

## Типы данных в теле запроса. Файл Cookie.

В теме «Запросы к серверу. XHR и Fetch API» мы указали, какие типы данных можно передать в тело запроса с методами POST, PUT или PATCH. Среди них мы подробно рассмотрим:

- String,
- Blob,
- File,
- FormData,

поскольку именно они используются в подавляющем большинстве запросов и с каждым из них вы обязательно столкнетесь в реальной работе.

Тип String – это простая строка в JS. Вы можете отправить в запросе любую валидную строку, если это согласовано с документацией серверного API, которую вы используете в работе. Однако чаще всего такая строка представлена в формате JSON (JavaScript Object Notation), который очень сильно похож на структуру обычного объекта в JS, но более простой, поскольку имеет некоторые ограничения. Ниже представлена строка в формате валидного JSON:

```
1  {
2      "uid": 5,
3      "notes": [
4          {
5              "title": "Note #3",
6              "text": "Third note",
7              "image": null,
8              "id": 2,
9              "date": "2025-09-11T17:47:03.810Z"
10         }
11     ],
12     "indexes": { "2": 1, "3": 1, "4": 2, "5": 3 },
13     "changed": true
14 }
```

Синтаксис формата JSON описан в статье [1]. Этот формат способны распарсить все web-сервера, он не является уникальным для JS.

Как правило, на стороне клиента (фронтенда) приходится превращать нативные JS-объекты в строки JSON для отправки запроса. Такая операция называется сериализацией. И наоборот, преобразовывать JSON-строку в нативный объект. Такая операция называется десериализацией. Для этого существует глобальный объект JSON с методами `stringify()` и `parse()`, реализующими сериализацию и десериализацию соответственно.

Объект Blob представляет простую комбинацию необработанных байтов, содержащая также тип (как правило MIME-type). Вы можете создать объект Blob посредством конструктора `new Blob()`, передав в него массив строк или типизированных массивов (`ArrayBuffer`). Как и строка, Blob объект может быть передан в тело запроса, а также получен из тела ответа запроса. Подробнее о Blob в [2].

Объект File наследует свойства и методы объекта Blob и имеет дополнительные свойства в качестве информации о файлах, такие как «name», «lastModified», «lastModifiedDate» и другие. Более полная информация содержится в [3]. Объекты File могут быть получены через событие загрузки документа в элемент `<input type="file">`. Используется точно также, как и Blob.

Объект FormData представляет собой комбинацию пар «ключ-значение», а также предоставляет методы для добавления пары – `append()` и `set()`, для удаления пары – `delete()`, для получения значения ключа – `get()` и т. д. Полный список методов и их синтаксис представлен в [4].

Объект FormData может быть сформирован с помощью конструктора `new FormData` двумя способами: как добавлением полей через методы объекта, так и путем передачи в DOM-элемента формы в конструктор – форма распарсится в объект FormData автоматически. Ниже представлен один из примеров использования с HTML-формами, взятый с ресурса [5]:

```
1 <form id="formElem">
2   <input type="text" name="firstName" value="John">
3   Картинка: <input type="file" name="picture" accept="image/*">
4   <input type="submit">
5 </form>
6
7 <script>
8   formElem.onsubmit = async (e) => {
9     e.preventDefault();
10
11     let response = await fetch('/article/formdata/post/user-avatar', {
12       method: 'POST',
13       body: new FormData(formElem)
14     });
15
16     let result = await response.json();
17
18     alert(result.message);
19   };
20 </script>
```

С основами формирования тела запроса мы разобрались. Еще одна важная тема, касающаяся обмена данными с сервером – это куки (Cookies). Cookie – это текстовые данные небольшого размера, которые (как правило) приходят в ответе (Response) на HTTP-запрос в заголовке «Set-Cookie» и записываются в локальный файл браузером автоматически, обнаружив такой заголовок в ответе. Необходимые Cookie задаются на сервере.

Заголовок Set-Cookie содержит строку в формате «name=value; attr1; attr2; ...», где name – имя Cookie, value – значение, attr – атрибут (например, Expires, Secure, HttpOnly и др.). Ниже представлен в качестве примера скриншот из DevTools одного из ответов, который, помимо основной задачи – получения словаря – устанавливает Cookie цветовой темы приложения (theme) и язык (language) через заголовки Set-Cookie:

Name	×	Headers	Preview	Response	Initiator	Timing	Cookies
get_language_dict...							
options/							
color_scheme/							
get_user_data/							
login/							
get_user_data/							
timers/							
currencies/							
legal_agreements/							
not_completed_at...							
auto_assignment/							
onboarding/							
iAbYz8vOUjEOo...							

