1. Поясните понятие «TCP-порт».

TCP-порт — это числовой идентификатор, используемый для адресации процесса в компьютерной сети. В контексте Node.js и сетевого программирования, TCP-порты используются для определения конечной точки соединения между двумя узлами в сети. Когда приложение запускает сервер на определенном порту, оно слушает входящие соединения на этом порту, чтобы устанавливать обмен данными с другими приложениями.

1. Поясните понятие «сетевой сокет».

Сетевой сокет – комбинация ip-адреса и номера порта, которые представляют собой номер адресации.

Сетевой сокет представляет собой программный интерфейс для обеспечения обмена данными между процессами, работающими на разных узлах сети. В контексте Node.js, сетевые сокеты могут использоваться для установки соединения между сервером и клиентом через протоколы, такие как TCP или UDP.

1. Поясните понятие «WebSocket».

WebSocket — это протокол обмена сообщениями между клиентом и сервером через одно постоянное соединение в режиме реального времени. Этот протокол обеспечивает более эффективную и меньшую задержку связь по сравнению с традиционными методами HTTP. В Node.js существуют библиотеки, такие как ws, которые позволяют легко реализовывать WebSocket-серверы и клиенты.

1. Поясните процедуру установки соединения между WS-сервером и WS-клиентом.

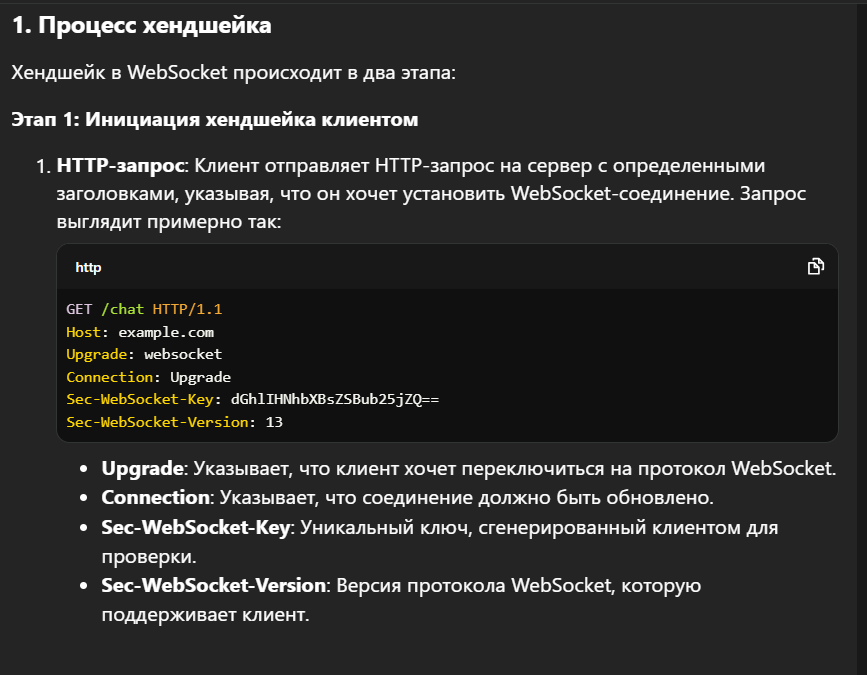
Процедура установки соединения между WS-сервером и WS-клиентом:

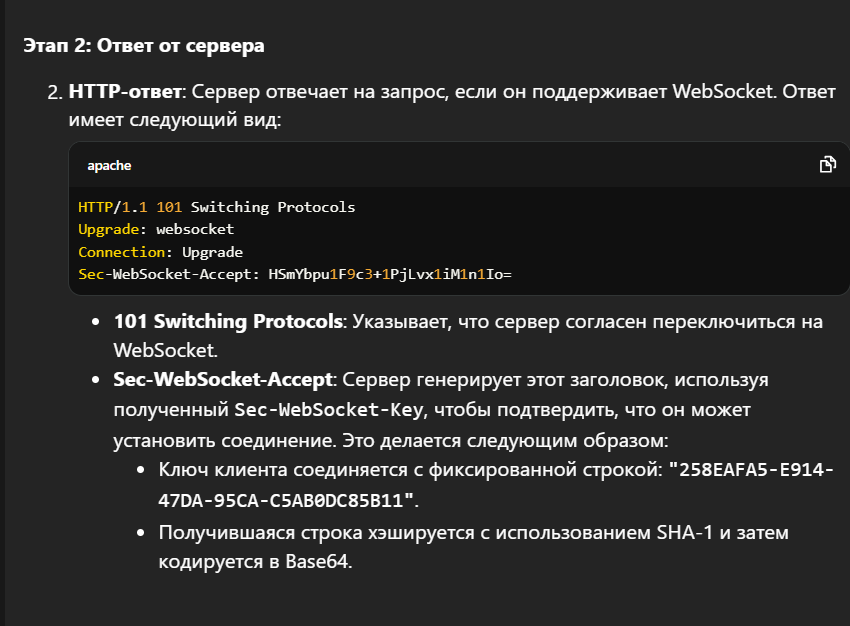
WS-сервер слушает определенный порт.

