Callback and Promises quiz

17 out of 30 correct

1. What is a JavaScript callback function?

- ○ A function that is called when a web page is loaded.
- ○ A function that is executed immediately after it is defined.
- ◉ A function that is passed as an argument to another function and is executed when that function completes.
- ○ A function that is used to display a message to the user.

2. Which of the following is an example of a callback function in JavaScript?

- ○ document.getElementById('example').innerHTML = "Hello World";
- ○ setInterval(function() { alert("Hello"); }, 1000);
- ◉ function add(a, b) { return a + b; }
- ○ console.log("Hello World");

3. What is the purpose of a callback function in JavaScript?

- ○ To execute a function immediately.
- ◉ To pass a function as an argument to another function.
- ○ To display a message to the user.
- ○ To create a loop in the code.

4. Can a callback function be asynchronous in JavaScript?

- ○ Yes, a callback function can be asynchronous.
- ◉ No, a callback function cannot be asynchronous.
- ○ It depends on the web browser being used.
- ○ It depends on the size of the function being called.

5. What is the difference between a synchronous and asynchronous callback function in JavaScript?

- ○ A synchronous callback function is executed immediately, while an asynchronous callback function is executed after a certain amount of time has passed.
- ○ A synchronous callback function is executed in a single thread, while an asynchronous callback function is executed in a separate thread.
- ◉ A synchronous callback function is executed in the order it is defined, while an asynchronous callback function is executed after all synchronous code has been completed.
- ○ There is no difference between synchronous and asynchronous callback functions.

6. What is a common use case for a callback function in JavaScript?

- ○ Animating web page elements.
- ◉ Displaying messages to the user.
- ○ Handling user input.
- ○ Making asynchronous requests to a server.

7. What is a Promise constructor in JavaScript?

- ◉ A method that is used to create a new Promise object.

- ○ A function that is used to execute a task asynchronously.
- ○ A method that is used to catch errors in JavaScript code.
- ○ A function that is used to create a new Promise prototype.

8. Which of the following is a method used with Promise objects in JavaScript?

- ○ .then()
- ○ .catch()
- ○ .finally()
- ◉ All of the above

9. What is the purpose of the .then() method in the Promise constructor in JavaScript?

- ◉ To handle successful Promise resolutions.
- ○ To handle Promise rejections.
- ○ To handle both successful Promise resolutions and rejections.
- ○ To create a new Promise object.

10. What is the purpose of the .catch() method in the Promise constructor in JavaScript?

- ○ To handle successful Promise resolutions.
- ◉ To handle Promise rejections.
- ○ To handle both successful Promise resolutions and rejections.
- ○ To create a new Promise object.

11. What is the purpose of the .finally() method in the Promise constructor in JavaScript?

- ◯ To handle successful Promise resolutions.
- ◯ To handle Promise rejections.
- ◯ To handle both successful Promise resolutions and rejections.
- ◉ To execute code after either a successful Promise resolution or rejection.

12. Which of the following statements is true regarding Promise objects in JavaScript?

- ◯ A Promise object can be in one of three states: pending, fulfilled, or rejected.
- ◉ A Promise object can be in one of two states: fulfilled or rejected.
- ◯ A Promise object can be in one of four states: initializing, processing, resolved, or rejected.
- ◯ A Promise object can only be in a single state at any given time.

13. Which of the following statements is true regarding the execution order of Promise methods in JavaScript?

- ◯ The .then() method is always executed before the .catch() method.
- ◯ The .catch() method is always executed before the .then() method.
- ◉ The order of execution depends on whether the Promise is resolved or rejected.
- ◯ The order of execution depends on the size of the Promise object.

14. What is a common use case for the Promise constructor in JavaScript?

- ◉ Handling user input in web forms.

- ◯ Animating web page elements.
- ◯ Making asynchronous requests to a server.
- ◯ Displaying messages to the user.

15. What is the purpose of async/await in JavaScript?

- ◉ To make code execution faster.
- ◯ To handle errors in JavaScript code.
- ◯ To write asynchronous code that looks and behaves like synchronous code.
- ◯ To create new JavaScript objects.

16. Which of the following keywords is used to mark a function as an async function in JavaScript?

- ◉ async
- ◯ await
- ◯ promise
- ◯ function

17. Which of the following keywords is used to wait for a Promise to resolve in an async function in JavaScript?

- ◯ async
- ◉ await
- ◯ promise
- ◯ function

18. Which of the following statements is true about async functions in JavaScript?

- ○ Async functions always return a Promise.
- ○ Async functions never return a value.
- ○ Async functions can only be called from within other async functions.
- ◉ Async functions always execute synchronously.

19. What is the purpose of try/catch blocks in async/await functions in JavaScript?

- ○ To handle successful Promise resolutions.
- ○ To handle Promise rejections.
- ◉ To handle both successful Promise resolutions and rejections.
- ○ To create new Promise objects.

20. Which of the following statements is true regarding the order of execution in async/await functions in JavaScript?

- ○ All code in an async function executes asynchronously.
- ◉ The code in an async function executes synchronously until an await keyword is encountered.
- ○ The order of execution depends on the size of the async/await function.
- ○ The order of execution depends on the value returned by the async/await function.

21. Which of the following is true about error handling in async/await functions in JavaScript?

- ○ Errors in async/await functions must always be handled using try/catch blocks.
- ◉ Errors in async/await functions can only be handled using the .catch() method.

- ○ Errors in async/await functions can be handled using try/catch blocks or the .catch() method.
- ○ Errors in async/await functions do not need to be handled.

22. Which of the following is an advantage of using async/await over Promises in JavaScript?

- ○ Async/await is more efficient than Promises.
- ◉ Async/await is easier to understand and write than Promises.
- ○ Async/await is less flexible than Promises.
- ○ Async/await is less commonly used than Promises.

23. What is the purpose of the fetch() method in JavaScript?

- ◉ To make HTTP requests and handle responses.
- ○ To access and manipulate DOM elements.
- ○ To create and manipulate new JavaScript objects.
- ○ To write asynchronous code that looks and behaves like synchronous code.

24. What does the fetch() method return when a request is made?

- ◉ A string
- ○ An object
- ○ A Promise
- ○ A boolean

25. What is the format of a fetch() response in JavaScript?

- ○ A resolved Promise with a response object.

- ○ A rejected Promise with an error object.
- ◉ A string with the response data.
- ○ An object with a status code and response data.

26. Which of the following is a valid way to handle a fetch() response in JavaScript?

- ○ Using the .then() method on the Promise returned by fetch().
- ○ Using the .catch() method on the Promise returned by fetch().
- ○ Using async/await syntax.
- ◉ All of the above.

27. What is the purpose of the async keyword in JavaScript?

- ◉ To define a function as asynchronous.
- ○ To define a function as synchronous.
- ○ To define a variable as asynchronous.
- ○ None of the above.

28. Can async/await be used with synchronous functions in JavaScript?

- ◉ Yes, async/await can be used with synchronous functions.
- ○ No, async/await can only be used with asynchronous functions.
- ○ It depends on the version of JavaScript being used.
- ○ None of the above.

29. How do you handle errors with async/await in JavaScript?

- ○ Using the try...catch statement.
- ○ Using the .catch() method on the Promise returned by the asynchronous function.
- ◉ Both a) and b).
- ○ Neither a) nor b).

30. Is it possible to use multiple await statements in a single async function in JavaScript?

- ◉ Yes, multiple await statements can be used in a single async function.
- ○ No, only one await statement can be used in a single async function.
- ○ It depends on the specific use case.
- ○ None of the above.

Submit