JavaScript Promise

Promises in real-life express a trust between two or more persons and an assurance that a particular thing will surely happen. In javascript, a Promise is an object which ensures to produce a single value in the future (when required). Promise in javascript is used for managing and tackling asynchronous operations.

Need for JavaScript Promise

Till now, we learned about events and callback functions for handling the data. But, its scope is limited. It is because events were not able to manage and operate asynchronous operations. Thus, Promise is the simplest and better approach for handling asynchronous operations efficiently.

There are two possible differences between Promise and Event Handlers:

1. A Promise can never fail or succeed twice or more. This can happen only once.
2. A Promise can neither switch from success to failure, or failure to success. If a Promise has either succeeded or failed, and after sometime, if any success/failure callback is added, the correct callback will be invoked, no matter the event happened earlier.

Terminology of Promise

A promise can be present in one of the following states:

1. **pending:** The pending promise is neither rejected nor fulfilled yet.
2. **fulfilled:** The related promise action is fulfilled successfully.
3. elated promise action is failed to be fulfilled.

4. **settled:** Either the action is fulfilled or rejected.

Thus, a promise represents the completion of an asynchronous operation with its result. It can be either successful completion of the promise, or its failure, but eventually completed. Promise uses a **then()** which is executed only after the completion of the promise resolve.

Promises of Promise

A JavaScript Promise promises that:

1. Unless the current execution of the js event loop is not completed (success or failure), callbacks will never be called before it.
2. Even if the callbacks with then() are present, but they will be called only after the execution of the asynchronous operations completely.
3. When multiple callbacks can be included by invoking then() many times, each of them will be executed in a chain, i.e., one after the other, following the sequence in which they were inserted.

Methods in Promise

The functions of Promise are executable almost on every trending web browsers such as Chrome, Mozilla, Opera, etc. The methods list is:

Method Name	Summary
Promise.resolve(promise)	This method returns promise only if promise.constructor==Promise.
Promise.resolve(thenable)	Makes a new promise from thenable containing then().
Promise.resolve(obj)	Makes a promise resolved for an object.
Promise.reject(obj)	Makes a promise rejection for the object.
Promise.all(array)	Makes a promise resolved when each item in an array is fulfilled or rejects when items in the array are not fulfilled.
Promise.race(array)	If any item in the array is fulfilled as soon, it resolves the promise, or if any item is rejected as soon, it rejects the promise.

Constructor in Promise

⇧ SCROLL TO TOP

new Promise(function(resolve, reject){});	Here, resolve(thenable) denotes that the promise will be resolved with then(). Resolve(obj) denotes promise will be fulfilled with the object Reject(obj) denotes promise rejected with the object.

Promise Implementation

```
<html>
<head>
<h2> Javascript Promise</h2>
</br> </head>
<body>
<script>
var p=new Promise(function(resolve, reject){
var x= 2+3;
if(x==5)
   resolve(" executed and resolved successfully");
else
   reject("rejected");
});
 p.then(function(fromResolve){
 document.write("Promise is"+fromResolve);
 }).catch(function(fromReject){
 document.write("Promise is "+fromReject);
 });
</script>
</body>
</html>
```

Test it Now

In the above Promise implementation, the Promise constructor takes an argument that callbacks the function. This callback function takes two arguments, i.e.,

1. **Resolve:** When the promise is executed successfully, the resolve argument is invoked, which provides the result.

ne promise is rejected, the reject argument is invoked, which results in an error.

It means either resolve is called or reject is called. Here, then() has taken one argument which will execute, if the promise is resolved. Otherwise, catch() will be called with the rejection of the promise.

Advantages of using Promises

1. A better option to deal with asynchronous operations.
2. Provides easy error handling and better code readability.

← Prev Next →