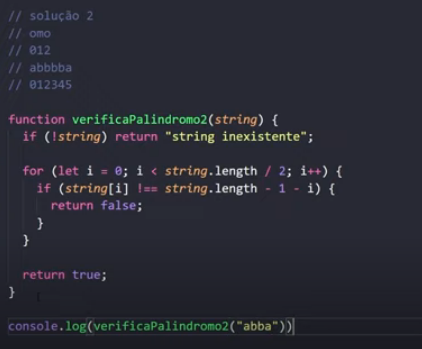
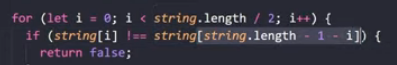


Como funciona no boolean



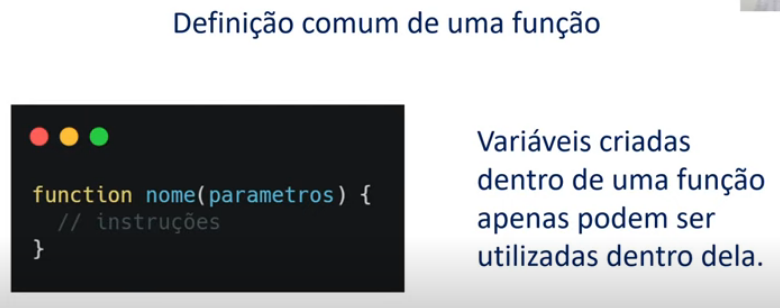
## Atividade palíndromo





# DOMINANDO FUNÇÕES EM JS

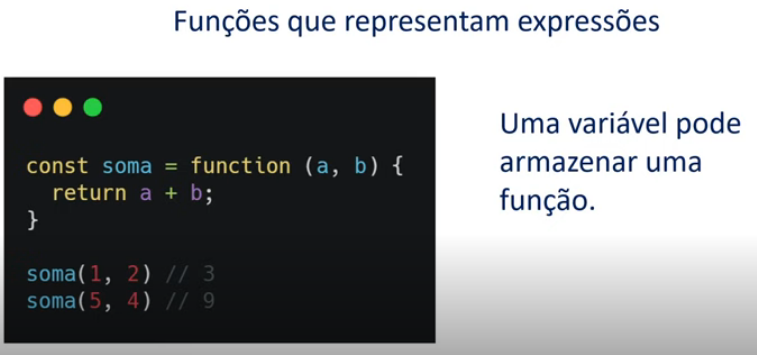
## Estrutura da função



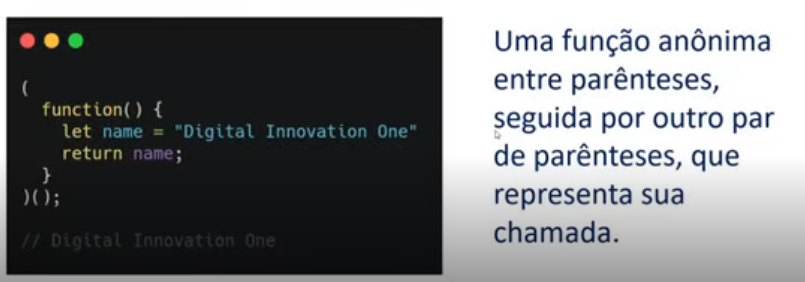


## Função anônima

Funções que representam expressões



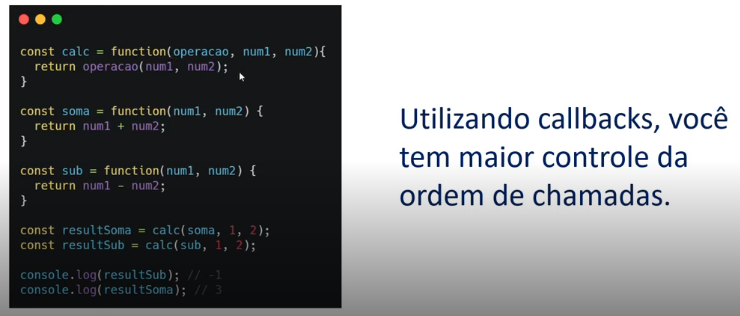
## Função autoinvocável (IIFE)



