Javascript Promises

Sudeep Shetty

A JavaScript Promise is an object that represents a future value. Promises play an important role in JavaScript in managing asynchronous operations such as getting data from the server, reading data or executing deferred operations. They provide more integration and optimization than the callback-centric approach to working with asynchronous code.
Promises have three states:

1. **Pending:** The initial state, representing the promise that hasn't been fulfilled or rejected yet.
2. **Fulfilled:** The state where the promise has successfully resolved with a value.
3. **Rejected:** The state where the promise has encountered an error or has been explicitly rejected with a reason.

- **How to create a promise**

```
const myPromise = new Promise((resolve,reject)=>{
        //some code
})
```

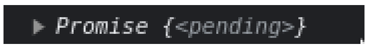
- **How to resolve a promise**

When we call a promise, it will be in pending unless it is resolved or rejected

Example

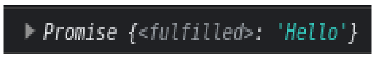
```
const myPromise = new Promise((resolve,reject)=>{
  setTimeout(()=>{
    resolve("Hello");
  },1000)
})
console.log(myPromise)
```

Then the output would be



```
▶ Promise {<pending>}
```

But if we do console.log(myPromise) after 1sec then the output would be this, because the promise got resolved after 1sec



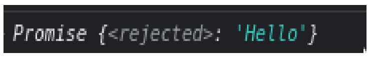
```
▶ Promise {<fulfilled>: 'Hello'}
```

- **How to reject a promise**

Now, The same goes for reject, if in the above code we change reject with resolve

```
const myPromise = new Promise((resolve,reject)=>{
  setTimeout(()=>{
    reject("Hello");
  },1000)
})
```

And then we console.log(myPromise) after 1 sec, we would get this output



```
Promise {<rejected>: 'Hello'}
```