Promises Basics, Promise.then() & Promise.catch() | JavaScript Tutorial In Hindi #39

Promises Basics, Promise.then() & Promise.catch() | JavaScript Tutorial In Hindi #39

Promises is a way through which we can deal with asynchronous operations in JavaScript. To understand this tutorial, check out my tutorial about ***JavaScript Callbacks***. As we know that **callback functions** were initially used to handle asynchronous operations. However, callbacks functions were limited in terms of functionality and often led to confusing code, so, **promises** were introduced to deal with these problems. Many people struggle with understanding how Promises work especially beginners, so in this tutorial, we will try to understand promises by making it as simple as we can. In this tutorial, you will find an introduction to what Promises are, explanations of terms like resolve, reject along with coding examples.

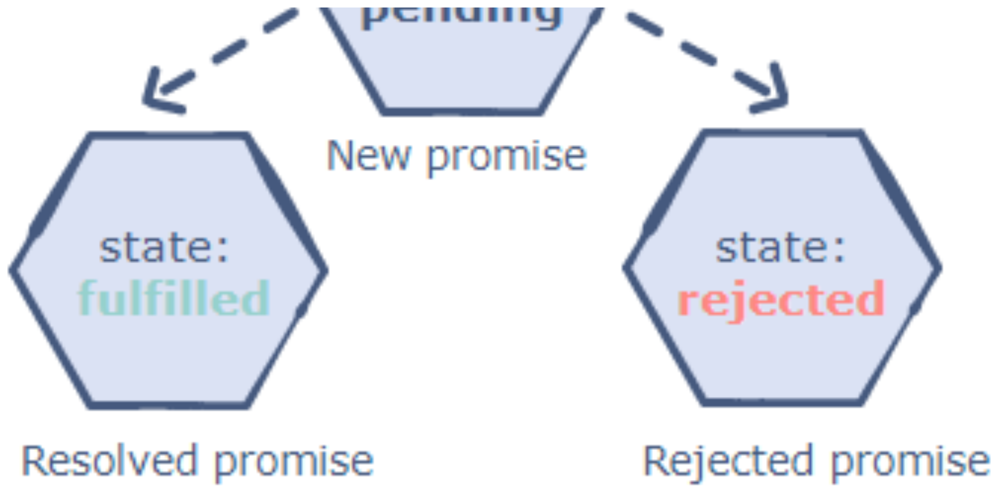
What is a Promise?

A promise in JavaScript is similar to a promises we do in real life. When we make a promise, it is a guarantee that in future, we are going to do something. A promise has two possible outcomes: it will be kept when the time comes, or it will not. Similarly, in JavaScript, when we define a promise, either it will be resolved when the time comes, or it will get rejected. According to MDN, “the Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.”

A promise has three states:

- **pending**: It is the initial state.
- **Fulfilled**: It indicates that the promised operation was successful.
- **Rejected**: It indicates that the promised operation was unsuccessful.

state:
pending



The constructor syntax for a promise object is:

```
let myPromise = new Promise(function(resolve, reject) {
// code here
});
```

When new Promise is created, the function passed into it runs automatically. It contains the producing code which produces the result. Its arguments resolve and reject. Here is an example of simple promise.

```
var promise = new Promise(function(resolve, reject) {
  const x = "geeksforgeeks";
  const y = "geeksforgeeks"
  if(x === y) {
    resolve();
  } else {
    reject();
  } });
promise.then(function(){
console.log('Success, You are a GEEK');}).catch(function () {
console.log('Some error has occurred');});
```

A promise is created using a constructor that takes a call back function with two arguments resolve and reject in line 1. If the task is successful(x===y), the promise is resolved. If the task is unsuccessful(x is not equal to y), then the promise is rejected. The then() method is called if the promise is resolved, and the catch() method is called when the promise is rejected or if an error occurred during the code execution.

Promises are the ideal choice for asynchronous programming. Promises can handle multiple asynchronous operations easily and are better at error handling as compared to callbacks and events.

Benefits of Promises:-

1. It improves the code readability
2. It is better in the handling of asynchronous operations
3. It has a better flow of control definition in asynchronous logic
4. It is better in error handling

Wrap Up:-

So, in this tutorial, we have learned how we create a Promise in JavaScript and use it for the resolved and rejected cases. If you are a beginner, understanding how promises they work might take time.