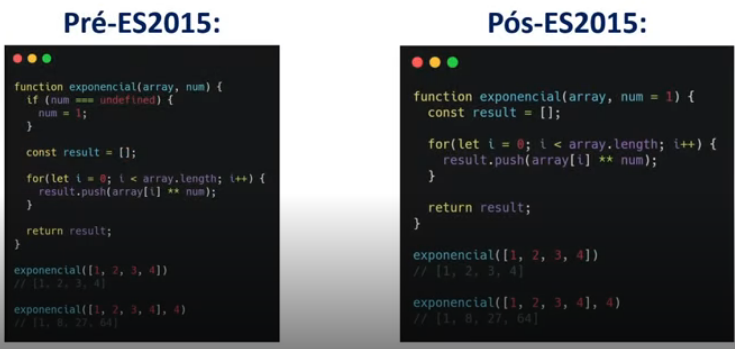


## Função Callbacks

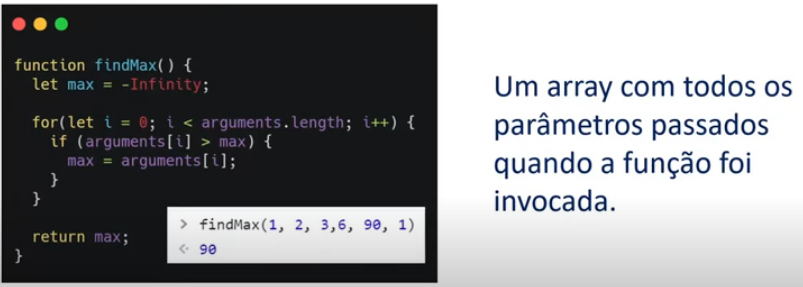
Uma função passada como argumento para outra

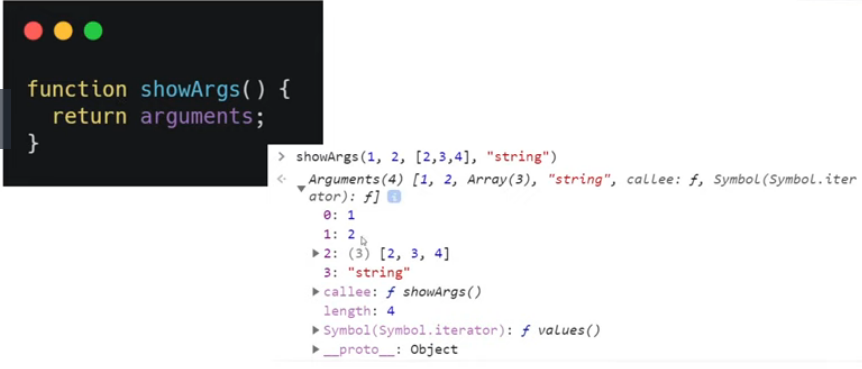


## Parâmetros - Valores padrão



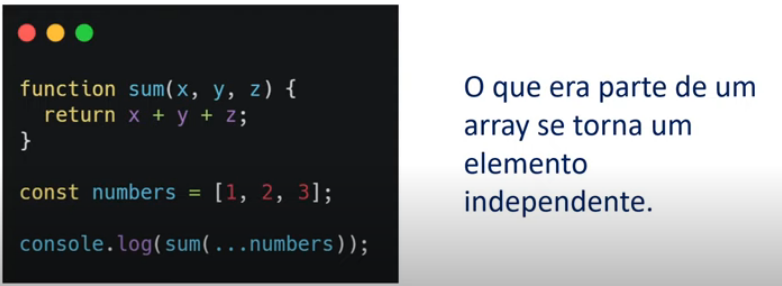
## Parâmetros – objeto ‘arguments’



