PROMISES

Stage1

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

Output

```
Promise { <pending> }
```

Stage2

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

Output

```
Promise { 3 }
```

Stage3

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

Output

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

Stage4

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

put

```
Promise { 3 }
```

Stage5

```
let p = new Promise(
    (e, f) => {
        console.log("Entering Executor");
        f(-3);
        console.log("Leaving Executor");
    }
);
console.log(p);
let q = p.catch(e => {
    console.log("In Catch");
    console.log(e)
    console.log("Leaving catch");
});
console.log(p);
console.log(q);
setImmediate(() => {
    console.log("Entering setImmediate");
    console.log("Leaving SetImmediate");
    console.log("q is ", q);
})
```

out

```
Entering Executor
Leaving Executor
Promise { <rejected> -3 }
Promise { <rejected> -3 }
Promise { <pending> }
In Catch
-3
Leaving catch
Entering setImmediate
Leaving SetImmediate
q is  Promise { undefined }
```

Stage6

```
et 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");
})
```

ut

```
Entering Executor
Leaving Executor
Promise { <pending> }
Entering ProcessTick
Promise { <pending> }
Leaving ProcessTick
Entering SuccessHandler
3
Leaving SuccessHandler
Entering SetImmediate
Promise { 7 }
Leaving SetImmediate
```

Stage7

```
et 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

```
Before Executor
After executor
Entering work
End of Script
This is data of work
successHello World
```

Stage8

```
let fs = require("fs");
let fs1 = require("fs/promises");

```

```
sync function work1(n)
{
   console.log("Entering work1");
   data = await fs1.readFile(n,"utf-8");
   console.log(data);
   console.log("That ends work1");
}
work1("./Input.txt");

function work(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);
           }
       });
}

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

let x = 13;
let p = new Promise(
   (resolve, reject) => {
       if (x > 5) {
           console.log(resolve);
           console.log(resolve.toString());
           resolve(x)
```

```
        }
        else {
            console.log(reject);
            console.log(reject.toString());
            reject(0);
        }
    }
);
console.log(p);
console.log("task1");
console.log("task2");
function f1(v) {
   console.log("Successs......", v);
   d = Promise.resolve(7);
   return d;
}
function f2(v) {
   console.log("Failed....", v)
   return Promise.reject(-3);
}
p1 = p.then(f1,f2);
console.log(p1);
process.nextTick(() => { console.log("process tick"); });
setImmediate(() => {
   console.log("Set Immediate");
   console.log(p1);
});
console.log("After then");
```

out

```
Entering work1
Entering work
[Function (anonymous)]
function () { [native code] }
Promise { 13 }
task1
task2
Promise { <pending> }
After then
process tick
Successs...... 13
Set Immediate
Promise { 7 }
This is data of work
successHello World

Hello World

That ends work1
```