  

▼ Response Headers	<input type="checkbox"/> Raw
Connection	keep-alive
Content-Encoding	gzip
Content-Type	application/json
Cross-Origin-Opener-Policy	same-origin
Date	Mon, 15 Sep 2025 16:22:40 GMT
Referrer-Policy	origin-when-cross-origin
Server	nginx
Set-Cookie	theme=dark; Path=/
Set-Cookie	language=ru-RU; Path=/
Transfer-Encoding	chunked

Также куки можно создать и записать в файл посредством JS кода: `document.cookie = "name=value; attr1; attr2; ..."`. Только в данном случае не удастся установить атрибут `HttpOnly`. Об атрибутах Вы узнаете чуть позже.

А теперь главное, для чего нужны Cookie – в каждый запрос (Request), за исключением некоторых случаев, браузер по умолчанию добавляет заголовок «Cookie» со значением, равным конкатенированной строке из пар «name=value» через символ «;». Пример одного из запросов на скриншоте ниже, где на сервер передаются следующие Cookie: `theme`, `language`, `user_id` и `csrftoken`:

×	Headers	Preview	Response	Initiator	Timing	Cookies
Accept-Encoding			gzip, deflate, br, zstd			
Accept-Language			ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7			
Cache-Control			no-cache			
Connection			keep-alive			
Content-Type			application/json			
Cookie			theme=dark; language=ru-RU; user_id="2 1:0 10:1757953620 7:user_id 8:MS0xNA== 5029a0403348d282eab4f22c4ffbb34cc1d16669ce09adc1624205a5ad0f85a3"; csrftoken=LsAbqrusJYpYfSR1FQQh7R3GZXL1WrYm			
Host			box-test.boxbattle.ru			

Таким образом, мы имеем возможность хранить состояние, например, тему или токен авторизации пользователя, которое не исчезнет при перезагрузке страницы и необходимо для передачи в каждый запрос, чтобы сервер реагировал соответствующим образом. Например, если

передается валидный токен авторизации, то запрос будет выполнен успешно и учтет, что запрос был выполнен именно с этого конкретного аккаунта.

Отметим, что браузеры хранят куки в файле на компьютере пользователя. Например, браузеры на основе Chromium (Google Chrome, Yandex Browser и другие) хранят куки в файле, являющимся базой данных SQLite. Например, для браузера Google Chrome в ОС Windows Вы можете найти этот файл под именем «Cookies» в директории `\AppData\Local\Google\Chrome\User Data\Default\Network`.

Настоятельно рекомендуется ознакомиться с документацией [6] об использовании Cookie и разобрать следующие атрибуты:

- SameSite
- Secure
- HttpOnly
- Expires
- Domain
- Path

которые имеют непосредственное отношение к обеспечению безопасности данных web-приложения.

### Вопросы и задания

1. Какие из следующих строк (a, b, c или d) в файле [8] являются невалидными с точки зрения формата JSON и почему?
2. Что будет на выходе после сериализации объекта «{a: [10, undefined, NaN], b: 10, c: () => {}, d: '2'}»?
3. Напишите фрагмент кода, состоящий из элемента `<input>` типа «file», элемента `<img>` и функции-обработчика события `input`, которая выгружает изображение, прикрепленное в `<input>` с последующим выводом изображения в элемент `<img>`. Установите ограничение на максимальный размер загружаемого файла в 1МБ, а также на допустимые MIME-типы: только `png`, `jpeg`, `jpg`. Выведите ошибки в консоль. Стилизация элементов не требуется, кроме тэга `<img>` для установки позиционирования и размеров выводимого изображения.

4. В чем заключается отличие методов добавления `append()` и `set()` объекта `FormData`?
5. Исследуйте Cookie в приложении Вовлекай [7]. Какие из них имеют атрибут «SameSite = Lax»? А какие «HttpOnly»? Сколько действует Cookie с именем «csrftoken»? Сможете ли вы обнаружить, какой запрос устанавливает Cookie с именами «csrftoken» и «user\_id»? Очистите Cookie и перезагрузите страницу – что произошло и почему?
6. \* Какую схему можно использовать, чтобы украсть Cookie с некоторого сайта? Приведите пример хотя бы одной. Как называется такая атака?

#### Ссылки

- [1]. [https://developer.mozilla.org/en-US/docs/Learn\\_web\\_development/Core/Scripting/JSON#json\\_syntax\\_restrictions](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/JSON#json_syntax_restrictions)
- [2]. <https://learn.javascript.ru/blob>
- [3]. <https://learn.javascript.ru/file>
- [4]. <https://developer.mozilla.org/en-US/docs/Web/API/FormData>
- [5]. <https://learn.javascript.ru/formdata>
- [6]. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Cookies>
- [7]. <https://labmedia.vovlekay.online/>
- [8]. [https://codepen.io/diana\\_kondakova/pen/dPYxzoz?editors=0010](https://codepen.io/diana_kondakova/pen/dPYxzoz?editors=0010)