**JavaScript (JS)**

**documento de ano 2021 elaborado por Rodrigo Ehlert Cavalheiro**

# Afirmações e comentários:

#### Breve História:

Em 1989 Tim Bernners-Lee cria web (www) e no Cern ele criou 1º navegador.

Em 1991 lei criada por Al Gore de incentivo a tecnologia.  
Foi criado o mosaic web browser.

Em 1993 netscape contrata Brendan Eich que cria Mocha(móca)  
Em 1995 é renomeada para Live Script e logo depois para Java Script  
 - a microsoft lança o JScript. ("plágio")

Em 1997 netscape solicita a padronização do código onde é fundado a ECMA script responsável pela padronização.

#### ECMA (1997)

**E**urope **C**o**M**puter **A**ssossiation   
padronização do JavaScript  
  
ES - **E**cma**S**cript  
ES8 (2018)

#### Afirmações:

coloque o arquivo .js no final do <body> do arquivo.HTML  
<script src="app.js"></script> </body> // src == 'source'  
  
usa { } para definir o bloco de código.  
  
usa ; para definir o fim de uma instrução.  
  
Console:  
abra um site no google chrome e com o botão direito vá em inspecionar e entre na guia console para executar os códigos em .js   
clear() == para limpar a tela.

No VScode pode-se utilizar o plugin RunCode. (nodejs já deve estar instalado)  
  
identação: geralmente quatro espaços em branco. (tecla tab)  
Servem para deixa o código mais limpo e visualmente interpretável.

#### Comentários:

Uma linha de cometário é definida pelo caracter //   
Exemplo:  
// uma linha de comentário  
  
Várias linhas de comentário ficam entre /\* comentário \*/  
Exemplo:  
/\* três  
linhas  
comentário \*/

# Declaração de váriaveis

let city = 'Tokyo'; // block scoped  
const country = "Japan"; // váriavel imutável (block scoped)  
var state = 'São Paulo'; // all scoped  
a = 1; // não declarado não da erro mas pode causar resultados inesperados.  
usa tipagem dinâmica. Exemplo: a = 1; b = bola  
já associa a como tipo númerico e b como texto(string) não declara no .JS  
  
Não pode declarar começando com nº  
let 1user = 45; // error: Invalid or unexpected token  
  
1 + 'hi';  
output: '1hi' // concatena e não da erro  
  
JS usa-se camelCase (tituloDoConteudo)

## tipos primitivos

string (o que estiver entre "aspas" vira texto)  
var nomeCompleto = 'RodrigoCavalheiro';  
typeof nomeCompleto; // "string"

number (10, -10, 0)  
var idade = 37.38;  
typeof idade; // "number"

boolean (true, false)  
var aprovado = true; // ou false  
typeof aprovado; // "boolean"

indefinido (não existe)  
let name;  
typeof name; // "undefined"

nulo (existe mas esta vazio)   
let color = null;  
typeof name; // "null"

link: (para mais informações sobre variáveis)  
<https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Variables>

# Hoisting (içar)

default behavior of moving declarations to the top  
var e functions fazem hoisting

# Tipo de operadores:

## Aritméticos:

|  |  |
| --- | --- |
| **OPERADORES** | **DESCRIÇÃO** |
| **+** | Soma |
| **-** | Subtração |
| **\*** | Multiplicação |
| **/** | Divisão |
| **//** | Divisão de inteiros |
| **\*\*** | Exponenciação |
| **%** | Módulo |
| **++** | incremento i++ | i = i + 1) |
| **--** | decremento i-- | i = i - 1) |

var dividendo = 11;

var divisor = 4;

var resultadoDoModulo = dividendo % divisor;    
// Retorna: 3 (que é a sobra)

## Relacionais: (comparações)

|  |  |  |
| --- | --- | --- |
| **OPERADORES** | **DESCRIÇÃO** | **Retorno** |
| **<** | Menor. | 2 < **'1'** **false** |
| **>** | Maior. | 2 > **'1'** **true** |
| **>=** | Maior igual. | 2 >= **'2'** **true** |
| **<=** | Menor igual. | 2 >= **'2'** **true** |
| **==** | valor é igual. | (**'1'** == 1)**true** |
| **===** | valor e tipo são estritamente iguais. | (**'1'** === 1)**false** |
| **!=** | valor é diferente. | (**'1'** != 1)**false** |
| **!==** | valor e tipo são estritamente diferentes. | (**'1'** !== 1)**true** |

Tipo de dado bool ou boolean o Output sempre True(verdadeiro) ou False  
valores falso:  
false, 0, "" (empty string), null, undefined, NaN (not an number)  
todo restante é verdadeiro.  
  
