

JAVASCRIPT PARA INICIANTES

Variáveis

Antes de Iniciarmos

- Criar um documento HTML (index.html)
- Criar um documento JavaScript (script.js)
- Linkar o script ao html

```
<script src="script.js"></script>
```

Variáveis

Responsáveis por guardar dados na memória.

Inicia com a palavra `var` , `let` ou `const`

```
var nome = 'André';
let idade = 28;
const possuiFaculdade = true;
```

Evitam repetições

DRY (Don't repeat yourself)

```
var preco = 20;  
var totalComprado = 5;  
var precoTotal = preco * totalComprado;
```

Sintaxe

Palavra chave `var` seguida do nome, sinal de igual e o valor.

```
var nome = 'André';
var idade = 28;
var possuiFaculdade = true;
```

Vírgula

Utilizei a vírgula para criar mais de uma variável, sem repetir a palavra chave var.

```
var nome = 'André',  
    idade = 28,  
    possuiFaculdade = true;
```

Sem valor

Pode declarar ela e não atribuir valor inicialmente.

```
var precoAplicativo;  
// retorna undefined
```

Nome

—

- Os nomes podem iniciar com \$, _, ou letras.

Podem conter números mas não iniciar com eles

- Case sensitive

nome é diferente de Nome

- Não utilizar palavras reservadas

https://www.w3schools.com/js/js_reserved.asp

- Camel case

É comum nomearmos assim: abrirModal

Nome

```
// Inválido
var $nome;
var function;
var 1possuiFaculdade;

// Válido
var $selecionar;
var _nome;
var possuiFaculdadeNoExterior;
```

Declarar

Erro ao tentar utilizar uma variável que não foi declarada.

```
console.log(nome);  
// retorna nome is not defined
```

Hoisting

São movidas para cima do código, porém o valor atribuído não é movido.

```
console.log(nome);
var nome = 'André';
// Retorna undefined

var profissao = 'Designer';
console.log(profissao);
// Retornar Designer
```

Mudar o valor atribuído

É possível mudar os valores atribuídos a variáveis declaradas com `var` e `let`. Porém não é possível modificar valores das declaradas com `const`

```
var idade = 28;  
idade = 29;  
  
let preco = 50;  
preco = 25;  
  
const possuiFaculdade = true;  
possuiFaculdade = false;  
// Retorna um erro
```

Exercício

Dica rápida, frases (string) devem ser colocadas entre aspas (simples ou duplas) e números não levam aspas.

// Declarar uma variável com o seu nome

// Declarar uma variável com a sua idade

// Declarar uma variável com a sua comida

// favorita e não atribuir valor

// Atribuir valor a sua comida favorita

// Declarar 5 variáveis diferentes sem valores

JAVASCRIPT PARA INICIANTES

Tipos de Dados

7 tipos de Dados

Todos são primitivos exceto os objetos.

```
var nome = 'André'; // String
var idade = 28; // Number
var possuiFaculdade = true; // Boolean
var time; // Undefined
var comida = null; // Null
var simbolo = Symbol() // Symbol
var novoObjeto = {} // Object
```

| Primitivos são dados imutáveis.

Verificar tipo de Dado

```
var nome = 'André';
console.log(typeof nome);
// retorna string
```

| `typeof null` retorna object

String

Você pode somar uma string e assim concatenar as palavras.

```
var nome = 'André';
var sobrenome = 'Rafael';
var nomeCompleto = nome + ' ' + sobrenome;
```

String

Você pode somar números com strings, o resultado final é sempre uma string.

```
var gols = 1000;  
var frase = 'Romário fez ' + gols + ' gols';
```

Aspas Duplas, Simples e Template String

```
'JavaScript é "super" fácil';
"JavaScript é 'super' fácil";
"JavaScript é \"super\" fácil";
`JavaScript é "super" fácil`;
"JavaScript é "super" fácil"; // Inválido
```

Não necessariamente precisamos atribuir valores a uma variável

Template String

```
var gols = 1000;
var frase1 = 'Romário fez ' + gols + ' gols';
var frase2 = `Romário fez ${gols} gols`; // Utilizando Template S
```

Você deve passar expressões /
variáveis dentro de `${}`

Exercício

Dica rápida, frases (string) devem ser colocadas entre aspas (simples ou duplas) e números não levam aspas.

