Curso Web Moderno Completo com JavaScript 2021 + Projetos

# Introdução Curso Desenvolvimento Web

## A Importância dos Fundamentos

Para desenvolver você precisa ter uma base sólida,

Para lutar você precisa ter uma base sólida.

# Configuração do Ambiente

* Usando o terminal, pode testar o node, usando: *node --version,* este comando informa a versão do Node.js
* Também podemos usar o comando: *code fonte.js*, neste caso, o comando abre o Visual Studio Code com este arquivo, podendo também abrir a pasta, passando o caminho da mesma como parâmetro.
* Alguns comandos do Terminal:
  + mkdir: cria uma pasta
  + cd: abre uma pasta
  + ls: mostra os arquivos dentro de uma pasta
  + &&: executa dois comandos na mesma linha
* F12: para abrir o console no Browser
* Plugin “Runner” no Visual Studio Code executa o programa em JavaScript, através do Node, usando Ctrl+Alt+N

# JavaScript: Fundamentos

## Comentários

// Comentário de uma linha

/\*Comentário do

múltiplas linhas

\*/

/\*

\* Comentário de

\* múltiplas linhas

\*

\*

\*

\*/

## Variáveis e Constantes

Declarando Variáveis:

var a = 3

let b = 4

console.log(a, b)

var a = 30 // é redeclarar usando ‘var’

b = 40 // redeclarar ‘b’ com ‘let’ gera um erro

console.log(a, b)

a = 300

b = 400

console.log(a, b)

Saída:

3 4

30 40

300 400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Declarando Constantes

const c = 5

// c = 50, não é possível mudar o valor de uma constante

console.log(c)

Saída:

5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Tipagem Fraca

Ser uma linguagem de tipagem fraca significa que os tipos primitivos não precisam ser declarado, o JavaScript se encarrega de manipular todas elas.

let qualquer = 'Legal'

console.log(qualquer)

console.log(typeof qualquer)

qualquer = 3.1516

console.log(qualquer)

console.log(typeof qualquer)

## Tipos em JavaScript

### Numbers

Podendo ser números reais em geral, não se diferencia de número inteiros do números de ponto fluante.

A função *Number*() recebe um valor qualquer como parâmetro, e o transforma para do tipo number se o mesmo representar um valor numérico

const peso1 = 1.0

const peso2 = *Number*('2.0')

//Number é uma função

Temos várias funções em JavaScript para manipular números:

//.isInteger devolve se é inteiro ou não (true ou false)

console.log(*Number*.isInteger(peso2))

//para fixar o número de casas decimais após a vírgula

console.log(media.toFixed(2))

//para mostrar o valor da variável como string em outra base númerica

(binário por exemplo)

console.log(media.toString(2))

Alguns cuidados com o tipo Numbers:

//O tipo "Infinity"

console.log(7/0)

Saída:

Infinity

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Exemplo de Tipagem Fraca:

console.log("10" / 2)

Saída:

5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Impossível Múltiplar Strings:

console.log("Show!" \* 2)

Saída:

NaN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Não há 100% de precisão

console.log(0.1 + 0.7)

Saída:

0.7999999999999999

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// console.log(10.toString()) << aqui temos um erro

console.log((10.345).toString())

Utilizando o “Math”

const raio = 5.6

const area = Math.PI \* Math.pow(raio, 2)

console.log(area)

console.log(typeof Math)

Saída:

98.5203456165759

object

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 

### Strings

Manipulando Strings:

const escola = "C0d3r"

//Obtendo uma caractere através da posição do índice

console.log(escola.charAt(4))

console.log(escola.charAt(5)) //aqui não da erro, e imprime um espaço em branco

//Obtendo o valor ASCII do caractere selecionado

console.log(escola.charCodeAt(3))

//busca a posição do caractere buscado

console.log(escola.indexOf('3'))

//retornando intervalos da string

console.log(escola.substring(1))

console.log(escola.substring(0, 3))

//Concatenação

console.log('Escola '.concat(escola).concat("!"))

console.log('Escola ' + escola + "!")

//Substituindo um caractere

console.log(escola.replace(3, 'e'))

//Separando Strings dentro de vírgulas (por exemplo) em um Array

console.log('Ana,Maria,Pedro'.split(','))

Saída:

r

51

3

0d3r

C0d

Escola C0d3r!

Escola C0d3r!

C0der

[ 'Ana', 'Maria', 'Pedro' ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Utilizando templates

const nome = 'Rebeca'

const concatenacao = 'Olá ' + nome + '!'

const template = `

Olá

${nome}!`

console.log(concatenacao, template)

//expressoes...

console.log(`1 + 1 = ${1 + 1}`)

//chamando um função dentro de um template

const up = *texto* => *texto*.toUpperCase() //texto é o parâmetro

console.log(`Ei... ${up('cuidado')}!`) //chamada da função

Saída:

Olá Rebeca!

Olá

Rebeca!

1 + 1 = 2

Ei... CUIDADO!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Boolean

:

//declarando e alterando variáveis boolean

let isAtivo = false

console.log(isAtivo)

isAtivo = true

console.log(isAtivo)

//Usando através de outros valores

isAtivo = 1

console.log(!!isAtivo) //negamos duas vezes para termos o resultado em boolean

console.log(*Boolean*(isAtivo)) //usando a função Boolean

Valores que retornam verdadeiro:

console.log('os verdadeiros...')

console.log(!!3)

console.log(!!-1)

console.log(!!' ')

console.log(!!'texto')

console.log(!![])

console.log(!!{})

console.log(!!Infinity)

console.log(!!(isAtivo = true))

Valores que retornam falso:

console.log('os falsos...')

console.log(!!0)

console.log(!!'')

console.log(!!null)

console.log(!!NaN)

console.log(!!undefined)

console.log(!!(isAtivo = false))

### 

### Array

//declarando um array

const value = [7.7, 8.9, 6.3, 9.2]

// imprime nas posições citadas

console.log(value[0], value[3])

// undefined para quando tentamos acessar uma posição que não existe

console.log(value[4])

// definindo um novo valor na posição citada

value[5] = 10

//imprime todo o array

console.log(value)

//mostra o tamanho, número de itens no array

console.log(value.length)

//função para adicionar nos valores

value.push({id:3}, false, null, 'teste')

console.log(value)

//função para remover o último valor, e retorna o valor

console.log(value.pop())

//removendo um item de algum item (deixa o espaço vazio no array)

delete value[0]

console.log(value)

//tipo do Array

console.log(typeof value)

Saída:

7.7 9.2

undefined

[ 7.7, 8.9, 6.3, 9.2, <1 empty item>, 10 ]

6

[

7.7,

8.9,

6.3,

9.2,

<1 empty item>,

10,

{ id: 3 },

false,

null,

'teste'

]

teste

[

<1 empty item>,

8.9,

6.3,

9.2,

<1 empty item>,

10,

{ id: 3 },

false,

null

]

object

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Object

Objeto em JavaScript é uma combinação de chave e valor (Dicionário em Python)

//forma literal de declarar um objeto

const prod1 = {}

prod1.nome = 'Celular Ultra Mega' //nome: chave, 'Celular Ultra Mega': valor

prod1.preco = 4998.90

prod1['Desconto legal'] = 0.40 //evitar atributos com espaço

console.log(prod1)

//outra forma de criar de literalmente

const prod2 = {

nome: 'Camisa Polo',

preco: 79.90,

obj: { //object em cadeia

blabla: 1,

obj: {

blabla: 2

}

}

}

console.log(prod2)

Saída:

{ nome: 'Celular Ultra Mega', preco: 4998.9, 'Desconto legal': 0.4 }

{

nome: 'Camisa Polo',

preco: 79.9,

obj: { blabla: 1, obj: { blabla: 2 } }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Objetos recebem seus dados por referência

// objetos recebem seus dados por referência

const a = {name: 'Teste'}

const b = a

console.log(a)

console.log(b)

//perceba que após a mudança de "b", o valor é alterado em "a" também

b.name = "Opa"

console.log(a)

console.log(b)

/\*

\* Isso acontece porque quando alteramos em "b",

\* estamos alterando dentro do valor de memória onde está armazenado

\* e como "a" aponta para o mesmo espaço de memória,

\* temos que a mudança acontece em "a" também

\*/

Saída:

{ name: 'Teste' }

{ name: 'Teste' }

{ name: 'Opa' }

{ name: 'Opa' }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Entendendo o Null & Undefined

let valor // não inicializada

console.log(valor) // aqui temos undefined

valor = null // null indica que não aponta para nenhum local de memória

console.log(valor)

Saída:

undefined

null

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Caso necessário definir uma variável sem um valor, prefira usar o null,

para que o programa use o undefined

## Funções

// Object em JavaScript é uma função

console.log(typeof *Object*)

// class também é uma funcão

class Produto {}

console.log(typeof Produto)

Saída:

function

function

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Função sem retorno

// função sem retorno

function imprimirSoma(*a*,*b*){

console.log(*a* + *b*)

}

imprimirSoma(2, 5)

imprimirSoma(2)

imprimirSoma(2, 3, 4, 5, 6, 7)

imprimirSoma()

Saída:

7

NaN

5

NaN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Função com retorno

//função com retorno

function soma(*a*, *b* = 1){

// b terá o valor 1 como padrão (caso não seja passado como parâmetro)

return *a* + *b*

}

//imprime o resultado

console.log(soma(2, 3))

//'b' não foi passado como parâmetro, b assume ser igual a 1

console.log(soma(2))

//caso 'a' não seja passado como parâmetro

console.log(soma())

Saída:

5

3

NaN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Arrow Function

// Armazenando uma função em uma variável

const imprimirSoma = function (*a*, *b*) {

console.log(*a* + *b*)

}

imprimirSoma(2, 3)

// Armazenando uma função arrow em uma variável

const soma = (*a*, *b*) => {

return *a* + *b*

}

console.log(soma(5, 3))

// Retorno implícito

const subtracao = (*a*, *b*) => *a* - *b*

console.log(subtracao(2, 3))

## Declarações de variáveis com var

// Declarando uma variável var dentro um escopo

{

{

{

{

var sera = 'Será???'

}

}

}

}

/\* uma variável var declarada dentro de um escopo

será visível fora do escopo \*/

console.log(sera)

function teste(){

// Declarando uma variável var dentro de uma função

var local = 123

// Uma variável var será visível dentro da própria função

console.log(local)

}

teste()

/\* Uma variável var declarada dentro de uma função

não poderá ser acessada fora da função \*/

console.log(local) //Erro

Saída:

Será???

