Promise

Tags	Tools
Created time	@April 23, 2023 11:12 PM
Last edited time	@April 23, 2023 11:15 PM

In JavaScript, a promise is an object representing the eventual completion or failure of an asynchronous operation and its resulting value. Promises provide a way to handle asynchronous operations without blocking the main thread of execution.

Promises have three states:

1. `Pending`: The initial state of a promise, which means that the asynchronous operation hasn't been completed yet.
2. `Fulfilled`: The state of a promise when the asynchronous operation has completed successfully, and the resulting value is available.
3. `Rejected`: The state of a promise when the asynchronous operation has failed, and an error is available.

Promises are created using the `Promise`

constructor, which takes a function with two parameters: `resolve` and `reject`. The `resolve` function is called when the asynchronous operation has completed successfully, and the `reject` function is called when the operation has failed.

Here's an example of creating and using a promise in JavaScript:

```
const myPromise = new Promise((resolve, reject) => {
  // asynchronous operation
  setTimeout(() => {
    const randomNumber = Math.floor(Math.random() * 10);
    if (randomNumber % 2 === 0) {
      resolve(randomNumber);
    } else {
      reject(new Error('Odd number generated'));
    }
  }, 1000);
});

myPromise
```

```
  .then(result => {
    console.log(`Promise fulfilled with result: ${result}`);
  })
  .catch(error => {
    console.error(`Promise rejected with error: ${error.message}`);
  });
```

`resolve` is not a function that takes the `randomNumber` as a parameter, rather it is a function that when called, changes the state of the Promise to `fulfilled` and sets the value of the promise to the provided value (`randomNumber` in this case).

So in the example I provided, if the `randomNumber` is even, the `resolve` function is called with `randomNumber` as its argument, which fulfills the promise with `randomNumber` as its value. On the other hand, if the `randomNumber` is odd, the `reject` function is called with a new `Error` object as its argument, which rejects the promise with the provided error.

In this example, if `randomNumber` is even, then the `resolve` function is called with `randomNumber` as its argument, which fulfills the promise with `randomNumber` as its value. When the promise is fulfilled, the `.then()` method is called with `result` as its argument, and the code inside the `.then()` block is executed. In this case, the output will be `Promise fulfilled with result: ${randomNumber}`.

If `randomNumber` is odd, then the `reject` function is called with a new `Error` object as its argument, which rejects the promise with the provided error. When the promise is rejected, the `.catch()` method is called with `error` as its argument, and the code inside the `.catch()` block is executed. In this case, the output will be `Promise rejected with error: Odd number generated`.