PROMISE IN JAVASCRIPT

PRAVIN VELUSAMY

JS Promise

Promise is an object that represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

It provides a cleaner and more efficient way to handle asynchronous code compared to traditional callback-based approaches.

PRAVIN VELUSAMY

Three Stages of Promise

Pending: The initial state when a Promise is created. The Promise is neither fulfilled nor rejected.

Fulfilled: The state when the asynchronous operation is completed successfully.

Rejected: The state when the asynchronous operation encounters an error or is rejected explicitly.

PRAVIN VELUSAMY

.then() & .catch ()

Inbuilt call back functions that allows you to handle the resolve and reject states of a promise respectively.

PRAVIN VELUSAMY

```
const promise= new Promise((resolve, reject)=>{
    const sum= 1 + 1;
    if(sum == 2){
        resolve("Success");
    }
    else{
        reject("Error")
    }
});

 promise
    .then((msg)=>{
    console.log(msg);
 })
    .catch((msg)=>{
        console.log(msg);
});
        // Output: Success
```

PRAVIN VELUSAMY

Promise.all()

The Promise.all() method accepts an iterable Object, such as an Array of promises as an input and returns a single promise that resolves to a result array of the input promises.

It rejects immediately when an input promise rejects or non-promise throws an error and will reject with that first rejection message / error message.

PRAVIN VELUSAMY

```
------------------------------

        Promise.all([Promise.resolve("Good"), Promise.resolve("Good-1"), Promise.reject("Good-2")])
        .then((msg)=>{
            console.log("then() :",msg)
        })
        .catch((msg)=>{
            console.error("catch() :",msg)
        });

        // Output: catch() : Good-2
```

PRAVIN VELUSAMY

```
Promise.all([Promise.resolve("Good"), Promise.resolve("Good-1"), Promise.resolve("Good-2")])
        .then((msg)=>{
            console.log(msg)
        })
        .catch((msg)=>{
            console.error(msg)
        });
            // output: ["Good","Good-1","Good-2"]
```

PRAVIN VELUSAMY

Promise.any()

The Promise.any() takes an iterable Object, such as an Array of promises as an input.
Once a promise is fulfilled, a single promise is returned and the promise is resolved using the value of the promise.
If no promises in the iterable fulfill (if all of the given promises are rejected), then the returned promise is rejected with an AggregateError (that groups together individual errors).
Unlike Promise.all(), this method is used to return the first promise that fulfills.

PRAVIN VELUSAMY

```
Promise.any([Promise.reject("Best"), Promise.reject("Best-1"), Promise.resolve("Best-2")])
.then((msg)=>{
    console.log(msg)
})
.catch((msg)=>{
    console.error(msg)
});
    // Output: Best-2
```

PRAVIN VELUSAMY

Promise.race()

The Promise.race() method returns a Promise that is resolved or rejected, as soon as one of the promises in an iterable, such as an array, fulfills or rejects, with the value or reason from that Promise.

The promise returned will be forever pending, if the iterable passed is empty.

Promise.race() will resolve to the first value found in the iterable, if the iterable contains one or more non-promise value or an already settled promise.

PRAVIN VELUSAMY

```
Promise.race([Promise.resolve("Well Done 1st"), Promise.resolve("Well Done 2nd"), Promise.reject("Well Done 3rd")])
.then((msg)=>{
    console.log(msg)
})
.catch((msg)=>{
    console.error(msg)
});
    // Output: Well Done 1st
```

PRAVIN VELUSAMY

```
Promise.race([Promise.reject("Race error 1"), Promise.resolve("Race error 2"), Promise.reject("Race error 3")])
.then((msg)=>{
    console.log(msg)
})
.catch((msg)=>{
    console.error(msg)
});
    // Output: Race error 1
```

PRAVIN VELUSAMY

Promise.allsettled()

The Promise.allSettled() method in JavaScript is used to get a promise when all inputs are settled that is either fulfilled or rejected.

It basically returns a promise that gets resolved when all other promises which are passed are either fulfilled or rejected and in output it displays the array of objects which particularly displays the status and the value of each promise individually.

PRAVIN VELUSAMY

```javascript
Promise.allSettled([Promise.resolve("All Settled 1st"), Promise.resolve("All Settled 2nd"), Promise.reject("All Settled 3rd")])
.then((msg)=>{
    console.log(msg)
})
.catch((msg)=>{
    console.error(msg)
});

/*     Output: [
     {
         "status": "fulfilled",
         "value": "All Settled 1st"
     },
     {
         "status": "fulfilled",
         "value": "All Settled 2nd"
     },
     {
         "status": "rejected",
         "reason": "All Settled 3rd"
     }
]
```

PRAVIN VELUSAMY

Promise.prototype.finally()

The finally() method returns a Promise.
When a Promise is completed, either resolved or rejected,this specified callback function is executed.

This helps to avoid duplicating code in both the promise's then() and catch() handlers

.

The finally() method will help if you're going to do any processing or cleanup work once a promise is made, regardless of the outcome.

PRAVIN VELUSAMY

```
const pro = Promise.resolve("Done");

pro.then((msg)=>{
    console.log(msg);
})
    .catch((err)=>{
    console.error(err);

    })
    .finally(()=>{
    console.log("All Work Completed");
    });
        // Done
        // All Work Completed
```