123

ReferenceError: local is not defined

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Podemos ver que as variáveis var ignoram os blocos

var numero = 1

{

var numero = 2

console.log('dentro = ', numero)

}

console.log('fora =', numero)

Saída:

dentro = 2

fora = 2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Declarações de variáveis com let

//variáveis do tipo let respeitam o escopo de bloco

var numero = 1

{

let numero = 2

console.log('dentro =', numero)

}

console.log('fora =', numero)

Saída:

dentro = 2

fora = 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//variáveis let dentro do escopo terão preferência

let numero1 = 10

let numero2 = 20

{

let numero1 = 100

console.log('numero1 =', numero1)

console.log('numero2 =', numero2)

}

console.log('numero1 =', numero1)

console.log('numero2 =', numero2)

Saída:

numero1 = 100

numero2 = 20

numero1 = 10

numero2 = 20

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Usando Var ou Let em Loops

/\*Podemos ver a vantagem de usar var em um loop

onde a variável após o término do loop ainda é visível\*/

for (var i = 0; i < 10; i++) {

console.log(i)

}

console.log('i =', i)

Saída:

0

1

2

3

4

5

6

7

8

9

i = 10

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*A variável let não é visível após o término do loop\*/

for (let i = 0; i < 10; i++) {

console.log(i)

}

console.log('i =', i)

Saída:

0

1

2

3

4

5

6

7

8

9

ReferenceError: i is not defined

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Usamos var neste exemplo:

const funcs = []

for (var i = 0; i < 10; i++){

funcs.push(function(){

console.log(i)

})

}

/\*Já que var não respeita o escopo

teremos para todas as funções o último valor atribuido a variável i

funcs[2]()

funcs[8]()

Saída:

10

10

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Usamos let neste exemplo:

const funcs = []

for (var i = 0; i < 10; i++){

funcs.push(function(){

console.log(i)

})

}

/\*Já que let respeita o escopo

teremos para todas as funções cada valor atribuido a ela\*/

funcs[2]()

funcs[8]()

Saída:

2

8

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Hoisting

//É um efeito que faz o içamento do código

console.log('a =', a) // a = undefined porque a variável já existe

var a = 2

console.log('a =', a)

//Este efeito só acontece com var, usando let irá causar um erro

Saída:

a = undefined

a = 2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Função vs Objeto

console.log(typeof *Object*) // já sabemos de Object é um função

console.log(typeof **new** *Object*) // instanciando uma função

const Cliente = function() {} //Usando uma função criada

console.log(typeof Cliente)

console.log(typeof **new** Cliente)

class Produto {} // ES 2015 (ES6)

console.log(typeof Produto)

console.log(typeof **new** Produto())

Saída:

function

object

function

object

function

object

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Par Nome/Valor

// par nome/valor

const saudacao = 'Opa' //contexto léxico 1

function exec() {

const saudacao = 'Falaaa' // contexto léxico 2

return saudacao

}

//Objetos são grupos aninhados de pares nome/valor

const cliente = {

nome: 'Pedro',

idade: 32,

peso: 90,

endereco: {

logradouro: 'Rua Muito Legal',

numero: 123

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Notação Ponto

console.log(typeof Math) // Math é um objeto

console.log(typeof Math.ceil) // .ceil é uma função

console.log(Math.ceil(6.1)) //Usando uma função apartir de um objeto

const obj1 = {}

/\*guardando um valor a um atributo,

até mesmo criando um atributo que ainda não exista\*/

obj1.nome = 'Bola'

//obj1 ['nome'] = 'Bola2' << outra notação possível

console.log(obj1.nome)

function Obj(*nome*) { //criação de função para os exemplos abaixo

*this*.nome = *nome*

*this*.exec = function(){

console.log('Exec...')

}

}

const obj2 = **new** Obj('Cadeira') //Instanciando objetos

const obj3 = **new** Obj('Mesa')

console.log(obj2.nome) //chamando os atributos do objeto

console.log(obj3.nome)

obj3.exec() //chamando atributos/métodos/funções do objeto

Saída:

object

function

7

Bola

Cadeira

Mesa

Exec...

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Operadores

### Atribuição

const a = 7

let b = 3

b += a // acrescentando a variável 'a' a 'b', (b = b + a)

console.log(b)

b -= a // b = b - a

console.log(b)

b \*= 2 // b = b \* 2

console.log(b)

b /= 2 // b = b / 2

console.log(b)

b %= 2 // b = b % 2

console.log(b)

Saída:

10

3

6

3

1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Destructuring

// novo resurso do ES2015

//Usando o operador destruturing com objeto

const pessoa = {

nome: 'Ana',

idade: 5,

endereço: {

logradouro: 'Rua ABC',

numero: 1000

}

}

// operador destructuring

const { nome, idade} = pessoa

console.log(nome, idade)

//definindo outros nomes para as futuras variáveis

const { *nome*: n, *idade*: i} = pessoa

console.log(n, i)

// buscando no objeto atributos que não existem

const { sobrenome, bemHumorada = true} = pessoa

console.log(sobrenome, bemHumorada)

// acessando atributos dentro de outros

const { *endereço*: {logradouro, numero, cep}} = pessoa

console.log(logradouro, numero, cep)

Saída:

Ana 5

Ana 5

undefined true

Rua ABC 1000 undefined

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Usando o operador destruturing com array

const [a] = [10]

console.log(a)

const [n1, , n3, , n5, n6 = 0] = [10, 7, 9, 8]

console.log(n1, n3, n5, n6)

//cria a variável nota atribuindo o valor no array passado na posição definida

const [, [, nota]] = [[, 8, 8], [9, 6, 8]]

console.log(nota)

Saída:

10

10 9 undefined 0

6

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Destruturing em funções

function rand({ *min* = 0, *max* = 1000}) {

// define que será passado um objeto como parâmetro

const valor = Math.random() \* (*max* - *min*) + *min*

return Math.floor(valor)

}

const obj = { max: 50, min: 40 } // declarando um objeto

console.log(rand(obj)) // passando um objeto como parâmetro

console.log(rand({min: 955})) // max é padrão 1000

console.log(rand({})) // passando um objeto vazio

Saída:

46

973

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Destruturing em funções passando um array como parâmetro

function rand([*min* = 0, *max* = 1000]) { // min e max são atributos que foram desestructurizados

// inverte caso min seja maior que max

if (*min* > *max*) [*min*, *max*] = [*max*, *min*]

const valor = Math.random() \* (*max* - *min*) + *min*

return Math.floor(valor) //retorna o valor arredondado para baixo

}

//passando um array como parâmetro

console.log(rand([50, 40]))

//passando um array somente com o primeiro parâmetro, o segundo atribui o valor padrão

console.log(rand([992]))

//passando somente o segundo parâmetro

console.log(rand([, 10]))

// passando um array vazio

console.log(rand([]))

Saída:

47

996

5

796

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Aritméticos

//declarando as variáveis a, b, c, d com a desestructuruzação

const [a, b, c, d] = [3, 5, 1, 15]

const soma = a + b + c + d // adição

const subtracao = d - b // subtração

const multiplicacao = a \* b // multiplicação

const divisao = d / a // divisão

const modulo = a % 2 // resto da divisão

console.log(soma, subtracao, multiplicacao, divisao, modulo)

Saída:

24 10 15 5 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Relacionais

// TESTE RELACIONAIS

console.log('01)', '1' == 1) // == testa se os valores são iguais

console.log('02)', '1' === 1) // === testa se valores e tipos são iguas (estritamente)

console.log('03)', '3' != 3) // != testa se são valores diferentes

console.log('04)', '3' !== 3) // !== testa se valores e tipos são diferentes (estritamente)

// TESTE COMPARATIVOS

console.log('05)', 3 < 2)

console.log('06)', 3 > 2)

console.log('07)', 3 <= 2)

console.log('08)', 3 >= 2)

//COMPARANDO OBJETOS

const d1 = **new** *Date*(0) //Data de referência do JavaScript: 01/01/1970

const d2 = **new** *Date*(0)

console.log('09)', d1 === d2) // compara referências de memória

console.log('10)', d1 == d2)

console.log('11)', d1.getTime() === d2.getTime()) // compara o mesmo valor e tipo

//COMPARANDO UNDEFINED E NULL

console.log('12)', undefined == null)

console.log('13)', undefined === null)

Saída:

01) true

02) false

03) false

04) true

05) false

06) true

07) false

08) true

09) false

10) false

11) true

12) true

13) false

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Lógicos

/\*

\* AND: &&

\* OR: ||

\* XOR: ^

\* NOT: !

\*/

function compras(*trabalho1*, *trabalho2*) {

const comprarSorvete = *trabalho1* || *trabalho2*

const comprarTv50 = *trabalho1* && *trabalho2*

//const comprarTv32 = !!(trabalho1 ^ trabalho2) // bitwise XOR

const comprarTv32 = *trabalho1* != *trabalho2* //simula um XOR

const manterSaudavel = !comprarSorvete // operador unário

return { comprarSorvete, comprarTv50, comprarTv32, manterSaudavel } //retorna um objeto

}

console.log(compras(true, true))

console.log(compras(true, false))

console.log(compras(false, true))

console.log(compras(false, false))

Saída:

{

comprarSorvete: true,

comprarTv50: true,

comprarTv32: false,

manterSaudavel: false

}

{

comprarSorvete: true,

comprarTv50: false,

comprarTv32: true,

manterSaudavel: false

}

{

comprarSorvete: true,

comprarTv50: false,

comprarTv32: true,

manterSaudavel: false

}

{

comprarSorvete: false,

comprarTv50: false,

comprarTv32: false,

manterSaudavel: true

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Unários

let num1 = 1

let num2 = 2

num1++ //incrementa

console.log(num1)

--num1 //decrementa

console.log(num1)

console.log(++num1 === num2--) //EVITAR

console.log(num1 === num2)

Saída:

2

1

true

false

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Ternário

//nome\_da\_função => condição ? se\_verdadeiro : se\_falso

const resultado = *nota* => *nota* >= 7 ? 'APROVADO' : 'REPROVADO'

//Em conjunto com uma função Arrow

console.log(resultado(7.1))

console.log(resultado(6.7))

Saída:

APROVADO

REPROVADO

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Browser vs Node

### Browser

‘window’ é um grande objeto no browser, e também podemos chama-lo como ‘this’

this === window (quando fora de um escopo de outro objeto)

Variáveis ou funções criadas como const ou let não irão estar relacionadas ao objeto global window, ao contrário quando criadas com var

### Node

‘global’ é usado como o objeto geral no Node, semanhante o ‘window’ do browser

this === module.exports

Cada arquivo é um módulo diferente

let a = 3

global.b = 123

*this*.c = 456

*this*.d = false

*this*.e = 'teste'

console.log(a)

console.log(global.b)

console.log(*this*.c)

console.log(*module*.*exports*.c)

console.log(*module*.*exports* === *this*)

console.log(*module*.*exports*) //module.exports é um objeto

//Criando uma variável sem var e let!

abc = 3 // EVITAR

console.log(global.abc) //Criando sem let ou var ela irá para o escopo global

Saída:

3

123

456

456

true

{ c: 456, d: false, e: 'teste' }

3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Tratamento de Erro

function tratarErroELancar(*erro*) {

//throw new Error('...')

//throw 10

//throw true

throw { //Lançando o erro

nome: *erro*.name,

msg: *erro*.message,

date: **new** *Date*

}

}

function imprimirNomeGritado(*obj*) {

try{ //Tentativa do código

console.log(*obj*.name.toUpperCase() + '!!!')