// Declare uma variável contendo uma string

// Declare uma variável contendo um número dentro de uma string

// Declare uma variável com a sua idade

// Declare duas variáveis, uma com seu nome

// e outra com seu sobrenome e some as mesmas

// Coloque a seguinte frase em uma variável: It's time

// Verifique o tipo da variável que contém o seu nome

JAVASCRIPT PARA INICIANTES

Números e Operadores

Números

```
var idade = 28;  
var gols = 1000;  
var pi = 3.14; // ponto para decimal  
var exp = 2e10; // 20000000000
```

| *Precisão para até 15 dígitos*

Operadores Aritméticos

```
var soma = 100 + 50; // 150
var subtracao = 100 - 50; // 50
var multiplicacao = 100 * 2; // 200
var divisao = 100 / 2; // 50
var expoente = 2 ** 4; // 16
var modulo = 14 % 5; // 4
```

Lembrando que `soma`  em `Strings`
serve para concatenar

Operadores Aritméticos (Strings)

```
var soma = '100' + 50; // 10050
var subtracao = '100' - 50; // 50
var multiplicacao = '100' * '2'; // 200
var divisao = 'Comprei 10' / 2; // NaN (Not a Number)
```

É possível verificar se uma variável é `NaN` ou não com a função `isNaN()`

NaN = Not a Number

```
var numero = 80;  
var unidade = 'kg';  
var peso = numero + unidade; // '80kg'  
var pesoPorDois = peso / 2 // NaN (Not a Number)
```

A ordem importa

Começa por multiplicação e divisão, depois por soma e subtração.

```
var total1 = 20 + 5 * 2; // 30
var total2 = (20 + 5) * 2; // 50
var total3 = 20 / 2 * 5; // 50
var total4 = 10 + 10 * 2 + 20 / 2; // 40
```

| *Parênteses para priorizar uma expressão*

Operadores Aritméticos Unários

```
var incremento = 5;  
console.log(incremento++); // 5  
console.log(incremento); // 6
```

```
var incremento2 = 5;  
console.log(++incremento2); // 6  
console.log(incremento2); // 6
```

Mesma coisa para o decremento

--x

Operadores Aritméticos Unários

O **+** e **-** tenta transformar o valor seguinte em número

```
var frase = 'Isso é um teste';
+frase; // NaN
-frase; // NaN
```

```
var idade = '28';
+idade; // 28 (número)
-idade; // -28 (número)
console.log(+idade + 5); // 33
```

```
var possuiFaculdade = true;
console.log(+possuiFaculdade); // 1
```

O **-** antes de um número torna
ele negativo

Guia Completo de Operadores

[https://developer.mozilla.org/pt-
BR/docs/Web/JavaScript/Guide/Expressions_and_Operators](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Expressions_and_Operators)

JAVASCRIPT PARA INICIANTES

Boolean e Condicionais

Boolean

Existem dois valores booleanos `false` ou `true`.

```
var possuiGraduacao = true;  
var possuiDoutorado = false;
```

Condicionais If e Else

Verificar se uma expressão é verdadeira com `if`, caso contrário o `else` será ativado.

```
var possuiGraduacao = true;

if(possuiGraduacao) {
    console.log('Possui graduação');
} else {
    console.log('Não possui graduação');
}
// retorna Possui Graduação e não executa o else
```

O valor dentro dos parênteses sempre será avaliado em `false` ou `true`.

Condicionais Else If

Se o `if` não for verdadeiro, ele testa o `else if`

```
var possuiGraduacao = true;
var possuiDoutorado = false;

if(possuiDoutorado) {
    console.log('Possui graduação e doutorado');
} else if(possuiGraduacao) {
    console.log('Possui graduação, mas não possui doutorado');
} else {
    console.log('Não possui graduação');
}

// retorna Possui Graduação, mas não possui doutorado
```

Switch

Com o `switch` você pode verificar se uma variável é igual à diferentes valores utilizando o `case`. Caso ela seja igual, você pode fazer alguma coisa e utilizar a palavra chave `break;` para cancelar a continuação. O valor de default ocorrerá caso nenhuma das anteriores seja verdadeira.

```
var corFavorita = 'Azul';

switch (corFavorita) {
  case 'Azul':
    console.log('Olhe para o céu.');
    break;
  case 'Vermelho':
    console.log('Olhe para rosas.');
    break;
  case 'Amarelo':
    console.log('Olhe para o sol.');
    break;
  default:
    console.log('Feche os olhos');
```

Truthy e Falsy

Existem valores que retornam `true` e outros que retornam `false` quando verificados em uma expressão booleana.

```
// Falsy
if(false)
if(0) // ou -0
if(NaN)
if(null)
if(undefined)
if('') // ou "" ou ``
```

| *Todo o resto é truthy*

Truthy

```
// Truthy
if(true)
if(1)
if(' ')
if('andre')
if(-5)
if({})
```

Operador Lógico de Negação !

