

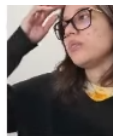
## Aula 1 | Etapa 1:

# Definição

## Javascript Assíncrono



## Definição



### Assíncrono

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

### SYNCHRONOUS

VERSUS

### ASYNCHRONOUS

COMPARING 2 APPROACHES TO  
REMOTE LEARNING

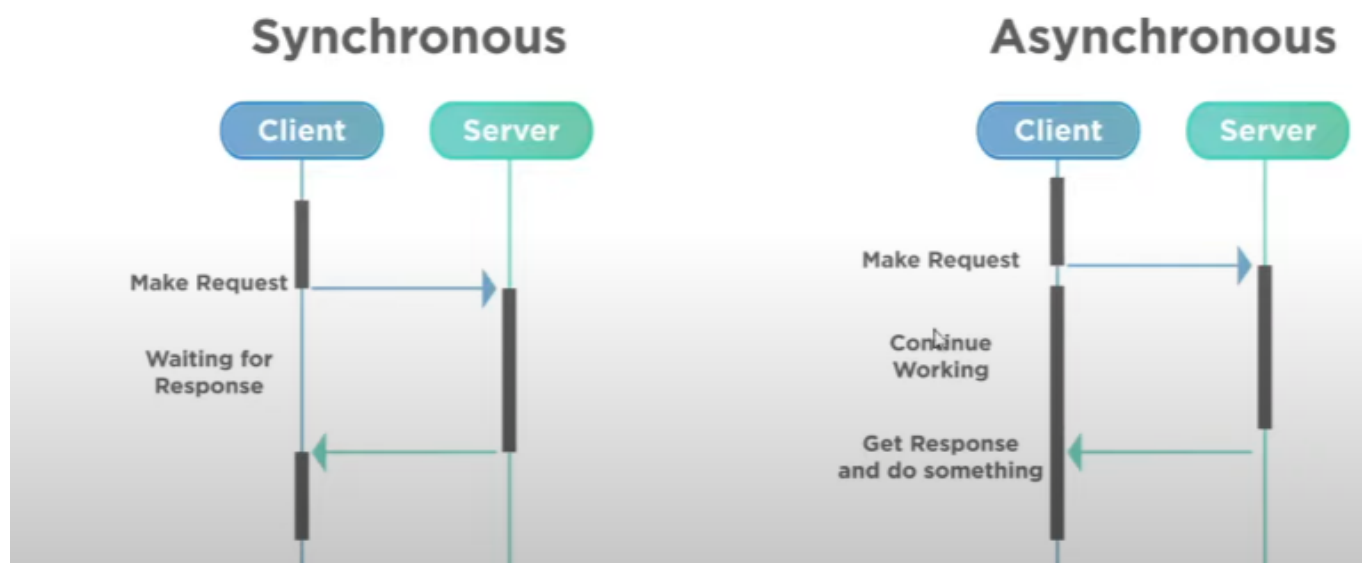
Students engage in course activities at a specific date/time, requiring that everyone be online for a scheduled event

Students engage in course activities at any time, contributing at their own pace

Síncrono é no mesmo tempo

# Definição

O Javascript roda de maneira **síncrona**.



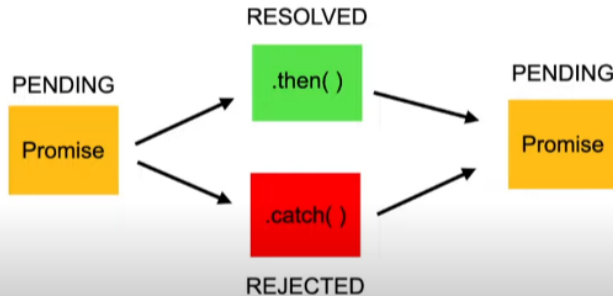
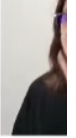
Aula 1 | Etapa 2:

# Promises

## Javascript Assíncrono

# Promises

Objeto de processamento assíncrono



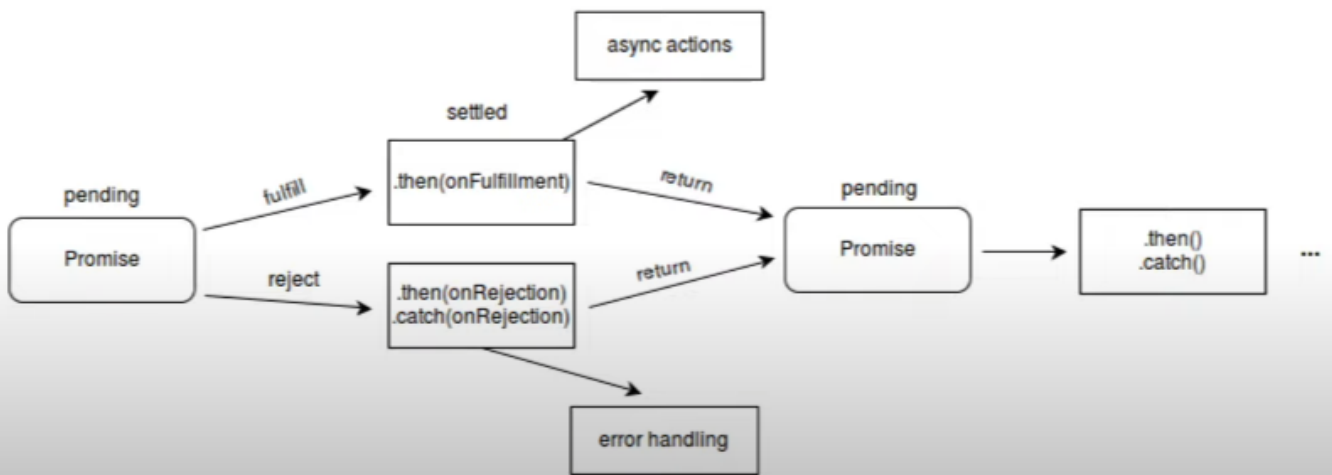
Inicialmente, seu valor é desconhecido. Ela pode, então, ser **resolvida** ou **rejeitada**.

ION

# Promises

Uma promise pode ter 3 estados

**1) Pending    2) Fulfilled    3) Rejected**



ela sempre está pendente e depois de um tempo pode ser rejeitada ou completada

# Promises

## Estrutura

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

depois de 2 segundo vai aparecer Resolvida

# 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

AL  
VATION

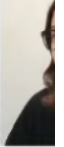
## Async/await

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;  
}
```

funções assíncronas precisa dessa palavra chave para que possa usar a palavra await

# 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!"
```

Quando você estiver fazendo uma função assíncrona não se pode chamá-la é preciso dar await nela também

# 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;  
}
```