

1. Dry Run of Previous code of

/A1. JavaScript/A5. Module5_Promises/A1. raw/A2. facts/A1.

Promises/A4.Promises.js

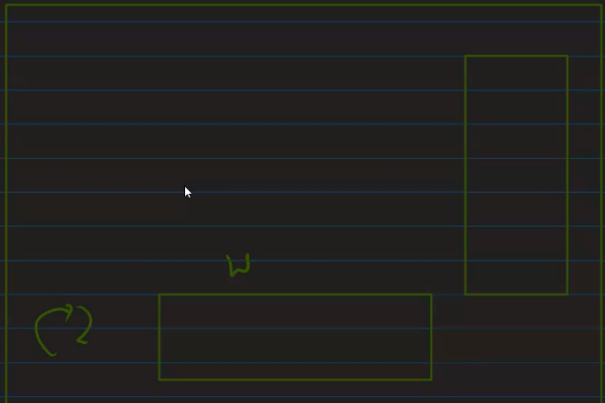
* Promise => assurance of a work that could be completed in future.

* promise-> initial state-> pending, value -> undefined,

final state-> settled resolved -> you have got the future value, rejected-> error

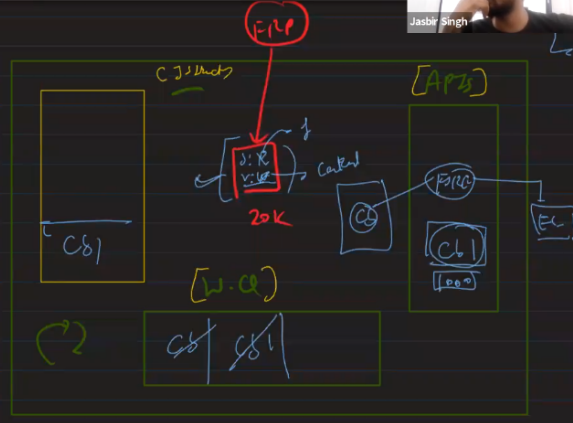
```
let fs = require("fs");
function myPromisedFsReader(filePath) {
  // using this existing function
  return new Promise(function (resolve, reject) {
    fs.readFile(filePath, function cb(err, data) {
      if (err) {
        reject(err);
      } else {
        resolve(data);
      }
    })
  })
}

let fileReadPromise = myPromisedFsReader("fi.txt");
// state -> pending
console.log("Before");
console.log("?", fileReadPromise);
// 1sec -> async (1sec)
setTimeout(function () {
  console.log("!!", fileReadPromise);
}, 1000);
console.log("after");
```



```
let fs = require("fs");
function myPromisedFsReader(filePath) {
  // using this existing function
  return new Promise(function (resolve, reject) {
    fs.readFile(filePath, function cb(err, data) {
      if (err) {
        reject(err);
      } else {
        resolve(data);
      }
    })
  })
}

let fileReadPromise = myPromisedFsReader("fi.txt");
// state -> pending
console.log("Before");
console.log("?", fileReadPromise);
// 1sec -> async (1sec)
setTimeout(function () {
  console.log("!!", fileReadPromise);
}, 1000);
console.log("after");
```



Before
? R L R
After
!! P L R

2. Dry Run of same file after adding then and catch.

```
function myPromisedFsReader(filePath) {
  // using this existing function
  return new Promise(function (resolve, reject) {
    console.log("28")
    fs.readFile(filePath, function cb(err, data) {
      console.log("9");
    })
  })
}

console.log("Before");
let fileReadPromise = myPromisedFsReader("fl.txt");
// state -> pending
console.log("24", fileReadPromise);
// 1sec -> async (1sec)
// setTimeout(function () {
//   console.log("11", fileReadPromise);
// }, 1000);
// function call -> then is synchronous
// this will always run async
function cb(data) {
  console.log("hello", data);
}
function fcb(err) {
  console.log("hello", err);
}
console.log("33")
fileReadPromise.then(cb);
fileReadPromise.catch(fcb);
console.log("35")

console.log("after");
```

Diagram labels: C (Callback), P (Promise), Q (Queue). Arrows show the flow from C to P, then to Q, and finally to P.

```
function myPromisedFsReader(filePath) {
  // using this existing function
  return new Promise(function (resolve, reject) {
    console.log("28")
    fs.readFile(filePath, function cb(err, data) {
      console.log("9");
    })
  })
}

console.log("Before");
let fileReadPromise = myPromisedFsReader("fl.txt");
// state -> pending
console.log("24", fileReadPromise);
// 1sec -> async (1sec)
// setTimeout(function () {
//   console.log("11", fileReadPromise);
// }, 1000);
// function call -> then is synchronous
// this will always run async
function cb(data) {
  console.log("hello", data);
}
function fcb(err) {
  console.log("hello", err);
}
console.log("33")
fileReadPromise.then(cb);
fileReadPromise.catch(fcb);
console.log("35")

console.log("after");
```

Diagram labels: C (Callback), P (Promise), Q (Queue), Data, P (Promise). Arrows show the flow from C to P, then to Q, and finally to P.

Time	State	Value
0	Pending	
1	Resolved	hello, data
2	Rejected	hello, err

jo then ka callback register kara hua hai woh tabhi chalta hai jab, jisme attack kiya gya tha woh callback woh promise resolve hojaye, catch keliye bhi same case hai.

3. Final dry run of the same file after

```

function myPromisiedFsReader(filePath) { ...
}
console.log("Before");
let fileReadPromise = myPromisiedFsReader("f1.txt");
// state -> pending
console.log("24", fileReadPromise);
// 1sec -> async (1sec)
// setTimeout(function () {
//   console.log("11", fileReadPromise);
// }, 1000);
// function call -> then is synchronous
// this will always run async
function scb(data) {
  console.log("hello", data);
  // return new Error("This is a error");
  return "Hello value returend from scb"
}
console.log("33");
// let thenNPromise = fileReadPromise.then(scb)
fileReadPromise.then(scb).then(scb2);
function scb2(data) {
  console.log("42", data);
}

```

C.J. Dacko

```

function myPromisiedFsReader(filePath) { ...
}
console.log("Before");
let fileReadPromise = myPromisiedFsReader("f1.txt");
// state -> pending
console.log("24", fileReadPromise);
// 1sec -> async (1sec)
// setTimeout(function () {
//   console.log("11", fileReadPromise);
// }, 1000);
// function call -> then is synchronous
// this will always run async
function scb(data) {
  console.log("hello", data); ✓
  // return new Error("This is a error");
  return "Hello value returend from scb" ✓
}
console.log("33");
// let thenNPromise = fileReadPromise.then(scb)
fileReadPromise.then(scb).then(scb2);
function scb2(data) {
  console.log("42", data); ✓
}

```

C.J. Dacko

Done
PC ready
33
hello data
42 -> hello value

Activate Windows
Go to Settings to activate Windows.