} catch (e) { //Tratamento em caso de erro

tratarErroELancar(e)

} finally { //Finalização independente se houve erro ou não

console.log('final')

}

}

const obj = { nome: 'Roberto'}

imprimirNomeGritado(obj)

Saída:

final

c:\Users\J0N45\MyProjects\JavaScript-Udemy\exercicios-js\Fundamentos\tratamentoErros.js:5

throw { //Lançando o erro

^

{

nome: 'TypeError',

msg: "Cannot read property 'toUpperCase' of undefined",

date: 2021-07-15T13:06:16.932Z

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Estruturas de Controle

## Estrutura IF

function soBoaNoticia(*nota*) {

if(*nota* >= 7) {

console.log('Aprovado com ' + *nota*)

}

}

soBoaNoticia(8.1)

soBoaNoticia(6.1)

function seForVerdadeEuFalo(*valor*) {

if(*valor*) {

console.log('É verdade... ' + *valor*)

}

}

seForVerdadeEuFalo()

seForVerdadeEuFalo(null)

seForVerdadeEuFalo(undefined)

seForVerdadeEuFalo(NaN)

seForVerdadeEuFalo('')

seForVerdadeEuFalo(0)

seForVerdadeEuFalo(-1)

seForVerdadeEuFalo(' ')

seForVerdadeEuFalo('?')

seForVerdadeEuFalo([])

seForVerdadeEuFalo([1, 2])

seForVerdadeEuFalo({})

Saída:

Aprovado com 8.1

É verdade... -1

É verdade...

É verdade... ?

É verdade...

É verdade... 1,2

É verdade... [object Object]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function teste1(*num*) {

if(*num* > 7) //Caso não use {chaves}, somente a primeira linha entra na condional

console.log(*num*)

console.log('Final') //JavaScript não considera identação

}

teste1(6)

teste1(8)

function teste2(*num*) {

if (*num* > 7); { // ; é uma sentença de código vazia que neste caso finaliza a condicional

console.log(*num*)

}

}

teste2(6)

teste2(8)

Saída:

Final

8

Final

6

8

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Estrutura IF/ELSE

const imprimirResultado = function(*nota*) {

if (*nota* >= 7) {

console.log('APROVADO')

} else {

console.log('REPROVADO')

}

}

imprimirResultado(10)

imprimirResultado(4)

imprimirResultado('Epa!') //cuidado com a tipagem fracamente tipada

Saída:

APROVADO

REPROVADO

REPROVADO

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Estrutura IF/ELSE IF

*Number*.prototype.entre = function (*inicio*, *fim*) {

return *this* >= *inicio* && *this* <= *fim*

}

const imprimirResultado = function (*nota*) {

if (*nota*.entre(9, 10)) {

console.log('Quadro de Honra')

} else if (*nota*.entre(7, 8.99)) {

console.log('Aprovado')

} else if (*nota*.entre(4, 6.99)) {

console.log('Recuperação')

} else if (*nota*.entre(0, 3.99)) {

console.log('Reprovado')

} else {

console.log('Nota inválida')

}

console.log('fim')

}

imprimirResultado(10)

imprimirResultado(8.9)

imprimirResultado(6.5)

imprimirResultado(2.3)

imprimirResultado(-1)

imprimirResultado(11)

Saída:

Quadro de Honra

fim

Aprovado

fim

Recuperação

fim

Reprovado

fim

Nota inválida

fim

Nota inválida

fim

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Estrutura SWITCH

const imprimirResultado = function (*nota*) {

switch (Math.floor(*nota*)) { //swith (valor) {casos}

case 10:

case 9:

console.log('Quadro de Honra')

break //para sair não os próximos

case 8: case 7: //na mesma linha

console.log('Aprovado')

break

case 6: case 5: case 4:

console.log('Recuperação')

break

case 3: case 2: case 1: case 0:

console.log('Reprovado')

break

default: //caso não entre em nenhum

console.log('Nota inváliva')

}

}

imprimirResultado(10)

imprimirResultado(8.9)

imprimirResultado(6.5)

imprimirResultado(2.3)

imprimirResultado(-1)

imprimirResultado(11)

Saída:

Quadro de Honra

Aprovado

Recuperação

Reprovado

Nota inváliva

Nota inváliva

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Estruturas de Repetição While

function getInteiroAleatorioEntre(*min*, *max*) {

const valor = Math.random() \* (*max* - *min*) + *min*

return Math.floor(valor)

}

let opcao = 0

while (opcao != -1) {

opcao = getInteiroAleatorioEntre(-1, 10)

console.log(`Opção escolhida foi ${opcao}`)

}

console.log('Até a próxima')

Saída:

Opção escolhida foi 4

Opção escolhida foi 9

Opção escolhida foi 0

Opção escolhida foi 9

Opção escolhida foi 8

Opção escolhida foi 5

Opção escolhida foi 6

Opção escolhida foi 5

Opção escolhida foi 2

Opção escolhida foi 7

Opção escolhida foi 7

Opção escolhida foi 9

Opção escolhida foi 6

Opção escolhida foi 7

Opção escolhida foi -1

Até a próxima

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Estruturas de Repetição Do/While

function getInteiroAleatorioEntre(*min*, *max*) {

const valor = Math.random() \* (*max* - *min*) + *min*

return Math.floor(valor)

}

let opcao = 0

do { // primeiro executa, no final ele testa

opcao = getInteiroAleatorioEntre(-1, 10)

console.log(`Opção escolhida foi ${opcao}`)

} while (opcao != -1)

console.log('Até a próxima')

Saída:

Opção escolhida foi 0

Opção escolhida foi 9

Opção escolhida foi 1

Opção escolhida foi 1

Opção escolhida foi 4

Opção escolhida foi 0

Opção escolhida foi 1

Opção escolhida foi 0

Opção escolhida foi -1

Até a próxima

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Estruturas de Repetição For

for (let i = 1; i <= 10; i++) {

console.log(`i = ${i}`)

}

const notas = [6.7, 7.4, 9.8, 8.1, 7.7]

for (let i = 0; i < notas.length; i++) {

console.log(`nota = ${notas[i]}`)

}

Saída:

i = 1

i = 2

i = 3

i = 4

i = 5

i = 6

i = 7

i = 8

i = 9

i = 10

nota = 6.7

nota = 7.4

nota = 9.8

nota = 8.1

nota = 7.7

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Estruturas de Repetição For/In

const notas = [6.7, 7.4, 9.8, 8.1, 7.7]

for (i in notas) { //percorrendo um array

console.log(i, notas[i])

}

const pessoa = {

nome: 'Ana',

sobrenome: 'Silva',

idade: 29,

peso: 64

}

for (let atributo in pessoa) { // percorrendo um objeto

console.log(`${atributo} = ${pessoa[atributo]}`)

}

Saída:

0 6.7

1 7.4

2 9.8

3 8.1

4 7.7

nome = Ana

sobrenome = Silva

idade = 29

peso = 64

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Break e Continue

const nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for (x in nums) {

if (x == 5) break // ele para a repetição for (ou while)

console.log(`${x} = ${nums[x]}`)

}

for (y in nums) {

if (y == 5) {

continue //interrope o restante do laço, e continua pro próximo ciclo

}

console.log(`${y} = ${nums[y]}`)

}

externo:

for (a in nums) {

for (b in nums) {

if (a == 2 && b == 3) break externo //EVITAR

console.log(`Par = ${a},${b}`)

}

}

console.log('Fim!')

Saída:

0 = 1

1 = 2

2 = 3

3 = 4

4 = 5

0 = 1

1 = 2

2 = 3

3 = 4

4 = 5

6 = 7

7 = 8

8 = 9

9 = 10

Par = 0,0

Par = 0,1

Par = 0,2

Par = 0,3

Par = 0,4

Par = 0,5

Par = 0,6

Par = 0,7

Par = 0,8

Par = 0,9

Par = 1,0

Par = 1,1

Par = 1,2

Par = 1,3

Par = 1,4

Par = 1,5

Par = 1,6

Par = 1,7

Par = 1,8

Par = 1,9

Par = 2,0

Par = 2,1

Par = 2,2

Fim!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Função

// Função em JS é First-Class Object (Citizens)

// Higher-order function

// criar de forma literal

function fun1() { }

// Armazenar em uma variável

const fun2 = function () { }

// Armazenar em um array

const array = [function (*a*, *b*) { return *a* + *b* }, fun1, fun2]

console.log(array[0](2,3))

// Armazenar em um atributo de objeto

const obj = {}

obj.falar = function() {return 'Opa'}

console.log(obj.falar())

// Passar função como parâmetro

function run(*fun*) {

fun()

}

run(function() {console.log('Executando...')})

// Uma função pode retornar/conter uma função

function soma(*a*, *b*){

return function (*c*){

console.log(*a* + *b* + *c*)

}

}

soma(2, 3)(4)

Saída:

5

Opa

Executando...

9

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Parâmetro Variáveis

function soma() {

let soma = 0

for (i in *arguments*) { // arguments é um array dos parâmetro passados na chamada

soma += *arguments*[i]

}

return soma

}

console.log(soma())

console.log(soma(1))

console.log(soma(1.1, 2.2, 3.3))

Saída:

0

1

6.6

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## “this” pode variar

this se refere ao objeto do espoço atual, portanto ele pode variar

Em caso de funções arrow, o this nunca muda, ele sempre será determinado na definição da variável

## “this” e a função Bind

const pessoa = {

saudacao: 'Bom dia!',

falar() {

console.log(*this*.saudacao)

}

}

pessoa.falar()

const falar = pessoa.falar

falar() // o this desse método aponta para outro lugar

// conflito entre paradigmas: funcional e orientação a objeto

const falarDePessoa = pessoa.falar.bind(pessoa) // força o this para o objeto que se deseja

falarDePessoa()

Saída:

Bom dia!

undefined

Bom dia!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function Pessoa() {

*this*.idade = 0

setInterval(function(){

*this*.idade++

console.log(*this*.idade)

}.bind(*this*), 1000)

}

**new** Pessoa

Saída:

1

2

3

4

5

...

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Função Arrow

let dobro = function (*a*) {

return 2 \* *a*

}

// Função Arrow

dobro = (*a*) => {

return 2 \* *a*

}

// Função Arrow reduzida

dobro = *a* => 2 \* *a*

console.log(dobro(Math.PI))

let ola = function() {

return 'Olá'

}

ola = () => 'Olá'

ola = *\_* => 'Olá' //Outra forma

console.log(ola())

Saída:

6.283185307179586

Olá

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function Pessoa() {

*this*.idade = 0

setInterval(() => {

*this*.idade++

console.log(*this*.idade)

}, 1000)

}

**new** Pessoa

Saída:

1

2

3

4

...

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

let comparaComThis = function (*param*) {

console.log(*this* == *param*)

}

comparaComThis(global)

const obj = {}

comparaComThis = comparaComThis.bind(obj) //Muda para qual objeto a função aponta

comparaComThis(global)

comparaComThis(obj)

let comparaComThisArrow = *param* => console.log(*this* === *param*)

comparaComThisArrow(global) // Uma função arrow não aponta para o objeto global

comparaComThisArrow(*module*.*exports*)

comparaComThisArrow = comparaComThisArrow.bind(obj)

comparaComThisArrow(obj) // as Arrow Functions são mais fortes, não altera com o bind

comparaComThisArrow(*module*.*exports*)

// O This em uma função Arrow está associado ao contexto que a função arrow foi escrita.

