## МЕТОДЫ ЗАПРОСОВ

## Метод fetch

По умолчанию – передает запрос GET

fetch('https://api.kanye.rest')

.then(res => res.json())

.then((data) => {

console.log(data);

});

## Метод JSON.stringify

Метод JSON.stringify делает из объекта строку JSON

const songsJSON = JSON.stringify(songs);

## Метод JSON.parse

Метод JSON.parse делает обратное — преобразовывает JSON-строку в объект JavaScript

const songsJSON = JSON.stringify(songs);

const songsObject = JSON.parse(songsJSON);

## Метод res.json

Метод json читает ответ от сервера в формате json и возвращает промис. Из этого промиса потом можно доставать нужные нам

данные.

fetch('https://api.kanye.rest')

.then(res => res.json())

.then((data) => {

console.log(data);

});

## МЕТОДЫ HTTP

## GET

Если метод не прописать явно, fetch будет отправлять запросы методом GET

## POST

Его используют для отправки данных на сервер. Чтобы послать запрос методом POST, нужно указать этот метод в объекте опций — втором аргументе метода fetch

fetch('https://example.com', {

method: 'POST'

});

## PUT

предназначен для полного обновления указанного ресурса. Например, если в каталоге книг вы решили заменить одну книгу другой;

## PATCH

для частичного обновления ресурса. Например, при обновлении профиля пользователя

## DELETE

для удаления ресурса с сервера

## MIME Type

MIME типов десятки — с большинством из них вам работать не придётся.

Для отправки данных формы обычно используют один из трёх типов:

* application/json — для передачи данных в формате JSON;
* application/x-www-form-urlencoded — формат, который кодирует поля формы так, чтобы их можно было отправить в URL;
* multipart/form-data — для отправки файлов на сервер. Подойдёт, если среди прочего вы отправляете через форму картинку.

Вот самые распространённые:

text/plain

text/html

text/javascript

text/css

image/jpeg

image/png

audio/mpeg

audio/ogg

video/mp4

application/json

application/octet-stream

application/x-www-form-urlencoded

multipart/form-data

fetch('https://example.com/users', {

method: 'POST',

headers: {

'Content-Type': 'application/json; charset=UTF-8'

},

body: JSON.stringify({

username: 'ivan'

})

});

## Передать данные на сервер

Еще можно в параметрах запроса

fetch('https://example.com/images/random?type=portrait', {

method: 'GET'

});

Имя параметра и его значение указывают друг за другом через знак равенства: name=value. Такие пары имя атрибута=значение разделяют амперсандами &:

Скопировать кодJAVASCRIPT

fetch('https://example.com/images/random?type=portrait&name=DorianGrey', {

method: 'GET'

});

## МЕТОДЫ ОТВЕТОВ

## Статус

Каждому статусу запроса соответствует уникальный код.

* 2 — значит, запрос прошёл успешно;
* 3 — запрос был перенаправлен;
* 4 — с запросом что-то не так: ресурс не найден или у вас нет к нему доступа;
* 5 — на сервере произошла какая-то ошибка.

fetch('https://api.kanye.rest')

.then(res => {

console.log(res.status, res.statusText); *// 200 OK*

});

fetch('https://api.kanye.rest')

.then(res => {

console.log(res.ok); *// true*

});

const quoteElement = document.querySelector('div.quote');

fetch('https://api.kanye.rest')

.then((res) => {

if (res.ok) {

return res.json();

}

*/\* отклоняем промис, чтобы перейти*

*в блок catch, если сервер вернул ошибку \*/*

return Promise.reject(`Что-то пошло не так: ${res.status}`);

})

.then((data) => {

quoteElement.textContent = data.quote;

})

.catch((err) => {

console.log(err); *// "Что-то пошло не так: ..."*

});

## Данные заголовков

Чтобы получить значение заголовка, есть метод get.

У объекта заголовков есть и [другие методы](https://developer.mozilla.org/en-US/docs/Web/API/Headers).

Единственное отличие заголовков ответа от заголовков запроса: первые можно только читать, но нельзя редактировать

fetch('https://api.kanye.rest')

.then((res) => {

if (res.headers.get('Content-Type').contains('application/json')) {

return res.json();

}

});

## Тело ответа

Чтобы разобрать тело ответа, есть специальные методы:

* res.json — разбирает JSON в объект, этот метод вы уже знаете;
* res.text — разбирает тело как текст;
* res.blob — разбирает тело ответа как бинарные данные: это нужно при получении файлов (изображений, видео, pdf-документов).

*/\* методы разбора тела возвращают промис, их нужно использовать асинхронно \*/*

fetch('https://api.kanye.rest')

.then(res => res.json())

.then((result) => {

console.log(result);

});

fetch('https://api.kanye.rest')

.then(res => res.text())

.then((result) => {

console.log(result);

});

## *// простое и полезное окончание цепочки промисов:* .catch(console.log.bind(console));