Async/await

DigitalHouse >
Coding School

Índice

then() vs async/await

As promises vieram de início para evitar o chamado **callback hell**.
Entretanto, logo se percebeu que, apesar de ajudar bastante um código com alta quantidade de **then()** encadeado, se tornava difícil de ler e sua manutenção não era das melhores.

Para resolver essa situação, foi criado no **ES8** (ou ES2017) as funções **async**.
No próximo slide, vamos comparar as duas sintaxes.

then() vs async/await

Aqui temos uma comparação da nossa promise sendo resolvida com **async function** e da forma **tradicional**.

```
{}
const resolvePromise = async() => {
    try{
        const resultado = await minhaPromise
        console.log(resultado)
    }
    catch(err) {
        console.log(err)
    }
}
resolvePromise()
```

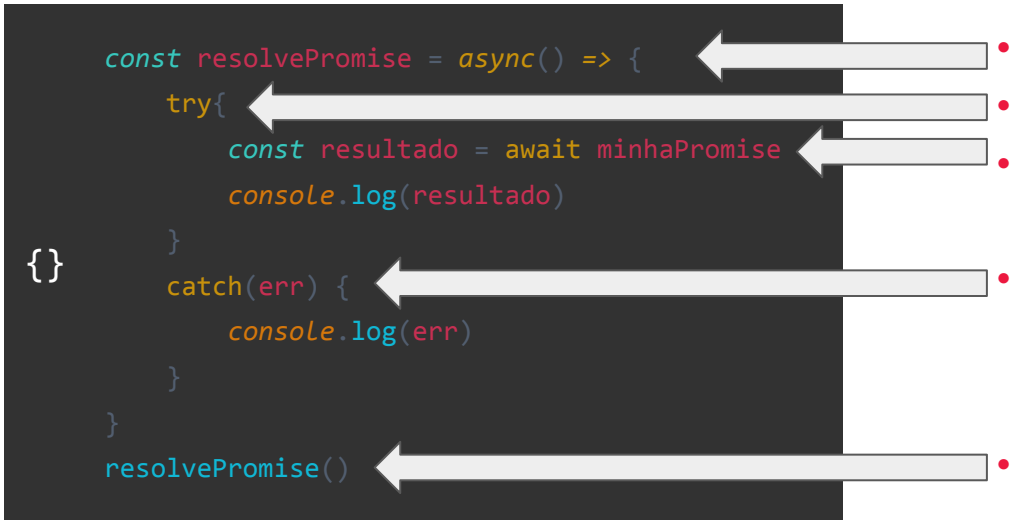
```
{}
minhaPromise
    .then(res => console.log(res))
    .catch(err => console.log(err))
```

then() vs async/await

Apesar de parecer **mais verboso**, o formato com **async function** é mais fácil de se manter, uma vez que consegue resolver promises **sem** necessidade de **concatenações**, tornando o código mais limpo.

Observando os detalhes

Agora vamos dissecar a async function para entender melhor a lógica!



- O código deve estar dentro de uma função marcada como **async**.
- O bloco de fluxo normal deve estar envolto em um **try**.
- A chamada à promise deve estar marcada como **await**.
- Todo bloco try precisa de um bloco **catch**.
- Como o código está em uma função, você deve chamar a mesma para que o código seja executado.


DigitalHouse>
Coding School