Saída:

true

false

true

false

true

false

true

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Funções anônimas

// Função Anônima é uma função sem nome

const soma = function (*x*, *y*) {

return *x* + *y*

}

const imprimirResultado = function (*a*, *b*, *operacao* = soma) {

// O parâmetro operacao por padrão chama a função soma

console.log(operacao(*a*, *b*))

}

imprimirResultado(3, 4)

imprimirResultado(3, 4, soma)

imprimirResultado(3, 4, function (*x*, *y* ) {

return *x* - *y*

}) // podemos passar uma função anônima em uma chama de função

imprimirResultado(3, 4, (*x*, *y*) => *x* \* *y*) // passando uma função arrow

// Função Anônima dentro de um objeto

const pessoa = {

falar: function () {

console.log('Opa')

}

}

pessoa.falar()

Saída:

7

7

-1

12

Opa

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Funções CallBack

const fabricantes = ["Mercedes", "Audi", "BMW"]

function imprimir(*nome*, *indice*) {

console.log(`${*indice* + 1}. ${*nome*}`)

}

fabricantes.forEach(imprimir)

// para cada elemento do array, e chama a função imprimir, passando como parâmetro valor e indice

fabricantes.forEach(*fabricante* => console.log(*fabricante*))

Saída:

1. Mercedes

2. Audi

3. BMW

Mercedes

Audi

BMW

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

const notas = [7.7, 6.5, 5.2, 8.9, 3.6, 7.1, 9.0]

// Sem Callback

let notasBaixas1 = []

for (let i in notas) {

if (notas[i] < 7) {

notasBaixas1.push(notas[i])

}

}

console.log(notasBaixas1)

// Com callback

notasBaixas2 = notas.filter(function(*nota*) {

return *nota* < 7

}) // .filter() Não altera o array que foir feita a leitura (notas)

console.log(notasBaixas2)

// Usando função arrow e callbakc

const notasBaixas3 = notas.filter(*nota* => *nota* < 7)

console.log(notasBaixas3)

// Criando a função antes, e reutilizando

const notasMenorQue7 = *nota* => *nota* < 7

const notasBaixas4 = notas.filter(notasMenorQue7)

console.log(notasBaixas4)

Saída:

[ 6.5, 5.2, 3.6 ]

[ 6.5, 5.2, 3.6 ]

[ 6.5, 5.2, 3.6 ]

[ 6.5, 5.2, 3.6 ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Funções Construtoras

function Carro(*velocidadeMaxima* = 200, *delta* = 5) {

// atributo privado

let velocidadeAtual = 0

// metodo publico

*this*.acelerar = function () {

if (velocidadeAtual + *delta* <= *velocidadeMaxima*) {

velocidadeAtual += *delta*

} else {

velocidadeAtual = *velocidadeMaxima*

}

}

// metodo publico

*this*.getVelocidadeAtual = function () { // this. torna algo publico

return velocidadeAtual

}

}

const uno = **new** Carro // instanciando um obejto sem parâmetros

uno.acelerar()

console.log(uno.getVelocidadeAtual())

const ferrari = **new** Carro(350, 20) // instanciando outro objeto com parâmetros

ferrari.acelerar()

ferrari.acelerar()

ferrari.acelerar()

console.log(ferrari.getVelocidadeAtual())

console.log(typeof Carro)

console.log(typeof ferrari)

Saída:

5

60

function

object

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Tipos de declaração

// function declaration

console.log(soma(3, 4)) // chamando ela antes de ser declarada

function soma(*x*, *y*) {

return *x* + *y*

}

// function expression

const sub = function (*x*, *y*) {

return *x* - *y*

}

console.log(sub(3, 4))

// named function expression

const mult = function mult(*x*, *y*) {

return *x* \* *y*

}

console.log(mult(3, 4))

Saída:

7

-1

12

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Contexto Léxico

const valor = 'Global'

function minhaFuncao() { // A função carrega consigo o contexto que ela foi escrita

console.log(valor)

}

function exec() {

const valor = 'local'

minhaFuncao()

}

exec()

Saída:

Global

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Closures

// Closure é o escopo criado quando uma função é declarada

// Esse escopo permite a função acessar e manipular variáveis externas à função

// Contexto léxico em ação!

const x = 'Global'

function fora() {

const x = 'Local'

function dentro() { // Aqui ela leva o contexto (escopo) junto na declaração

return x

}

return dentro

}

const minhaFuncao = fora()

console.log(minhaFuncao())

Saída:

Local

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Função Factory

// Função que retorna um objeto

function criarPessoa() {

return {

nome: 'Ana',

sobrenome: 'Silva'

}

}

console.log(criarPessoa())

Saída:

{ nome: 'Ana', sobrenome: 'Silva' }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function criarProduto(*nome*, *preco*) {

return {

nome,

preco

}

}

console.log(criarProduto('Carro', 14999))

Saída:

{ nome: 'Carro', preco: 14999 }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Classe

class Pessoa {

constructor(*nome*) {

*this*.nome = *nome*

}

falar(){

console.log(`Meu nome é ${*this*.nome}`)

}

}

const p1 = **new** Pessoa('João')

p1.falar()

Saída:

Meu nome é João

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## IIFE

// IIFE -> Immediately Invoked Function Expresion

(function() {

console.log('Será executando na hora!')

console.log('Foge do escopo mais abrangente!')

})()

Saída:

Será executando na hora!

Foge do escopo mais abrangente!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Call & Apply

function getPreco(*imposto* = 0, *moeda* = 'R$') {

return `${*moeda*} ${*this*.preco \* (1 - *this*.desc) \* (1 + *imposto*)}`

}

const produto = {

nome: 'Notebook',

preco: 4589,

desc: 0.15,

getPreco

}

globalThis.preco = 20

globalThis.desc = 0.1

console.log(getPreco())

console.log(produto.getPreco())

const carro = { preco: 49990, desc: 0.2 }

console.log(getPreco.call(carro)) //passa o contexto para a função

console.log(getPreco.apply(carro)) //passa o contexto para a função

console.log(getPreco.call(carro, 0.17, '$'))

// passando o contexto com parâmetros

console.log(getPreco.apply(carro, [0.17, '$']))

// passando o contexto com parâmetros (dentro de um array)

Saída:

R$ 18

R$ 3900.65

R$ 39992

R$ 39992

$ 46790.64

$ 46790.64

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Objetos

// coleção dinâmica de pares chave/valor

const produto = **new** *Object*

produto.nome = 'Caldeira'

produto['marca do produto'] = 'Generica'

produto.preco = 220

console.log(produto)

delete produto.preco

delete produto['marca do produto']

console.log(produto)

const carro = {

modelo: 'A4',

valor: 89000,

proprietario: {

nome: 'Raul',

idade: 56,

endereco: {

logradouro: 'Rua ABC',

numero: 123

}

},

condutores: [{

nome: 'Junior',

idade: 19

}, {

nome: 'Ana',

idade: 42

}],

calcularValorSeguro: function() {

// ...

}

}

carro.proprietario.endereco.numero = 1000

carro['proprietario']['endereco']['logradouro']

console.log(carro)

delete carro.condutores

delete carro.proprietario.endereco

delete carro.calcularValorSeguro

console.log(carro)

console.log(carro.condutores)

Saída:

{ nome: 'Caldeira', 'marca do produto': 'Generica', preco: 220 }

{ nome: 'Caldeira' }

{

modelo: 'A4',

valor: 89000,

proprietario: {

nome: 'Raul',

idade: 56,

endereco: { logradouro: 'Rua ABC', numero: 1000 }

},

condutores: [ { nome: 'Junior', idade: 19 }, { nome: 'Ana', idade: 42 } ],

calcularValorSeguro: [Function: calcularValorSeguro]

}

{

modelo: 'A4',

valor: 89000,

proprietario: { nome: 'Raul', idade: 56 }

}

undefined

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Criação de Objetos

// usando a notação literal

const obj1 = {}

console.log(obj1)

// Object em JS

console.log(typeof *Object*, typeof **new** *Object*)

const obj2 = **new** *Object*

console.log(obj2)

// Funções Construtoras

function Produto (*nome*, *preco*, *desc*) {

*this*.nome = *nome*

*this*.getPrecoComDesconto = () => {

return *preco* \* (1 - *desc*)

}

}

const p1 = **new** Produto('Caneta', 7.99, 0.15)

const p2 = **new** Produto('Notebook', 2998.99, 0.25)

console.log(p1.getPrecoComDesconto(), p2.getPrecoComDesconto())

// Função Factory

function criarFuncionario(*nome*, *salarioBase*, *faltas*) {

return {

nome,

salarioBase,

faltas,

getSalario() {

return (*salarioBase* / 30) \* (30 - *faltas*)

}

}

}

const f1 = criarFuncionario('João', 7980, 4)

const f2 = criarFuncionario('Maria', 11400, 1)

console.log(f1.getSalario(), f2.getSalario())

// Object.create

const filha = *Object*.create(null)

filha.nome = 'Ana'

console.log(filha)

// Uma funcão famosa que retorna Object

const fromJSON = JSON.parse('{"info": "Sou um JSON"}')

Saída:

{}

function object

{}

6.7915 2249.2425

6916 11020

[Object: null prototype] { nome: 'Ana' }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Objetos Constantes

// pessoa -> 123 -> {...}

const pessoa = { nome: 'Joao'}

pessoa.nome = 'Pedro'

console.log(pessoa)

//pessoa = { nome: 'Ana'} --pessoa é uma constante, é uma tentativa de escrever em outro endereço, causa Erro

*Object*.freeze(pessoa) // "congela o objeto", impossível alterar

pessoa.nome = 'Maria' //impossível alterrar o atributo

pessoa.end = 'Rua ABC' //impossível adicionar atributos novos

delete pessoa.nome // impossível excluir o atributo

console.log(pessoa.nome) // o atributo continua sendo o mesmo

console.log(pessoa) //objeto continua sendo o mesmo

// criando um objeto já sendo uma constante

const pessoaConstante = *Object*.freeze({ nome: 'Joao'})

pessoaConstante.nome = 'Maria'

console.log(pessoaConstante)

Saída:

{ nome: 'Pedro' }

Pedro

{ nome: 'Pedro' }

{ nome: 'Joao' }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Notação Literal

const a = 1

const b = 2

const c = 3

const obj1 = { a: a, b: b, c: c } // caso se queira trocar o nome do atributo

const obj2 = { a, b, c } //novo formato, EC2015

console.log(obj1, obj2)

const nomeAttr = 'nota'

const valorAttr = 7.87

const obj3 = {}

obj3[nomeAttr] = valorAttr

console.log(obj3)

const obj4 = {[nomeAttr]: valorAttr}

console.log(obj4)

const obj5 = {

funcao1: function() { //forma tradicional

//...

},

funcao2() { //forma reduzida EC2015

//...

