**Requisições Ajax**

Requisição assíncrona que é realizada ao back-end. Então no JavaScript ao invés de ter que sempre recarregar a página para trazer as novas informações do servidor, conseguimos usar o AJAX, que é uma forma de requisitar informações do servidor, sem precisar atualizar a página.

A requisição assíncrona, não acontece no mesmo fluxo do script do js, ela responde um pouco depois.

Ex:

// Para começar uma requisiçao ajax,

// Definir uma variavel, com a classe "new XMLHttpRequest()

var xhr = new XMLHttpRequest();

//Pegar o url para trazer os dados

xhr.open('GET','https://api.github.com/users/LucasPires50');

xhr.send(null);

//Validação para retonar as informação do servidor

xhr.onreadystatechange = function(){

    // "4" é a variavel que siginifica, quando a resposta do servidor volto.

    if (xhr.readyState === 4) {

        console.log(JSON.parse(xhr.responseText));

    }

}

var xhr = new XMLHttpRequest();

* Essa classe XMLHttpRequest() do js é que vai nos dar acesso a funcinalidade do ajax, para pode recuperar a informação de algum servidor;

**Promises (promessas):**

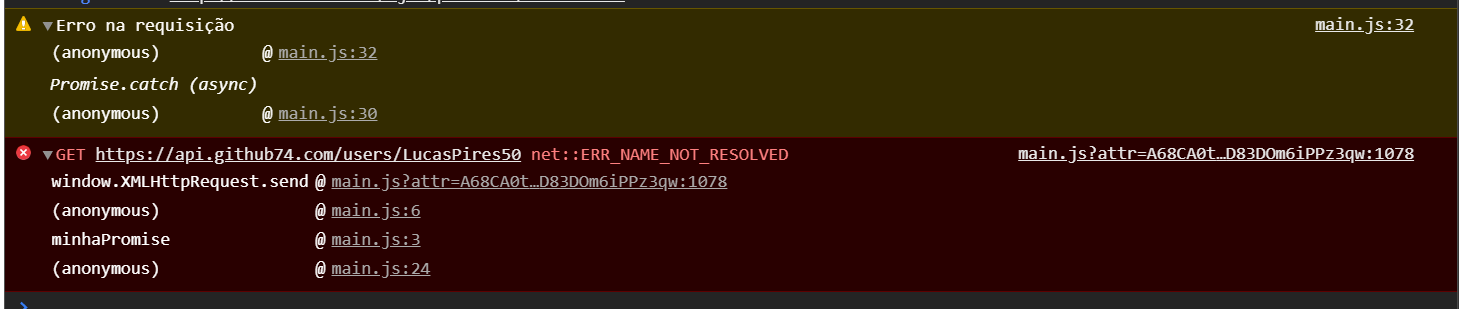
São códigos que não vão influenciar na linha do tempo do js, são funções que vão devolver o valor do resultado seja ele de sucesso ou erro, só depois de um tempo. E não é necessário se preocupar quando esse valor vai ser retornado, pois o js vai continuar executando normalmente.

    return new Promise(function(resolver, reject){

    })

* Resolve: também é uma função, é a função que utilizamos dentro da promesi, quando o resultado, que obtivemos foi de sucesso.
* Reject: também é uma função, é a função que utilizamos dentro da promesi, quando o resultado, que obtivemos não foi de sucesso.

console.warn -> Alerta



Ex:

var minhaPromise = function () {

    //o retorno desse função será uma promise

    return new Promise(function (resolve, reject) {

        var xhr = new XMLHttpRequest();

        xhr.open('GET', 'https://api.github74.com/users/LucasPires50');

        xhr.send(null);

        xhr.onreadystatechange = function () {

            // "4" é a variavel que siginifica, quando a resposta do servidor volto.

            if (xhr.readyState === 4) {

                //cod. 200 quer dizer que a requisição teve sucesso.

                if (xhr.status === 200) {

                    resolve(JSON.parse(xhr.responseText));

                    //Caso o resultado não seja de sucesso na requisição

                } else {

                    reject('Erro na requisição');

                }

            }

        }

    })

}

//Exibir o resultado que a promise retornou

minhaPromise()

    //".then", vai ser execultado quando chamar o resolve na promisse

    .then(function (response) {

        console.log(response);

    })

    //".catch", vai ser execultado quando chamar o reject na promisse

    .catch(function (error) {

        //".warn" é para avisar do erro

        console.warn(error);

    });

Utilizando Axios

Axios é uma biblioteca, utilizada para fazer as requisições assíncronas do js.

O axios nada mais é do que um encapsulamento por volta do XMLHttpRequest, com isso o axios só vai nos ajudar a retornar os valores de forma mais fácil.

Ex:

axios.get('https://api.github.com/users/LucasPires50')

    //".then", vai ser execultado quando chamar o resolve na promisse

    .then(function (response) {

        console.log(response);

    })

    //".catch", vai ser execultado quando chamar o reject na promisse

    .catch(function (error) {

        //".warn" é para avisar do erro

        console.warn(error);

    });