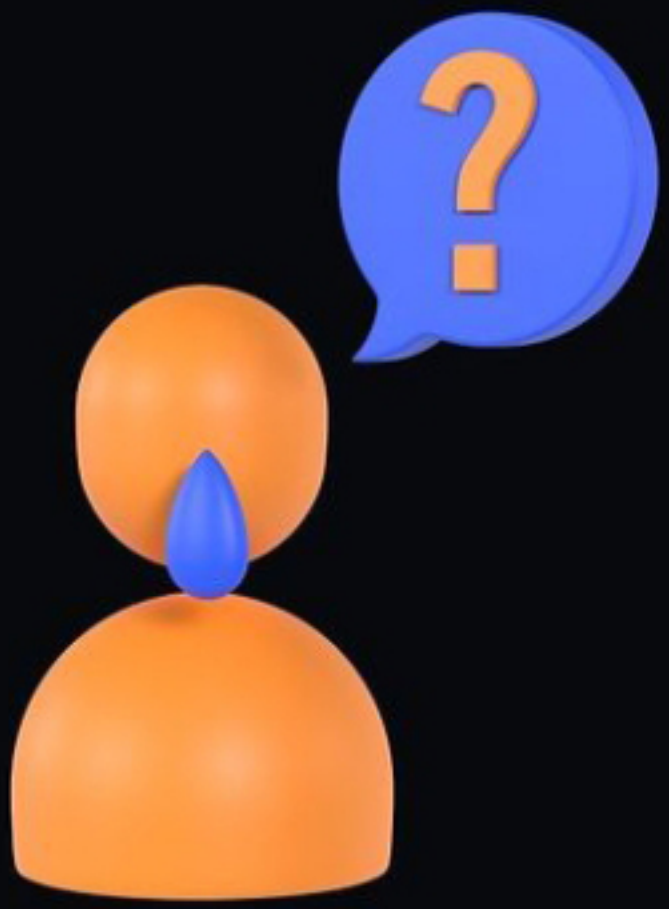
deeecode

PROMISES IN JS

JS



swipe left

deeecode

JS

PROMISES IN JS

What is a promise?

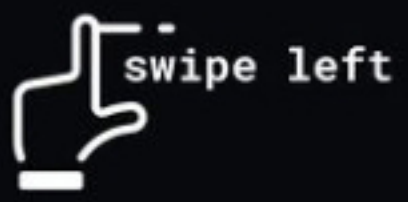
A promise is something that doesn't happen immediately but **MAY** happen in the future.

Emphasis on "**MAY**". This means it may happen (**promise fulfilled**) or it may not (**promise rejected**)

In JavaScript, promises are used to handle asynchronous events in JavaScript.

According to Wikipedia,

A promise acts as a proxy for a result that is initially unknown, usually because the computation of its value is not yet complete.



deeecode

JS

PROMISES IN JS

There are times when programming whereby some data would not be gotten immediately. This can be API requests, or some asynchronous operation. JavaScript uses promises to handle such situations.

A promise has four states (using API request for example):

- **pending**: the promise is processing (request has been sent to the server)
- **fulfilled**: the promise is successful (data gotten successfully from server)
- **rejected**: the promise fails (maybe the server crashed or 404 error)
- **settled**: a decision has been made (this can be success or failed, but not pending)

swipe left

deeecode

JS

PROMISES IN JS

You can create your own promise with the **Promise** constructor like this:

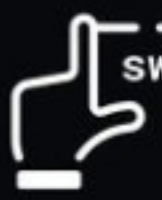
```
const score = 120

const promise = new Promise((resolve, reject) => {
  if (score > 100) {
    resolve("Hurray, you got more than 100!")
  } else {
    reject("Sorry, low score!")
  }
})

promise
  .then(value => {
    console.log(value)
  })
  .catch(error => {
    console.log(error)
  })

// "Hurray, you got more than 100!""
```

A promise has the **resolve** and **reject** arguments. The resolve method handles the promise if successful while the reject method handles the promise if rejected. The **then** function handles for success and the **catch** for errors.



swipe left

deeecode

JS

PROMISES IN JS

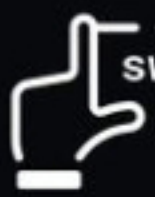
Most times, you don't have to create your own promise. **fetch** for example create promises internally and you can use them like this:

```
function result() {
  return fetch("https://backend.com/api")
}

result()
  .then(value => console.log(value))
  .catch(error => console.log(error))
```

As you can see in this example, the fetch expression returns an promise in the result function. Then you can use the then and catch method to handle success and failed promises respectively.

Instead of handling promises like this, you can also use **async/await** but that's a post for another day.



JS

Did you find this helpful?

Kindly **LIKE**, **SAVE** &
SHARE this post

Follow me **@deeecode** for
HTML / CSS / React / Web
and more **JS Tips**.

@deeecode