Node.js MCQ Test - 50 Questions

Based on your 4-day study material

Section A: Synchronous vs Asynchronous Programming

1. What does synchronous programming mean in JavaScript? a) Tasks execute simultaneously b) Each task executes one at a time, blocking subsequent tasks c) Tasks execute in random order d) Tasks execute only when called

2. In the following code, what will be the output order?

```
console.log("start");
setTimeout(() => console.log("timeout"), 0);
console.log("end");
```

a) start, timeout, end b) start, end, timeout c) timeout, start, end d) end, start, timeout

3. Which queue has higher priority in JavaScript event loop? a) Callback queue b) Microtask queue c) Web API queue d) All have equal priority

4. What does the `0` represent in `setTimeout(() => {}, 0)`? a) Immediate execution b) Maximum time to wait c) Minimum time to wait d) No time delay

5. Asynchronous code waits in which part of the browser architecture? a) Call stack b) Event loop c) Web API d) Microtask queue

Section B: Event Loop and JavaScript Engine

6. Which part is considered as V8 engine in browser architecture? a) Web API b) Event loop c) Call stack d) Callback queue

7. When does the event loop move callbacks from queue to call stack? a) Immediately after timeout b) When call stack is empty c) After all synchronous code d) Both b and c are correct

8. In this code, what executes first?

```
setTimeout(() => console.log("timeout"), 1000);
for (let i = 0; i < 2000; i++) {
    console.log(i);
}
```

a) setTimeout callback b) for loop c) Both execute simultaneously d) Depends on system

9. What will be the execution order?

```
console.log(1);
setTimeout(() => console.log(2), 2000);
setTimeout(() => console.log(3), 1000);
console.log(4);
```

a) 1, 2, 3, 4 b) 1, 4, 3, 2 c) 1, 4, 2, 3 d) 4, 1, 3, 2

10. Which components are part of the browser (not JS engine)? a) Call stack only b) Web API, Event loop, Queues c) Only Event loop d) All components

Section C: Promises

11. What is a Promise in JavaScript? a) A function b) An object representing eventual completion of async task c) A variable d) A callback function

12. How many states does a Promise have? a) 2 (resolved, rejected) b) 3 (pending, fulfilled, rejected) c) 4 (pending, resolved, rejected, completed) d) 1 (resolved)

13. In this code, what does `response.json()` return?

```
fetch("https://api.example.com/data")
    .then(response => response.json())
```

a) JSON data directly b) Another promise c) String data d) Response object

14. Which is the correct way to handle promise rejection? a) .then() only b) .catch() only c) .then() and .catch() d) try-catch block

15. What executes first in this code?

```
let promise = fetch("https://api.example.com");
promise.then(data => console.log("Data"));
console.log("Immediate");
```

a) "Data" b) "Immediate" c) Both execute together d) Depends on network speed

Section D: Async/Await

16. Where do you use the `async` keyword? a) Inside function body b) In function declaration c) With variable declaration d) With return statement

17. Where do you use the `await` keyword? a) In function declaration b) Inside async function body c) Outside any function d) With variable declaration

18. What does an async function always return? a) undefined b) The actual return value c) A Promise d) An object

19. In this code, what is the execution order?

```
console.log("1");
async function test() {
    console.log("2");
    await fetch("https://api.example.com");
    console.log("3");
}
test();
console.log("4");
```

a) 1, 2, 3, 4 b) 1, 2, 4, 3 c) 1, 4, 2, 3 d) 2, 1, 4, 3

20. How do you handle errors in async/await? a) .catch() method b) try-catch block c) .error() method d) Both a and b

Section E: Modules - CommonJS

21. What is a module in Node.js? a) A database b) A logical piece of code that can be reused c) A server d) A browser feature

22. Which format does Node.js use by default for modules? a) ES Modules b) CommonJS modules c) AMD modules d) UMD modules

23. What is used for importing in CommonJS? a) import b) require() c) include() d) load()

24. What is used for exporting in CommonJS? a) export b) exports or module.exports c) return d) send()

25. What happens when multiple `module.exports` are used? a) All are exported b) First one is used c) Last one overrides previous ones d) Error occurs

26. Which is correct destructuring import in CommonJS? a) `import {greet} from './file'` b) `let {greet} = require('./file')` c) `const greet = import('./file')` d) `require {greet} from './file'`