}

}

console.log(obj5)

Saída:

{ a: 1, b: 2, c: 3 } { a: 1, b: 2, c: 3 }

{ nota: 7.87 }

{ nota: 7.87 }

{ funcao1: [Function: funcao1], funcao2: [Function: funcao2] }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Getters e Setters

// Usando getters e setters, é possível acessarmos os atributos usando em conjunto algum tipo de validação ou processamento

const sequencia = {

\_valor: 1, // convenção, não significa ser privado

get valor() { return *this*.\_valor++ },

set valor(*valor*) {

if(*valor* > *this*.\_valor) {

*this*.\_valor = *valor*

}

}

}

console.log(sequencia.valor, sequencia.valor)

sequencia.valor = 1000

console.log(sequencia.valor, sequencia.valor)

sequencia.valor = 900 // o atributo não é modificado porque não foi validaddo

console.log(sequencia.valor, sequencia.valor)

Saída:

1 2

1000 1001

1002 1003

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Funções Importantes

const pessoa = {

nome: 'Rebeca',

idade: 2,

peso: 13

}

console.log(*Object*.keys(pessoa)) // retorna os nomes das chaves

console.log(*Object*.values(pessoa)) // retorna os valores

console.log(*Object*.entries(pessoa)) // retorna um array do objeto, tendo nome e valor

*Object*.entries(pessoa).forEach(*e* => { // percorrendo o array criado com entries

console.log(`${*e*[0]}: ${*e*[1]}`)

})

//------------------------------------------------

*Object*.defineProperty(pessoa, 'dataNascimento', {

enumerable: true,

writable: false,

value: '01/01/2019'

})

pessoa.dataNascimento = '01/01/2017'

console.log(pessoa.dataNascimento) // podemos ver que a data não foi reescrita

// podemos que o atributo 'dataNascimento' aparece "enumerado"

console.log(*Object*.keys(pessoa)) // se o "enumerable" for false, ele não irá aparecer

//------------------------------------------------

//Object.assign (ECMAScript 2015)

const dest = {a: 1}

const o1 = {b: 2}

const o2 = {c: 3, a: 4}

const obj = *Object*.assign(dest, o1, o2) // os atributos são concatenados no primeiro objeto passado como parâmetro (dest)

console.log(obj) // os atributos são sobreescrito caso tenham os mesmo nomes.

*Object*.freeze(obj) // "congela" o objeto

obj.c = 1234

console.log(obj)

Saída:

[ 'nome', 'idade', 'peso' ]

[ 'Rebeca', 2, 13 ]

[ [ 'nome', 'Rebeca' ], [ 'idade', 2 ], [ 'peso', 13 ] ]

nome: Rebeca

idade: 2

peso: 13

01/01/2019

[ 'nome', 'idade', 'peso', 'dataNascimento' ]

{ a: 4, b: 2, c: 3 }

{ a: 4, b: 2, c: 3 }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Herança

const ferrari = {

modelo: 'F40',

velMax: 324

}

const volvo = {

modelo: 'V40',

velMax: 200

}

console.log(ferrari.\_\_proto\_\_) // objeto existe

console.log(ferrari.\_\_proto\_\_ === *Object*.prototype)

console.log(volvo.\_\_proto\_\_ === *Object*.prototype)

console.log(*Object*.prototype.\_\_proto\_\_) // aponta para null

function MeuObjeto() {}

console.log(typeof *Object*, typeof MeuObjeto)

console.log(*Object*.prototype, *MeuObjeto*.prototype)

Saída:

[Object: null prototype] {}

true

true

null

function function

[Object: null prototype] {} {}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Cadeia de protótipos (prototype chain)

*Object*.prototype.attr0 = '0' // evitar

const avo = { attr1: 'A' }

const pai = { \_\_proto\_\_: avo, attr2: 'B', attr: '3' }

const filho = { \_\_proto\_\_: pai, attr3: 'C' }

console.log(filho.attr0, filho.attr1, filho.attr2, filho.attr3)

const carro = {

velAtual: 0,

velMax: 200,

aceleraMais(*delta*) {

if (*this*.velAtual + *delta* <= *this*.velMax) {

*this*.velAtual += *delta*

} else {

*this*.velAtual = *this*.velMax

}

},

status() {

return `${*this*.velAtual}Km/h de ${*this*.velMax}Km/h`

}

}

const ferrari = {

modelo: 'F40',

velMax: 324 // shadowing

}

const volvo = {

modelo: 'V40',

status() {

return `${*this*.modelo}: ${*super*.status()}`

}

}

*Object*.setPrototypeOf(ferrari, carro) // estabele uma relação de protótipo

*Object*.setPrototypeOf(volvo, carro)

console.log(ferrari) // isso só mostra os atributos locais

console.log(volvo)

volvo.aceleraMais(100)

console.log(volvo.status())

ferrari.aceleraMais(300)

console.log(ferrari.status())

Saída:

0 A B C

{ modelo: 'F40', velMax: 324 }

{ modelo: 'V40', status: [Function: status] }

V40: 100Km/h de 200Km/h

300Km/h de 324Km/h

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

const pai = { nome: 'Pedro', corCabelo: 'preto'}

const filha1 = *Object*.create(pai) // cria um objeto tendo como protótipo o objecto passado com parâmetro

filha1.nome = 'Ana'

console.log(filha1.corCabelo)

const filha2 = *Object*.create(pai, {

nome: {value: 'Bia', writable: false, enumerable: true}

})

console.log(filha2.nome)

filha2.nome = 'Carla'

console.log(`${filha2.nome} tem cabelo ${filha2.corCabelo}`)

console.log(*Object*.keys(filha1))

console.log(*Object*.keys(filha2))

for(let key in filha2) {

filha2.hasOwnProperty(key) ? // para saber se um atributo é do próprio objeto

console.log(key) : console.log(`Por herança: ${key}`)

}

Saída:

preto

Bia

Bia tem cabelo preto

[ 'nome' ]

[ 'nome' ]

nome

Por herança: corCabelo

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function MeuObjeto() {

console.log(MeuObjeto.prototype)

}

const obj1 = **new** MeuObjeto

const obj2 = **new** MeuObjeto

console.log(obj1.\_\_proto\_\_ === obj2.\_\_proto\_\_)

console.log(MeuObjeto.prototype === obj1.\_\_proto\_\_)

MeuObjeto.prototype.nome = 'Anônimo'

MeuObjeto.prototype.falar = function() {

console.log(`Bom dia! Meu nome é ${*this*.nome}!`)

}

obj1.falar()

obj2.nome = 'Rafael'

obj2.falar()

const obj3 = {}

obj3.\_\_proto\_\_ = MeuObjeto.prototype

obj3.nome = 'Obj3'

obj3.falar()

// Resumindo a loucura...

console.log((**new** MeuObjeto).\_\_proto\_\_ === MeuObjeto.prototype)

console.log(MeuObjeto.\_\_proto\_\_ === *Function*.prototype)

console.log(*Function*.prototype.\_\_proto\_\_ === *Object*.prototype)

console.log(*Object*.prototype.\_\_proto\_\_ === null)

Saída:

{}

{}

true

true

Bom dia! Meu nome é Anônimo!

Bom dia! Meu nome é Rafael!

Bom dia! Meu nome é Obj3!

{ nome: 'Anônimo', falar: [Function (anonymous)] }

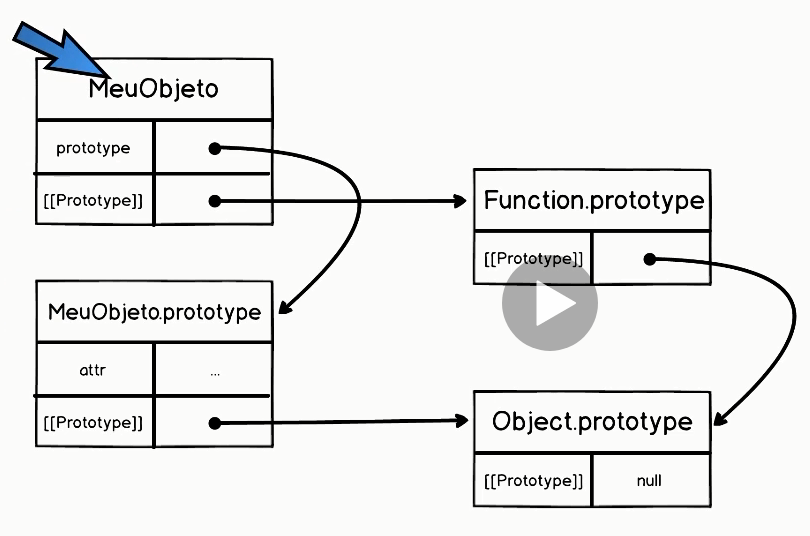
true

true

true

true

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



//são funções:

console.log(typeof *String*)

console.log(typeof *Array*)

console.log(typeof *Object*)

//toda função possuem .prototype

*String*.prototype.reverse = function () {

return *this*.split('').reverse().join('')

}

console.log('Escola Cod3r'.reverse())

*Array*.prototype.first = function () {

return *this*[0]

}

console.log([1, 2, 3, 4, 5].first())

console.log(['a', 'b', 'c'].first())

*String*.prototype.toString = function() { //alterando funções já existentes, evitar

return 'Lascou Tudo'

}

console.log('Escola Cod3r'.reverse()) //cuidado! resultado não esperados.

Saída:

function

function

function

r3doC alocsE

1

a

oduT uocsaL

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function Aula(*nome*, *videoID*) {

*this*.nome = *nome*

*this*.videoID = *videoID*

}

const aula1 = **new** Aula('Bem Vindo', 123)

const aula2 = **new** Aula('Até Breve', 456)

console.log(aula1, aula2)

//simulando o new

function novo(*f*, ...*params*) {

const obj = {}

obj.\_\_proto\_\_ = *f*.prototype

*f*.apply(obj, *params*)

return obj

}

const aula3 = novo(Aula, 'Bem Vindo', 123)

const aula4 = novo(Aula, 'Até Breve', 456)

console.log(aula3, aula4)

Saída:

Aula { nome: 'Bem Vindo', videoID: 123 } Aula { nome: 'Até Breve', videoID: 456 }

Aula { nome: 'Bem Vindo', videoID: 123 } Aula { nome: 'Até Breve', videoID: 456 }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Evitando Modificações

// Object.preventExtensions

const produto = *Object*.preventExtensions({ // impede de adcionar atributos

nome: 'Qualquer', preco: 1.99, tag: 'promoção'

})

console.log('Extensível:', *Object*.isExtensible(produto))

console.log(produto)

produto.nome = 'Borracha' // conseguimos alterar um valor de um atributo

console.log(produto)

produto.descricao = 'Borracha escolar branca' // Não podemos adicionar novos atributos

console.log(produto)

delete produto.tag // Podemos excluir atributos

console.log(produto)

// Object.seal (selar)

const pessoa = { nome: 'Juliana', idade: 35 }

*Object*.seal(pessoa) // Não podemos adicionar atributos novos, nem excluir atributos existentes

console.log('Selado:', *Object*.isSealed(pessoa))

pessoa.sobrenome = 'Silva' // Não podemos adicionar atributos novos

delete pessoa.nome // Não podemos excluir atributos existentes

pessoa.idade = 29 // Podemos alterar atributos já existentes

console.log(pessoa)

Saída:

Extensível: false

{ nome: 'Qualquer', preco: 1.99, tag: 'promoção' }

{ nome: 'Borracha', preco: 1.99, tag: 'promoção' }

{ nome: 'Borracha', preco: 1.99, tag: 'promoção' }

{ nome: 'Borracha', preco: 1.99 }

Selado: true

{ nome: 'Juliana', idade: 29 }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## JSON vc Objeto

Json é um formato textual de objeto

const obj = { a: 1, b: 2, c: 3, soma() { return a + b + c } }

console.log(JSON.stringify(obj)) // converte em formato textual, a função é excluida

