Por: Alice Dantas

Promise em JS

`Promise` (com P maiúsculo) é uma **classe** nativa do JavaScript, quando você usa `new Promise(...)`, está **criando um objeto** dessa classe. Esse objeto representa **algo que vai acontecer no futuro** (pode dar certo ou dar erro).

Uma *promisse* **recebe como atributo uma função executora**. Essa função executora, por sua vez, será uma **arrow function de dois parâmetros**, os quais serão, respectivamente, **um para caso a operação interna termine com sucesso**, e **outro para caso a operação interna falhe**.

```
const promessa = new Promise((resolve, reject) => {
  // algum processamento demorado...
  const sucesso = true; //simulando um resultado positivo

  if (sucesso) {
    resolve("Deu tudo certo!");
  } else {
    reject("Algo deu errado...");
  }
});
```

Métodos de uma Promisse

- O método **`.then()`** é usado para **pegar o resultado** de uma *Promise* **quando ela for resolvida com sucesso**. Ou seja, o `.then()` **espera a promessa terminar** e, quando terminar, ele **executa uma função callback com o resultado**.

```
const promessa = new Promise((resolve) => {
  setTimeout(() => {
    resolve("Tarefa concluída!");
  }, 2000);
});

promessa.then((resultado) => {
  console.log(resultado); // "Tarefa concluída!" (após 2 segundos)
});
```

// A `Promise` leva 2 segundos para ser resolvida. Quando o *`resolve("Tarefa concluída!")`* é chamado, o *`.then()`* é ativado e executa o *`console.log(resultado)`*.

- Quando ocorre um erro ou a promise é rejeitada, usamos o **`.catch()`**:

```
const promessa = new Promise((resolve, reject) => {
  const sucesso = false; //simulando um resultado negativo

  if (sucesso) {
    resolve("Deu certo!");
  } else {
    reject("Erro!");
  }
});

promessa
  .then((resultado) => console.log(resultado))
  .catch((erro) => console.log("Falha:", erro));
```

- Os principais métodos:

Método	O que faz
`.then()`	Executa quando a Promise é **resolvida com sucesso**
`.catch()`	Executa quando a Promise é **rejeitada (erro)**
`.finally()`	Executa **sempre**, independente do resultado

Por: Alice Dantas

Funções que retornam uma Promise

Função Assíncrona (async / await)

Funções assíncronas sempre retornam uma **Promise**. Usamos `await` dentro dela para esperar o **resultado de promessas**. Sem `await`, a função retorna uma Promise imediatamente.

```
async function pegarDados() {
    let resposta = await fetch("https://api.exemplo.com/dados");
    let dados = await resposta.json();
    return dados;
}

pegarDados().then(dados => console.log(dados));
```

Função fetch

`fetch()` é uma função nativa do JavaScript usada para **fazer requisições HTTP** (GET, POST, PUT, DELETE...) e **buscar dados de servidores ou APIs**. Ela sempre retorna uma **Promise**, porque a resposta não chega na hora — depende da internet/servidor. Estrutura básica:

```
fetch("URL")
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.log("Erro:", error));
```

`fetch("URL")` → envia uma requisição HTTP para a URL especificada e **sempre retorna uma *Promise***. A requisição padrão é **GET**, por isso, nesse caso, não precisa informar o método. Quando a Promise do fetch é resolvida, ela retorna um **objeto Response**.

`.then()` → **acessa os resultados de uma *Promise***. Nesse caso, vai acessar o objeto Response. Então, ele **passa o resultado dessa *Promisse* como argumento para uma função**.

`response.json()` → Lê o corpo da resposta, converte para JSON e retorna outra Promise (por isso precisamos de outro `.then()`).

`data` → dados prontos para usar.

`catch()` → trata erros.