Code as described/written in the video

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iX
    <title>Document</title>
</head>
<body>
    <div class="container">
        <h1 id="heading" class='yourhead rhia is'> Welcome to Code With Harry</h1>
        <div id="myfirst" class="child red good" id="first">child 1

            <ul class="this" id='myul'>
                <li class="childul" id='fui'>this</li>
                <li class="childul">is</li>
                <li class="childul">a</li>
                <li class="childul">list </li>
                <li class="childul" id='lui'>of my dreams</li>
        </div>
```

```
            <div class="child">child 2</div>
            <div class="child red">child 3</div>
            <div class="child">child 4</div>
            <form action="none.html" method="post">
                <a href="//codewithharry.com">Go to Code With Harry</a>
                <br>
                <br>
                Search this website: <input type="text" name="Hello" id="">
                <button id="btn">Submit form</button>
                <!-- <input type="button" id='btn' value="submit"> -->
            </form>
        </div>
        <br>
        <div class="no">this is a dummy div1</div>
        <div class="no">this is a dummy div2</div>
        <div class="no">this is a dummy div3</div>
        <div class="container">
        <h1>Student list</h1>
        <ul id="students"></ul>

            <button id="myBtn" class="btn btn-primary">Your Button</button>
            <button class="btn btn-primary">Fetch Data</button>
            <div id="content"></div>
        </div>
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhz
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ
    </body>
    <!-- <script src="js/tut12.js"></script> -->
    <!-- <script src="js/tut14.js"></script> -->
    <!-- <script src="js/tut15.js"></script> -->
    <!-- <script src="js/tut16.js"></script> -->
    <!-- <script src="js/tut17.js"></script> -->
    <!-- <script src="js/tut18.js"></script> -->
    <!-- <script src="js/tut20.js"></script> -->
    <!-- <script src="js/tut21.js"></script> -->
    <!-- <script src="js/tut23.js"></script> -->
    <!-- <script src="js/tut24.js"></script> -->
```

```html
<!-- <script src="js/tut25.js"></script> -->
<!-- <script src="js/tut27.js"></script> -->
<!-- <script src="js/tut28.js"></script> -->
<!-- <script src="js/tut30.js"></script> -->
<!-- <script src="js/tut31.js"></script> -->
<!-- <script src="js/tut32.js"></script> -->
<!-- <script src="js/tut34.js"></script> -->
<!-- <script src="js/tut37.js"></script> -->
<!-- <script src="js/tut39.js"></script> -->
<!-- <script src="js/tut39b.js"></script> -->
<!-- <script src="js/tut41.js"></script> -->
<!-- <script src="js/tut43.js"></script> -->
<script src="js/tut44.js"></script>
</html>
```

Code as described/written in the video

```javascript
// Promise: Best video on promises
// -resolve
// -reject
// -pending

function func1() {
    return new Promise(function (resolve, reject) {
        setTimeout(() => {
            const error = true;
            if (!error) {
                console.log('Function: Your promise has been resolved')
                resolve();
            }
            else {
                console.log('Function: Your promise has not been resolved')
                reject('Sorry not fulfilled');
            }
        }, 2000);
    })
}
```

```
func1().then(function(){
    console.log("Harry: Thanks for resolving")
}).catch(function(error){
    console.log("Harry: Very bad bro. Reason: " + error)
})
```

Code tut 39b.js as described/written in the video

```
console.log("This is tutorial 37");

// Pretend that this response is coming from the server
const students = [
    { name: "harry", subject: "JavaScript" },
    { name: "Rohan", subject: "Machine Learning" }
]


function enrollStudent(student) {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            students.push(student);
            console.log("Student has been enrolled");
            const error = false;
            if (!error) {
                resolve();
            }
            else {
                reject();
            }
        }, 1000);
    })
}

function getStudents() {
    setTimeout(function () {
        let str = "";
        students.forEach(function (student) {
            str += `<li> ${student.name}</li>`
        });
        document.getElementById('students').innerHTML = str;
        console.log("Students have been fetched");
    }, 5000);
}

let newStudent = { name: "Sunny", subject: "Python" }
enrollStudent(newStudent).then(getStudents).catch(function () {
    console.log("Some error occured");
});
// getStudents();

// function inside then is ran as - resolve()
// function inside catch is ran as - reject()
```

Previous   Next

CodeWithHarry |