// console.log(JSON.parse("{ a: 1, b: 2, c: 3 }"))

//console.log(JSON.parse("{ 'a': 1, 'b': 2, 'c': 3 }"))

console.log(JSON.parse('{ "a": 1, "b": 2, "c": 3 }')) // convertendo para objeto através de um JSON

// atributos dentro de "aspas duplas"

console.log(JSON.parse('{ "a": 1.7, "b": "string", "c": true, "d": {}, "e": [] }'))

Saída:

{"a":1,"b":2,"c":3}

{ a: 1, b: 2, c: 3 }

{ a: 1.7, b: 'string', c: true, d: {}, e: [] }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Classe

class Lancamento {

constructor(*nome* = 'Genérico', *valor* = 0) {

*this*.nome = *nome*

*this*.valor = *valor*

}

}

class CicloFinanceiro {

constructor(*mes*, *ano*) {

*this*.mes = *mes*

*this*.ano = *ano*

*this*.lancamentos = []

}

addLancamentos(...*lancamentos*) {

*lancamentos*.forEach(*l* => *this*.lancamentos.push(*l*))

}

sumario() {

let valorConsolidado = 0

*this*.lancamentos.forEach(*l* => {

valorConsolidado += *l*.valor

})

return valorConsolidado

}

}

const salario = **new** Lancamento('Salario', 45000)

const contaDeLuz = **new** Lancamento('Luz', -220)

const contas = **new** CicloFinanceiro(6, 2018)

contas.addLancamentos(salario, contaDeLuz)

console.log(contas.sumario())

Saída:

44780

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class Avo {

constructor(*sobrenome*) {

*this*.sobrenome = *sobrenome*

}

}

class Pai extends Avo { // usando herança em classes

constructor(*sobrenome*, *profissao* = 'Professor') {

*super*(*sobrenome*)

*this*.profissao = *profissao*

}

}

class Filho extends Pai {

constructor() {

*super*('Silva')

}

}

const filho = **new** Filho

console.log(filho)

Saída:

Filho { sobrenome: 'Silva', profissao: 'Professor' }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Array

// um array é um objeto

console.log(typeof *Array*, typeof **new** *Array*, typeof [])

let aprovados = **new** *Array*('Bia', 'Carlos', 'Ana')

console.log(aprovados)

aprovados = ['Bia', 'Carlos', 'Ana']

console.log(aprovados[0])

console.log(aprovados[1])

console.log(aprovados[2])

console.log(aprovados[3]) //undefined

aprovados[3] = 'Paulo'

aprovados.push('Abia') // adiciona no array

console.log(aprovados.length) // .length retorna o tamanho

aprovados[9] = 'Rafael'

console.log(aprovados.lenght)

console.log(aprovados[8] === undefined)

console.log(aprovados)

aprovados.sort() // ordena o array

console.log(aprovados) // o elementos vazios vão para o final

delete aprovados[1] // o elemento fica undefined, não reordena o array

console.log(aprovados[1])

console.log(aprovados[2])

aprovados = ['Bia', 'Carlos', 'Ana']

aprovados.splice(1, 1, 'Elemento1', 'Elemento2') // exclui elementos apartir do índice 1, 1 elemento apartir desta posição. o Elementos são adicionados na posição

console.log(aprovados)

Saída:

function object object

[ 'Bia', 'Carlos', 'Ana' ]

Bia

Carlos

Ana

undefined

5

undefined

true

[ 'Bia', 'Carlos', 'Ana', 'Paulo', 'Abia', <4 empty items>, 'Rafael' ]

[ 'Abia', 'Ana', 'Bia', 'Carlos', 'Paulo', 'Rafael', <4 empty items> ]

undefined

Bia

[ 'Bia', 'Elemento1', 'Elemento2', 'Ana' ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

const pilotos = ['Vettel', 'Alonson', 'Raikkonen', 'Massa']

pilotos.pop() // remove da última posição

console.log(pilotos)

pilotos.push('Verstappen') // adiciona na última posição

console.log(pilotos)

pilotos.shift() // remove o primeiro

console.log(pilotos)

pilotos.unshift('Hamilton') // adiciona na primeira posição

console.log(pilotos)

// splice pode adicionar e remover elementos

// nome\_do\_array.splice(posição, n\_elementos a ser excluido, novos elementos...)

// adicionar

pilotos.splice(2, 0, 'Bottas', 'Massa')

console.log(pilotos)

// remover

pilotos.splice(3, 1)

console.log(pilotos)

const algunsPilotos1 = pilotos.slice(2) // novo array, apartir do segundo elemento

console.log(algunsPilotos1)

const algunsPilotos2 = pilotos.slice(1, 4) // novo array, entre os índices passados

console.log(algunsPilotos2)

Saída:

[ 'Vettel', 'Alonson', 'Raikkonen' ]

[ 'Vettel', 'Alonson', 'Raikkonen', 'Verstappen' ]

[ 'Alonson', 'Raikkonen', 'Verstappen' ]

[ 'Hamilton', 'Alonson', 'Raikkonen', 'Verstappen' ]

[ 'Hamilton', 'Alonson', 'Bottas', 'Massa', 'Raikkonen', 'Verstappen' ]

[ 'Hamilton', 'Alonson', 'Bottas', 'Raikkonen', 'Verstappen' ]

[ 'Bottas', 'Raikkonen', 'Verstappen' ]

[ 'Alonson', 'Bottas', 'Raikkonen' ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Simulando Array

const quaseArray = { 0: 'Rafael', 1: 'Ana', 2: 'Bia' }

console.log(quaseArray)

*Object*.defineProperty(quaseArray, 'toString', {

value: function () { return *Object*.values(*this*) },

enumerable: false

})

console.log(quaseArray[0])

const meuArray = ['Rafael', 'Ana', 'Bia']

console.log(quaseArray.toString(), meuArray)

Saída:

{ '0': 'Rafael', '1': 'Ana', '2': 'Bia' }

Rafael

[ 'Rafael', 'Ana', 'Bia' ] [ 'Rafael', 'Ana', 'Bia' ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## ForEach

const aprovados = ['Agatha', 'Aldo', 'Daniel', 'Raquel']

// For Each percorre o array

aprovados.forEach(function(*nome*, *indice*, *array*) { // chamamos uma callBack

// primeiro parâmetro é o valor, o segundo é o índice

console.log(`${*indice* + 1}: ${*nome*}`)

console.log(*array*)

})

aprovados.forEach(*nome* => console.log(*nome*))

const exibirAprovados = *aprovado* => console.log(*aprovado*)

aprovados.forEach(exibirAprovados)

Saída:

1: Agatha

[ 'Agatha', 'Aldo', 'Daniel', 'Raquel' ]

2: Aldo

[ 'Agatha', 'Aldo', 'Daniel', 'Raquel' ]

3: Daniel

[ 'Agatha', 'Aldo', 'Daniel', 'Raquel' ]

4: Raquel

[ 'Agatha', 'Aldo', 'Daniel', 'Raquel' ]

Agatha

Aldo

Daniel

Raquel

Agatha

Aldo

Daniel

Raquel

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Array*.prototype.forEach2 = function(*callback*) { // essa função recebe um callback

for (let i = 0; i < *this*.length; i++) {

callback(*this*[i], i, *this*)

}

}

const aprovados = ['Agatha', 'Aldo', 'Daniel', 'Raquel']

aprovados.forEach(function(*nome*, *indice*) {

console.log(`${*indice* + 1}: ${*nome*}`)

})

Saída:

1: Agatha

2: Aldo

3: Daniel

4: Raquel

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Map

// Um Map transforma um array em outro, com o mesmo tamanho

const nums = [1, 2, 3, 4, 5]

// For com propósito

let resultado = nums.map(function(*e*) { // passando um callback

return *e* \* 2

})

console.log(resultado, nums)

const soma10 = *e* => *e* + 10

const triplo = *e* => *e* \* 3

const paraDinheiro = *e* => `R$ ${parseFloat(*e*).toFixed(2).replace('.', ',')}`

resultado = nums.map(triplo).map(soma10).map(paraDinheiro)

console.log(resultado)

Saída:

[ 2, 4, 6, 8, 10 ] [ 1, 2, 3, 4, 5 ]

[ 'R$ 13,00', 'R$ 16,00', 'R$ 19,00', 'R$ 22,00', 'R$ 25,00' ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

const carrinho = [

'{ "nome": "Borracha", "preco": 3.45 }',

'{ "nome": "Caderno", "preco": 13.90 }',

'{ "nome": "Kit de Lapis", "preco": 41.22 }',

'{ "nome": "Caneta", "preco": 7.50 }'

]

// Retornar um array apenas com os preços

const paraObjeto = *json* => JSON.parse(*json*)

const apenasPreco = *produto* => *produto*.preco

resultado = carrinho.map(paraObjeto).map(apenasPreco)

console.log(resultado)

Saída:

[ 3.45, 13.9, 41.22, 7.5 ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Array*.prototype.map2 = function(*callback*) { //implementação de um map

const newArray = []

for(let i = 0; i < *this*.length; i++) {

newArray.push(callback(*this*[i], i, *this*))

}

return newArray

}

const carrinho = [

'{ "nome": "Borracha", "preco": 3.45 }',

'{ "nome": "Caderno", "preco": 13.90 }',

'{ "nome": "Kit de Lapis", "preco": 41.22 }',

'{ "nome": "Caneta", "preco": 7.50 }'

]

const paraObjeto = *json* => JSON.parse(*json*)

const apenasPreco = *produto* => *produto*.preco

resultado = carrinho.map2(paraObjeto).map2(apenasPreco)

console.log(resultado)

Saída:

[ 3.45, 13.9, 41.22, 7.5 ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Filter

const produtos = [

{ nome: 'Notebook', preco: 2499, fragil: true },

{ nome: 'iPad Pro', preco: 4199, fragil: true },

{ nome: 'Copo de Video', preco: 12.49 , fragil: true },

{ nome: 'Copo de Plástico', preco: 18.99, fragil: false },

]

console.log(produtos.filter(function(*e*) { //essa função é uma callback

// se retornar true ele leva pro array final

// se false não será adicionado no array final

return (*e*.preco >= 500 && *e*.fragil == true) ? true : false

}))

// ou

const caro = *produto* => *produto*.preco >= 500 // retorna true ou false

const fragil = *produto* => *produto*.fragil

console.log(produtos.filter(caro).filter(fragil))

Saída:

[

{ nome: 'Notebook', preco: 2499, fragil: true },

{ nome: 'iPad Pro', preco: 4199, fragil: true }

]

[

{ nome: 'Notebook', preco: 2499, fragil: true },

{ nome: 'iPad Pro', preco: 4199, fragil: true }

]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Array*.prototype.filter2 = function(*callback*) {

const newArray = []

for (let i = 0; i < *this*.length; i++) {

if (callback(*this*[i], i, *this*)) {

newArray.push(*this*[i])

}

}

return newArray

}

const produtos = [

{ nome: 'Notebook', preco: 2499, fragil: true },

{ nome: 'iPad Pro', preco: 4199, fragil: true },

{ nome: 'Copo de Video', preco: 12.49 , fragil: true },

{ nome: 'Copo de Plástico', preco: 18.99, fragil: false },

]

const caro = *produto* => *produto*.preco >= 500 // retorna true ou false

const fragil = *produto* => *produto*.fragil

console.log(produtos.filter2(caro).filter2(fragil))

