**Лабораторная работа 3.**

**Использование протокола WebSocket для обратной связи с пользователем сайта.**

**Цель работы:** Знакомство с протоколом связи WebSocket. Применение WebSocket для обмена сообщениями между пользователями сайта.

**Теоретический материал**

**Как работает WebSocket и в чем его отличия от HTTP**

*Протокол HTTP однонаправленный*. После цикла «запрос — ответ» соединение закрывается, а любой следующий запрос каждый раз устанавливает новое соединение с сервером: сколько запросов, столько и соединений. Процесс передачи данных происходит с некоторыми задержками за счет того, что есть накладные расходы на установку нового соединения при каждом запросе/ответе, а также сетевая и серверная нагрузка из-за обилия периодических запросов.

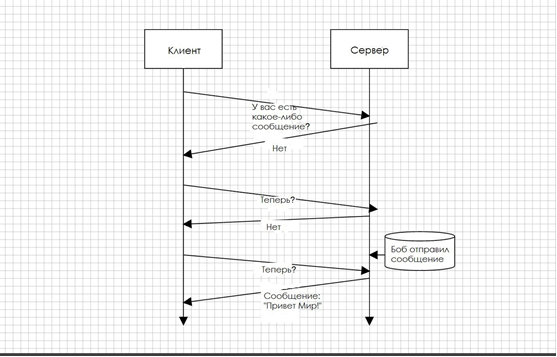


Рисунок 1 - Схема обмена сообщениями по HTTP

*Протокол WebSocket двунаправленный*, полнодуплексный, что означает, что он может одновременно и получать, и передавать информацию. Веб-сокет делает это множество раз в одном открытом соединении. У такого соединения и скорость выше, чем у HTTP.

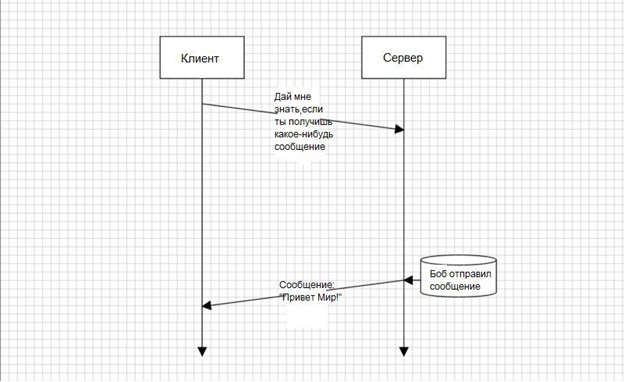


Рисунок 2 - Схема обмена сообщениями при использовании веб-сокетов

У веб-сокетов также есть возможность шифровать передаваемые данные, для этого используется надстройка над протоколом — WSS. Если передаваемые данные не зашифрованы, они становятся объектом для привлечения таких угроз, как несанкционированный доступ к клиенту третьих сторон, использование вредоносного ПО. Специальные надстройки протоколов передачи данных кодируют информацию на стороне отправителя и раскодируют на стороне получателя, оставляя ее зашифрованной для любых посредников. Так достигается безопасный транспортный уровень.

Спецификация WebSocket определяет API для установки соединения между веб-браузером и сервером, основанного на «сокете». Проще говоря, это — постоянное соединение между клиентом и сервером, пользуясь которыми клиент и сервер могут отправлять данные друг другу в любое время.

Клиент устанавливает соединение, выполняя процесс так называемого рукопожатия WebSocket. Для установления соединения веб-сокет применяет метод открывающего рукопожатия. Он заключается в том, что клиент предваряет отправку/получение сообщений предварительным запросом, в котором клиент и сервер «договариваются» использовать веб-сокеты. Это и есть «рукопожатие». Структура такого запроса похожа на HTTP, но немного отличается от него. Затем клиент и сервер обмениваются данными уже в рамках этого соединения.

Кроме того, у веб-сокетов есть дополнительные расширения, которые дополняют и расширяют протокол.

Чтобы открыть веб-сокет-соединение, нам нужно создать объект new WebSocket, указав в url-адресе специальный протокол ws:

*let socket = new WebSocket(“ws://test.com”);*

Когда new WebSocket(url) создан, он тут же сам начинает устанавливать соединение. Браузер, при помощи специальных заголовков, спрашивает сервер: «Ты поддерживаешь Websocket?» и, если сервер отвечает «да», они начинают работать по протоколу WebSocket, который уже не является HTTP.

Если сервер согласен переключиться на WebSocket, то он должен отправить в ответ код 101:

101 Switching Protocols

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Accept: hsBlbuDTkk24srzEOTBUlZAlC2g=

Sec-WebSocket-Accept браузер использует, чтобы убедиться, что ответ соответствует запросу.

Как только объект WebSocket создан, мы должны слушать его события. Их всего 4:

* open – соединение установлено,
* message – получены данные,
* error – ошибка,
* close – соединение закрыто.

Поток данных в WebSocket состоит из «фреймов», фрагментов данных, которые могут быть отправлены любой стороной, и которые могут быть следующих видов:

* «текстовые фреймы» – содержат текстовые данные, которые стороны отправляют друг другу.
* «бинарные фреймы» – содержат бинарные данные, которые стороны отправляют друг другу.
* «пинг-понг фреймы» используется для проверки соединения; отправляется с сервера, браузер реагирует на них автоматически.
* также есть «фрейм закрытия соединения» и некоторые другие служебные фреймы.

В браузере мы напрямую работаем только с текстовыми и бинарными фреймами.

Метод WebSocket *.send()* может отправлять и текстовые, и бинарные данные. Вызов *socket.send(body)* принимает *body* в виде строки или любом бинарном формате включая *Blob*, *ArrayBuffer* и другие. Дополнительных настроек не требуется, просто отправляем в любом формате.

При получении данных, текст всегда поступает в виде строки. А для бинарных данных мы можем выбрать один из двух форматов: *Blob* или *ArrayBuffer*.

Обычно, когда сторона хочет закрыть соединение (браузер и сервер имеют равные права), они отправляют «фрейм закрытия соединения» с кодом закрытия и указывают причину в виде текста.

Метод для этого:

socket.close([code], [reason]);

* *code* – специальный WebSocket-код закрытия (не обязателен).
* *reason* – строка с описанием причины закрытия (не обязательна).

Затем противоположная сторона в обработчике события *close* получит и код *code* и причину *reason*.

*code* – это не любое число, а специальный код закрытия WebSocket.

Наиболее распространённые значения:

* 1000 – по умолчанию, нормальное закрытие,
* 1006 – невозможно установить такой код вручную, указывает, что соединение было потеряно (нет фрейма закрытия).

**Контрольное задание**

Взяв за основу готовую ранее реализацию сайта, добавить реализацию чата «службы поддержки» с применением протокола WebSocket.