**What is promise?**

* A promise is simply an object in JS
* A promise is always one of the three states
  + Pending : which is initial state, neither fulfilled nor rejected.
  + full-filled: meaning that the operation completed successfully
  + rejected: meaning that the operation failed

**Why Promises:**

* Promises help us deal with asynchronous code in JS

**How to create a Promise, How to full-fill or reject, How to execute callback functions based on Promise value**

const promise = new Promise()

* How to fulfil or reject?
  + We need to pass a function to the Promise Constructor. That function is called the executor function(Remember, fetching the water?). The executor function takes two arguments, resolve and reject. These two are callback functions for the executor to announce an outcome.

let promise = new Promise(function(resolve, reject) {

// Do something and either resolve or reject

});

console.log(promise) // undefined

* The resolve method indicates successful completion of the task(got drinks), and the rejectmethod indicates an error(no drink). You do not implement the resolve/reject method. JavaScript provides it to you. You need to call them from the executor function.
* resolve and reject both are promises
* Resolve will be called when change of status from pending to full-filled.
* reject will be called when change of status from pending to rejected.

```jsx

const promise = new Promise((resolve,reject)=>{

//change of status from pending to fullfilled.

resolve("Got drink, prepare table")

})

console.log(promise) //value: When the promise is resolved(value).

```

const promise = new Promise((resolve,reject)=>{

//change of status from pending to rejected.

reject("No drink, prepare deserts")

})

console.log(promise) // error: When the promise is rejected.

```

## How to execute callback functions based on status change

const promise = new Promise((resolve,reject)=>{

//change of status from pending to fullfilled.

resolve("Got drink, prepare table")

})

const promise = new Promise((resolve,reject)=>{

//change of status from pending to rejected.

reject("No drink, prepare deserts")

})

promise.then()

promise.catch()

* Promise status : pending to fullfilled ? .then() is executed
* Promise status : pending to rejected ? .catch() is executed
* We get a .then()method from every promise. The sole purpose of this method is to let the consumer know about the outcome of a promise. It accepts two functions as arguments, resultand error.

const promise = new Promise((resolve,reject)=>{

//change of status from pending to fullfilled.

resolve("Brining Soft drinks")

})

const promise = new Promise((resolve,reject)=>{

//change of status from pending to rejected.

reject("Not brining Tacos")

})

promise.then((fromResolved)=>{

console.log(fromResolved)

})

promise.catch((fromReject)=>{

console.log(fromReject)

})

Example-1

let completedAssignment = new Promise((resolve, reject) => {

let isCompleted = true;

if (isCompleted) {

resolve("Completed assignment");

} else {

reject("Not completed assignment");

}

});

console.log(completedAssignment);

---------------------------------------------------------------------

let completedAssignment = new Promise((resolve, reject) => {

let isCompleted = false;

if (isCompleted) {

resolve("Completed assignment");

} else {

reject("Not completed assignment");

}

});

console.log(completedAssignment);

completedAssignment

.then((fromResolve) => console.log(fromResolve))

.catch((fromReject) => console.log(fromReject));

WE

const countValue = 2;

let promiseCount = new Promise(function (resolve, reject) {

if (countValue>0) {

resolve("The count value is positive.");

} else {

reject("Negative count value!");

}

});

console.log(promiseCount);

const countValue = 2;

let promiseCount = new Promise(function (resolve, reject) {

if (countValue>0) {

resolve("The count value is positive.");

} else {

reject("Negative count value!");

}

});

promiseCount

.then(response => console.log(response))

.catch(reason => console.log(reason))

function wait(t) {

return new Promise((res, rej) => {

setTimeout(() => res("Promise Resolved!"), t);

});

}

wait(5000).then((val) => console.log(val));

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Document</title>

</head>

<body>

<input type="number" />

<button onClick="checkOTP()">Submit</button>

</body>

</html>

<script>

function checkOTP() {

let otp = document.querySelector("input").value;

let otpPromise = new Promise((resolve, reject) => {

setTimeout(() => {

if (otp == 1234) {

resolve("correct OTP pin");

} else {

reject("incorrect OTP pin");

}

}, 2000);

});

otpPromise.then((val) => console.log(val)).catch((err) => console.log(err));

}

</script>