Saída:

[

{ nome: 'Notebook', preco: 2499, fragil: true },

{ nome: 'iPad Pro', preco: 4199, fragil: true }

]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Reduce

// transforma em único elemento

// array.reduce(callback, valor\_inicial)

const alunos = [

{ nome: 'João', nota: 7.3, bolsista: false },

{ nome: 'Maria', nota: 9.2, bolsista: true },

{ nome: 'Pedro', nota: 9.8, bolsista: false },

{ nome: 'Ana', nota: 8.7, bolsista: true }

]

console.log(alunos.map(*a* => *a*.nota))

const resultado = alunos.map(*a* => *a*.nota).reduce(function(*acumulador*, *atual*) {

console.log(*acumulador*, *atual*)

return *acumulador* + *atual*

}, 0) // 0 valor inicial (opcional)

console.log(resultado)

Saída:

[ 7.3, 9.2, 9.8, 8.7 ]

0 7.3

7.3 9.2

16.5 9.8

26.3 8.7

35

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

const alunos = [

{ nome: 'João', nota: 7.3, bolsista: false },

{ nome: 'Maria', nota: 9.2, bolsista: true },

{ nome: 'Pedro', nota: 9.8, bolsista: false },

{ nome: 'Ana', nota: 8.7, bolsista: true }

]

const isBolsista = *a* => *a*.bolsista

// Desafio 1: Todos os alunos são bolsistas?

const todosBolsistas = alunos.reduce(function (*acumulador*, *atual*) {

return *acumulador* && isBolsista(*atual*)

}, true)

console.log(todosBolsistas)

// Desafio 2: Algum aluno é bolsista?

const algumBolsista = alunos.reduce(function (*acumulador*, *atual*) {

return *acumulador* || isBolsista(*atual*)

}, false)

console.log(algumBolsista)

Saída:

false

true

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Array*.prototype.reduce2 = function (*callback*, *valorInicial*) {

const indiceInicial = *valorInicial* ? 0 : 1

let acumulador = *valorInicial* || *this*[0]

for (let i = indiceInicial; i < *this*.length; i++) {

acumulador = callback(acumulador, *this*[i], i, *this*)

}

return acumulador

}

const soma = (*total*, *valor*) => *total* + *valor*

const nums = [1, 2, 3, 4, 5, 6]

console.log(nums.reduce2(soma, 100))

Saída:

121

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Imperativo vs Declarativo

const alunos = [

{ nome: 'João', nota: 7.9 },

{ nome: 'Maria', nota: 9.2 }

]

// Imperativo, se preocupa mais em como fazer uma tarefa

let total1 = 0

for (let i = 0; i < alunos.length; i++) {

total1 += alunos[i].nota

}

console.log(total1/ alunos.length)

// Declarativo, se preocupa mais como o que tem que ser feito, além do reuso.

const getNota = *aluno* => *aluno*.nota

const soma = (*total*, *atual*) => *total* + *atual*

const total2 = alunos.map(getNota).reduce(soma)

console.log(total2 / alunos.length)

Saída:

8.55

8.55

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Concat (concatenação)

const filhas = ['Ualeskah', 'Cibalena']

const filhos = ['Uoxiton', 'Uesclei']

const todos = filhas.concat(filhos, 'Fulano')

console.log(todos, filhas, filhos)

console.log([].concat([1, 2], [3, 4], 5, [[6, 7]]))

Saída:

[ 'Ualeskah', 'Cibalena', 'Uoxiton', 'Uesclei', 'Fulano' ] [ 'Ualeskah', 'Cibalena' ] [ 'Uoxiton', 'Uesclei' ]

[ 1, 2, 3, 4, 5, [ 6, 7 ] ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## FlatMap

const escola = [{

nome: 'Turma M1',

alunos: [{

nome: 'Gustavo',

nota: 8.1

}, {

nome: 'Ana',

nota: 9.3

}]

}, {

nome: 'Turma M2',

alunos: [{

nome: 'Rebeca',

nota: 8.9

}, {

nome: 'Roberto',

nota: 7.3

}]

}]

const getNotaDoAluno = *aluno* => *aluno*.nota //a é o objeto aluno

const getNotasDaTurma = *turma* => *turma*.alunos.map(getNotaDoAluno)

const notas1 = escola.map(getNotasDaTurma)

console.log(notas1)

*Array*.prototype.flatMap = function(*callback*) {

return *Array*.prototype.concat.apply([], *this*.map(*callback*))

}

const notas2 = escola.flatMap(getNotasDaTurma)

console.log(notas2)

Saída:

[ [ 8.1, 9.3 ], [ 8.9, 7.3 ] ]

[ 8.1, 9.3, 8.9, 7.3 ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Node

É um runtime.

## Módulos

moduloA.js

// Em node um arquivo representa um módulo

*this*.ola = 'Fala Pessoal'

*exports*.bemVindo = 'Bem vindo ao node!'

*module*.*exports*.ateLogo = 'Até próximo exemplo'

moduloB.js

**let a = 2 // apenas visível dentro do módulo**

***module*.*exports* = {**

**bomDia: 'Bom dia',**

**boaNoite() {**

**return 'Boa noite'**

**}**

**}**

moduloCliente.js

// acessando outros módulos

const moduloA = require('./moduloA') //acesso o moduloA

const moduloB = require('./moduloB.js')

console.log(moduloA.ola)

console.log(moduloA.bemVindo)

console.log(moduloA.ateLogo)

console.log(moduloA)

console.log(moduloB.bomDia)

console.log(moduloB.boaNoite())

console.log(moduloB)

Saída:

Fala Pessoal

Bem vindo ao node!

Até próximo exemplo

{

ola: 'Fala Pessoal',

bemVindo: 'Bem vindo ao node!',

ateLogo: 'Até próximo exemplo'

}

Bom dia

Boa noite

{ bomDia: 'Bom dia', boaNoite: [Function: boaNoite] }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Usando Módulos de Terceiros

usamos o gerenciador de pacotes npm pelo terminal

>> npm i lodash (instala o módulo lodash na pasta do projeto)

>> npm i -g nodemon (instala de forma global no sistema operacional)

nodemon reinicia o programa a cada mudança no código

Ctrl + C, para o nodemon

usamos no terminal >> nodemon nome\_do\_arquivo

## Sistema de Módulos: Require

// ../ sai de uma pasta

//respeitar letras maiúsculas e minúsculas no nome do arquivo

const moduloA = require('../../moduloA')

console.log(moduloA.ola)

const saudacao = require('saudacao') // importando da pasta node\_modules

console.log(saudacao.ola)

const c = require('./pastaC/index') // ou somente './pastaC'

console.log(c.ola2)

const *http* = require('http')

*http*.createServer((*req*, *res*) => {

*res*.write('Bom dia!')

*res*.end()

}).listen(8080) // abra no browser localhost:8080

Saída:

Fala Pessoal

Bom dia Pessoal!

Legal

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Sistema de Módulos: Exports

console.log(*module*.*exports* === *this*)

console.log(*module*.*exports* === *exports*)

*this*.a = 1 // é apenas referência para modele.exports

*exports*.b = 2 // é apenas referência para modele.exports

*module*.*exports*.c = 3

*exports* = null

console.log(*module*.*exports*)

*exports* = { // não usar assim !!!, não exporta dessa forma para outro módulo

nome: 'Teste'

}

console.log(*module*.*exports*)

*module*.*exports* = { // forma correta

nome: 'Teste'

}

console.log(*module*.*exports*)

Saída:

true

true

{ a: 1, b: 2, c: 3 }

{ a: 1, b: 2, c: 3 }

{ nome: 'Teste' }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Arquivo package.json

Este arquivo tem a descriçãode todas as dependências do projeto

importante estar dentro da pasta do projeto no terminal

>> npm init (cria o arquivo package.json na pasta do projeto)

>> npm init -y (cria o arquivo sem as perguntas, ele responde ‘yes’ para todas as perguntas)

>> npm i --save axios (instalando um dependências, neste caso axios)

>> npm i --save-dev axios@0.17.1 -E

(--save-dev instala uma depedência somente necessária em desenvolvimento)

(@0.17.1 -E, instala exatamente a versão passada)

package-lock.json precisa ir para o repositório,

## Obtendo arquivo JSON remotamente

const url = 'http://files.cod3r.com.br/curso-js/funcionarios.json'

const axios = require('axios') // axios é um client http

const china = *funcionario* => *funcionario*.pais == 'China'

const mulher = *funcionario* => *funcionario*.genero == 'F'

const menorSalario = (*func*, *funcAtual*) => {

return *func*.salario < *funcAtual*.salario ? *func* : *funcAtual*

}

axios.get(url).then(*response* => { // obter o conteudo do arquivo

const funcionarios = *response*.data

const mulherChinesaMenorSalario = funcionarios

.filter(china)

.filter(mulher)

.reduce(menorSalario)

console.log(mulherChinesaMenorSalario)

})

// Desafio, obter a mulher chinesa com o menor salário

Saída:

{

id: 443,

nome: 'Shela',

sobrenome: 'Nowell',

email: 'snowellca@upenn.edu',

genero: 'F',

cidade: 'Yufa',

pais: 'China',

empresa: 'Quinu',

salario: 2435.61

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Instância Única vs Nova Instância

instanciaUnica.js

// node faz cache dos módulos

*module*.*exports* = {

valor: 1,

inc() {

*this*.valor++

}

}

instanciaNova.js

**// Uma factory retorna um novo objeto (uma nova instância)**

***module*.*exports* = () => {**

**return {**

**valor: 1,**

**inc() {**

***this*.valor++**

**}**

**}**

**}**

instanciaCliente.js

const contadorA = require('./instanciaUnica')

const contadorB = require('./instanciaUnica')

const contadorC = require('./instanciaNova')() // () já que é uma função "factory"

const contadorD = require('./instanciaNova')()

// note que invocando o contadorA, o contadorB também se altera

contadorA.inc()

contadorA.inc()

console.log(contadorA.valor, contadorB.valor)

// note que invocando o contadorC, o contadorD não se altera

contadorC.inc()

contadorC.inc()

console.log(contadorC.valor, contadorD.valor)

Saída:

**3 3**

**3 1**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Objeto Global no Node

console.log(global)

Saída:

<ref \*1> Object [global] {

global: [Circular \*1],

clearInterval: [Function: clearInterval],

clearTimeout: [Function: clearTimeout],

setInterval: [Function: setInterval],

setTimeout: [Function: setTimeout] {

[Symbol(nodejs.util.promisify.custom)]: [Getter]

},

queueMicrotask: [Function: queueMicrotask],

clearImmediate: [Function: clearImmediate],

setImmediate: [Function: setImmediate] {

[Symbol(nodejs.util.promisify.custom)]: [Getter]

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

global.js

global.MinhaApp1 = {

saudacao() {

return 'Estou em todos os lugares!'

},

nome: 'Sistema Legal'

}

global.MinhaApp2 = *Object*.freeze({ // impede que se faça alterações no objeto

saudacao() {

return 'Estou em todos os lugares!'

