1. Что такое callback?

В Node.js обратный вызов — это функция, которая передается в качестве параметра другой функции и вызывается, когда эта функция завершает свое выполнение. Это позволяет выполнять асинхронные операции неблокирующим образом, гарантируя, что выполнение кода не будет заблокировано до завершения асинхронной операции.

2. В чем минусы использования коллбэков? Какие есть способы их решения?

Одним из недостатков использования обратных вызовов является ад обратных вызовов, когда код становится трудно читать и поддерживать из-за вложенных обратных вызовов. Другим недостатком является то, что обработка ошибок может стать громоздкой при использовании обратных вызовов. Одним из способов решения этих проблем является использование промисов или async/await вместо обратных вызовов.

3. Что такое Promise и как он работает?

Обещание — это объект в JavaScript, представляющий значение, которое может быть еще недоступно, но будет разрешено в какой-то момент в будущем. Он предоставляет способ обработки асинхронных операций более структурированным способом, избегая ада обратных вызовов.

4. В каких состояниях может находиться Promise?

Обещание может находиться в трех состояниях:

Ожидание: начальное состояние до того, как обещание будет разрешено или отклонено.

Выполнено: состояние, когда обещание успешно выполнено, со значением.

Отклонено: состояние, когда обещание отклонено с ошибкой.

5. Как изменить состояние Promise?

Состояние обещания не может быть изменено после его установки. Его можно только решить или отвергнуть.

6. Как изменить значение Promise?

Значение Promise можно изменить, разрешив или отклонив его со значением или ошибкой соответственно.

7. Что такое цепочки промисов и как они работает?

Цепочки промисов — это способ выполнения последовательности асинхронных операций с использованием промисов. Он включает

объединение нескольких обратных вызовов Promise, чтобы гарантировать, что результат одного Promise будет передан следующему Promise в цепочке. Это позволяет получить более читаемый и поддерживаемый код.

8. Назовите два способа обработки ошибок в Promise.

Два способа обработки ошибок в промисе: использование метода .catch() для перехвата любых ошибок, возникающих во время выполнения промиса, или отклонение промиса с ошибкой, которая обрабатывается методом .catch().

9. Для чего нужен метод Promise.all()?

Метод Promise.all() используется для параллельного выполнения нескольких промисов и возврата массива их результатов. Он разрешается, когда все обещания в массиве разрешены, и отклоняется, если какое-либо из обещаний отклонено.

10. В чем отличия методов Promise.race() и Promise.any()?

Метод Promise.race() возвращает обещание, которое разрешается или отклоняется, как только одно из обещаний в массиве разрешается или отклоняется соответственно. Метод Promise.any() аналогичен, но разрешается только тогда, когда разрешается одно или несколько обещаний в массиве.

11. Что такое async/await?

Async/await — это способ написания асинхронного кода, который выглядит синхронно. Это позволяет разработчикам писать асинхронный код, используя синхронный стиль, без необходимости обратных вызовов или цепочек промисов. Async/await использует ключевое слово await для ожидания разрешения промиса перед продолжением выполнения и может использоваться с любой функцией, которая возвращает промис.