**Fetch: запросы на другие сайты**

Если мы сделаем запрос fetch на другой веб-сайт, он, вероятно, завершится неудачей. Ключевым понятием здесь является ***источник* (origin)** – комбинация домен/порт/протокол. Запросы на другой источник – отправленные на другой домен (или даже поддомен), или протокол, или порт – требуют специальных заголовков от удалённой стороны. Эта политика называется «**CORS**»: Cross-Origin Resource Sharing («совместное использование ресурсов между разными источниками»).

## Простые запросы

Есть два вида запросов на другой источник:

* Простые.
* Все остальные.

Простой запрос – это запрос, удовлетворяющий следующим условиям:

* Простой метод: **GET**, **POST** или **HEAD**
* Простые заголовки – разрешены только:
  + **Accept**,
  + **Accept-Language**,
  + **Content-Language**,
  + **Content-Type** со значением **application/x-www-form-urlencoded**, **multipart/form-data** или **text/plain**.

Любой другой запрос считается «непростым».Принципиальное отличие между ними состоит в том, что «простой запрос» может быть сделан через <**form**> или <**script**>, без каких-то специальных методов.

### CORS для простых запросов

При запросе на другой источник браузер всегда ставит «от себя» заголовок Origin - источник (домен/протокол/порт), без пути. Например, если мы запрашиваем https://anywhere.com/request со страницы https://javascript.info/page, заголовки будут такими:

GET /request

Host: anywhere.com

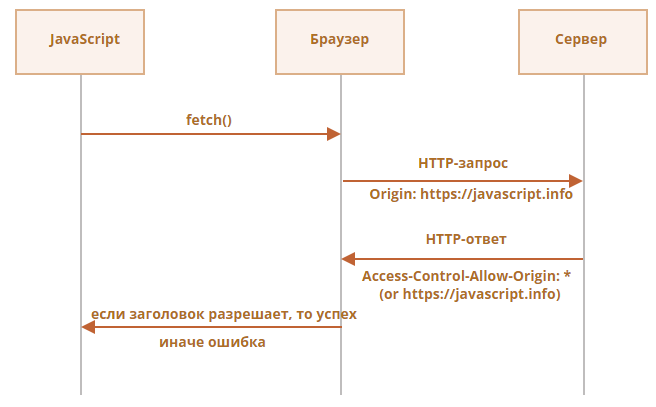
Origin: <https://javascript.info>

Сервер может проверить Origin и, если он согласен принять такой запрос, добавить особый заголовок Access-Control-Allow-Origin к ответу. Этот заголовок должен содержать **разрешённый источник** (в нашем случае https://javascript.info) или **звёздочку \***. Тогда ответ успешен, в противном случае возникает ошибка. Здесь браузер играет роль доверенного посредника: Вот пример ответа сервера, который разрешает доступ:

200 OK

Content-Type:text/html; charset=UTF-8

Access-Control-Allow-Origin: <https://javascript.info>



### [Заголовки ответа](https://learn.javascript.ru/fetch-crossorigin" \l "zagolovki-otveta)

По умолчанию при запросе к другому источнику JavaScript может получить доступ только к так называемым «простым» заголовкам ответа:

* **Cache-Control**
* **Content-Language**
* **Content-Type**
* **Expires**
* **Last-Modified**
* **Pragma**

**!** Обратите внимание: нет **Content-Length**

Чтобы разрешить JavaScript доступ к **любому другому заголовку ответа**, сервер должен указать заголовок Access-Control-Expose-Headers. Он содержит список, через запятую, заголовков, которые не являются простыми, но доступ к которым разрешён. Например:

200 OK

Content-Type:text/html; charset=UTF-8

Content-Length: 12345

API-Key: 2c9de507f2c54aa1

Access-Control-Allow-Origin: https://javascript.info

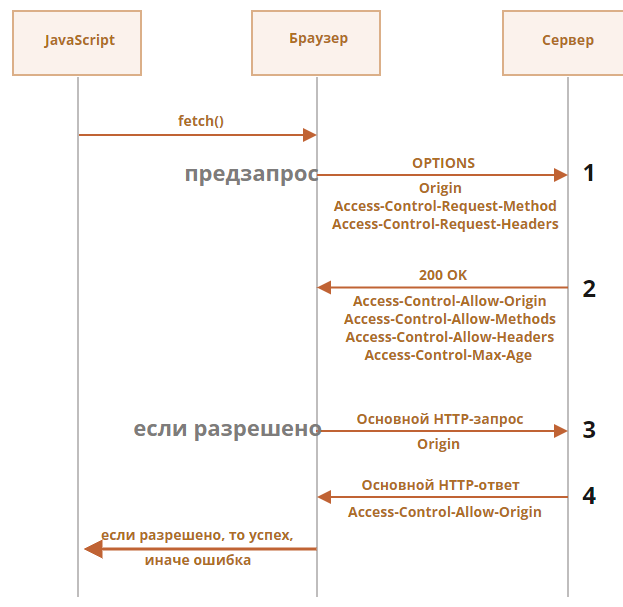
Access-Control-Expose-Headers: Content-Length,API-Key

При таком заголовке Access-Control-Expose-Headers, скрипту разрешено получить заголовки Content-Length и API-Key ответа.

## «Непростые» запросы

Для непростых запросов перед основным запросом отправляется **предзапрос**:

* → Браузер посылает запрос **OPTIONS** на тот же адрес с заголовками:
  + Access-Control-Request-Method – содержит запрашиваемый метод,
  + Access-Control-Request-Headers – перечисляет непростые запрашиваемые заголовки.
* ← Сервер должен ответить без тела, **со статусом 200** и заголовками:
  + Access-Control-Allow-Methods со списком разрешённых методов,
  + Access-Control-Allow-Headers со списком разрешённых заголовков,
  + Access-Control-Max-Age с количеством секунд для кеширования разрешений
* → Затем отправляется основной запрос, применяется предыдущая «простая» схема.



**!** Предзапрос осуществляется «за кулисами», невидимо для JavaScript. JavaScript получает только ответ на основной запрос или ошибку, если со стороны сервера нет разрешения.

### Авторизационные данные

Запрос на другой источник по умолчанию не содержит авторизационных данных (credentials), под которыми здесь понимаются куки и заголовки HTTP-аутентификации.

Чтобы **включить** отправку авторизационных данных в fetch, нам нужно добавить опцию credentials: "include", вот так:

fetch('http://another.com', {

credentials: "include"

});

Если сервер согласен принять запрос с авторизационными данными, он должен добавить заголовок Access-Control-Allow-Credentials: true к ответу, в дополнение к Access-Control-Allow-Origin. Например:

200 OK

Access-Control-Allow-Origin: https://javascript.info

Access-Control-Allow-Credentials: true

**!** Пожалуйста, обратите внимание: в Access-Control-Allow-Origin запрещено использовать **звёздочку \*** для запросов с авторизационными данными. Там должен быть именно источник, как показано выше. Это дополнительная мера безопасности, чтобы гарантировать, что сервер действительно знает, кому он доверяет делать такие запросы.