**operadores de filiação (in, not in)**usados para verificar se o algum valor ou variável se encontra na sequência  
  
let trees = ['redwood', 'bay', 'cedar', 'oak', 'maple']  
0 in trees; // true  
!(0 in trees); // false (not in)  
3 in trees; // true  
6 in trees; // false  
'bay' in trees;   
// false (must specify the index number, not the value at that index)

!('bay' in trees); // true ! == not (not in)  
'length' in trees; // true (length is an Array property)

**operadores de identificação (is, is not)**  
blabalablablabalbalbabalalabablbalblalabblablablabalbablablablalbalbalbalbalb

## Lógico:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OPERADORES** | **Descrição** | **Valor 1** | **Valor 2** | **Resultado** |
| **&&** | **and** | True | True | True |
| True | False | False |
| False | False | False |
| **||** | **or** | True | True | True |
| True | False | True |
| False | False | False |
| **!()** | **not** | True | | False |
| False | | True |

1 == 1; // True  
1 == 1 || 1 > 2; // True  
1 == 1 && 1 > 2; // False  
1 == 1 && !(1 > 2); // True

# Console (print):

console.log("hello", 1 + 2);  
console.warn("exibe msg no console formatada como aviso (!)");  
console.error('exibe msg no console formatada como erro (X)');  
alert('exibe msg como pop up ao abrir o site');  
variavelx = prompt('abre uma janela para o usuário digitar');  
  
link:  
<https://developer.mozilla.org/en-US/docs/Web/API/console>

# Metódos ()

## string:

### link:

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/substr>

### Template Literals

let a = 5;  
let b = 10;  
// concatenado  
console.log('cinquenta é ' + (a + b) + ' e não ' + (2 \* a + b) + '.');  
  
// usando template literals (`) ${}  
console.log(`cinquenta é ${a + b} e não ${2 \* a + b}.`);

// pular linha sem template literals coloca \n  
console.log(`Primeira linha;  
Segunda linha;  
Terceira linha;`);

### parseInt()

parseInt('1.8') + 1; // 2 converte para inteiro

### parseFloat()

parseFloat('1.8') + 1; // 2.8 converte para decimal

### .toUpperCase()

'hello'.toUpperCase(); // "HELLO"

### .toLowerCase()

'LOL'.toLowerCase(); // "lol"

### .trim()

' omg '.trim(); // "omg"

### .length

'city'.length; // 4 contagem de letras

### .indexOf

// retorna posição do caracter sendo 0 a 1º (ou -1 se não encontrado)  
'spider'.indexOf('s'); // 0  
'carro'.indexOf('r'); // 2 retorna posição da 1º letra encontrada  
'cactus'.indexOf('z'); // -1 não encontrado  
'hello world'.indexOf('ld') // 9

### .substr()

// .substr("início", "contagem de caracter opcional)  
'Mozilla'.substr(1, 2); // "oz"  
'Mozilla'.substr(2); // "zilla"

### .slice()

// retorna fatiamento do texto  
"pancake".slice(3); // "cake" corte apartir da posição 3  
"pancake".slice(0, 3); // "pan" corte da posição 0 à 3

### .replace()

// retorna novo caracter, substituindo o 1º caracter encontrado.  
'bomba'.replace("b", "p"); // "pomba"

### metódo dentro de metódo

var facialHair = ("skateboard".slice(5, 10)).replace("o", "e"); // 'beard'

## Math

### Math.floor()

// retorna o maior inteiro arredondado para menos de um número informado.  
Math.floor(5.95); // 5  
Math.floor(5.05); // 5  
Math.floor(-5.05); // -6

### Math.random()

// retorna número aleatório de 0 à 0.9999999999999999  
Math.floor(Math.random() \* 3) + 1 // random 1 to 3  
Math.floor(Math.random() \* 10) + 1 // random 1 to 10  
Math.floor(Math.random() \* 10) + 21 // random 21 to 30

### converter número para texto:

// somar o número com aspas  
1 + (3 + ''); // 13

### Math.max()

// retorna o maior número

#### Exemplo: 1

Math.max(3, 10, 18, 20); // 20

#### Exemplo 2: (com array)

// será necessário usar o spread " ... "  
let numbers = [3, 10, 18, 20];  
Math.max(numbers); // NaN (tenta executar no array)  
Math.max(...numbers); // 20 (executa nos argumentos expandidos do array)

### Math.min()

// retorna o maior número

#### Exemplo: 1

Math.min(3, 10, 18, 20); // 3

#### Exemplo 2: (com array)

// será necessário usar o spread " ... "  
let numbers = [3, 10, 18, 20];  
Math.min(numbers); // NaN (tenta executar no array)  
Math.min(...numbers); // 3 (executa nos argumentos expandidos do array)

### Math.abs()

// Retorna valor absoluto (sempre positivo)

#### Exemplo 1:

Math.abs(1 - 4) // 3

# Controle de fluxo

## if...else

syntax:  
if(condição === true){faça}; else(condição === true){faça};

#### Exemplo 1:

let password = prompt("Enter your password");  
if (password.length >= 6 && password.indexOf(' ') === -1) {  
 console.log("password aceito!");  
} else {  
 alert("formato incorreto de password!");  
};

#### Exemplo 2: (if dentro de if)

let age = prompt("digite sua idade");  
if (age != null && age != '') {  
 if ((age >= 0 && age < 6) || age >= 60) {  
 console.log("grátis, criança ou idoso");  
 } else if (age >= 6 && age < 18) {  
 console.log("R$12,50 estudante menor de 18");  
 } else if (age >= 18 && age < 60) {  
 console.log("R$25,00 adulto");  
 } else {  
 alert("digite à idade como número positivo");  
 }  
} else {  
 alert("campo não preenchido");  
};

