

Раздел

[Сетевые запросы](#)

Навигация по уроку

Частые опросы


Длинные опросы

Демо: чат

Область применения

Комментарии

Поделиться

[Редактировать на GitHub](#) → [Сетевые запросы](#) 28-го июля 2020

## Длинные опросы

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

Его очень легко реализовать, и он хорошо подходит для многих задач.

### 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

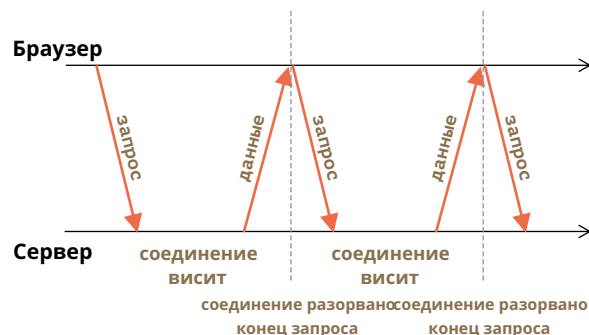
«Длинные опросы» – гораздо лучший способ взаимодействия с сервером.

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, ожидая ответа, является стандартной. Соединение прерывается только доставкой сообщений.



Если соединение будет потеряно, скажем, из-за сетевой ошибки, браузер немедленно посылает новый запрос.

Примерный код клиентской функции `subscribe`, которая реализует длинные опросы:



Раздел

[Сетевые запросы](#)

Навигация по уроку

Частые опросы

Длинные опросы

Демо: чат

Область применения

Комментарии

Поделиться



[Редактировать на GitHub](#)

```
1 async function subscribe() {
2   let response = await fetch("/subscribe");
3
4   if (response.status == 502) {
5     // Статус 502 - это таймаут соединения;
6     // возможен, когда соединение ожидало слишком долго
7     // и сервер (или промежуточный прокси) закрыл его
8     // давайте восстановим связь
9     await subscribe();
10  } else if (response.status != 200) {
11    // Какая-то ошибка, покажем её
12    showMessage(response.statusText);
13    // Подключимся снова через секунду.
14    await new Promise(resolve => setTimeout(resolve, 1000));
15    await subscribe();
16  } else {
17    // Получим и покажем сообщение
18    let message = await response.text();
19    showMessage(message);
20    // И снова вызовем subscribe() для получения следующей
21    await subscribe();
22  }
23 }
24
25 subscribe();
```

Функция `subscribe()` делает запрос, затем ожидает ответ, обрабатывает его и снова вызывает сама себя.



#### Сервер должен поддерживать много ожидающих соединений.

Архитектура сервера должна быть способна работать со многими ожидающими подключениями.

Некоторые серверные архитектуры запускают отдельный процесс для каждого соединения. Для большого количества соединений будет столько же процессов, и каждый процесс занимает значительный объём памяти. Так много соединений просто поглотят всю память.

Часто такая проблема возникает с бэкендом, написанными на PHP или Ruby, но технически дело не в языке, а в реализации. На большинстве современных языков можно написать подходящий сервер, но на некоторых это проще сделать.

Бэкенды, написанные с помощью Node.js, обычно не имеют таких проблем.



## Демо: чат

Вот демо-чат, вы также можете скачать его и запустить локально (если вам знаком Node.js и можете поставить модули):

Результат    browser.js    server.js    index.html



Раздел

[Сетевые запросы](#)

Навигация по уроку

Частые вопросы

Длинные опросы

Демо: чат

Область применения

Комментарии

Поделиться



[Редактировать на GitHub](#)



Все посетители этой страницы будут видеть сообщения друг друга.

Send

Браузерный код находится в `browser.js`.

## Область применения

Длинные опросы прекрасно работают, когда сообщения приходят редко.

Если сообщения приходят очень часто, то схема приёма-отправки сообщений, приведённая выше, становится похожей на «пилу».

Каждое сообщение – это отдельный запрос, с заголовками, авторизацией и так далее.

Поэтому в этом случае предпочтительнее использовать другой метод, такой как [Websocket](#) или [Server Sent Events](#).

Проводим [курсы по JavaScript и фреймворкам](#).



## Комментарии

перед тем как писать...

15 Комментариев [Learn.JavaScript.RU](#)

Политика конфиденциальности Disqus

Войти ▾

Рекомендовать

Твитнуть

Поделиться

Лучшее в начале ▾



Присоединиться к обсуждению...

войти с помощью

или через DISQUS

Имя

[Никита](#) · 7 месяцев назад

Сломал мозг серверным кодом

3 ^ | ▾ · [Ответить](#) · [Поделиться](#) ›

[Джони](#) · год назад

Фрагмент из index.html. Почему функции вызываются через конструкторы ?

```
new PublishForm(document.forms.publish, 'publish');
new SubscribePane....
```

Нет ведь необходимости создавать объект. А если есть, почему их значение никуда не записывается ?