**WebSocket (Веб-сокет)** - это протокол связи, который позволяет установить **постоянное, двунаправленное (full-duplex)** соединение между клиентом (например, браузером) и сервером через одно TCP-соединение.

1. **Постоянное соединение (Persistent Connection):** Устанавливается один раз и остается открытым, в отличие от HTTP, где для каждого запроса/ответа может требоваться новое соединение (или используются keep-alive с ограничениями).
2. **Двунаправленная связь (Full-Duplex):** И клиент, и сервер могут отправлять данные друг другу *независимо* и *в любое время* после установки соединения. Серверу не нужно ждать запроса от клиента, чтобы отправить данные.
3. **Низкие накладные расходы (Low Overhead):** После первоначального "рукопожатия" (handshake), заголовки для каждого сообщения очень маленькие по сравнению с HTTP-заголовками, что экономит трафик и уменьшает задержку.
4. **Работа поверх HTTP/HTTPS:** Начинается как обычный HTTP-запрос с просьбой "обновить" (upgrade) соединение до WebSocket, что позволяет проходить через стандартные файрволы и прокси (использует порты 80/443).

ws = new WebSocket('ws://localhost:4000'); — это строка кода на **клиентской стороне** (обычно в JavaScript, выполняемом в браузере, но может быть и в Node.js-клиенте), которая делает следующее:

1. **new WebSocket(...)**: Вызывает встроенный в браузер (или предоставленный библиотекой) конструктор WebSocket. Этот конструктор предназначен для **инициации** нового WebSocket-соединения *от клиента к серверу*.
2. **'ws://localhost:4000'**: Это **URL WebSocket-сервера**, к которому клиент пытается подключиться. Разберем его части:
3. **ws = ...**: Результат вызова new WebSocket(...) — это **объект WebSocket**. Этот объект присваивается переменной ws. Через этот объект ws клиент будет взаимодействовать с установленным (или устанавливаемым) соединением:
   * Отправлять сообщения на сервер (ws.send('Привет, сервер!')).
   * Назначать обработчики для событий, таких как:
     1. ws.onopen: Соединение успешно установлено.
     2. ws.onmessage: Получено сообщение от сервера.
     3. ws.onclose: Соединение закрыто.
     4. ws.onerror: Произошла ошибка соединения.

Отлично, давайте разберем эти понятия:

**20. Поясните разницу между полудуплексным и дуплексным каналами данными.**

* **Полудуплексный канал (Half-Duplex):** Передача данных возможна в **обоих** направлениях, но **не одновременно**. В каждый конкретный момент времени данные могут идти только в одну сторону. Представьте рацию: один говорит, другой слушает, потом они меняются ролями.
* **Дуплексный (или Полнодуплексный) канал (Full-Duplex):** Передача данных возможна в **обоих** направлениях **одновременно**. Участники обмена могут и отправлять, и получать данные в одно и то же время. Представьте телефонный разговор: оба собеседника могут говорить и слышать друг друга одновременно.

**21. Какой тип канала применяется HTTP-протоколом.**

Классический HTTP (особенно HTTP/1.0 и 1.1 без pipelining/multiplexing) работает по принципу **запрос-ответ**. Клиент отправляет запрос, затем ждет и получает ответ от сервера. Хотя базовая технология (TCP) под HTTP является полнодуплексной, сам протокол HTTP на уровне одного запроса/ответа использует канал скорее в **полудуплексном режиме** с точки зрения приложения: сначала говорит клиент (запрос), потом говорит сервер (ответ). Нельзя одновременно отправлять запрос и получать ответ на *этот же самый* запрос.

*Примечание:* HTTP/2 и HTTP/3 вводят мультиплексирование, позволяя передавать несколько запросов и ответов "параллельно" по одному соединению, что делает использование базового дуплексного канала TCP более эффективным, но фундаментальная модель "запрос-ответ" для каждой отдельной транзакции сохраняется.

**22. Поясните понятие WebSocket.**

**WebSocket** — это **протокол** связи, который обеспечивает установку **постоянного**, **двунаправленного (полнодуплексного)** канала связи между клиентом и сервером поверх одного TCP-соединения. Он разработан для веб-приложений, требующих обмена данными в реальном времени.

Ключевые моменты:

* Начинается как обычный HTTP-запрос (для совместимости с существующей инфраструктурой), который затем "обновляется" до WebSocket-соединения.
* После установки соединения оно остается открытым.
* Клиент и сервер могут отправлять друг другу сообщения в любой момент времени, независимо друг от друга.
* Имеет низкие накладные расходы на передачу данных (маленькие заголовки) по сравнению с постоянными HTTP-запросами.

**23. Какой тип канала применяется WebSocket-протоколом.**

WebSocket использует **полнодуплексный (Full-Duplex)** канал связи после успешного установления соединения.

**24. Поясните понятия «широковещательное сообщение» (broadcast), «широковещательный сервер».**

* **Широковещательное сообщение (Broadcast Message):** Это сообщение, которое отправляется от одного источника (например, сервера или одного из клиентов через сервер) **всем** (или определенной большой группе) подключенным получателям **одновременно**. Вместо того чтобы отправлять одно и то же сообщение каждому клиенту по отдельности (множество unicast-сообщений), сервер отправляет его "в эфир" для всех слушателей.
* **Широковещательный сервер (Broadcast Server):** Это сервер, основной функцией которого является прием сообщений от одного или нескольких источников и их последующая **рассылка** (трансляция) всем остальным подключенным клиентам. Сервер 10-03-server.js, который мы создали, является примером простого широковещательного WebSocket-сервера: он получает сообщение от одного клиента и рассылает его всем подключенным клиентам.