

Promises In Javascript

A promise is a javascript object that allows you to make asynchronous calls.

It produces a value when async operation completes successfully or produces an error if it doesn't complete.

You can create promise using constructor

```
let promise = new Promise(function(resolve, reject)
    ↑
    { } );
```

Executor function

Executor fn takes 2 arguments :-

- resolve — indicate successful completion
- reject — indicates an error

The Promise objects and states

The promise object should be capable of informing consumers when execution has been started, completed or returned with an error

1. State → pending - When execution fn starts
 - Fulfilled - When promise resolved successfully
 - Rejected - When the promise rejects
2. Result →
 - undefined - Initially when state value is pending
 - Value - When promise is resolved
 - Error - When the promise is rejected

A promise that is either resolved or rejected are settled

Handling Promises by Consumer

Three important handler methods

- then()
- catch()
- Finally

These methods helps us create a link between executor and consumer ↗

The .then() Promise Handler

It is used to let consumer know outcome of promise. It accept 2 arguments

- result
- error.

Eg - `promise.then (`
`(result) => {`
`console.log (result);`
`},`
`(error) => {`
`console.log (error);`
`};`

The catch Promise Handler

To handle errors (rejections) from promises.
It's better syntax to handle error than handling it with :then().

Eg \Rightarrow `Promise.catch (function (error) {`
`console.log (Error);`
`});`

The .finally () Promise Handler

The finally () handler method performs cleanups like stopping a loader, closing a live connection and so on

Irrespective of whether promise resolve or rejects, the finally () method will run

Eg - promise.finally (()) => {

```
    console.log ("Promise settled");
```

```
}).then ((result) => {
```

```
    console.log ({result});
```

```
});
```

Imp point to note,

the finally () method passes through result or error to the next handler

which can call a .then () or

.catch() again.