1. **Перечислите основные свойства глобальные объекты Node.js и поясните их предназначение.**

объект global предоставляет доступ к глобальным, то есть доступным из каждого модуля приложения, переменным и функциям. Примерным аналогом данного объекта в JS для браузера объект window. Смысл в том, что переменные\объекты не являются полностью глобальными, а имеют видимость на уровне модуля. То есть в разных модулях можно объявить две переменные с одним именем без конфликта. Чтобы переменная\объект была видна вообще во всем приложении и всех модулях, надо объявить ее через global.varName = “value”

global - хранит var-данные на уровне модуля. Все переменные и функции, объявленные без ключевого слова var, let, или const, автоматически становятся членами объекта global.

process - информация о текущем процессе, среде выполнения

buffer - класс Buffer – предназначен для работы с двоичными данными.

console - используется для печати в stdout и stderr

Существуют только в составе модулей:

exports - ссылка на module.exports

module - ссылка на текущий модуль

require - используется для импорта модуле

1. **Поясните понятие «асинхронная функция»**

**Promise – это специальный объект, который содержит своё состояние**

**Асинхронная функция** – это функция, если ее выполнение осуществляется в 2 фазы: 1) заявка на исполнение; 2) получение результата; при этом участвуют два механизма: A-механизм, формирующий заявку и потом получающий результат; B-механизм, получающий заявку от A, исполняющий операцию и отправляющий результат A; продолжительность исполнения операции B-механизмом, как правило, непредсказуемо; в то время пока B-механизм исполняет операцию, А-механизм выполняет собственную работу.

Асинхронная функция ожидает (await) исполнения промиса и всегда возвращает промис. Промис, возвращаемый асинхронной функцией, будет разрешен (resolve) с тем значением, которое вернет функция.

1. **Поясните понятие стандартные «системные потоки».**

Поток – байтовая последовательность, передаваемая в процессе ввода-вывода.

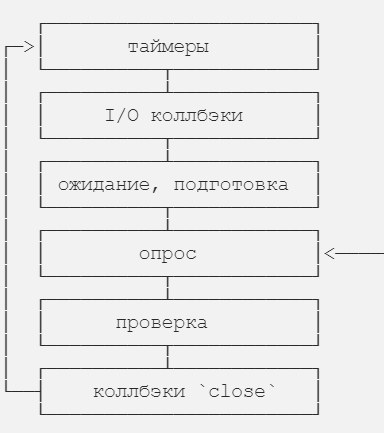
Стандартные системные потоки - основные потока ввода-вывода, предоставляемые ОС для взаимодействия с программами:

stdin (стандартный поток ввода): Этот поток предназначен для приема ввода от пользователя или других программ. В Node.js, он доступен через объект process.stdin.

stdout (стандартный поток вывода): Этот поток используется для вывода данных и сообщений программы. В Node.js, он доступен через объект process.stdout.

stderr (стандартный поток ошибок): Этот поток используется для вывода ошибок и диагностических сообщений. В Node.js, он доступен через объект process.stderr.

1. **Поясните назначение функций process.nextTick, setImmediate, поясните в чем разница.**



Process.nextTick выполняется между фазами EventLoop. Он является микротаском, поэтому находится в отдельной очереди и выполняется до макротасков EventLoop.

Макрозадачи — выполняются по одной за один проход цикла, микрозадачи — на каждом проходе цикла выполняет все накопившиеся

* setImmediate() предназначен для выполнения сценария после завершения текущей фазы опроса.
* setTimeout() планирует запуск сценария после истечения минимального порога в миллисекундах.

Каждый раз, когда вы вызываете process.nextTick() на данной фазе, все коллбэки, переданные процессу process.nextTick(), будут разрешаться до того, как цикл событий продолжится. Это может создать некоторые плохие ситуации, потому что это позволяет «замораживать» ваш ввод/вывод, делая рекурсивные вызовы process.nextTick(), что не даёт циклу событий достичь фазы опроса.

Основное различие между setImmediate и process.nextTick заключается в том, что обратные вызовы setImmediate помещаются в цикл обработки событий, а process.nextTick нет.

setImmediate() — это специальный таймер, который выполняется в отдельной фазе цикла событий. Он использует API libuv, чтобы запланировать коллбэки для выполнения после завершения фазы опроса.

Вообще, когда код исполняется, цикл событий, в конечном счете, попадает на фазу опроса, где он будет ждать входящего соединения, запроса и т.д. Однако, если коллбэк был запланирован с помощью setImmediate(), и фаза опроса переходит в нерабочее состояние, она закончится и начнётся фаза проверки, вместо того, чтобы ждать событий опроса.

После помещения обратного вызова в process.nextTick(), сценарий всё еще имеет возможность выполниться до конца, позволяя инициализировать все переменные, функции и т.д. до вызова коллбэка. Это также имеет то преимущество, что не позволяет циклу событий перейти на следующую фазу. Может быть полезно, чтобы пользователь был предупрежден об ошибке до того, как цикл событий продолжится