Введение

Предположим, что нам нужно разработать чат в браузере.

Стандартный и привычный интерфейс таков: лента чатов слева, сообщения выбранного чата справа. Уже привычно, что сообщения приходят сразу, как собеседник его отправил, без обновления страницы. Но сделать такой механизм средствами http может стать проблемой. Дело в том, что протокол http спроектирован так, что инициировать запрос может только клиент. Сервер не может отправить данные клиенту в случайный момент времени, а только тогда, когда клиент обратился к нему.

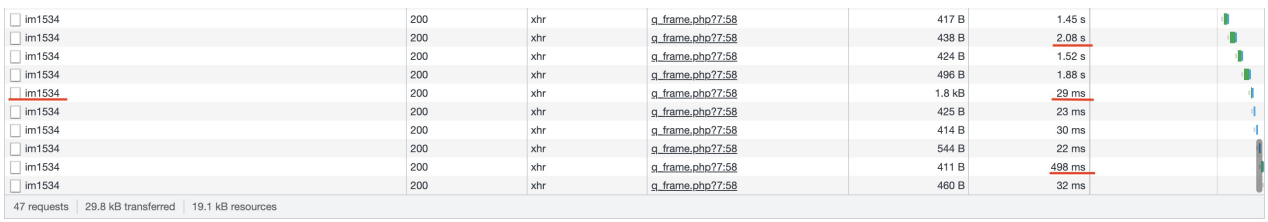
Исходя из этих ограничений, можно придумать следующие решения:

Polling. Клиент бесконечно спрашивает сервер, есть ли для него новые уведомления. Если да — сервер пришлет их клиенту и отобразит в браузере. Минус этого подхода в том, что клиенты отправляют большое количество «пустых» запросов. Они захламляют канал передачи данных и бессмысленно загружают сервер.

Long polling. Клиент так же бесконечно спрашивает сервер если ли для него новые уведомления, но при этом, если уведомлений нет, сервер удерживает соединение до тех пор, пока они не появятся.

То есть клиент сделал запрос, который висит до тех пор, пока не появится информация для клиента. Как только информация появилась, сервер отправляет данные и закрывает соединение. Клиент получает ответ и отображает изменения. Такой подход решает проблемму большого количества «пустых» запросов, но при этом остаются накладные расходы на инициализацию долгих запросов. К тому же раз в какой-то период соединение все равно обрывается и устанавливается новое. Так работает чат vk.

Если зайти на страницу vk и открыть *chrome dev tools*, вкладка *network*, можно увидеть long polling запросы. Слева красным выделен long polling запрос (im1534), справа выделена различная длительность. Она разная, так как в какой-то момент времени не было событий 2.08 секунды, а в какой-то появлялись каждые 28 ms. Так или иначе, если уведомлений много, то запросов тоже будет много.



WebSocket (ws). Протокол двухсторонней передачи данных. Это новый протокол, который является решением проблем в подходах выше.

Он обеспечивает полнодуплексные каналы связи через одно TCP-соединение. Протокол WebSocket обеспечивает взаимодействие между веб-браузером (или другим клиентским приложением) и сервером с меньшими издержками, чем HTTP, облегчая передачу данных в реальном времени от и к серверу.

Это стало возможным благодаря тому, что сервер предоставляет стандартизированный способ отправки контента клиенту без предварительного запроса клиента. Также сервер позволяет передавать сообщения туда и обратно, сохраняя соединение открытым. Поэтому между клиентом и сервером может происходить постоянный двусторонний диалог.

Связь обычно осуществляется через TCP-порт с номером 443, или 80 — в случае незащищенных подключений. То есть протокол работает на тех же портах, что и HTTP, и его внедрение не требует дополнительной конфигурации серверов.

Узнать подробнее про устройство web socket можно в курсе по компьютерным сетям.

Задачи, в которых может быть полезен WS

- браузерные игры
- чаты и уведомления на сайте в реальном времени
- биржевые сервисы для отображения изменения котировок
- ws применимы в mercury. В следующих итерациях в браузере будет отображаться ход запуска компонентов, чтобы пользователь видел прогресс