},

nome: 'Sistema Legal'

})

globalClient.js

require('./global')

console.log(MinhaApp1.saudacao())

console.log(MinhaApp1.nome)

console.log(MinhaApp2.saudacao())

console.log(MinhaApp2.nome)

MinhaApp1.nome = 'Eita!'

MinhaApp2.nome = 'Eita!'

console.log(MinhaApp1.nome)

console.log(MinhaApp2.nome) // este objeto está com o método freeze

Saída:

Estou em todos os lugares!

Sistema Legal

Estou em todos os lugares!

Sistema Legal

Eita!

Sistema Legal

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Entendendo o This

console.log(`this === global: ${*this* === global}`)

console.log(`this === module: ${*this* === *module*}`)

console.log(`this === module.exports: ${*this* === *module*.*exports*}`)

console.log(`this === exports: ${*this* === *exports*}`)

function logThis() {

console.log('Dentro de um função:')

console.log(`this === exports: ${*this* === *exports*}`)

console.log(`this === module.exports: ${*this* === *module*.*exports*}`)

console.log(`this === global: ${*this* === global}`)

*this*.perigo = [1, 2, 3] // aponta para global

}

logThis()

console.log(perigo) // após a chamada da função é possível acessar 'perigo'

Saída:

this === global: false

this === module: false

this === module.exports: true

this === exports: true

Dentro de um função:

this === exports: false

this === module.exports: false

this === global: true

[ 1, 2, 3 ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Passando parâmetros entre Módulos

passandoParametros.js

*module*.*exports* = function(...*nomes*) {

// ...nomes define que será passado vários parâmetros

return *nomes*.map(*nome* => `Boa semana ${*nome*}!`)

}

passandoParamentrosCliente.js

const saudacoes = require('./passandoParametros')('Ana', 'Lucas', 'João')

console.log(saudacoes)

Saída:

[ 'Boa semana Ana!', 'Boa semana Lucas!', 'Boa semana João!' ]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Instalando Deps & Scripts

package.json

{

"name": "funcionarios",

"version": "1.0.0",

"description": "Descrição",

"main": "funcionarios.js", // arquivo principal

"scripts": {

"start": "nodemon", // no terminal >> npm start

"dev": "nodemon", // no terminal >> npm run dev

"test": "cmd"

},

"author": "jonas",

"license": "ISC",

"dependencies": {

"axios": "^0.21.1"

}

}

no terminal >> npm i (lê o arquivo package.json, e instala as dependências descritas no arquivo, refaz a pasta node\_modules)

## Lendo Arquivos

arquivo.json

{

"db": {

"host": "localhost",

"port": 5432,

"user": "usuario",

"pass": "123465"

}

}

arquivoLeitura.js

const *fs* = require('fs')

const caminho = \_\_dirname + '/arquivo.json'

// sincrono...

const conteudo = *fs*.readFileSync(caminho, 'utf-8')

console.log(conteudo)

// assincrono...

*fs*.readFile(caminho, 'utf-8', (*err*, *conteudo*) => {

const config = JSON.parse(*conteudo*)

console.log(`${config.db.host}:${config.db.port}`)

})

const config = require('./arquivo.json')

console.log(config.db)

*fs*.readdir(\_\_dirname, (*err*, *arquivos*) => { // leitura de um diretório

console.log('Conteúdo da pasta...')

console.log(*arquivos*)

})

Saída:

{

"db": {

"host": "localhost",

"port": 5432,

"user": "usuario",

"pass": "123465"

}

}

{ host: 'localhost', port: 5432, user: 'usuario', pass: '123465' }

Conteúdo da pasta...

[

'arquivo.json',

'arquivoLeitura.js',

'exportar.js',

'funcionarios',

'global.js',

'globalCliente.js',

'instanciaCliente.js',

'instanciaNova.js',

'instanciaUnica.js',

'moduloA.js',

'moduloB.js',

'moduloCliente.js',

'passandoParametroCliente.js',

'passandoParametros.js',

'pastaA.js',

'somaSimples.js',

'this.js',

'usandoModulosTerceiros.js'

]

localhost:5432

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Escrevendo em Arquivos

arquivoEscrita.js

const *fs* = require('fs')

const produto = {

nome: 'Celular',

preco: 1249.99,

desconto: 0.15

}

// fs.writeFile(\_\_dirname + '/nome\_do\_arquivo.json', JSON.stringify(produto), callback)

// fs.writeFile(caminho + nome do arquivo, conversão para JSON, callback que será executada)

*fs*.writeFile(\_\_dirname + '/arquivoGerado.json', JSON.stringify(produto), *err* => {

// passando o erro (err) para callback

console.log(*err* || 'Arquivo salvo!') // em caso de erro, escreve o erro

})

arquivoGerado.json

{"nome":"Celular","preco":1249.99,"desconto":0.15}

Saída:

Arquivo salvo!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Frameworks Web

É algo que estrutura a aplicação

Exemplos:

* Frameworks front-end:
  + Angular.js <https://angularjs.org/> ;
  + Vue.js <https://vuejs.org/> ;
  + React.js <https://reactjs.org/>
* Frameworks back-end:
  + Express.js <https://expressjs.com/> ;
  + Sails.js <https://sailsjs.com/> ;
  + Koa <https://koajs.com/>

## Padrão Middleware

ou Padrão Chain of Responsability

## Projeto: API com Express

no terminal, e na pasta do projeto, criamos o arquivo package.json com

>> npm init -y

ainda no terminal, criamos a pasta node\_modules e instalamos o express (neste exemplo foi instalado a versão 4.16.2)

>> npm i --save express@4.16.2 -E

Instalamos também o Body-Parser (neste exemplo foi instalado a versão 1.18.2

>> npm i --save body-parser@1.18.2 -E

criamos uma pasta src dentro da pasta do projeto, e um arquivo server.js

criamos um módulo para simular um banco de dados (database.js)

Executamos pelo terminal >> npm start (ele irá rodar o nodemon)

Abrir o postman para acessar e fazer requisições da porta http://localhost:3003/produtos

database.js

const sequence = {

\_id: 1,

get id() { return *this*.\_id++ } // retorna o próximo valor de id

}

const produtos = {}

function salvarProduto(*produto*) {

if (!*produto*.id) *produto*.id = sequence.id

produtos[*produto*.id] = *produto*

return *produto*

}

function getProduto(*id*) {

return produtos[*id*] || {}

}

function getProdutos() {

return *Object*.values(produtos) // retorna todos os valores

}

function excluirProduto(*id*) {

const produto = produtos[*id*]

delete produtos[*id*] // exclui na posição

return produto

}

*module*.*exports* = { salvarProduto, getProduto, getProdutos, excluirProduto }

server.js

const porta = 3003 // declarando a porta (porta é um processo dentro do computador)

const express = require('express') // importando o express

const app = express() // instanciando em uma variável

const bodyParser = require('body-parser')

const database = require('./database') // importando um módulo

app.use(bodyParser.urlencoded({ extended: true }))

/\*

app.get('/produtos', (req, res, next) => { // .get é uma forma de requisição

console.log('Middleware 1...')

next() // passa para a próxima requisição

})

app.use((req, res, next) => { // .use responde para todas as requisições

console.log('Middleware 2...')

next() // passa para a próxima requisição

})

app.get('/produtos', (req, res, next) => { // .get é uma forma de requisição

res.send({ nome: 'Notebook', preco: 123.45}) // converter para JSON

})\*/

app.get('/produtos', (*req*, *res*, *next*) => {

*res*.send(database.getProdutos()) // retorna todos os produtos

})

app.get('/produtos/:id', (*req*, *res*, *next*) => { // passando a url com parâmetros

*res*.send(database.getProduto(*req*.params.id)) // passando parâmetro obtido como requisição

})

app.post('/produtos', (*req*, *res*, *next*) => {

const produto = database.salvarProduto({

nome: *req*.body.nome,

preco: *req*.body.preco

})

*res*.send(produto) // JSON

})

app.put('/produtos/:id', (*req*, *res*, *next*) => { // substituir um id, alterar um registro

const produto = database.salvarProduto({

id: *req*.params.id,

nome: *req*.body.nome,

preco: *req*.body.preco

})

*res*.send(produto) // JSON

})

app.delete('/produtos/:id', (*req*, *res*, *next*) => { // substituir um id, alterar um registro

const produto = database.excluirProduto(*req*.params.id)

*res*.send(produto) // JSON

})

app.listen(porta, () => {

console.log(`Servidor executando na porta ${porta}.`)

})

Saída:

Servidor executando na porta 3003.

Abrir o postman para acessar e fazer requisições da porta <http://localhost:3003/produtos>

get para obter registros,

<http://localhost:3003/produtos> para visualizar todos

[http://localhost:3003/produtos/1](http://localhost:3003/produtos) para visualizar o respectivo “id”

post para adicionar novos registros, passando as informações em body, na opção x-www-form-urlendoded,

adicionando chave de valor que serão adicionados (<http://localhost:3003/produtos>)

put irá fazer alteração de um registro já adicionado (na posição “id”) ([http://localhost:3003/produtos/1](http://localhost:3003/produtos)), levando os dados em x-www-form-urlendoded

delete irá excluir o registro passado como parâmetro o “id” em [http://localhost:3003/produtos/1](http://localhost:3003/produtos)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Temporizador

const schedule = require('node-schedule')

// '\*/5 \* 20 \* \* 2' de 5 em 5 segundos, em qualquer minuto, as 20 horas, ignorando o dia do mes, ignorando o mês, e nas terça feiras

const tarefa1 = schedule.scheduleJob('\*/5 \* 20 \* \* 2', function () {

console.log('Executando Tarefa 1!', **new** *Date*().getUTCSeconds())

})

setTimeout(function () { // temporarizador padrão do JS

tarefa1.cancel()

console.log('Cancelando Tarefa 1')

}, 20000)

// setImmediate, similar ao setTimeout, com o tempo sempre em zero 0

// setInterval, executa uma função de tempos em tempos

const regra = **new** schedule.RecurrenceRule()

regra.dayOfWeek = [**new** schedule.Range(1, 5)]

regra.hour = 20

regra.second = 30

const tarefa2 = schedule.scheduleJob(regra, function () {

console.log('Executando Tarefa 2!', **new** *Date*().getSeconds())

})

Saída:

Executando Tarefa 1! 10

Executando Tarefa 1! 15

Executando Tarefa 1! 20

Executando Tarefa 1! 25

Cancelando Tarefa 1

Executando Tarefa 2! 30

Executando Tarefa 2! 30

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Process: Entrada e Saída Padrão

// pegando parâmetros apartir de uma chamada pelo terminal

// >> node entradaESaida.js -a

const anonimo = process.argv.indexOf('-a') !== -1

// console.log(anonimo)

if (anonimo) {

process.stdout.write('Fala Anônimo!\n')

process.exit()

} else {

process.stdout.write('Informe o seu nome: ') // aparece no terminal

process.stdin.on('data', *data* => { // pega o que foi digitado no terminal

const nome = *data*.toString().replace('\n', '') // removendo o Enter que o usuário digita

process.stdout.write(`Fala ${nome}\n`)

process.exit()

})

}

Saída:

...\MyProjects\JavaScript-Udemy\exercicios-js\8\_node>node entradaESaida.js

Informe o seu nome: Jonas

Fala Jonas

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.

Saída:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_