

JS: Fetch

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

Повторим;)

■ Какой аргумент принимает Promise.all?

■ В чём отличие Promise.all от Promise.race?

■ На что указывает ключевое слово await?

■ Три главных компонента в клиент-серверной архитектуре

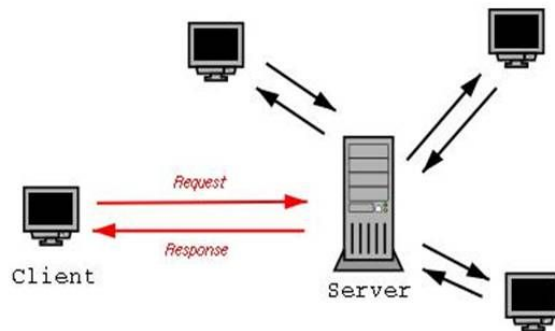
ЦЕЛЬ

Изучить способ отправки запросов на сервер - Fetch

ПЛАН ЗАНЯТИЯ

- Клиент-серверная архитектура
- JSON
- Fetch

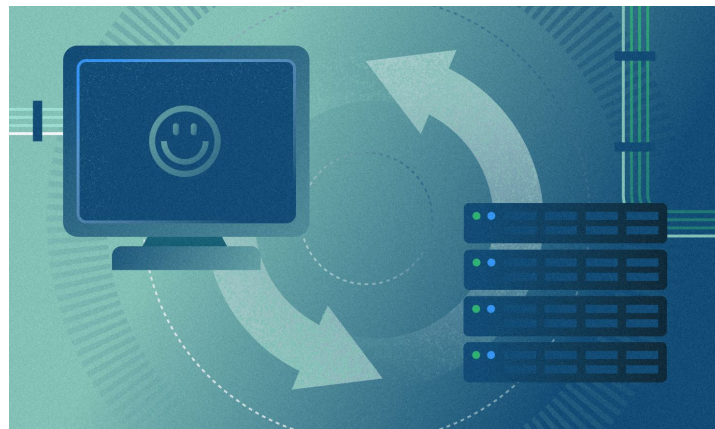
Клиент-серверная архитектура



JavaScript может отправлять сетевые запросы на сервер и подгружать новую информацию по мере необходимости.

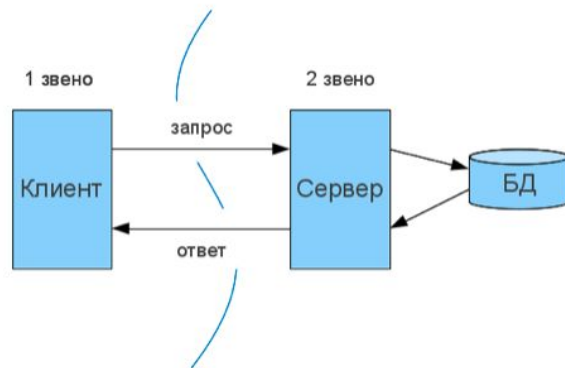
Например, мы можем использовать сетевой запрос, чтобы:

- Отправить заказ,
- Загрузить информацию о пользователе,
- Запросить последние обновления с сервера,
- ...и т.п.