## switch

let sexo = prompt('você é "homem" ou "mulher" ???')  
switch ((sexo.toLowerCase()).trim()) {  
 case "":  
 alert("digite seu gênero")  
 break; // se o break for esquecido executa até o próximo break  
 case "homem":  
 alert("Bem-vindo!")  
 break;  
 case "mulher":  
 alert("Bem-vinda!")  
 break;  
 default: // default == padrão  
 alert("Sem preconceito... Bem-vindo/a!")  
};

## ternary operator

// sintaxe:

`expressão` ? `retorno se verdadeira` : `retorno se falsa`;

// Exemplo 1:

const trueExpression = (1 + 1 === 2) ? `isso é verdade` : `isso é mentira`;

console.log(trueExpression) // isso é verdade

const falseExpression = (2 + 2 === 3) ? `isso é verdade` : `isso é mentira`;

console.log(falseExpression) // isso é mentira

// Exemplo 2:

// Situação em que é mais simples usar o operador ternário:

const checkIfElse = (age) => {

  if (age >= 18) {

    return `Você tem idade para dirigir!`;

  } else {

    return `Você ainda não tem idade para dirigir...`;

  }

};

const checkTernary = (age) => (

  age >= 18 ? `Você tem idade para dirigir!` : `Você ainda não tem idade para dirigir...`

);

// Exemplo 3:

// Situação em que usar o operador ternário não faz muito sentido:

// código truncado e difícil de ler não é uma boa prática

const checkIfElse = (fruit) => {

  if (fruit === `maçã`) {

    return `Essa fruta é vermelha`;

  } else if (fruit === `banana`) {

    return `Esta fruta é amarela`;

  } else {

    return `Não sei a cor`;

  }

};

const checkTernary = (fruit === `maçã`) ? `Essa fruta é vermelha`

  : ((fruit === `banana`) ? `Esta fruta é amarela` : `Não sei a cor`);

# Array []

## [1] == [1]; // False

let nums = [1];  
let nums2 = [1];  
nums == nums2; // **False**; não possui mesma referencia na memória:   
nums.push(2); // nums se torna [1,2] mas nums2 ainda é [1]  
  
let nums = [1];  
let copia = nums;  
copia == nums; // **True**; copia passa a ter mesma referencia na memória.  
nums.push(2); //nums se torna [1,2]  
copia == nums; // True; copia se torna [1,2] porque copia = nums

## acesso

let meses = ['Jan', 'March', 'April', 'Peru'];  
meses[0]; // "Jan" (1º posição é 0)  
meses[-1]; // undefined  
meses[meses.length - 1]; // "Peru"

## metódos

### .length

// conta quantidade de elementos no array  
let meses = ['Jan', 'March', 'April', 'Peru'];  
meses.length; // 4

### .push(value)

// adiciona value no fim do array  
let meses = ['Jan', 'March', 'April', 'Peru'];  
meses.push('adicionado');  
meses; // ["Jan", "March", "April", "Peru", "adicionado"]

### .pop()

// remove e retorna o ultimo valor dentro do array  
let meses = ['Jan', 'March', 'April', 'Peru'];  
let captura = meses.pop();  
meses; // ['Jan', 'March', 'April']  
captura; // "Peru"

### .unshift(value)

// adiciona value no ínicio do array  
let meses = ['Jan', 'March', 'April', 'Peru'];  
meses.unshift('adicionado');  
meses; // ["adicionado", "Jan", "March", "April", "Peru"]

### .shift()

// remove e retorna o 1º valor dentro do array  
let meses = ['Jan', 'March', 'April', 'Peru'];  
let captura = meses.shift();  
meses; // ['March', 'April', 'Peru']  
captura; // "Jan"

### .concat()

// concatena arrays tornandos um só  
let num1 = [1, 2, 3];  
let num2 = [4, 5, 6];  
let num3 = [7, 8, 9];  
let numbers = num1.concat(num2, num3);  
numbers; // [1, 2, 3, 4, 5, 6, 7, 8, 9]

### .includes()

// verifica se o array possui um determinado elemento  
let cats = ['Tom', 'Garfield'];  
cats.includes('Tom'); // True  
cats.includes('tom'); // False  
[1, 2, 3].includes(3, 3) // false (elemento, posição)  
[1, 2, 3].includes(3, -1) // true

### .indexOf()

// verifica a posição do elemento determinado se não encontrado retorna -1  
let bichos = ['formiga', 'peru', 'camelo', 'peru', 'pato'];  
bichos.indexOf('peru'); // 1  
bichos.indexOf('peru', 2); // 4 (elemento, índice em que ínicia a pesquisa)  
bichos.indexOf('peru', 4); // -1  
bichos.indexOf('girafa'); // -1 não encontrado