O operador `!`, nega uma operação booleana. Ou seja, `!true` é igual a `false`

```
// Truthy
if(!true) // false
if(!1) // false
if(!'') // true
if(!undefined) // true
if (!! '') // true
if (!! '') // false
```

Dica, você pode utilizar o `!!` para verificar se uma expressão

Operadores de comparação

Vão sempre retornar um valor booleano

```
10 > 5; // true
5 > 10; // false
20 < 10; // false
10 <= 10 // true
10 >= 11 // false
```

Operadores de comparação

O `==` faz uma comparação não tão estrita e o `===` faz uma comparação estrita, ou seja, o tipo de dado deve ser o mesmo quando usamos `==`

```
10 == '10'; // true
10 == 10; // true
10 === '10'; // false
10 === 10 // true
10 != 15 // true
10 != '10' // false
10 !== '10' // true
```

Operadores Lógicos &&

`&&` Compara se uma expressão `e` a outra é verdadeira

```
true && true; // true
true && false; // false
false && true; // false
'Gato' && 'Cão'; // 'Cão'
(5 - 5) && (5 + 5); // 0
'Gato' && false; // false
(5 >= 5) && (3 < 6); // true
```

*Se ambos os valores forem true
ele irá retornar o último valor
verificado Se algum valor for
false ele irá retornar o mesmo e*

Operadores Lógicos ||

|| Compara se uma expressão ou outra é verdadeira

```
true || true; // true
true || false; // true
false || true; // true
'Gato' || 'Cão'; // 'Gato'
(5 - 5) || (5 + 5); // 10
'Gato' || false; // Gato
(5 >= 5) || (3 < 6); // true
```

Retorna o primeiro valor true que encontrar

Exercício

```
// Verifique se a sua idade é maior do que a de algum parente  
// Dependendo do resultado coloque no console 'É maior', 'É igual'  
  
// Qual valor é retornado na seguinte expressão?  
var expressao = (5 - 2) && (5 - ' ') && (5 - 2);  
  
// Verifique se as seguintes variáveis são Truthy ou Falsy  
var nome = 'Andre';  
var idade = 28;  
var possuiDoutorado = false;  
var empregoFuturo;  
var dinheiroNaConta = 0;  
  
// Compare o total de habitantes do Brasil com China (valor em mi
```

```
// O que irá aparecer no console?  
if('Gato' === 'gato') && (5 > 2) {  
    console.log('Verdadeiro');  
} else {  
    console.log('Falso');  
}
```

```
// O que irá aparecer no console?  
if('Gato' === 'gato') || (5 > 2) {  
    console.log('Gato' && 'Cão');  
} else {  
    console.log('Falso');  
}
```

JAVASCRIPT PARA INICIANTES

Funções

Funções

Bloco de código que pode ser executado e reutilizado. Valores podem ser passados por uma função e a mesma retorna outro valor.

```
function areaQuadrado(lado) {  
    return lado * lado;  
}  
  
areaQuadrado(4) // 16  
areaQuadrado(5) // 25  
areaQuadrado(2) // 4
```

| *Chamada de function declaration*

Funções

```
function pi() {  
    return 3.14;  
}  
  
var total = 5 * pi(); // 15.7
```

| Parênteses `()` executam uma
| função

Parâmetros e Argumentos

Ao **criar** uma função, você pode definir **parâmetros**.

