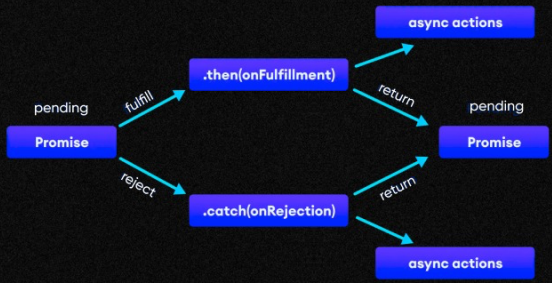

JS
Promise
Chaining
in JavaScript
codewithsloba.com
Swipe

JS

Promise chaining is a technique for handling asynchronous operations in a sequential manner using .then() and .catch() methods.

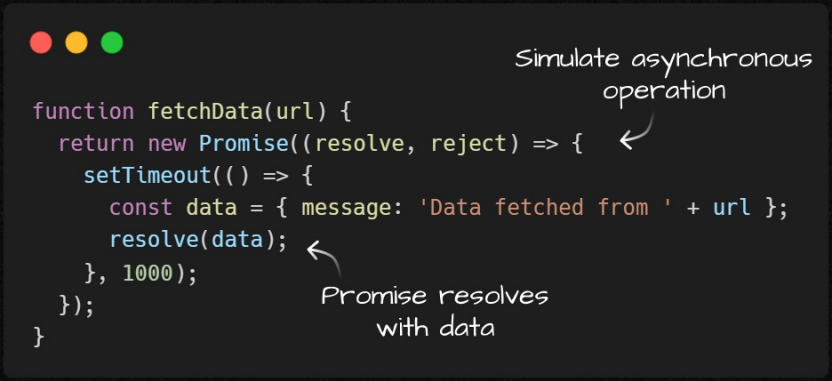


It allows you to create a chain of promises, where the result of one promise is passed to the next as input.

This helps avoid callback hell, a situation where nested callbacks become difficult to read and manage.

JS

The **fetchData** function returns a promise that resolves with data after a simulated delay.



The first `.then()` attaches a callback to the **fetchData** promise. This callback receives the resolved data (**data**).

JS

Inside the callback, we call processData(data), which also returns a promise.
The second .then() attaches another callback to the processData promise. This callback receives the processed data (processedData).

```
function processData(data) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const processedData = data.message.toUpperCase();
      resolve(processedData);
    }, 500);
  });
}

fetchData('https://api.example.com/data')
  .then(data => processData(data))
  .then(processedData => console.log(processedData))
  .catch(error => console.error(error));
```

Simulate processing

Promise resolves with processed data

Handle errors

Inside the second callback, we log the uppercase processed data.
The **`.catch()`** method is used to handle any errors that might occur in the promise chain.


Subscribe on YouTube to learn more
Code with Sloba
@CodewithSloba • 21.8K subscribers
Hi, my name is Slobodan (Sloba) Gajic, and I'm a JavaScript software developer. Building a
Subscribed