### .slice()

// retorna a cópia de uma porção do array determinado (ínicio, fim)  
let bichos = ['formiga', 'peru', 'camelo', 'peru', 'pato'];  
bichos.slice(1,3); // ['peru', 'camelo']  
bichos.slice(-1); // ['pato']  
bichos.slice(-2); // ['peru', 'pato']  
bichos.slice(-2,-1); // ['peru']

### .splice()

// altera o array adicionando ou substituindo seu conteúdo  
// syntax ('na posição', 'nº delete', 'insert values')  
let months = ['Jan', 'March', 'April', 'Peru'];  
months.splice(1, 0, 'Feb'); // ["Jan", "Feb", "March", "April", "Peru"]  
months.splice(4, 1, 'May'); // ["Jan", "Feb", "March", "April", "May"]

### .sort()

// ordena elementos do array

#### exemplo 1: (strings)

let months = ['March', 'Jan', 'Feb', 'Dec'];  
months.sort(); // ["Dec", "Feb", "Jan", "March"]

#### exemplo 2: (numeros)

/\* o sort transforma os elementos do array em string para funcionar com numeros deve-se acrescentar a função "Bubble Sort" (MDN) \*/  
let numArray = [140000, 104, 99];  
numArray.sort(function(a, b) {  
 return a - b;  
});  
console.log(numArray); // [99, 104, 140000]  
  
numArray.sort((a, b) => a - b); // For ascending sort  
numArray.sort((a, b) => b - a); // For descending sort

### .forEach()

// cada elemento do array é retornado após passar pela função

#### Exemplo 1:

let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12];  
  
numbers.forEach(function (elemento) {  
 if (elemento % 2 === 0) {  
 console.log(elemento)  
 }  
}); // Output: 2 4 6 8 10 12

#### Exemplo 2:

// com object literal  
let movies = [  
 {  
 title: 'Amadeus',  
 score: 99  
 },  
 {  
 title: 'Stand By Me',  
 score: 85  
 },  
 {  
 title: 'Parasite',  
 score: 95  
 },  
 {  
 title: 'Alien',  
 score: 90  
 }  
];  
movies.forEach(function (movie) {  
 console.log(`${movie.title} - ${movie.score}/100`)  
});  
/\* Output:  
Amadeus - 99/100  
Stand By Me - 85/100  
Parasite - 95/100  
Alien - 90/100 \*/

### .map()

// retorna um novo array com os elementos executados em uma função

#### Exemplo 1:

let namesToBeTrim =   
[" timothee ", ' darth\_hater', 'sassyfrassy ', ' elton john '];  
  
function cleanNames(names) {  
 let newArrayOfStrings = names.map(function (str) {  
 return str.trim();  
 });  
 return newArrayOfStrings;  
};  
  
cleanNames(namesToBeTrim);  
// Output: ["timothee", "darth\_hater", "sassyfrassy", "elton john"]

#### Exemplo 2:

let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];  
  
let doubles = numbers.map(function (num) {  
 return num \* 2;  
});  
  
doubles; // [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30]

### .filter()

// cria um novo array com todos os elementos que passam no teste implementado

#### Exemplo 1:

let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];  
numbers.filter(n => {  
 return n < 5  
}); // [1, 2, 3, 4]

#### Exemplo 2: (array com object literal)

let movies = [  
 {  
 title: 'Stand By Me',  
 score: 85,  
 year: 1986  
 },  
 {  
 title: 'Waterworld',  
 score: 62,  
 year: 1995  
 },  
 {  
 title: 'Parasite',  
 score: 95,  
 year: 2019  
 },  
 {  
 title: 'Notting Hill',  
 score: 77,  
 year: 1999  
 },  
 {  
 title: 'Alien',  
 score: 90,  
 year: 1979  
 }  
];  
  
movies.filter(m => m.score < 70);   
// Output: {title: "Waterworld", score: 62, year: 1995}  
  
movies.filter(m => m.score > 80).map(m => m.title);  
// Output: ["Stand By Me", "Parasite", "Alien"]

#### Exemplo 3 (função no filter)

let arrayNums = [-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13];  
function numPrimo(num) {  
 for (let i = 2; num > i; i++) {  
 if (num % i == 0) {  
 return false;  
 }  
 }  
 return num > 1;  
};  
  
console.log(arrayNums.filter(numPrimo)); // [2, 3, 5, 7, 11, 13]

### .some()

// verifica se algum dos elementos passam no teste  
function checkAdult(age) {  
 return age >= 18;  
}  
  
let ages = [3, 10, 18, 20];  
ages.some(checkAdult); // true  
  
let ages = [3, 10, 15, 17];  
ages.some(checkAdult); // false

### .every()

// verifica se **todos** os elementos passam no teste

#### Exemplo 1

function goMotel(age) {  
 return age >= 18;  
}  
  
let ages = [23, 30, 18, 20];  
ages.every(goMotel); // true  
  
let ages = [23, 30, 17, 20];  
ages.every(goMotel); // false

#### Exemplo 2

// verificar se os elemntos do array são pares  
function allEvens(arr) {  
 return arr.every(function(num) {  
 return num % 2 === 0;  
 });  
}  
  