Ao **executar** uma função, você pode passar **argumentos**.

```
// peso e altura são os parâmetros
function imc(peso, altura) {
  const imc = peso / (altura ** 2);
  return imc;
}

imc(80, 1.80) // 80 e 1.80 são os argumentos
imc(60, 1.70) // 60 e 1.70 são os argumentos
```

Separar por vírgula cada parâmetro. Você pode definir mais de um parâmetro ou nenhum também

Parênteses executa a função

```
function corFavorita(cor) {  
    if(cor === 'azul') {  
        return 'Você gosta do céu';  
    } else if(cor === 'verde') {  
        return 'Você gosta de mato';  
    } else {  
        return 'Você não gosta de nada';  
    }  
}  
  
corFavorita(); // retorna 'Você não gosta de nada'
```

Se apenas definirmos a função com o `function` e não executarmos a mesma, nada que estiver dentro dela irá acontecer

Argumentos podem ser funções

Chamadas de Callback, geralmente são funções que ocorrem após algum evento.

```
addEventListener('click', function() {  
    console.log('Clicou');  
});  
// A função possui dois argumentos  
// Primeiro é a string 'click'  
// Segundo é uma função anônima
```

Funções anônimas são aquelas em que o nome da função não é definido, escritas como

`function() {} ou () => {}`

Pode ou não retornar um valor

Quando não definimos o return, ela irá retornar `undefined`. O código interno da função é executado normalmente, independente de existir valor de return ou não.

```
function imc(peso, altura) {  
  const imc = peso / (altura ** 2);  
  console.log(imc);  
}  
  
imc(80, 1.80); // retorna o imc  
console.log(imc(80, 1.80)); // retorna o imc e undefined
```

Valores retornados

Uma função pode retornar qualquer tipo de dado e até outras funções.

```
function terceiraIdade(idade) {  
    if(typeof idade !== 'number') {  
        return 'Informe a sua idade!';  
    } else if(idade >= 60) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

Cuidado, retornar diferentes tipos de dados na mesma função não é uma boa ideia.

Escopo

Variáveis e funções definidas dentro de um bloco `{ }`, não são visíveis fora dele.

```
function precisoVisitar(paisesVisitados) {  
  var totalPaises = 193;  
  return `Ainda faltam ${totalPaises - paisesVisitados} países pa  
}  
console.log(totalPaises); // erro, totalPaises não definido
```

Escopo Léxico

Funções conseguem acessar variáveis que foram criadas no contexto `pai`

```
var profissao = 'Designer';

function dados() {
  var nome = 'André';
  var idade = 28;
  function outrosDados() {
    var endereco = 'Rio de Janeiro';
    var idade = 29;
    return `${nome}, ${idade}, ${endereco}, ${profissao}`;
  }
  return outrosDados();
}

dados(); // Retorna 'André, 29, Rio de Janeiro, Designer'
outrosDados(); // retorna um erro
```

Hoisting

Antes de executar uma função, o JS 'move' todas as funções declaradas para a memória

```
imc(80, 1.80); // imc aparece no console

function imc(peso, altura) {
  const imc = peso / (altura ** 2);
  console.log(imc);
}
```

Exercício

// Crie uma função para verificar se um valor é Truthy

// Crie uma função matemática que retorne o perímetro de um quadrado
// lembrando: perímetro é a soma dos quatro lados do quadrado

// Crie uma função que retorne o seu nome completo
// ela deve possuir os parâmetros: nome e sobrenome

// Crie uma função que verifica se um número é par

// Crie uma função que retorne o tipo de
// dado do argumento passado nela (typeof)

// addEventListener é uma função nativa do JavaScript
// o primeiro parâmetro é o evento que ocorre e o segundo o callback
// utilize essa função para mostrar no console o seu nome completo

```
function precisoVisitar(paisesVisitados) {  
    var totalPaises = 193;  
    return `Ainda faltam ${totalPaises - paisesVisitados} países pa-  
}  
function jaVisitei(paisesVisitados) {  
    return `Já visitei ${paisesVisitados} do total de ${totalPaises}  
}  
precisoVisitar(20);  
jaVisitei(20);
```

JAVASCRIPT PARA INICIANTES

Objetos

Objetos

Conjunto de variáveis e funções, que são chamadas de propriedades e métodos.

```
var pessoa = {  
    nome: 'André',  
    idade: 28,  
    profissao: 'Designer',  
    possuiFaculdade: true,  
}  
  
pessoa.nome; // 'André'  
pessoa.possuiFaculdade; // true
```

*Propriedades e métodos consistem
em nome (chave) e valor*

Métodos

É uma propriedade que possui uma função no local do seu valor.

```
var quadrado = {  
    lados: 4,  
    area: function(lado) {  
        return lado * lado;  
    },  
    perimetro: function(lado) {  
        return this.lados * lado;  
    },  
}  
  
quadrado.lados; // 4  
quadrado.area(5); // 25  
quadrado.perimetro(5); // 20
```

Métodos

Abreviação de `area: function() {}` para `area()`, no ES6+

```
var quadrado = {
    lados: 4,
    area(lado) {
        return lado * lado;
    },
    perimetro(lado) {
        return this.lados * lado;
    },
}
```

Organizar o Código

Objetos servem para organizar o código em pequenas partes reutilizáveis.