27. In this code, what will `value` contain?

```
// file1.js
module.exports = {name: "John", age: 25};

// file2.js
let value = require('./file1');
```

a) "John" b) 25 c) {name: "John", age: 25} d) undefined

Section F: ES Modules

28. What is the correct syntax for named export in ES modules? a) `module.exports = function` b) `export function greet(){}` c) `exports.greet = function` d) `return function greet(){}`

29. What is required when importing ES modules? a) No file extension needed b) Must include .js extension c) Can use any extension d) Extension is optional

30. How many default exports can a file have? a) Unlimited b) Only one c) Two maximum d) None

31. Which is correct for importing named exports? a) `import greet from './file.js'` b) `import {greet} from './file.js'` c) `import * as greet from './file.js'` d) `import './file.js' as greet`

32. What's the difference between named and default export? a) No difference b) Named exports use {}, default doesn't c) Default exports are faster d) Named exports are deprecated

Section G: Module Wrapper & IIFE

33. What does IIFE stand for? a) Immediately Invoked Function Expression b) Internal Invoke Function Expression c) Import Invoke Function Execute d) Immediately Internal Function Expression

34. How many parameters does Node.js module wrapper pass? a) 3 b) 4 c) 5 d) 6

35. What are the 5 parameters in Node.js module wrapper? a) exports, require, module, __filename, __dirname b) import, export, module, file, directory c) exports, imports, module, name, path d) require, module, exports, path, file

36. Which parameter gives the current file's absolute path? a) __dirname b) __filename c) module d) exports

37. Every code in Node.js is wrapped inside: a) try-catch block b) IIFE (Immediately Invoked Function Expression) c) Promise d) Callback function

Section H: Architecture & General Concepts

38. What does 2-tier architecture consist of? a) Client, Server, Database b) Client and Server c) Server and Database d) Client, Middleware, Server

39. What is a client in 2-tier architecture? a) Database layer b) Presentation layer (UI) c) Business logic layer d) Server layer

40. What is a server? a) Only hardware b) Only software c) Combination of both hardware and software d) Just an application

41. What happens in one request-response cycle? a) Client sends request, server processes and sends response b) Server sends request to client c) Multiple requests sent simultaneously d) Database directly responds to client

42. What protocol is used for communication in the diagram? a) FTP b) HTTP c) SMTP d) TCP

Section I: Code Analysis

43. What will this code output?

```
setTimeout(() => console.log("timeout 1"), 0);
Promise.resolve().then(() => console.log("Promise"));
console.log("Hello world");
```

a) timeout 1, Promise, Hello world b) Hello world, Promise, timeout 1 c) Promise, Hello world, timeout 1 d) Hello world, timeout 1, Promise

44. In this module export, what can be imported?

```
function greet() { return "Hello"; }
let name = "John";
module.exports = { greet, name };
```

a) Only greet function b) Only name variable c) Both greet and name d) Nothing can be imported

45. What's wrong with this ES module import?

```
import {greet} from './file'  // Missing extension
```

a) Syntax error b) Missing .js extension c) Wrong import method d) Nothing is wrong

Section J: Best Practices & Error Handling

46. Which is better for readability? a) Promise chains with .then() b) Async/await c) Callback functions d) All are equal

47. What should you wrap await calls in? a) if-else blocks b) try-catch blocks c) for loops d) switch statements

48. For parallel async operations, which should you use? a) Sequential await calls b) Promise.all() c) Multiple setTimeout d) Callback functions

49. What principle do modules help avoid? a) KISS (Keep It Simple Stupid) b) DRY (Don't Repeat Yourself) c) YAGNI (You Aren't Gonna Need It) d) SOLID principles

50. What is the main benefit of using modules? a) Faster execution b) Less memory usage c) Clean, manageable, and reusable code d) Better error handling

---

```
## Answer Key:
1. b  2. b  3. b  4. c  5. c  6. c  7. d  8. b  9. b  10. b
11. b  12. b  13. b  14. c  15. b  16. b  17. b  18. c  19. b  20. d
21. b  22. b  23. b  24. b  25. c  26. b  27. c  28. b  29. b  30. b
31. b  32. b  33. a  34. c  35. a  36. b  37. b  38. b  39. b  40. c
41. a  42. b  43. b  44. c  45. b  46. b  47. b  48. b  49. b  50. c
```