

Sem For	
array.slice(inicio, fim?)	Cria uma nova lista contendo os elementos da string separados pelo demilitador
array.map(fn(item, index))	Aplica a função fn para todos os itens e retorna algo
array.filter(fn(item))	Filtra a array. Fn deve retornar true para o item que devera ser filtrado
array.reduce(fn(accumulator, current), valorInicial)	Para cada item na lista, a função é aplicada, retornando ao fim o valor total do accumulator
array.forEach(fn(item, index))	Aplica a função fn para cada item do array
array.join(delimitador)	Transforma a array em uma string

Strings	
str.split(delimitador)	Retorna uma lista contendo os elementos da string separados pelo demilitador
str.toUpperCase()	Retorna a string com todos os caracteres maiusculos

Strings (cont)	
str.toLowerCase()	

Eu sou uma \${vari alv elQ ual a uer}

str + str 2

const str = "ola"

const str = " ola '1' "

const str = ' ola "1" '

\

Funções	
(parametro1, parametro2) => {}	Cria uma nova arrow function
function(parametro1, parametro2) {}	Cria uma nova função
() => ({key: valor})	cria uma arrow function de uma linha que retorna um objeto

Funções (cont)	
() => {return string ola.}	Retorna arrow function
"- ola- "	Cria arrow function
"- ola- "	posso colocar uma linha, não precisa de {}

Interpolação de string

class Oi {}

new Oi()

constructor {}

Oi.metodo()

class ClassWithPrivateField { #privateField; }

const str = "ola" dentro da classe para ter um construtor

Coisas legais

const newObj = {...oldObj}

const newList = [...oldList]

Coisas legais (cont)	
const [primeiro, segundo, terceiro] = [1, 2, 3]	Desconstroi a lista em três constantes

const [primeiro, segundo, terceiro] = obj	Desconstroi um obj para as constantes. O Objeto precisa ter as chaves iguais para funcionar!
---	--

Encontre o que procura

array.findIndex(fn(item))	Retorna o primeiro item encontrado ou undefined
---------------------------	---

array.indexOf(fn(item))	Retorna o indice do primeiro item encontrado ou -1
-------------------------	--

array.includes(item)	Retornar True ou False
----------------------	------------------------

Regex no javascript

const regex = /novoRegex/gmi	Cria uma nova expressão regex que pode ser usada posteriormente
------------------------------	---

str.replace(regex, novoValor)	Substitui o que foi encontrado na string pelo novoValor
-------------------------------	---

regex.test(str)	Retorna verdadeiro caso de um match ou falso caso não
-----------------	---



By **fersasil**
cheatography.com/fersasil/

Published 13th July, 2021.
Last updated 13th July, 2021.
Page 1 of 2.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopadd.com>

Regex no javascript (cont)		Objetos (cont)		Promessas		Promessas (cont)	
regex.e xec(s- tring)	Retorna null em caso de não encontrar nada, retorna uma lista dos matchs e grupos de captura	Object.ke- ys(obj),m- ap(key => obj[key])	Acessando todos os valores de um objeto	promis- e.then- (fu- nction resolv- e(r- es)).c- atch(f- unction reject- (error))	Caso de sucesso, a função dentro do then sera executada. Em de erro, a função dentro do catch sera executada	Promis e.allS- ettled- (lista- DeP- rom- essas)	Executa todas as promises em paralelo e só executa o código abaixo quando todas são executas, com ou sem erro. Retorna uma lista com o resultado das promessas
str.ma- tch- (regex)	Retorna uma lista com todos os itens (Não suporta grupo de captura)	Object.as- sign({}, ...listaD- eObjetos)	Retornar um array de todos os valores de um objeto	async function fn(){}		new Promis	Cria uma nova promessa. Chame
str.ma- tchAll- (regex)	Retorna um iterator que pode ser usado com for de todos os matchs(Suporta lista de captura)	Object.fr- eeze(obj);	Cria um novo object mesclando todos os objetos da lista	await promise	Sugar sintatico que permite a utilização do await	e((res- olve, reject) => {})	resolve e passe o resultado em caso de sucesso. Chame o reject em caso de falha.
[...st- r.mat- chAll(r- egex)]	Lista de todos os matchs (suporta grupo de captura)	Objetos são como dicionários (python). Eles possuem uma chave única (key) e um valor (Qualquer coisa). Podemos cirar um objeto com a seguinte sintaxe:		Faz com que todo o código abaixo do await espere para ser executado. Por baixo dos panos, coloca tudo que esta abaixo dentro de um then.		Uma promessa é um código assincrono, que sera mandado ao event loop. O resultado só estara disponível depois, mas a aplicação continuara executando o código. Ele é não bloqueante	
Objetos		Listas		Promis- e.all(lis- taDePr- ome- ssas)			
const obj = {}	Criando um novo objeto	array.push(n- ovoltem)	Adiciona um novo item ao array	Executa todas as promises em paralelo e só executa o código abaixo quando todas são executas e bem-su- cedidas. Retorna uma lista com o resultado das promessas			
const obj = {chave- Unica: 'valorQua- lquer'}	Criar objeto com valor inicial	array.pop()	Remove o último elemento do array				
obj.chave- Univa ou obj['chav- eUnica']	Acessar o valor de um objeto	array.shift()	Remove o primeiro elemento do array				
delete obj.chave- Unica	Deleta uma chave	array.unshif- t(item)	Remove o primeiro elemento do array				
Object.ke- ys(obj)	Retornar um array de todas as chaves de um elemento	array.splice(in- dexParaRe- mover, numero- DelItemsParaR- emover)	Adiciona um item ao inicio do array				
		fruits.slice()	Remove um item de determinada posição de um array				
			Copia o array				

