Introduction to Promises Part 1

24 June 2023 22:53

Notes by: Mohini Mahato

Introduction to Promises

it's native to JS

↳ Coding pattern / Readibility Enhancer

↳ Readibility Enhancer

↳ Can solve the problem of Inversion of Control

↳ In JS promises are special types of object that get returns immediately when we call them

↳ Promises act as a placeholder for the data we hope to get back sometimes in future

for the result we expect

↳ In the promise objects we can attach the functionality we want to execute once the future task is done.

↳ Once the future task is done, promises will automatically execute the attached functionality

↳ Promises have two property.

(i) State property

(ii) Value property

↳ Creation of promise object is sync in nature

↳ Any promise object is in one of the three mutually exclusive State

represents work in progress

(i) Pending – When we create a new promise this is the default state

(ii) fulfilled – if the operation is completed successfully

(iii) rejected – If ops not successful

from pending either → fulfilled

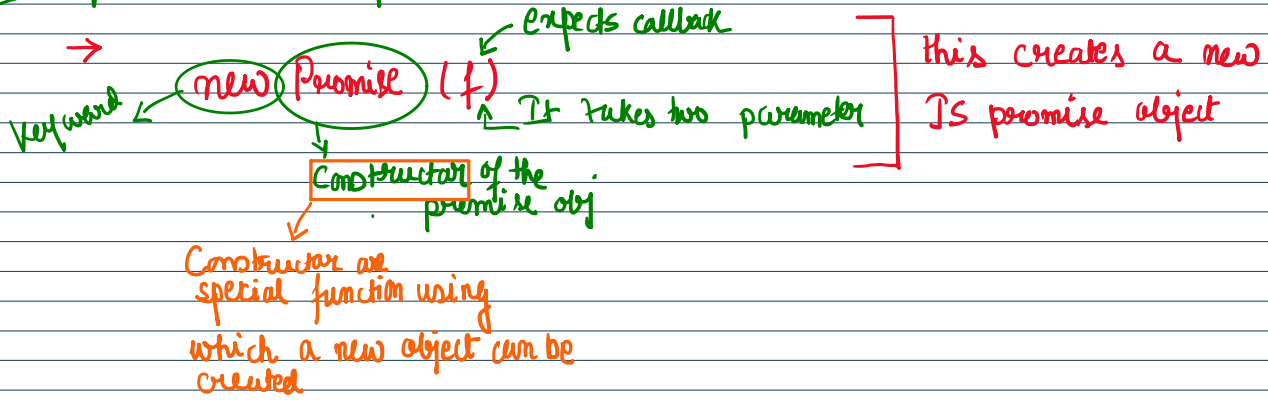
or → rejected

In order to understand promises extremely well we need to understand technically two things only

technically two things only

(i) How we can create a promise ?
(ii) How we can consume a promise ?

Q. How to create a promise ?




```
new Promise ( function (resolve, reject)) {
    // time consuming task
})
```

↳ Whenever in the implementation of executor callback :
(a) We call the resolve function, the promise goes to a fulfilled state
(b) If we call the reject function, it goes to a rejected state
(c) And if we don't call anything then promise remains in the pending state
↳ value: undefined

↳ With whatever argument we call resolve or reject with it gets assigned to the value property

```
function createPromise() {
    return new promise( function exec()(resolve, reject){
        for (let i=0 ; i< 1000000000 ; i++) {
            // some task
        }
        let num = getRandomInt(10) ;
        if (num % 2 == 0) {
            // some task
```

```
    if (num % 2 == 0) {
        // some task
    }
    else {
        // some task
    }
});
);
```

Quick Summary of promise constructor :

(i) At the time when the constructor generates a new promise object, it also generates a pair of function called resolve or reject

(ii) Generally the executor callback wraps some async / sync operation

(iii) The executor is called synchronously

Note

(i) The state of promise will only change if we call the resolve or reject func^n, if we don't call the resolve or reject function it's never going to change the state of promise.

(ii) Once we resolve or reject the promise object, it can never be updated again. We can resolve 10 times it doesn't matter first resolve will only be going to be effective.

— To be Continued...