**PROMISES**

**ASSIGNMENT-19**

1.What is the output of the code ?

function job() {

return new Promise(function(resolve, reject) {

**reject();**

});

}

let promise = job();

promise

.then(function() {

console.log('Success 1');

})

.then(function() {

console.log('Success 2');

})

.then(function() {

console.log('Success 3');

})

**.catch(function() {**

**console.log('Error 1');**

**})**

**.then(function() {**

**console.log('Success 4');**

**});**

**In the above program, return new Promise(){reject();} is called..so it wont execute promise part..if resolve is given then it will execute promise.then()part so the output here is**

**Output:**

Error 1

Success 4

2). Write a sleep function using a promise in javascript?

setTimeout ---- **defer** execution of a function for x milliseconds.

Example:

Java script does not have sleep function..but with the use of async() and await ..sleep function is created on custom basis.

const sleep=(delay)=>new Promise((resolve)=>setTimeout(resolve,delay))

const repeat =async()=>

{

await sleep(1000)

console.log("First")

await sleep(2000)

console.log("second")

await sleep(3000)

console.log("Third")

}

repeat(); //calling repeat function

//First will be printed after a sec…”second” will be printed after 2 second and Third will be printed after 3 seconds..this is done with the help of setTimeout.

**3). What is the output of the following code?**

const promise = new Promise(res => res(2));

promise.then(v => {

console.log(v); //2 is printed

return v \* 2; //v becomes 4 and this 4 is returned to next then

})

After executing above ,then value of v is 2, then v=> v\*2 so v--4

.then(v => { //4 is provided from previous block

console.log(v); //4 is printed

return v \* 2; //here 4\*2 –8 is returned as v to next block

})

.finally(v => {

console.log(v);

return v \* 2;

})

.then(v => {

console.log(v);

});

//above will get v value as 8 (not from finally)previous then block.

Output:

2

4

undefined

8

The finally block doesn’t receive any value and if any value is returned from the finally wont be considered,in the then block so the output from the last then is used. Any value printed from the finally will give us undefined value.

**4.What is the output of this code snippet?**

console.log('start')

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

console.log(1);

resolve('success')

}))

console.log('middle')

fn().then(res => {

    console.log(res)

    })

console.log('end')

output:

start

middle

1

End

Success

Synchronous function starts executing first so start is printed first then next synchronous function middle is executed next ..then anonymous function

Enters call stack and then next promise will enter microtask queue..this will be given preference so next 1 is printed then next synchronous enters event loop and print End.

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

console.log(1);

resolve('success')

}))

Next,

fn().then(res => {

    console.log(res)

this will enter call stack then res value ‘success’ is printed.

**5). Write a function delay that returns a promise. And that promise**

**should return a setTimeout that calls resolve after 1000ms.**

const sleep=(delay)=>new Promise((resolve)=>setTimeout(resolve,delay))

const repeat =async()=>

{

//await sleep(1000)

console.log("First")

await sleep(1000)

console.log("second")

await sleep(3000)

console.log("Third")

}

repeat();

Another simple example:

function delay (time) {

  return new Promise((resolve) => setTimeout(resolve, time));

}

delay(1000).then(() => {

    // Do something after the delay

    console.log("This will execute after 1000ms specified in the delay function")

});

**6.What will the output be?**

Promise.resolve('Success!')

.then(data => {

return data.toUpperCase()

})

.then(data => {

console.log(data)

})

Output:

SUCCESS!

After resolving promise..then data is returned as success in uppercase..then when printing data it will print SUCCESS!

**7. What will the output be?**

var p = new Promise((resolve, reject) => {

reject(Error('The Fails!'))

})

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

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

Output:

Error: The Fails!

In the above program,reject holds a error message..then .then will be executed so the error The fails will be printed along with error messages in the console.

**8.What will the output be?**

console.log('start')

setTimeout(() => {

console.log('setTimeout')

})

Promise.resolve().then(() => {

console.log('resolve')

})

console.log('end')

output:

start

end

resolve

setTimeout

first execute in synchronous manner: start then setTimeout enters task queue and promise is there…it enters microtask queue so next synchronously end will be executed…after that promise will be given preference so resolve will be executed and then setTimeout function will be executed.

**10. What will the output be?**

console.log('start')

Promise.resolve(1).then((res) => {

console.log(res)

})

Promise.resolve(2).then((res) => {

console.log(res)

})

console.log('end')

output:

start

end

1

2

In order to execute synchronously, start will be printed then end will be printed..then promise.resolve(1), enters the microtask queue..and next execution is also a microtask queue…microtask queue follows First in first out..so according to sequence promise.resolve(1) will executed first then last promise.resolve(2)prints 2.