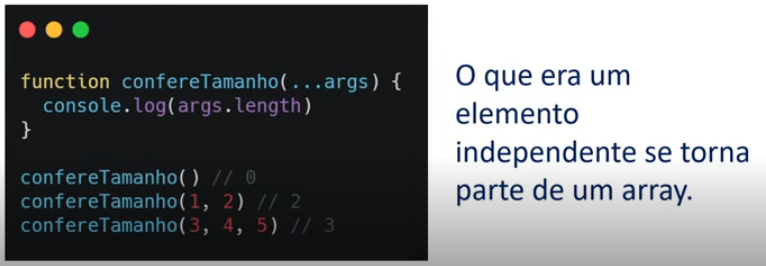


## Parâmetros – Arrays

Spread – uma forma de lidar separadamente com elementos

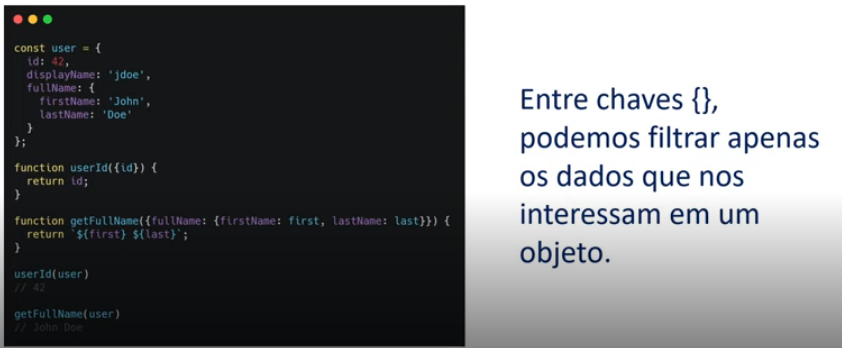


Rest – combina os argumentos em uma array

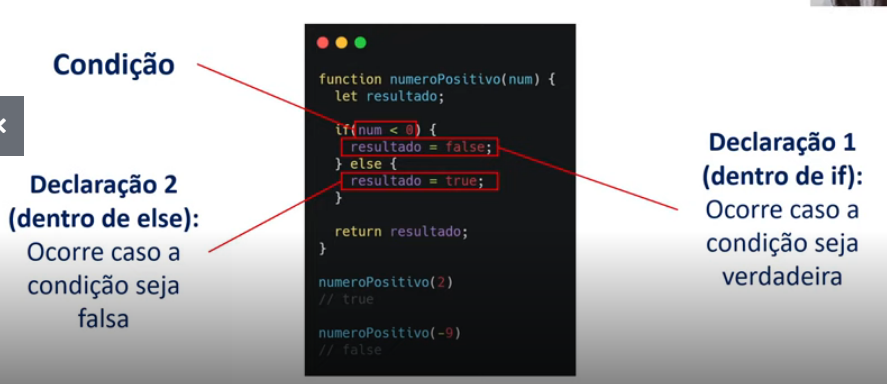


## Parâmetros – objetos

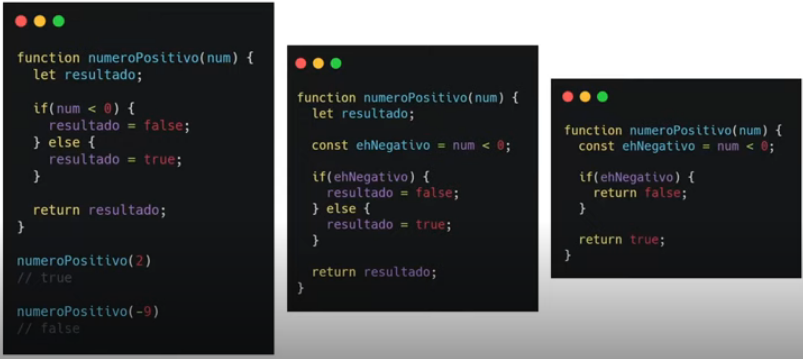
Object destructuring



## Loops – if/else

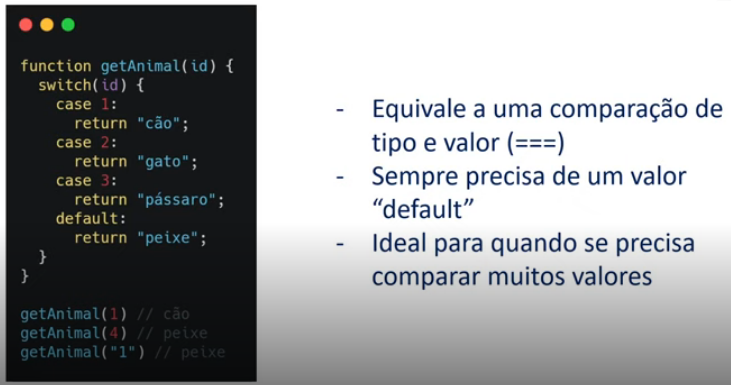


Exemplos de como usar



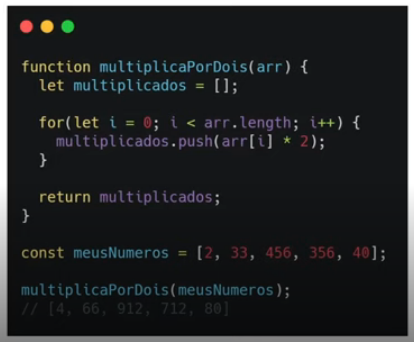


## Loops – switch-case



## Loops – For

É um loop dentro dos elementos iteráveis (arrays, strings)



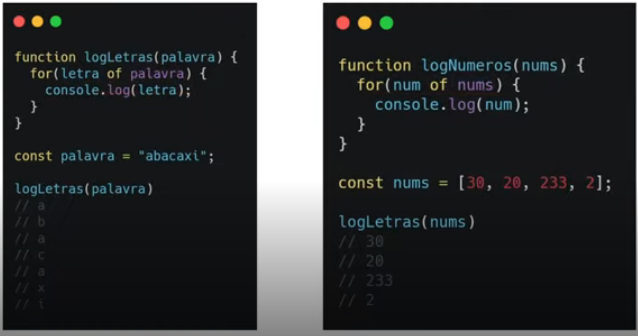
Variações do FOR...IN

Loop entre propriedades enumeráveis de um objeto

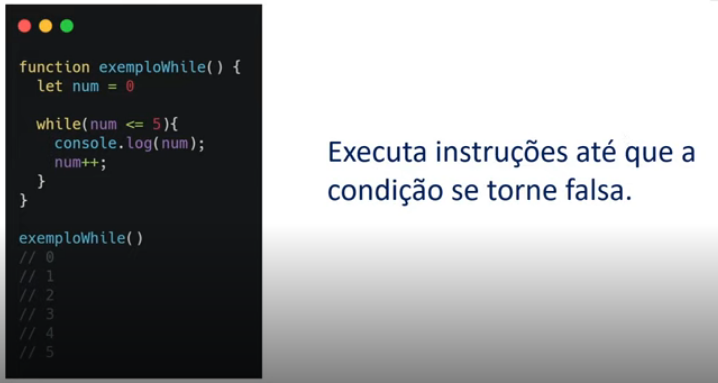


Variações de FOR...OF

Loop entre estruturas iteráveis (arrays, strings)



## Loops – While



## Loops – Do...while