allEvens([2, 4, 6, 8]); // true  
allEvens([1, 4, 6, 8]); // false

### .reduce()

/\* reduz o array em um simples valor, o valor do retorno deste metódo é armazenado em um acumulador (resultado/total). \*/

let prices = [9.99, 1.50, 19.99, 49.99, 30.50];

function minPrice(arr) {  
 return arr.reduce((min, price) => {  
 if (price < min) {  
 return price;  
 }  
 return min;  
 });  
}  
  
minPrice(prices); // 1,5

# object literal {key:value}

o "object literal" não é igual ao "**J**ava**S**cript **O**bject **N**otation (JSON)".  
Eles são similar, mas existe direferença entre eles.  
// todos os valores dentro da chave do objeto se tornarão string!   
// incluindo boolean, numbers and null  
// exceto special symbols

link:  
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Object_initializer>

## criação e acesso

console.table(singer); // imprime object em forma de tabela

// for que imprime chave:valor  
for(let key in object) {  
 console.log(key, object[key]);  
}

// adicionando propriedade  
// object['key'] = 'value'   
player['recorrente'] = 'sim'; // recorrente: sim

#### Exemplo 1

let dog = {  
 name: "Rusty",  
 breed: "unknown",  
 isAlive: false,  
 age: 7,  
 tags: ['#hilarius', '#funny', '#silly'],  
 commentText: 'Hello World',  
 true: 'what hell?'  
};  
  
dog.true; // 'what hell?'  
dog['true']; // 'what hell?'  
dog.age; // 7  
  
dog.age = 8;  
dog['breed'] = 'mutt';

#### Exemplo 2: object com array

let comentarios = [  
 { nomeusuario: 'peixe boi', texto: 'glub glub', votes: 9 },  
 { nomeusuario: 'rodrigo', texto: 'ha ha - ha ha', votes: 69 }  
];  
comentarios[1].texto; // 'ha ha - ha ha'

#### Exemplo 3:

function techList(tech, name) {  
 let lista = [];  
 if (tech.length <= 0) {  
 return 'Vazio!';  
 }  
 tech = tech.sort();  
 // add parans in object  
 for (let index in tech) {  
 lista.push(  
 object = {  
 name: name,  
 tech: tech[index]  
 }  
 );  
 }  
 return lista;  
}

## Destructuring

// extrair dados de arrays, e das propriedades dos objetos em váriaveis

#### Exemplo 1: (array)

let colocacao = ['Rodrigo', 'Sandra', 'Aledxandre', 'Raphael', 'Felipe'];  
// let highScore = scores[0];  
// let secondHighScore = scores[1];  
let [gold, silver, bronze, ...everyoneElse] = colocacao;

#### Exemplo 2: (object)

let user = {  
 email: 'harvey@gmail.com',  
 password: 'sCoTt1948sMiTh',  
 firstName: 'Harvey',  
 lastName: 'Milk',  
 born: 1930,  
 died: 1978,  
 city: 'San Francisco',  
 state: 'California'  
}  
  
let user2 = {  
 email: 'Stacy@gmail.com',  
 firstName: 'Stacy',  
 lastName: 'Gonzalez',  
 born: 1987,  
 city: 'Tulsa',  
 state: 'Oklahoma'  
}  
// let firstName = user.firstName;  
// let email = user.email;  
let { email, firstName, lastName, city } = user;  
  
// let { born: birthYear, died: deathYear = 'N/A' } = user;  
// let { city, state, died = 'N/A' } = user2;

#### Exemplo 3: (param)

let user = {  
 firstName: 'Harvey',  
 lastName: 'Milk'  
}  
3.1 regular use: (object acess)  
// function fullName(user) {  
// return `${user.firstName} ${user.lastName}`  
// }  
3.2 criando váriavel com destructuring  
// function fullName(user) {  
// let { firstName, lastName } = user;  
// return `${firstName} ${lastName}`  
// }  
3.3 destructuring param  
function fullName({ firstName, lastName }) {  
 return `${firstName} ${lastName}`  
}

#### Exemplo 4: (usando destructuring em metódos)

let movies = [  
 {  
 title: 'Waterworld',  
 score: 62,  
 year: 1995  
 },  
 {  
 title: 'Parasite',  
 score: 95,  
 year: 2019  
 },  
 {  
 title: 'Alien',  
 score: 90,  
 year: 1979  
 }  
]  
// movies.filter((movie) => movie.score >= 90)  
movies.filter(({ score }) => score >= 90)  
  
// movies.map(movie => {  
// return `${movie.title} (${movie.year}) is rated ${movie.score}`  
// })  
  
movies.map(({ title, score, year }) => {  
 return `${title} (${year}) is rated ${score}`  
})

## adicionar propriedade

Para adicionar propriedades ver o [SPREAD Exemplo 3: (object literal)](#_Exemplo_3:_(object)

## excluir propriedade

let meuObjeto = {  
 name: "Rodrigo",  
 idade: "40"};

delete myObject.idade; // True (deletou idade)  
meuObjeto; // {name: "Rodrigo"}

# Spread ...

/\* permite que um iterável seja expandido em argumentos.

