Mohan Sai Potla

29 Oct 2022

JavaScript Assignment 19

1.

```
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');
    });
```

What is the output of the code above ?

Output:

Error 1

Success 4

This is because the catch has responsibility only for then methods written before it. The then methods after the catch will be called no matter what happens. Therefore, after “Error 1”, “Success 4” also printed in the output.

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

```
//sleep function using a promise
function sleep(time) {
    return new Promise(resolve => setTimeout(resolve, time));
};

//driver code
console.log("start");
sleep(3000);
```

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

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

promise
.then(v => {
    console.log(v);
    return v * 2;
})
.then(v => {
    console.log(v);
    return v * 2;
})
.finally(v => {
    console.log(v);
    return v * 2;
})
.then(v => {
    console.log(v);
});
```

Output:

2

4

undefined

8

Finally method receives no parameters. That’s why it prints undefined.

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

5). Write a function delay that returns a promise. And that promise should return a setTimeout that calls resolve after 1000ms.

```
//delay function
function delay() {
    return new Promise(resolve => {
        setTimeout(resolve, 1000)
    });
};

delay();
```

6). What will the output be?

```
Promise
    .resolve('Success!')
    .then(data => {
        return data.toUpperCase()
    })
    .then(data => {
        console.log(data)
    })
```

Output:

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!

Undefined

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

This is because promises have highest priority then setTimeout functions.

9). 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 the given code only promises are there. Promises had given the equal priority in JavaScript. Therefore, first promise executes first and then second.