/\*\* Funções - Parte 2

- Funções anônimas

- Arrow Function - Funções de seta

- Promises - Promessas

- async/await

\*/

// setTimeout(() => console.log("Executado após 3 segundos"), 3000);

// [].filter(function () { });

// [].filter(() => { });

// const resultado = function (a, b) {

// return a \* b;

// }(10, 2);

// function somar(a, b) {

// return a + b;

// }

// const somar = (a, b) => a + b;

/\*\* Funções anônimas:

São funções que não possuem um nome identificador

São usadas principalmente quando precisamos de uma função

temporária ou quando queremos passar uma função como argumento

para outra função.

Exemplos:

setTimeout(function() {

console.log("Executado após 3 segundos");

}, 3000);

const resultado = function (a, b) {

return a \* b;

}(10, 2);

\*/

/\*\* Arrow Function - Funções de Seta

É uma sintaxe alternativa, mais concisa, para escrever funçõe em JS

A palavra-chave function não é necessária e o retorno da função é

implícito, a menos que seja usado um bloco de código

const digaOla = () => {

console.log('Olá');

};

const soma = (a, b) => {

return a + b

};

const soma = (a, b) => a + b;

const numeroPar = (numero) => numero % 2 === 0;

\*/

/\*\* Promises - Promesas

Usado para lidar com operações assíncronas (operações lentas), como uma requisição

HTTP, leitura de arquivo ou consulta à um banco de dados.

Representa a eventual conclusão ou falha de uma operação assíncrona.

function multiplicaPorDois(valor) {

return new Promise((resolve, reject) => {

if (typeof valor === 'number') {

resolve(valor \* 2);

} else {

reject(new Error('O argumento não é um número.'));

}

});

}

multiplicaPorDois('10')

.then(resulado => console.log(resulado))

.catch(console.error);

O método then() é usado quando a Promise é resolvida com sucesso e recebe uma função de retorno de chamada que será executada com o resultado.

O método catch() é usado para tratar erros e recebe uma função de retorno de chamada que será executada quando ocorrer um erro.

Um exemplo prático de uso de Promises é uma requisição assíncrona a uma API usando o fetch():

fetch('https://viacep.com.br/ws/78085720/json')

.then(resultado => resultado.json())

.then(dados => console.log(dados))

.catch(error => console.error('Erro consulta via CEP:', error));

\*/

/\*\* async / await:

Introduzido no ECMAScript 2017, é uma maneira mais fácil e legível para lidar com programação assíncrona.

Você escreve um código assíncrono como se estivesse escrevendo um código síncrino

A palavra-chave async é usada para declarar uma função assíncrona, que pode conter uma ou várias operações assíncronas.

async function consultarCep() {

try {

const resulta do = await fetch('https://viacep.com.br/ws/78085720/json');

const json = await resultado.json();

console.log(json);

} catch (error) {

console.error('Erro na consulta via CEP:', error);

}

}

É importante notar que o uso de await só é permitido dentro de uma função assíncrona.

\*/

// somar(10, '2')

// .then(resuladoSoma => console.log(resuladoSoma))

// .catch(error => console.error(error));

// Javascript object annotation

// fetch('https://viacep.com.br/ws/78085720/json')

// .then(resultado => {

// resultado.json()

// .then(json => console.log(json))

// .catch(console.error);

// })

// .catch(error => console.error(error));

// function somar(a, b) {

// return new Promise((resolve, reject) => {

// if (typeof a !== 'number' || typeof b !== 'number') {

// reject('Números invalido!');

// }

// resolve(a + b);

// });

// }

// async function somar(a, b) {

// try {

// if (typeof a !== 'number' || typeof b !== 'number') {

// throw new Error('Numero invalido')

// }

// return a + b;

// } catch (error) {

// console.error(error);

// }

// }

// fetch('https://viacep.com.br/ws/78085720/json')

// .then(resultado => {

// resultado.json()

// .then(json => console.log(json))

// .catch(console.error);

// })

// .catch(error => console.error(error));

// async function consultarCEP() {

// try {

// const resutado = await fetch('https://viacep.com.br/ws/78085720/json');

// const resultadoJSON = await resutado.json();

// console.log(resultadoJSON);

// } catch (error) {

// console.error('Error na consulta CEP', error);

// }

// }

const consultarCEP = async () => {

try {

} catch (error) {

}

};