WS-клиент устанавливает соединение с сервером, отправляя запрос с заголовком Upgrade: websocket.

Сервер подтверждает запрос, и соединение становится WebSocket-соединением.

Теперь сервер и клиент могут обмениваться данными в режиме реального времени через это соединение.





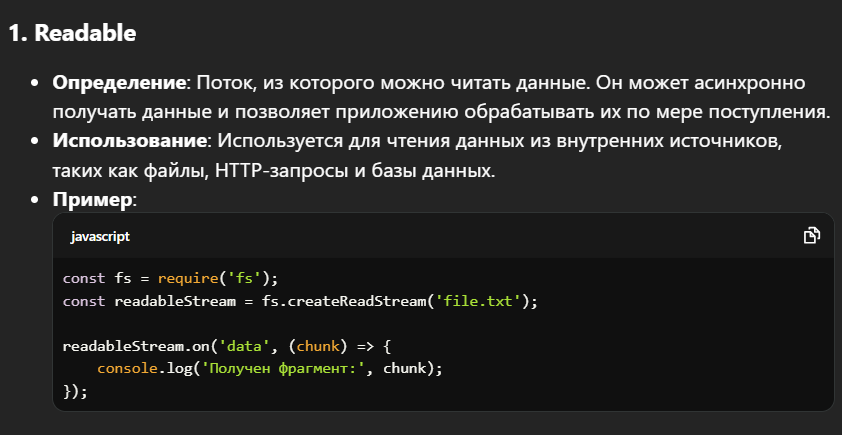
1. Поясните понятие «широковещательное сообщение».

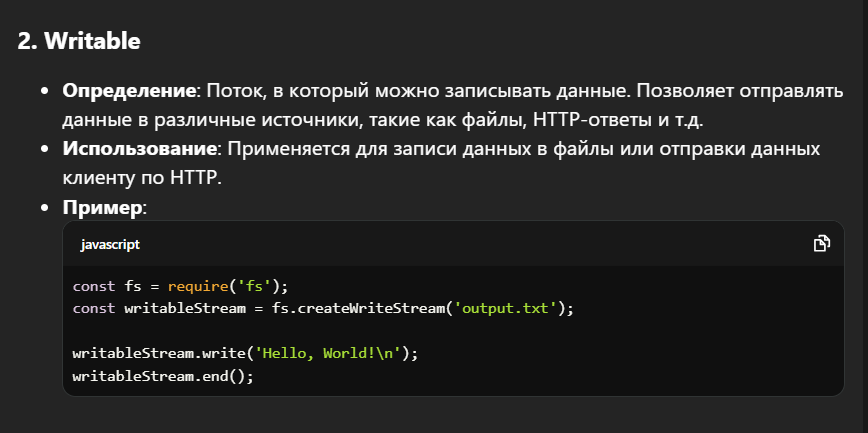
Широковещательное сообщение — это сообщение, отправляемое одним отправителем и получаемое всеми узлами в сети. В контексте Node.js и сетевого программирования, широковещательные сообщения могут использоваться для передачи данных от сервера ко всем подключенным клиентам.

