PROMISES IN JAVASCRIPT

@oluwakemioluwadahunsi

WHAT IS A PROMISE?

- A Promise is an object representing the eventual completion or failure of an asynchronous operation.

- Promises have three states: **pending**, **fulfilled**, and **rejected**.

@oluwakemioluwadahunsi

CREATING A PROMISE

```
const myPromise = new Promise((resolve, reject) => {
    let success = true; // Simulate an operation

    if (success) {
        resolve('Operation was successful!');
    } else {
        reject('Operation failed.');
    }
});
```

- In this example, **myPromise** will either resolve or reject based on the success condition.

@oluwakemioluwadahunsi

HANDLING PROMISES

- We use .then() to handle a resolved promise and .catch() for a rejected promise.

```
myPromise
    .then((message) => {
        console.log(message); // Logs: 'Operation was successful!'
    })
    .catch((error) => {
        console.error(error); // Logs: 'Operation failed.'
    });
```

- .then() is called if the promise resolves, and .catch() is called if it rejects.

@oluwakemioluwadahunsi

CHAINING PROMISES

- Promises can be chained to handle a sequence of asynchronous operations.

```
fetch('https://api.chaining.com/data')
    .then(response => response.json())
    .then(data => {
        console.log(data); // Handle the data from the API
    })
    .catch(error => {
        console.error('Error fetching data:', error);
    });
```

- Each .then() returns a new promise, allowing chaining.

@oluwakemioluwadahunsi

BETTER SYNTAX?

- Using ‘async’ and ‘await’ provide a cleaner syntax for working with Promises.

For example:

```
async function fetchData() {
    try {
        const response = await fetch('https://api.example.com/data');
        const data = await response.json();
        console.log(data);
    } catch (error) {
        console.error('Error fetching data:', error);
    }
}

fetchData();
```

- The await keyword pauses the function execution until the promise settles.

@oluwakemioluwadahunsi

IN SUMMARY

- Promises simplify handling asynchronous operations in JavaScript
- Use .then() and .catch() for promise handling and async/await for cleaner syntax.
- Understanding promises helps in writing efficient and maintainable asynchronous code.

@oluwakemioluwadahunsi

HI THERE!

I'm Oluwakemi Oluwadahunsi, a Frontend Software Developer. I enjoy sharing knowledge and helping others on their coding journey.

Did you enjoy this knowledge?

💼 Follow my LinkedIn page for more Coding tips, tutorials, and industry insights.

LinkedIn: Oluwakemi Oluwadahunsi

@oluwakemioluwadahunsi