#### Exemplo 1: (metódo)

/\* veja os exemplos [Math.max()](#_Exemplo_2:_(com) e [Math.min()](#_Exemplo_2:_(com_1) acima, onde é necessárrio expandir o array para que o metódo seja executado. \*/

#### Exemplo 2: (array)

let cats = ['Blue', 'Scout', 'Rocket'];  
let dogs = ['Rusty', 'Wyatt'];  
let allPets = [...cats, ...dogs]; [// concatena arrays tornandos um só](#_.concat())  
allPets; // ["Blue", "Scout", "Rocket", "Rusty", "Wyatt"]

#### Exemplo 3: (object literal)

let dataFromForm = {  
 email: 'blueman@gmail.com',  
 password: 'tobias123!',  
 username: 'tfunke'  
}  
let novoUsuario = { ...dataFromForm, id: 2345, isAdmin: false }  
// novo usuario terá os dados do "dataFromForm" + os adicionados  
  
// adicionando nova propriedade  
dataFromForm = { ...dataFromForm, itenQualquer: 'adicionado'};

# Loop

## for

syntax:  
for( [i Expressão]; [condição]; [incremento] ) { "faça dentro do loop"; }

#### exemplo 1:

for (let i = 1; i <= 10; i += 1) { // (i += 1) == ( i = i + 1)   
 console.log(i);  
}; // output: números do 1 ao 10

#### Exemplo 1.2:

resultado = [];  
frase = 'ola mundo';  
for(var i = 0; i < frase.length; i++){  
 resultado.push(frase[i])  
};  
resultado; // ["o", "l", "a", " ", "m", "u", "n", "d", "o"]

#### exemplo 2: (com array ordem crescente)

let bichos = ['formiga', 'peru', 'camelo', 'peru', 'pato'];  
for (let i = 0; i < bichos.length; i++) {  
 console.log(i, bichos[i]);  
}; // output: a posição e os bichos um a um

#### exemplo 3.1: (com array ordem decrescente)

let bichos = ['formiga', 'peru', 'camelo', 'peru', 'pato'];  
for (let i = bichos.length - 1; i >= 0; i--) {  
 console.log(bichos[i].toUpperCase())  
} // outpout: os bichos um a um na ordem decrescente em letra maiuscula

#### exemplo 3.2:

var palindromo = "";  
var palavra = 'juca';  
for(var i = palavra.length - 1; i >= 0; i--) {  
 palindromo += palavra[i]  
};  
palindromo; // 'acuj'

#### exemplo 4.1: (for dentro de for)

// tambem chamados de: (for aninhado) (nested for)  
for (let i = 1; i <= 10; i++) {  
 console.log(`i is: ${i}`)  
 for (let j = 1; j < 4; j++) {  
 console.log(` j is: ${j}`)  
 }  
}  
// Output:  
 i is: 1  
 j is: 1  
 j is: 2  
 j is: 3  
 i is: 2  
 j is: 1  
 j is: 2  
 j is: 3

// .... até o   
  
 i is: 10  
 j is: 1  
 j is: 2  
 j is: 3

#### exemplo 4.2:

function imprimaQuadradoAsterisco(n) {

for (var i = 0; i < n; i++){  
 var linhaDeAsteriscos = ' ';  
 for (var linha = 0; linha < n; linha++){  
 linhaDeAsteriscos = linhaDeAsteriscos + '\*';  
 };  
 console.log(linhaDeAsteriscos);  
}; // fecha for  
}; // fecha function

imprimaQuadradoAsterisco(3);  
// \*\*\*  
// \*\*\*  
// \*\*\*

#### exemplo 4.3: (for com array bidimensional)

let oceano = [  
 ['camarão', 'lagosta', 'carangueijo'],  
 ['tilapia', 'sardinha', 'tubarão', 'baleia'],  
 ['alga', 'esponja\_do\_mar', 'coral']  
]  
  
for (let i = 0; i < oceano.length; i++) {  
 let linha = oceano[i];  
 console.log(`linha #${i + 1}`);  
 for (let j = 0; j < linha.length; j++) {  
 console.log(linha[j]);  
 }  
console.log(' ');  
}

#### exemplo 5: (break)

for (let i = 0; i < 1000; i++) {  
 console.log(i);  
 if (i === 100) break;  
} // Output: nº do 1 ao 100

## for of

não suportado pelo internet explorer

#### Exemplo 1

let filmes = ['xuxa', 'kruul', 'matrix', 'piranhas', 'tubarão'];  
for (let i of filmes) {  
 console.log(`iten da lista: ${i}`)  
}

#### Exemplo 2

let oceano = [  
 ['camarão', 'lagosta', 'carangueijo'],  
 ['tilapia', 'sardinha', 'tubarão', 'baleia'],  
 ['alga', 'esponja\_do\_mar', 'coral']  
]  
  
for (let linha of oceano) {  
 console.log(`linha #`);  
 for (let elementos of linha) {  
 console.log(elementos);  
 }  
}