```
Math.PI; // 3.14  
Math.random(); // número aleatório  
  
var pi = Math.PI;  
console.log(pi); // 3.14
```

`Math` é um objeto nativo de JavaScript. Já percebeu que `console` é um objeto e `log()` um método?

Criar um Objeto

Um objeto é criado utilizando as chaves `{ }`

```
var carro = {};
var pessoa = {};

console.log(typeof carro); // 'object'
```

Dot Notation Get

Acesse propriedades de um objeto utilizando o ponto `.`

```
var menu = {  
    width: 800,  
    height: 50,  
    backgroundColor: '#84E',  
}
```

```
var bg = menu.backgroundColor; // '#84E'
```

Dot Notation Set

Substitua o valor de uma propriedade utilizando `.` e o `=` após o nome da mesma.

```
var menu = {  
    width: 800,  
    height: 50,  
    backgroundColor: '#84E',  
}  
  
menu.backgroundColor = '#000';  
console.log(menu.backgroundColor); // '#000'
```

Adicionar Propriedades e Métodos

Basta adicionar um novo nome e definir o valor.

```
var menu = {  
    width: 800,  
}  
  
menu.height = 50;  
menu.position = 'fixed';
```

Palavra-chave this

`this` irá fazer uma referência ao próprio objeto.

```
var height = 120;  
var menu = {  
    width: 800,  
    height: 50,  
    metadeHeight() {  
        return this.height / 2;  
    }  
}  
  
menu.metadeHeight(); // 25  
// sem o this, seria 60
```

`this` irá retornar o próprio
objeto

Protótipo e Herança

O objeto herda propriedades e métodos do objeto que foi utilizado para criar o mesmo.

```
var menu = {  
    width: 800,  
}  
  
menu.hasOwnProperty('width') // true  
menu.hasOwnProperty('height') // false
```

hasOwnProperty é um método de Object

JAVASCRIPT PARA INICIANTES

Tudo é Objeto

Tudo é Objeto

Strings, Números, Boolean, Objetos e mais, possuem propriedades e métodos. Por isso são objetos.

```
var nome = 'André';  
  
nome.length; // 5  
nome.charAt(1); // 'n'  
nome.replace('ré', 'rei'); // 'Andrei'  
nome; // 'André'
```

Uma string herda propriedades e métodos do construtor `String()`

Números

```
var altura = 1.8;  
  
altura.toString(); // '1.8'  
altura.toFixed(); // '2'
```

*Por um breve momento o número é
envolvido em um Objeto
(coerção), tendo acesso assim as
suas propriedades e métodos*

Funções

```
function areaQuadrado(lado) {  
    return lado * lado;  
}  
  
areaQuadrado.toString();  
// "function areaQuadrado(lado) {"  
//   return lado * lado;  
// }"  
  
areaQuadrado.length; // 1
```

Elementos do DOM

```
<a class="btn">Clique</a>
```

```
var btn = document.querySelector('.btn');

btn.classList.add('azul') // adiciona a classe azul
btn.innerText; // 'Clique'
btn.addEventListener('click', function() {
  console.log('Clicou')
})
```

Praticamente todos os efeitos com JS são feitos utilizando propriedades e métodos de objetos do DOM.

Objetos revolucionaram a programação

Web API's são métodos e propriedades que permitem a interação JavaScript e Browser.

Exercício

// nomeie 3 propriedades ou métodos de strings

// nomeie 5 propriedades ou métodos de elementos do DOM

// busque na web um objeto (método) capaz de interagir com o clipboard
// clipboard é a parte do seu computador que lida com o CTRL + C

JAVASCRIPT PARA INICIANTES

Arrays e Loops

Array

É um grupo de valores geralmente relacionados. Servem para guardarmos diferentes valores em uma única variável.

