# Способы REAL-TIME взаимодействия клиент-сервер

1. Long pulling
2. Event source
3. WebSocket

## Long polling

Длинные опросы – это самый простой способ поддерживать постоянное соединение с сервером, не используя при этом никаких специфических протоколов (типа WebSocket или Server Sent Events).

**Частые опросы**

Самый простой способ получать новую информацию от сервера – периодический опрос. То есть, регулярные запросы на сервер вида: «Привет, я здесь, у вас есть какая-нибудь информация для меня?». Например, раз в 10 секунд.

В ответ сервер, во-первых, помечает у себя, что клиент онлайн, а во-вторых посылает весь пакет сообщений, накопившихся к данному моменту.

Это работает, но есть и недостатки:

1. Сообщения передаются с задержкой до 10 секунд (между запросами).
2. Даже если сообщений нет, сервер «атакуется» запросами каждые 10 секунд, даже если пользователь переключился куда-нибудь или спит. С точки зрения производительности, это довольно большая нагрузка.

Так что, если речь идёт об очень маленьком сервисе, подход может оказаться жизнеспособным, но в целом он нуждается в улучшении.

**Длинные опросы**

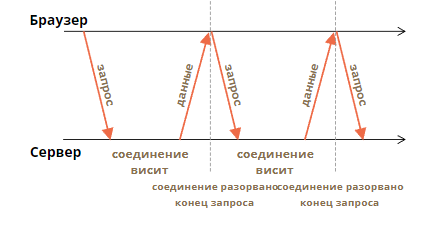
«Длинные опросы» – гораздо лучший способ взаимодействия с сервером.

Они также очень просты в реализации, и сообщения доставляются без задержек.

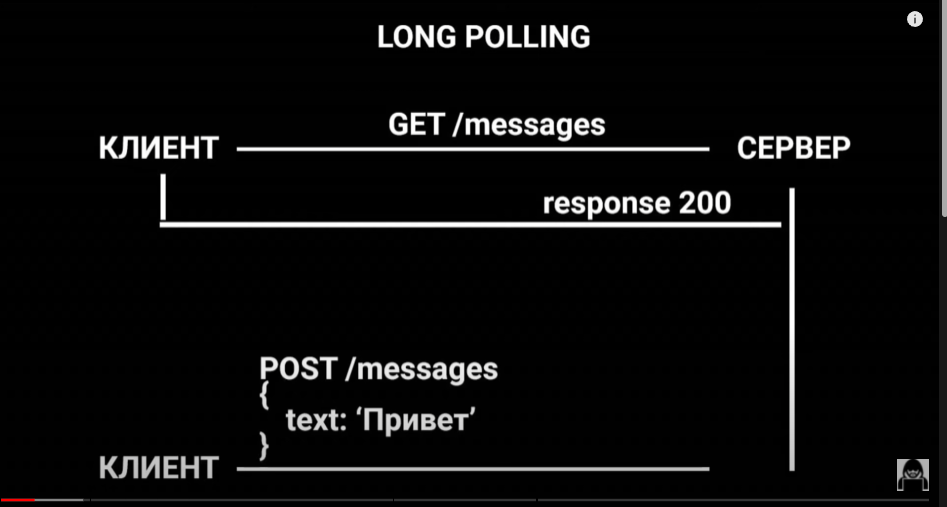
Как это происходит:

1. Запрос отправляется на сервер.
2. Сервер не закрывает соединение, пока у него не возникнет сообщение для отсылки.
3. Когда появляется сообщение – сервер отвечает на запрос, посылая его.
4. Браузер немедленно делает новый запрос.

Для данного метода ситуация, когда браузер отправил запрос и удерживает соединение с сервером в ожидании ответа, является стандартной. Соединение прерывается только доставкой сообщений.

****

**Т.е клиент отправляет запрос, но сервер ему ничего не возвращает, запрос повисает и ждёт ответа. Если время запроса вышло, то он отправляется заново. И так происходит, пока не произошло какое-либо событие (в чате это отправка сообщения другим пользователем). Когда происходит данное событие, сервер сразу же отправляет ответ на клиент. Далее мы снова отправляем запрос, который повисает до следующего события.**

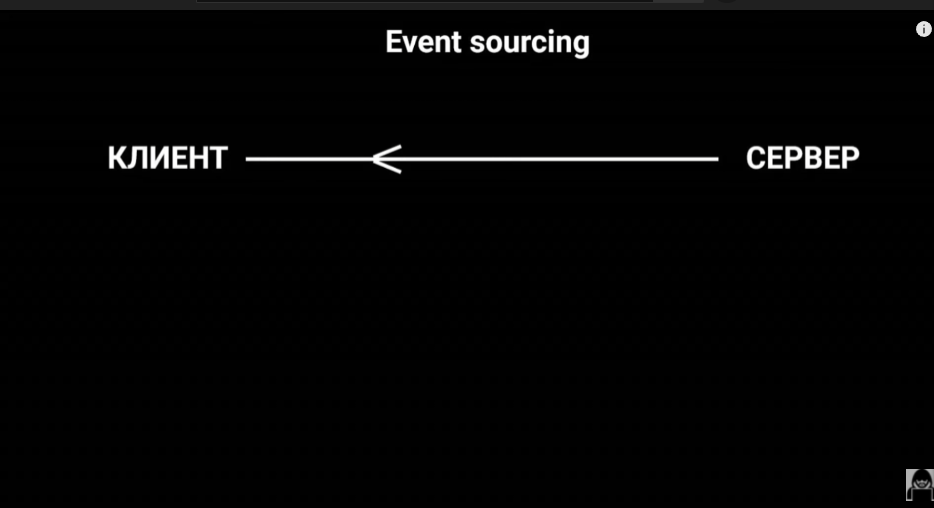
****

* **Самый простой в реализации способ**
* **Не требует особых протоколов**
* **Затрачивает меньше серверных мощностей**
* **Менее стабилен, т.к постоянное соединение не устанавливается**

## Event sourcing

**Event sourcing** (источники событий, регистрация событий, генерация событий) — это мощный архитектурный шаблон, работающий поверх http протокола, при котором все изменения, вносимые в состояние приложения, сохраняются в той последовательности, в которой они происходили. Эти записи служат как источником для получения текущего состояния, так и журналом аудита того, что происходило в приложении за время его существования. Event sourcing способствует децентрализованному изменению и чтению данных. Такая архитектура хорошо масштабируется и подходит для систем, которые уже работают с обработкой событий или хотят перейти на такую архитектуру.

**Другими словами, тут уже устанавливается постоянное подключение, но оно одностороннее, т.е только сервер может отправлять данные. Этот способ используется, когда вся мощь веб сокетов не нужна и достаточно односторонней связи (уведомления события и т.д)**

****

## Web sockets

**WebSocket** — протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени. Данные передаются по нему в обоих направлениях в виде «пакетов», без разрыва соединения и дополнительных HTTP-запросов.

WebSocket особенно хорош для сервисов, которые нуждаются в постоянном обмене данными, например онлайн игры, торговые площадки, работающие в реальном времени, и т.д.

**Это самый мощный способ real-time взаимодействия между клиентом и сервером. Используется Web socket – протокол. Между клиентом и сервером устанавливается постоянное соединение, обмен данными может происходить в двухстороннем режиме.**

* **Более ресурсоёмкий способ**
* **Требует поднятие веб-сокет сервера**

