



PROMISES

Stage1

```
1. let p = new Promise(()=>{});  
2. console.log(p);
```

Output

```
1. Promise { <pending> }
```

Stage2

```
1. let p = new Promise(e => e(3));  
2. console.log(p);
```

Output

```
1. Promise { 3 }
```

Stage3

```
1. let p = new Promise((e,f)=> f(3));  
2. console.log(p);
```

Output

```
1. Promise { <rejected> 3 }
```

Stage4

```
1. let p = new Promise((e,f)=> {e(3);f(-3);e(8);});  
2. console.log(p);
```



 out

```
1. Promise { 3 }
```

Stage 5

```
1. let p = new Promise(  
2.   (e, f) => {  
3.     console.log("Entering Executor");  
4.     f(-3);  
5.     console.log("Leaving Executor");  
6.   }  
7. );  
8. console.log(p);  
9. let q = p.catch(e => {  
10.   console.log("In Catch");  
11.   console.log(e)  
12.   console.log("Leaving catch");  
13. });  
14. console.log(p);  
15. console.log(q);  
16. setImmediate(() => {  
17.   console.log("Entering setImmediate");  
18.   console.log("Leaving SetImmediate");  
19.   console.log("q is ", q);  
20. })
```



 out

1. Entering Executor
2. Leaving Executor
3. Promise { <rejected> -3 }
4. Promise { <rejected> -3 }
5. Promise { <pending> }
6. In Catch
7. -3
8. Leaving catch
9. Entering setImmediate
10. Leaving SetImmediate
11. q is Promise { undefined }

Stage6



```
let p = new Promise(  
  (e, f) => {  
    console.log("Entering Executor");  
    e(3);  
    console.log("Leaving Executor");  
  }  
) .then((e) => {  
  console.log("Entering SuccessHandler");  
  console.log(e);  
  console.log("Leaving SuccessHandler");  
  return 7;  
})  
  
console.log(p);  
process.nextTick(() => {  
  console.log("Entering ProcessTick");  
  console.log(p);  
  console.log("Leaving ProcessTick");  
});  
  
setImmediate(() => {  
  console.log("Entering SetImmediate");  
  console.log(p);  
  console.log("Leaving SetImmediate");  
})
```



 out

1. Entering Executor
2. Leaving Executor
3. Promise { <pending> }
4. Entering ProcessTick
5. Promise { <pending> }
6. Leaving ProcessTick
7. Entering SuccessHandler
8. 3
9. Leaving SuccessHandler
10. Entering SetImmediate
11. Promise { 7 }
12. Leaving SetImmediate

Stage 7



```
let fs = require("fs");

console.log("Before Executor");

function executor(resolve, reject) {
  console.log("Entering work");
  fs.readFile("./Input.txt", "utf8",
    (err, data) => {
      if (err){
        console.log("This is err of work");
        reject(err);
      }
      else{
        console.log("This is data of work");
        resolve(data);
      }
    });
}

console.log("After executor");

let fp = new Promise(executor).then(
  (value)=>{console.log("success" + value);},
  (reason)=>{console.log("failed " + reason)});

console.log("End of Script");
```

Output

```
1. Before Executor
2. After executor
3. Entering work
4. End of Script
5. This is data of work
6. successHello World
```

Stages

```
1. let fs = require("fs");
2. let fs1 = require("fs/promises");
3.
```



```
sync function work1(n)
```



```
6.   console.log("Entering work1");
7.   data = await fs1.readFile(n,"utf-8");
8.   console.log(data);
9.   console.log("That ends work1");
10. }
11. work1("./Input.txt");
12.
13. function work(resolve, reject) {
14.   console.log("Entering work");
15.   fs.readFile("./Input.txt", "utf8",
16.     (err, data) => {
17.       if (err){
18.         console.log("This is err of work");
19.         reject(err);
20.       }
21.       else{
22.         console.log("This is data of work");
23.         resolve(data);
24.       }
25.     });
26. }
27.
28. let fp = new Promise(work).then(
29.   (v)=>{console.log("success" + v)};,
30.   (v)=>{console.log("failed " + v)});
31.
32. let x = 13;
33. let p = new Promise(
34.   (resolve, reject) => {
35.     if (x > 5) {
36.       console.log(resolve);
37.       console.log(resolve.toString());
38.       resolve(x)
```



```

    }
    else {
41.         console.log(reject);
42.         console.log(reject.toString());
43.         reject(0);
44.     }
45. }
46. );
47. console.log(p);
48. console.log("task1");
49. console.log("task2");
50. function f1(v) {
51.     console.log("Successs.....", v);
52.     d = Promise.resolve(7);
53.     return d;
54. }
55. function f2(v) {
56.     console.log("Failed....", v)
57.     return Promise.reject(-3);
58. }
59. p1 = p.then(f1,f2);
60. console.log(p1);
61. process.nextTick(() => { console.log("process tick"); });
62. setImmediate(() => {
63.     console.log("Set Immediate");
64.     console.log(p1);
65. });
66. console.log("After then");
```



 X out

```
1. Entering work1
2. Entering work
3. [Function (anonymous)]
4. function () { [native code] }
5. Promise { 13 }
6. task1
7. task2
8. Promise { <pending> }
9. After then
10. process tick
11. Successs..... 13
12. Set Immediate
13. Promise { 7 }
14. This is data of work
15. successHello World
16.
17. Hello World
18.
19. That ends work1
```

Add : 25, Patel Shopping Center, Sai Nath Road, Malad (west) ,Opp malad subway , Mumbai 64 Contact : 9820396074, 022-28809398, 9820860292
Copyright © 2011-2020 Rajesh Patkar, All rights reserved.