```
var videoGames = ['Switch', 'PS4', 'XBox'];

videoGames[0] // Switch
videoGames[2] // Xbox
```

Acesse um elemento da array
utilizando `[n]`

Métodos e Propriedades de uma Array

```
var videoGames = ['Switch', 'PS4', 'XBox'];

videoGames.pop(); // Remove o último item e retorna ele
videoGames.push('3DS'); // Adiciona ao final da array
videoGames.length; // 3
```

Existem diversos outros métodos,
como `map`, `reduce`, `forEach` e
mais que veremos mais à frente

For Loop

Fazem algo repetidamente até que uma condição seja atingida.

```
for (var numero = 0; numero < 10; numero++) {  
    console.log(numero);  
}  
// Retorna de 0 a 9 no console
```

O for loop possui 3 partes,
início, **condição** e **incremento**

While Loop

```
var i = 0;
while (i < 10) {
    console.log(i);
    i++;
}
// Retorna de 0 a 9 no console
```

| O for loop é o mais comum

Arrays e Loops

```
var videoGames = ['Switch', 'PS4', 'XBox', '3DS'];
for (var i = 0; i < videoGames.length; i++) {
    console.log(videoGames[i]);
}
```

O for loop é o mais comum

Break

O loop irá parar caso encontro e palavra `break`

```
var videoGames = ['Switch', 'PS4', 'XBox', '3DS'];
for (var i = 0; i < videoGames.length; i++) {
    console.log(videoGames[i]);
    if(videoGames[i] === 'PS4') {
        break;
    }
}
```

forEach

forEach é um método que executa uma função para cada item da Array. É uma forma mais simples de utilizarmos um loop com arrays (ou array-like)

```
var videoGames = ['Switch', 'PS4', 'XBox', '3DS'];
videoGames.forEach(function(item) {
  console.log(item);
});
// O argumento item será atribuído dinamicamente
```

Podemos passar os seguintes parâmetros `item`, `index` e `array`

Não se confunda com a sintaxe

```
var numero = 0;  
var maximo = 50;  
for(;numero < maximo;) {  
    console.log(numero);  
    numero++;  
}
```

*Não aconselho escrever da forma
acima, mas funciona normalmente.*

Exercício

```
// Crie uma array com os anos que o Brasil ganhou a copa  
// 1959, 1962, 1970, 1994, 2002  
  
// Interaja com a array utilizando um loop, para mostrar  
// no console a seguinte mensagem, 'O brasil ganhou a copa de ${a}  
  
// Interaja com um loop nas frutas abaixo e pare ao chegar em Pera  
var frutas = ['Banana', 'Maçã', 'Pera', 'Uva', 'Melância']  
  
// Coloque a última fruta da array acima em uma variável,  
// sem remover a mesma da array.
```

JAVASCRIPT PARA INICIANTES

Atribuição e Ternário

Comentários

Servem para explicar o código

```
// Comentário de uma linha  
  
/*  
Comentário  
com diversas  
linhas  
*/  
  
// var nome = 'André';
```

*Comentar uma linha de código
desativa a mesma. Não deixe
linhas de código comentadas no
arquivo final.*

Operadores de Atribuição

Podem funcionar como formas abreviadas

```
var x = 5;  
var y = 10;  
x += y; // x = x + y (15)  
x -= y; // x = x - y (-5)  
x *= y; // x = x * y (50)  
x /= y; // x = x / y (0.5)  
x %= y; // x = x % y (0)  
x **= y; // x = x ** y (9765625)
```

Operador Ternário

Abreviação de condicionais com `if` e `else`

```
var idade = 19;  
var podeBeber = (idade >= 18) ? 'Pode beber' : 'Não pode beber';  
console.log(podeBeber) // Pode beber  
  
// condição ? true : false
```

Geralmente utilizado quando precisamos atribuir um valor para uma variável, dependendo de uma condição

If Abreviado

Não é necessário abrir e fechar as chaves `{}` quando retornamos apenas uma linha de código

```
var possuiFaculdade = true;  
if(possuiFaculdade) console.log('Possui faculdade');  
else console.log('Não possui faculdade');  
  
// ou  
if(possuiFaculdade)  
    console.log('Possui faculdade');  
else  
    console.log('Não possui faculdade');
```

Eu particularmente prefiro a segunda opção aqui.

Exercício

```
// Some 500 ao valor de scroll abaixo,  
// atribuindo o novo valor a scroll  
var scroll = 1000;  
  
// Atribua true para a variável darCredito,  
// caso o cliente possua carro e casa.  
// E false caso o contrário.  
var possuiCarro = true;  
var possuiCasa = true;  
var darCredito;
```

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.

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

[COPIAR](#)

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);
```