

## Promise

Tuesday, March 15, 2022 5:17 PM

### ES2015 Promises

- ▶ Promises are standard way to handle async programming
- ▶ It represents a long running task.
- ▶ Promise is a JavaScript Object
- ▶ it takes a callback function representing long running task.
- ▶ This call back in turn takes two more call back function
  - ▶ resolve
  - ▶ reject
- ▶ The long running task should
  - ▶ return result by calling resolve function
  - ▶ return error by calling reject function
- ▶ The function should return a Promise

This is not really a long running task. It is a scheduler for a long running task

```

function MyLongRunningTask ( param ) {

  const promise = new Promise (
    function( resolve, reject ) {
      //real long running task
      if( something_went_wrong)
        reject(error);
      //success
      resolve(result);
    }
  );

  return promise;
}

```

We return immediately with a Promise that we will eventually return a result

- The function which we call schedules a task that may long time to finish.
- This function creates a promise and returns immediately: other functions can do what they want
- When we return a promise
  - We are not returning result
  - We are promising that we will get the result eventually

- Promise is a future tense.
- You promise to return back something in future
- Right now you just have a token

## searchAirlines using Promise

```

15
16 function airlinesSearch(name, params){
17
18   const p = new Promise ( function(resolve, reject){
19
20     //the real long running task. should be scheduled for later
21     setTimeout(function(){
22       //do your logic here
23
24       let cities=["delhi","mumbai","bangalore","kolkatta","chennai","hyderabad","jaipur"]
25
26       let flights=[] ; //we should return 5 results.
27
28       if( !params)
29         return reject(new Error("No parameter"));
30
31       if(!params.from || !cities.includes(params.from))
32         return reject(new Error("Missing invalid from city:"+params.from));
33
34       if( !params.to || !cities.includes(params.to) )
35         return reject( new Error("Missing Invalid to city:"+params.to));
36
37       if(params.date < new Date())
38         return reject(new Error("Invalid Date."));
39
40       let result= {
41         airlines:name,
42         date: params.date,
43         flights:[]
44       };
45
46       --
47       let i=0;
48       let iid= setInterval(()=>{
49         result.flights.push(`${name.toUpperCase().substring(0,3)}${i+1}-${params.from.toLowerCase().substring(0,3)}`);
50         i++;
51         if(i==5){
52           clearInterval(iid);
53           return resolve(result); //result will be sent after 5 seconds
54         }
55       },1000);
56

```

Reject a promise

- Resolve the promise

```

57
58
59
60
61
62     }, 10) // do it after 10ms
63 });
64
65
66     return p;
67 }
68
69
70

```

## Handling a Promise

- When we call an async function we return a Promise
  - Result is not ready yet
- We can configure what to do on
  - Success
  - Failure

# ES2015 Promise Handling

```

function myLongRunningTask ( param ) {

    const promise = new Promise (
        function(resolve, reject ) {
            //real long running task
            if( something_went_wrong)
                reject(error);
            //success
            resolve(result);
        }
    );

    return promise;
}

```

```

//client.js

let promise = myLongRunningTask();

//returns immediately
//Now we setup future of the promise
promise
    .then ( result=> {
        //handle success
    }
    .catch( error =>{
        //handle error
    }
    );

```

A Promise is resolved immediately. It returns a result. It's a promise.

You are screen sharing Stop Share