## for (com object literal)

let testScores = {  
 keenan: 80,  
 damon: 67,  
 kim: 89,  
 shawn: 91,  
 marlon: 72  
}  
  
for (let pessoa in testScores) {  
 console.log(`${pessoa} scored ${testScores[pessoa]}`);  
}  
  
let total = 0;  
let scores = Object.values(testScores);  
for (let i of scores) {  
 total += i;  
}  
console.log( total/scores.length ); // Output 79.7 (media do score)

## while

syntax:  
while ( [expressão] ) { "faça dentro do loop"; }

#### exemplo 1

let count = 0;  
while (count < 10) {  
 count++;  
 console.log(count)  
}; // output: nº do 1 ao 10

#### exemplo 2 (break)

let input = prompt( "Hey, say something!" )  
while (true) {  
 input = prompt(input);  
 if (input.toLowerCase() === "stop copying me" ) break;  
}  
console.log("ok, you win")

#### exemplo 3 (aceita somente nº)

let maximum = parseInt(prompt("Enter the maximum number!"));  
while (!maximum) { // fica perguntando até dígitar um número  
 maximum = parseInt(prompt("Enter a valid number!"));  
}  
console.log(maximum) // Output: nº dígitado

## do...while

syntax:  
do { "faça dentro do loop"; } while ( [expressão] )

#### exemplo 1:

let result = '';  
let i = 0;  
do {  
 i = i + 1;  
 result = result + i;  
} while (i < 5); // output: 12345

# function

sytanx:  
function funcName('arguments (opcional)') {'faça alguma coisa'}

## function (com loop)

function repeat(str, numTimes) {  
 let result = '';  
 for (let i = 0; i < numTimes; i++) {  
 result += str;  
 }  
 console.log(result);   
};  
repeat('repita ', 3);  
// Output: repita repita repita

## function (com return)

function capitalize(palavra) {  
 return palavra.charAt(0).toUpperCase() + palavra.slice(1);  
//console.log('erro: códigos após return nunca será executado')  
};  
capitalize('bicho'); // "Bicho"

## function (com if)

// validação númerica  
function add(x, y) {  
 if (typeof x !== 'number' || typeof y !== 'number') {  
 return false;  
 }  
 return x + y;  
};  
add(add(2,6),5); // 13

## function (com array)

#### exemplo 1:

function ultimoElemento(x) {  
 if (x.length === 0) {  
 return null;  
 }  
 else {  
 return x[x.length - 1];  
 }  
};  
ultimoElemento([3, 5, 7]); // 7

#### exemplo 2:

function sumArray(array) {  
 let x = 0;  
 for (let i = 0; i < array.length; i++) {  
 x = x + array[i];  
 }  
 return x;  
}  
sumArray([1, 2, 3, 1]); // 7

## function (com switch)

function returnDay(day) {  
 if (day < 1 || day > 7) {  
 return null;  
 } else {  
 switch (day) {  
 case 1:  
 return "monday - segunda";  
 case 2:  
 return "Tuesday - terça";  
 case 3:  
 return "Wednesday - quarta";  
 case 4:  
 return "Thursday - quinta";  
 case 5:  
 return "Friday - sexta";  
 case 6:  
 return "Saturday - sabado";  
 case 7:  
 return "Sunday - domingo";  
 } // fecha switch  
 } // fecha else  
}; // fecha function  
returnDay(1); // "monday - segunda"

## function (dentro de function)

function bankRobbery() {  
 let heroes = ['Spiderman', 'Wolverine', 'Black Panther', 'Batwoman']  
 function cryForHelp() {  
 let color = 'purple';  
 function inner() {  
 for (let hero of heroes) {  
 console.log( `PLEASE HELP US, ${hero.toUpperCase()}`)  
 }  
 }  
 inner(); // func chamado dentro de func. "lexical scope"   
 }  
 cryForHelp();  
}

## arrow

// forma mais compacta de criar função.  
// **toda arrow é uma function anonima**  
// não é suportado pelo internet explorer.

#### Exemplo 1:

let add = (x, y) => { return x + y; };  
add(1,2) // 3

#### Exemplo 2: implícito X explícito

// 2.1 explicity return  
let rollDie = () => {  
 return Math.floor(Math.random() \* 6) + 1  
};  
  
// 2.2 implicity return (2x parenteses ao invés de return))  
// funciona somente com retorno simples (1x)  
let rollDie = () => (  
 Math.floor(Math.random() \* 6) + 1   
);

#### Exemplo 3:

function contaPalavras(frase) {  
 return frase.split(' ').length;  
}  
// arrow: (sempre anonima)  
const contaPalavras = frase => frase.split(' ').length;

console.log(contaPalavras('Fala tribo, beleza?')); // 3

#### Exemplo 4: (object)

function objetoPessoa(nome, idade) {  
 return {  
 nome: nome,  
 idade: idade  
 }  
}  
  
const objetoPessoa = (nome, idade) => ({ nome: nome, idade: idade });

console.log(objetoPessoa('Ana', 25));  
console.table(objetoPessoa('Ana', 25));  
  
Mais sobre arrows:  
https://www.freecodecamp.org/news/when-and-why-you-should-use-es6-arrow-functions-and-when-you-shouldnt-3d851d7f0b26/