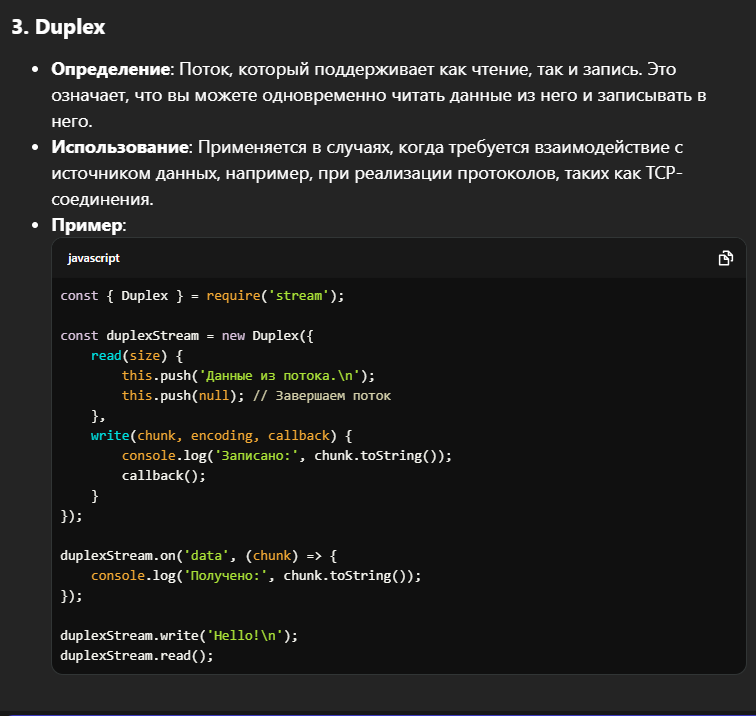
1. Поясните принцип организации потокового ввода/вывода через WS-соединение.

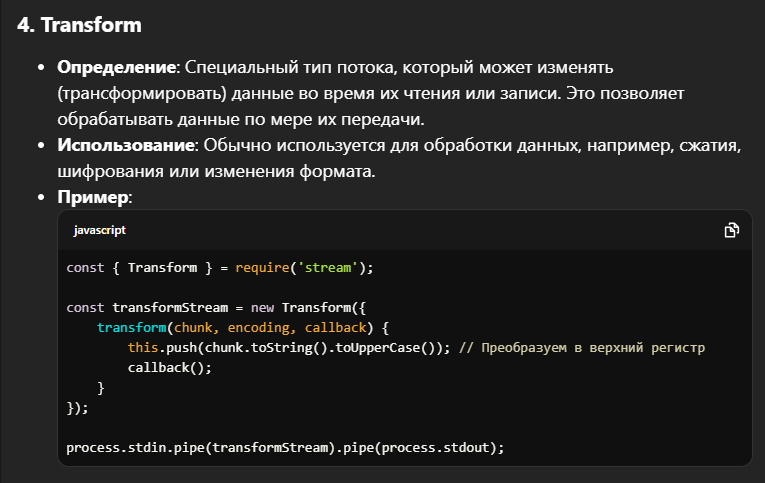
Принцип организации потокового ввода/вывода через WS-соединение:

Потоковый ввод/вывод через WebSocket-соединение в Node.js может быть реализован с использованием объектов потока (Stream). Данные, поступающие через соединение, могут быть обработаны как поток, что обеспечивает эффективность и возможность обработки больших объемов данных по мере их поступления.









1. Поясните принцип действия и назначение механизма «ping/pong».

Механизм "ping/pong" в WebSocket используется для поддержания активности соединения. Сервер или клиент могут отправлять "ping" (запрос) и ожидать "pong" (ответа). Если одна из сторон не получает "ping" в течение определенного времени, соединение может считаться разорванным.

1. Поясните аббревиатуру «RPC».

RPC (remote procedure call) – удаленный вызов процедур

RPC — это механизм взаимодействия между удаленными процессами или программами, который позволяет вызывать процедуры или функции на удаленном сервере так, как если бы они были вызваны локально.

1. Поясните принцип работы RPC-механизма, предоставляемого пакетом rpc-websockets.

rpc-websockets — это библиотека для реализации RPC в приложениях на основе WebSocket. С ее помощью можно определять удаленные процедуры, вызывать их на сервере через WebSocket-соединение и получать результаты

1. Поясните принцип действия и назначение механизма «subscriber/publisher».

Механизм "subscriber/publisher" используется для реализации шаблона "издатель-подписчик". Клиенты могут подписываться на определенные события (издатели), и при возникновении этих событий сервер отправляет уведомления подписчикам.

1. Поясните принцип действия и назначение механизма уведомлений.

Механизм уведомлений позволяет серверу отправлять сообщения клиентам о каких-то событиях, даже если клиент не явно запрашивает эти данные. Это особенно полезно в реальном времени, когда сервер хочет уведомить клиента о изменениях или событиях, произошедших на сервере.

Уведомления предполагают, что функция или метод уведомляет о завершении своей работы, передавая результат через аргументы обратного вызова или возвращая результат через промис.

События могут иметь несколько обработчиков, тогда как уведомления обычно имеют один обработчик (например, обратный вызов или промис).

Уведомления обычно используются для информирования о результате выполнения асинхронной операции. Например, после завершения запроса к базе данных сервер уведомляет клиента о результате.

