Seção 08 Javascript assíncrono – Promises, Ajax, Fecth e Async/Await

* Aula 86 Promises

function rand(min, max) {

    min \*= 1000;

    max \*= 1000;

    return Math.floor(Math.random() \* (max - min) + min);

}

function esperaAi(msg, tempo){

    setTimeout( () => {

        console.log(msg);

    }, tempo);

}

esperaAi('Frase 1', rand(1, 3));

esperaAi('Frase 2', rand(1, 3));

esperaAi('Frase 3', rand(1, 3));

aqui cada função esperai() levará um tempo para ser executada.

Para resolver isso vamos utilizar as promises.

Uma ação so será utilizada depois que a anterior tiver acabado.

function esperaAi(msg, tempo){

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

        setTimeout( () => {

            resolve(msg);

        }, tempo);

    });

}

Essa e uma função que retorna uma promessa.

Para que isso ocorra e nescessario chamar o new Promise(res,rej).

Resolve -> tente fazer isso, caso de certo use resolve

Reject -> tente fazer isso caso de errado use reject

esperaAi('Conexão com o BD', rand(1, 3))

.then(resposta => {

    console.log(resposta);

    return esperaAi('Buscando dado da base', rand(1, 3));

})

.then(resposta => {

    console.log(resposta);

    return esperaAi('Tratando os dados da base', rand(1, 3));

})

.then(resposta => {

    console.log(resposta);

})

.then(() => {

    console.log('Exibi os dados na tela');

})

.catch();

Aqui fica o complemento do código.

O uso do .then, indica uma promessa para esperar que deu certo.

O uso do .catch, indica que uma promessa que deu erro, se na cadeira ocorre um erro você vai direto pro catch.

Aqui vamos ver caso o tipo de msg for string ele ira cair no reject

function esperaAi(msg, tempo){

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

        if(typeof msg !== 'string'){

reject('BAD VALUE');

return;

};

        setTimeout( () => {

            resolve(msg);

        }, tempo);

    });

}

esperaAi('Conexão com o BD', rand(1, 3))

.then(resposta => {

    console.log(resposta);

    return esperaAi('Buscando dado da base', rand(1, 3));

})

.then(resposta => {

    console.log(resposta);

    return esperaAi(222222222222, rand(1, 3));

}).then(resposta => {

    console.log(resposta);

})

.then(() => {

    console.log('Exibi os dados na tela');

})

.catch(e => {

    console.log('ERRO', e);

});

Todo o resolve vai cair no then.

Todo o reject cai no catch.

* Aula 87 Metodos uteis para promises

function rand(min, max) {

    min \*= 1000;

    max \*= 1000;

    return Math.floor(Math.random() \* (max - min) + min);

}

function esperaAi(msg, tempo){

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

        if(typeof msg !== 'string') {

reject('BAD VALUE');

return;

}

        setTimeout( () => {

            resolve(msg.toUpperCase() + ' - Passei na promise');

        }, tempo);

    });

}

// Promise.all Promise.race Promise.resolve Promise.reject

const promises = [

    'Primeiro valor',

    esperaAi('Promise 1', 3000),

    esperaAi('Promise 2', 500),

    esperaAi('Promise 3', 1000),

    'Outro Valor'

];

Promise.all(promises)

.then(function(valor){

    console.log(valor);

})

.catch(function (erro){

    console.log(erro);

});

Ele vai resolver todas as promesas que tem dentro de promises,se uma de errado ele rejeita todas

Promise.race(promises)

.then(function(valor){

    console.log(valor);

})

.catch(function (erro){

    console.log(erro);

});

Nesse ele vai retorna sempre o primeiro valor. Como numa corrida.

Ele ira executar todas as tarefas, porem so ira retornar o primeiro valor o valor mais rápido

function baixaPagina() {

    const emChache = true;

    if(emChache) {

        return Promise.resolve('Pagina em chache');

    } else {

        return esperaAi('Baixei a pagina', 3000);

    }

}

Retorna uma promessa resolvida ou rejeitada de cara.