Better Stack

JS

How to use JavaScript to HANDLE ERRORS

1

Async Code

The code below isn’t handling errors causing an **Unhandled Promise Rejection** error.

```
async function fetchData() {
    const response = await fetch('https://api.example.com/data');
    const data = await response.json();
    return data.json();
}
```

2

Try Catch

Adding a try catch block can prevent unhandled promise rejections.

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

3

Split Response & Network Errors

Using response.ok can handle 4xx & 5xx errors and the catch can handle timeout & CORS errors.

```
try {
    const response = await fetch('https://api.example.com/data');
    if (!response.ok) {
        throw new Error('Network response was not ok');
    }
    const data = await response.json();
    return data.json();
}
```

4

Specific Errors

You can use **instanceof** to handle specific error types differently.

```
try {
  // ...
} catch (error) {
    if (error instanceof TypeError) {
        console.error('TypeError', error);
        return;
    }
    console.error('Error fetching data:', error);
}
```

Better Stack

Want to Remember this for Later?

Tap Save