NodeJS Pre-Mock

Total points 58/60

Test

A Promise can be in which of the following states? *

2/2

- a) Waiting, Running, Done
- b) Pending, Fulfilled, Rejected ✓
- c) Idle, Success, Error
- d) Start, Stop, Finish

What will be logged? *

2/2

```
setTimeout(() => console.log("timeout"), 0);
Promise.resolve().then(() => console.log("promise"));
console.log("end");
```

- a) timeout → promise → end
- b) promise → end → timeout
- c) end → promise → timeout ✓
- d) end → timeout → promise

✓ Which method runs **regardless of fulfillment or rejection**? * 2/2

- ○ .then()
- ○ b) .catch()
- ◉ c) .finally() ✓
- ○ d) .done()

Phone Number *

6204732828

✓ Why is JS called single-threaded? * 2/2

- ◉ a) It runs one task at a time in the Call Stack ✓
- ○ b) It can run multiple tasks in parallel
- ○ c) Only Node.js is single-threaded
- ○ d) Because Promises exist

✓ Which method is used for error handling in Promises? * 2/2

- ○ a) .fail()
- ◉ b) .catch() ✓
- ○ c) .error()
- ○ d) .finally()

What will be the output of this code? * 2/2

```
console.log("Start");
setTimeout(() => console.log("Timeout"), 0);
 console.log("End");
```

- a) Start → Timeout → End
- b) Timeout → Start → End
- c) Start → End → Timeout ✓
- d) End → Start → Timeout

What will be printed? * 2/2

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

setTimeout(() => console.log("B"), 0);

Promise.resolve().then(() => {
  console.log("C");
  setTimeout(() => console.log("D"), 0);
});

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

- a) A → B → C → D → E
- b) A → E → C → B → D ✓
- c) A → C → E → B → D
- d) A → E → B → D → C

Degree *

MCA

✓ Which Web API is used for localStorage? * 2/2

- (●) a) Storage API ✓
- ( ) b) Fetch API
- ( ) c) Timers API
- ( ) d) Cache API

✓ What happens if a Promise is rejected but no catch() is used? * 2/2

- ( ) a) Code still runs normally
- (●) b) Unhandled Promise Rejection error ✓
- ( ) c) Automatically retries
- ( ) d) Silent fail

✓ Which Web API manages event listeners like onclick? * 2/2

- (●) a) DOM API ✓
- ( ) b) Timers API
- ( ) c) Storage API
- ( ) d) Fetch API

✓ Which statement(s) is/are true? * 2/2

- ○ a) JS is multi-threaded by default
- ○ b) Web APIs execute synchronous tasks
- ◉ c) Microtasks execute before macrotasks ✓
- ○ d) setTimeout callback runs before promises

✓ Which of the following tasks goes into the **Macrotask Queue**? * 2/2

- ◉ a) setTimeout ✓
- ○ b) Promise.then
- ○ c) queueMicrotask
- ○ d) MutationObserver

✓ What problem do Promises solve? * 2/2

- ○ a) Memory leaks
- ◉ b) Callback Hell ✓
- ○ c) DOM manipulation
- ○ d) Variable hoisting

Branch *

computer science

Masters

MCA

✓ * 2/2

Output?

```
Promise.resolve("done")
  .then(res => console.log(res))
  .finally(() => console.log("cleanup"));
```

- ○ a) cleanup → done
- ◉ b) done → cleanup ✓
- ○ c) nothing
- ○ d) error

✓ * 2/2

Output?

```
setTimeout(() => console.log("A"), 0);
Promise.resolve().then(() => console.log("B"));
Promise.resolve().then(() => console.log("C"));
console.log("D");
```

- ○) D → A → B → C
- ◉ b) D → B → C → A ✓
- ○ c) D → C → B → A
- ○ d) B → C → A → D

✓ Which of the following creates a new Promise? * 2/2

-) Promise.create()
- b) new Promise(...) ✓
- c) Promise.resolveNew()
- d) makePromise()

✗ What is the output? * 0/2

```
console.log("1");
async function foo() {
  console.log("2");
  return "3";
}
foo().then(res => console.log(res));
console.log("4");
```

- a) 1 → 2 → 4 → 3
- b) 1 → 2 → 3 → 4 ✗
- c) 2 → 1 → 4 → 3
- d) 1 → 4 → 2 → 3

Correct answer

- a) 1 → 2 → 4 → 3

✓ Output? * 2/2

```
async function test() {
  return "Hello";
}
test().then(res => console.log(res));
```

- ○ b) Error
- ◉ c) Hello ✓
- ○ d) undefined
- ○ d) setTimeout

✓ Event Loop first checks which condition? * 2/2

- ○ a) Whether Microtask Queue is full
- ○ b) Whether Web APIs finished execution
- ◉ c) Whether Call Stack is empty ✓
- ○ d) Whether setTimeout expired

✓ Which of these is **NOT** handled directly by the JavaScript engine but by Web APIs? *2/2

- ◉ a) setTimeout ✓
- ○ b) Promise.resolve
- ○ c) console.log
- ○ d) Arithmetic operations

✓ Which Web API allows JavaScript to listen for user actions like clicks or keypresses? *2/2

- ○ a) Timer API
- ◉ b) DOM API ✓
- ○ c) Storage API
- ○ d) Fetch API

Name *

Avinash Ranjan

✓ Which statement is **false**? * 2/2

- ○ a) Microtasks execute before Macrotasks
- ○ b) A Promise once settled cannot change its state
- ◉ c) Event Loop executes tasks while Call Stack is not empty ✓
- ○ d) async/await is syntactic sugar over Promises

✓ Which of the following describes the Event Loop correctly? * 2/2

- (●) a) Moves callbacks from Queues → Call Stack when stack is empty ✓
- ( ) b) Executes timers directly
- ( ) c) Clears garbage memory
- ( ) d) None of the above

Email *

avinashranjan918@gmail.com

✓ * 2/2

Output?

```
console.log("A");
setTimeout(() => console.log("B"), 100);
console.log("C");
```

- ( ) a) A → B → C
- (●) b) A → C → B ✓
- ( ) c) B → A → C
- ( ) d) C → A → B

✓ * 2/2

Output?

```
console.log("1");
setTimeout(() => console.log("2"));
Promise.resolve().then(() => console.log("3"));
console.log("4");
```

- ○ a) 1, 2, 3, 4
- ◉ b) 1, 4, 3, 2 ✓
- ○ c) 1, 3, 4, 2
- ○ d) 1, 4, 2, 3

✓ * 2/2

Which best describes .finally() in Promises?

- ○ a) Runs only when fulfilled
- ○ b) Runs only when rejected
- ◉ c) Runs after settlement (either success or error) ✓
- ○ d) Ignores rejections

✓ * 2/2

What is the default return type of an async function?

- ○ a) undefined
- ◉ b) Promise ✓
- ○ c) Object
- ○ d) void

✓ Which is **NOT** a Microtask? * 2/2

- a) Promise.then
- b) queueMicrotask
- c) MutationObserver
- d) setTimeout ✓

Branch(Masters)

Master in computer application

✓ What will this log? * 2/2

```
setTimeout(() => console.log("T1"), 0);
setTimeout(() => console.log("T2"), 0);
console.log("X");
```

- a) T1 → T2 → X
- b) X → T1 → T2 ✓
- c) X → T2 → T1
- d) T2 → X → T1

Year of Passout *

2024

YOP(Masters)

Which queue has **higher priority** in the Event Loop? *

2/2

- a) Macrotask Queue
- b) Microtask Queue ✓
- c) Callback Queue
- d) Web APIs