**Requisição (Request) → Resposta (Response)**

**Cliente (Navegador) (Client Side) → Servidor (Server Side)**

Servidores HTTP atuais: Apache, IIS, Nginx

**Requisição:**

Header (Cabeçalho):

* Location (URL): [www.uol.com.br](http://www.uol.com.br)
* Method: GET, POST, DELETE, PUT

Body (Corpo):

JSON, HTML, ARQUIVOS

**Resposta:**

Header:

* Status Code: 200, 201, 400, 404 (Not Found)
* Content-type: Tipo do corpo (JSON, HTML, PDF...)

Body:

JSON, HTML, ARQUIVOS

Nós podemos ver as requisições HTTP do navegador acessando o DEVTOOLS na aba Network (Rede).

**FETCH**

Fetch é uma API nativa no JS para executar requisições HTTP.

O Fetch retorna uma promessa (Promise).

**Exemplo de promessa**

const minhaPromessa = new Promise((resolve, reject) => {

let erro = true;

if(erro){

reject("Ops, deu um erro com a promessa");

}else{

resolve("Tudo certo com a promessa");

}

})

**Resolvendo promessas com .then e .catch**

O argumento passado na função resolve da promise é recebido pela função de callback do then.

O argumento passado na função reject da promisse é recebido pela função de callback do catch.

fetch("<http://localhosttt:3333/notes>").then(res => res.json()).then(json => console.log(json)).catch(e => console.log(e))

console.log("algumas coisa"); //este código ainda é assíncrono

**Resolvendo promessas com async await**

O argumento passado no resolve da promisse é atríbuido à variável após o await.

async function buscarNotasAPI(){

const response = await fetch("<http://localhost:3333/notes>");

const json = await response.json();

console.log(json); // este código é síncrono

}

Para capturar o erro de uma promessa com async await, pode-se usar o catch ou um try/catch

async function buscarNotasAPI(){

try{

const response = await fetch("<http://localhost:3333/notes>");

const json = await response.json();

console.log(json); // este código é síncrono

}catch(error){

console.error(error);

}

}