

Раздел

[Сетевые запросы](#)



Навигация по уроку

Не очень полезное событие  
progress

Алгоритм

Комментарии

Поделиться

[Редактировать на GitHub](#) → [Сетевые запросы](#) 14-го августа 2019

## Возобновляемая загрузка файлов

При помощи `fetch` достаточно просто отправить файл на сервер.

Но как возобновить загрузку, если соединение прервалось? Для этого нет готовой настройки, но у нас есть все средства, чтобы решить эту задачу самостоятельно.

Возобновляемая загрузка должна сопровождаться индикацией прогресса, ведь, скорее всего, нам нужно отправлять большие файлы. Поскольку `fetch` не позволяет отслеживать прогресс отправки, то мы будем использовать [XMLHttpRequest](#).

### Не очень полезное событие progress

Чтобы возобновить отправку, нам нужно знать, какая часть файла была успешно передана до того, как соединение прервалось.

Можно установить обработчик `xhr.upload.onprogress`, чтобы отслеживать процесс загрузки, но, к сожалению, это бесполезно, так как этот обработчик вызывается, только когда данные *отправляются*, но были ли они получены сервером? Браузер этого не знает.

Возможно, отправленные данные оказались в буфере прокси-сервера локальной сети или удалённый сервер просто отключился и не смог принять их, или данные потерялись где-то по пути при разрыве соединения и так и не достигли пункта назначения.

В общем, событие `progress` подходит только для того, чтобы показывать красивый индикатор загрузки, не более.

Для возобновления же загрузки нужно *точно* знать, сколько байт было получено сервером. И только сам сервер может это сказать, поэтому будем делать для этого отдельный запрос.

### Алгоритм

- Во-первых, создадим уникальный идентификатор для файла, который собираемся загружать:

```
1 let fileId = file.name + '-' + file.size + '-' + +fil
```

Это нужно, чтобы при возобновлении загрузки серверу было понятно, какой файл мы продолжаем загружать.

Если имя или размер или дата модификация файла изменятся, то у него уже будет другой `fileId`.

- Далее, посылаем запрос к серверу с просьбой указать количество уже полученных байтов:

```
1 let response = await fetch('status', {
2   headers: {
3     'X-File-Id': fileId
4   }
5 });
6
7 // сервер получил столько-то байтов
8 let startByte = +await response.text();
```

Предполагается, что сервер учитывает загружаемые файлы с помощью заголовка `X-File-Id`. Это на стороне сервера должно быть реализовано.

Если файл серверу неизвестен, то он должен ответить 0.

3. Затем мы можем использовать метод `slice` объекта `Blob`, чтобы отправить данные, начиная со `startByte` байта:

```
1 xhr.open("POST", "upload", true);
2
3 // Идентификатор файла, чтобы сервер знал, что мы заг
4 xhr.setRequestHeader('X-File-Id', fileId);
5
6 // Номер байта, начиная с которого мы будем отправлят
7 // Таким образом, сервер поймёт, с какого момента мы
8 xhr.setRequestHeader('X-Start-Byte', startByte);
9
10 xhr.upload.onprogress = (e) => {
11   console.log(`Uploaded ${startByte + e.loaded} of ${
12   };
13
14 // файл file может быть взят из input.files[0] или др
15 xhr.send(file.slice(startByte));
```

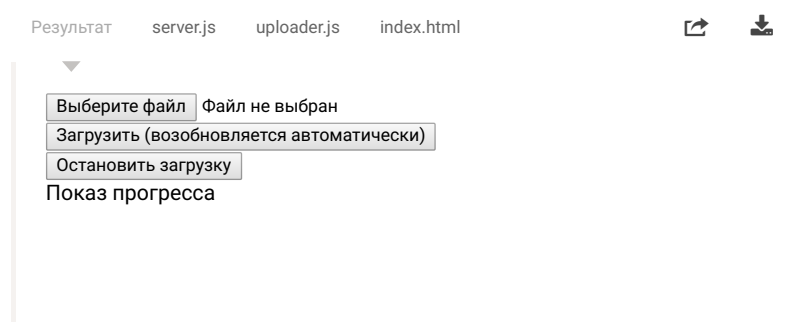
Здесь мы посылаем серверу и идентификатор файла в заголовке `X-File-Id`, чтобы он знал, что мы загружаем, и номер стартового байта в заголовке `X-Start-Byte`, чтобы он понял, что мы продолжаем отправку, а не начинаем её с нуля.

Сервер должен проверить информацию на своей стороне, и если обнаружится, что такой файл уже когда-то загружался, и его текущий размер равен значению из заголовка `X-Start-Byte`, то вновь принимаемые данные добавлять в этот файл.

Ниже представлен демо-код как для сервера (Node.js), так и для клиента.

Пример работает только частично на этом сайте, так как Node.js здесь располагается за другим веб-сервером Nginx, который сохраняет в своём буфере все загружаемые файлы и передаёт их дальше в Node.js только после завершения загрузки.

Но вы можете скачать код и запустить его локально, чтобы увидеть полный пример в действии:



Как видим, современные методы работы с сетью очень близки по своим возможностям к файловым менеджерам – контроль заголовков, индикация прогресса загрузки, отправка данных по частям и так далее.

Можно реализовать и возобновляемую отправку и многое другое.

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

## Комментарии

перед тем как писать...

Раздел

[Сетевые запросы](#)

Навигация по уроку

Не очень полезное событие  
progress

Алгоритм

Комментарии

Поделиться



[Редактировать на GitHub](#)

Раздел

[Сетевые запросы](#)

Навигация по уроку

Не очень полезное событие  
progress

Алгоритм

Комментарии

Поделиться



[Редактировать на GitHub](#)