## function (default)

// default == valor padrão

#### Exemplo 1: (default de "b")

function multiply(a, b = 1) {  
 return a \* b;  
}  
console.log(multiply(5, 2)); // 10  
console.log(multiply(5)); // 5 (porque por default b = 1)

#### Exemplo 2: (forma antiga)

function rollDie(numSides) {  
 if (numSides === undefined) {  
 numSides = 6 // default é 6  
 }  
 return Math.floor(Math.random() \* numSides) + 1  
}

## function (arguments)

function func1(a, b, c) {  
 console.log(arguments[0]); // 1  
 console.log(arguments[1]); // 2  
 console.log(arguments[2]); // 3  
func1(1, 2, 3);

## function (rest ... )

// aceita um número indefinido de argumentos  
// representado por ... mas não é spread

#### Exemplo 1:

function sum(...nums) {  
 return nums.reduce((total, el) => total + el)  
}  
sum(1,2,3,4,5) // 15

#### Exemplo 2:

function raceResults(gold, silver, ...todoRestante) {  
 console.log(`medalha de ouro vai para: ${gold}`)  
 console.log(`medalha de prata vai para: ${silver}`)  
 console.log(`e obrigado a todos os outros: ${todoRestante}`)  
}  
raceResults('Rodrigo', 'Sandra', 'Aledxandre', 'Raphael', 'Felipe');  
/\* Output:  
medalha de ouro vai para: Rodrigo  
medalha de prata vai para: Sandra  
e obrigado a todos os outros: Aledxandre,Raphael,Felipe \*/

# criação de metódos

// todo metódo é uma função; Mas nem toda função é um metódo.

#### Exemplo 1:

const myMath = {  
 PI: 3.14159,  
 square: function (num) { // com o "function" (explícito)  
 return num \* num;  
 },  
 cube(num) { // sem o function (implícito)  
 return num \*\* 3;  
 }  
}  
  
myMath.square(2); // 4   
myMath.cube(2); // 8  
myMath["cube"](2); // 8

#### Exemplo 2.1: (this)

// "this" depende do contexto de quando é chamado.  
let dog = {  
 name: 'Leticia',  
 color: 'preta',  
 latido() {  
 console.log( "THIS IS:" , this)   
 console.log(`${this.name} fala au au`);  
 }  
};  
  
let latido2 = dog.latido;  
**dog.**latido();  
// THIS IS: "keys:values" do "dog"   
// Leticia fala au au // this tem o value do dog.name  
latido2();  
// THIS IS: "todos os metódos do js" // this não tem o value do dog.name  
// fala au au

#### Exemplo 2.2: (this)

let hen = {  
 name: "Helen",  
 eggCount: 0,  
 layAnEgg() {  
 this.eggCount += 1;  
 return "ovo adicionado com sucesso";  
 }  
};  
  
hen.name; // "Helen"  
hen.eggCount; // 0  
hen.layAnEgg(); // "ovo adicionado com sucesso"  
hen.layAnEgg(); // "ovo adicionado com sucesso"  
hen.eggCount; // 2

# tratamento de erros

## try ...catch

syntax: try { 'tentativa' }; catch { 'se erro faça' }

#### Exemplo 1:

function grito(texto) {  
 try {  
 console.log(texto.toUpperCase().repeat(1));  
 } catch (e) {  
 console.log("escreva um texto!")  
 }  
};  
  
grito(2); // escreva um texto!  
grito('nois capota mais num para'); // NOIS CAPOTA MAIS NUM PARA

# Temporizador

// 1000 ms == 1s

## setTimeout("function", "time ms")

// executado somente uma vez  
setTimeout(() => {console.log("...are you still there?")}, 3000)  
setTimeout(function(){console.log ("Hello"); }, 3000);  
// "clearTimeout()" finaliza a contagem do setTimeout

## setInterval()

setInterval(() => {console.log("...are you still there?")}, 3000);  
setInterval(function(){console.log ("Hello"); }, 3000);

### clearInterval()

// finaliza a contagem do setInterval  
function carregar() {  
 let contador = 0;  
 let intervalo = setInterval(frame, 100);  
 function frame() {  
 if (contador == 3) {  
 clearInterval(intervalo);  
 console.log( "tarefa concluída, caiu no clearInterval" );  
 } else {  
 contador++;  
 console.log(`${contador} "...continue, contador = + setInterval" `);  
 }  
 }  
};  
  
carregar(); /\* Output:  
1 "...continue, contador = + setInterval"  
2 "...continue, contador = + setInterval"   
3 "...continue, contador = + setInterval"   
tarefa concluída, caiu no clearInterval \*/

# exercícios

## números primos

function buscaNumeroPrimos(n) {

var numerosPrimos = [];

for (var numeroAtual = 2; numeroAtual < n; numeroAtual++) {

  var quantidadeDeDivisores = 0;

  for(var numeroMenor = 2; numeroMenor < numeroAtual; numeroMenor++) {

    if (numeroAtual % numeroMenor === 0) {

      quantidadeDeDivisores++;

    }

  }

if (quantidadeDeDivisores === 0) {

    numerosPrimos.push(numeroAtual);

  }

}

return numerosPrimos;

};