



JavaScript Assíncrono

Stephany Nusch
Software Engineer @ QuintoAndar

Objetivo do Curso



1. Explicar o conceito de assincronicidade
2. Apresentar Promises
3. Ensinar como fazer requisições de APIs

**Aula 1**

Assincronicidade

Aula 2

Consumindo APIs

Aula 3

Atividade Prática



< Aula 1: Assincronicidade >

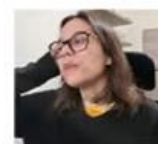
Javascript Assíncrono

Objetivos



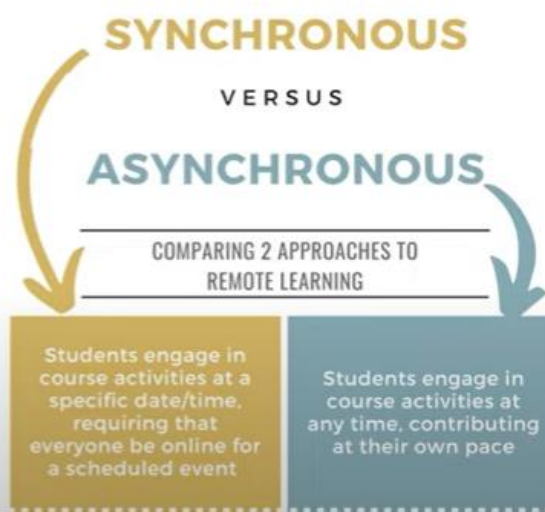
1. Explicar o que são Promises
2. Ensinar como manipulá-las
3. Apresentar as palavras-chave “async” e “await”

Definição



Assíncrono

“Que não ocorre ou não se efetiva ao mesmo tempo.”

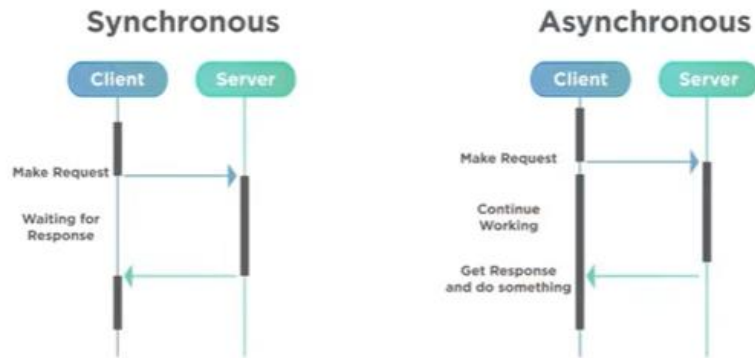




Definição



O Javascript roda de maneira **síncrona**.



Aula 1 | Etapa 2:



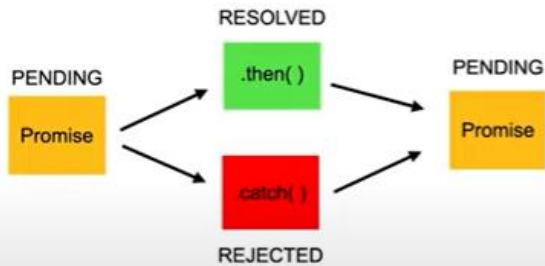
Promises

Javascript Assíncrono



Promises

Objeto de processamento assíncrono

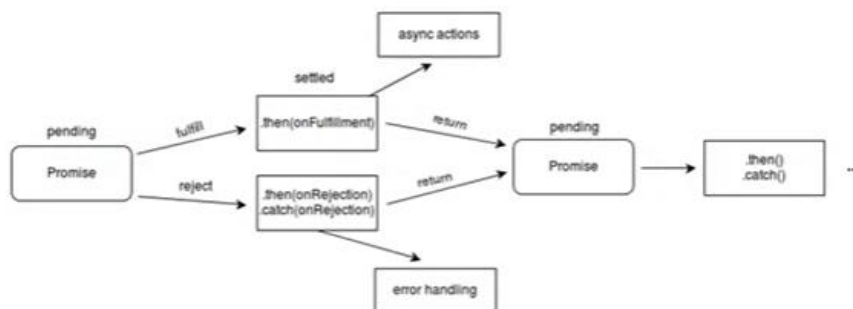


Inicialmente, seu valor é desconhecido. Ela pode, então, ser **resolvida** ou **rejeitada**.

Promises

Uma promise pode ter 3 estados

1) Pending 2) Fulfilled 3) Rejected





Promises

Estrutura



```
const myPromise = new Promise((resolve, reject) => {  
  window.setTimeout(() => {  
    resolve(console.log('Resolvida!'));  
  }, 2000);  
});
```



Promises

Manipulação



```
const myPromise = new Promise((resolve, reject) => {  
  window.setTimeout(() => {  
    resolve('Resolvida');  
  }, 2000);  
});  
  
await myPromise  
  .then((result) => result + ' passando pelo then')  
  .then((result) => result + ' e agora acabou!')  
  .catch((err) => console.log(err.message));  
  
// Após 2 segundos, retornará o valor  
// "Resolvida passando pelo then e agora acabou!"
```



Aula 1 | Etapa 3:

< Async/await >

Javascript Assíncrono



Async/await



Funções assíncronas precisam dessas duas palavras chave.

```
async function resolvePromise() {  
  const myPromise = new Promise((resolve, reject) => {  
    window.setTimeout(() => {  
      resolve('Resolvida');  
    }, 3000);  
  });  
  
  const resolved = await myPromise  
    .then((result) => result + ' passando pelo then')  
    .then((result) => result + ' e agora acabou!')  
    .catch((err) => console.log(err.message));  
  
  return resolved;  
}
```




Async/await

Funções assíncronas também retornam Promises!



```
async function resolvePromise() {
  const myPromise = new Promise((resolve, reject) => {
    window.setTimeout(() => {
      resolve('Resolvida');
    }, 3000);
  });

  const resolved = await myPromise
    .then((result) => result + ' passando pelo then')
    .then((result) => result + ' e agora acabou!')
    .catch((err) => console.log(err.message));

  return resolved;
}
```

```
> resolvePromise()
< Promise {<pending>}
> await resolvePromise()
< "Resolvida passando pelo then e agora acabou!"
```



Async/await

Utilizando try...catch



```
async function resolvePromise() {
  const myPromise = new Promise((resolve, reject) => {
    window.setTimeout(() => {
      resolve('Resolvida');
    }, 3000);
  });

  let result;

  try {
    result = await myPromise
      .then((result) => result + ' passando pelo then')
      .then((result) => result + ' e agora acabou!')
  } catch(err) {
    result = err.message;
  }

  return result;
}
```