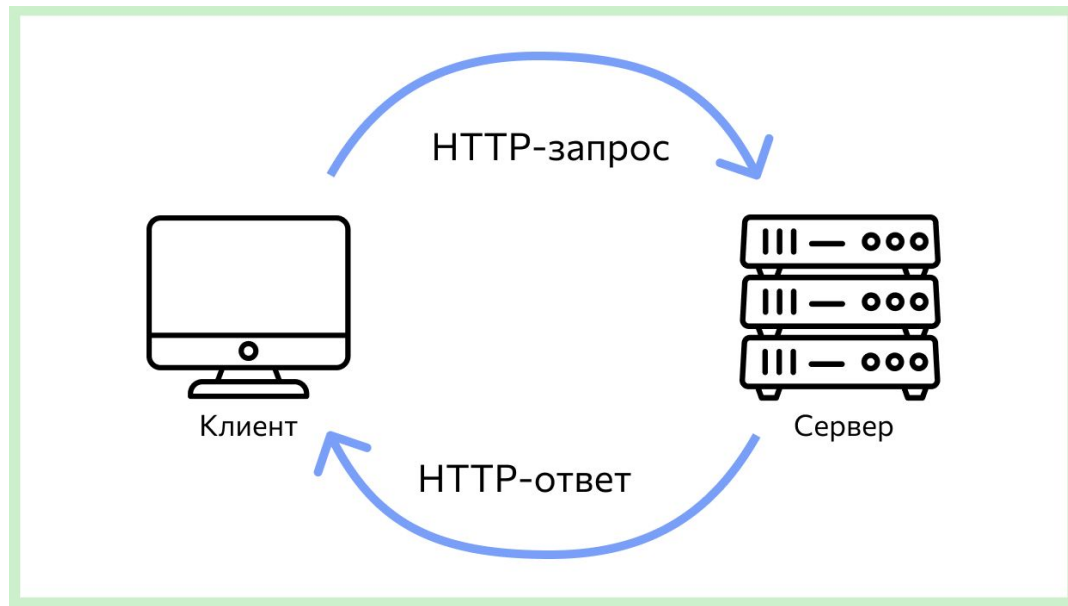


В клиент-серверной архитектуре используется три компонента:

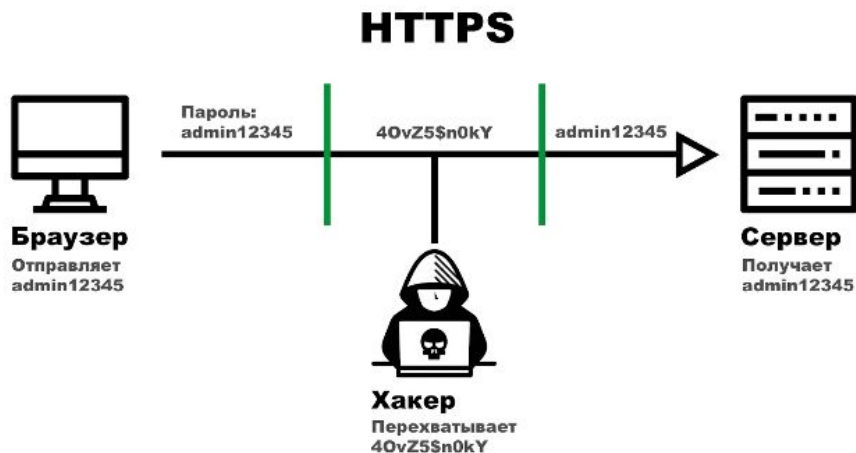
- **Клиент** — программа, которую мы используем в интернете. Чаще всего это браузер, но может быть и другая отдельная программа
- **Сервер** — компьютер, на котором хранится сайт или приложение. Когда мы заходим на сайт магазина, мы обращаемся к серверу, на котором находится сайт
- **База данных** — программа, в которой хранятся все данные приложения.



HTTP – это протокол передачи информации в интернете, который расшифровывается как «протокол передачи гипертекста» (HyperText Transfer Protocol).



HTTPS — это расширение для протокола HTTP, которое делает его безопасным. Дело в том, что данные передаются по HTTP в открытом виде. HTTPS решает эту проблему, добавляя в изначальный протокол возможность шифрования данных.

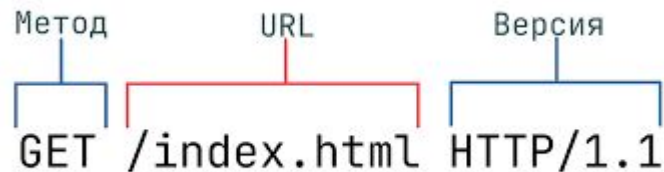


HTTP-запрос состоит из трех элементов:

- стартовой строки, которая задает параметры запроса или ответа,
- заголовка, который описывает сведения о передаче и другую служебную информацию.
- тело (его не всегда можно встретить в структуре). Обычно в нем как раз лежат передаваемые данные. От заголовка тело отделяется пустой строкой.



Стартовая строка



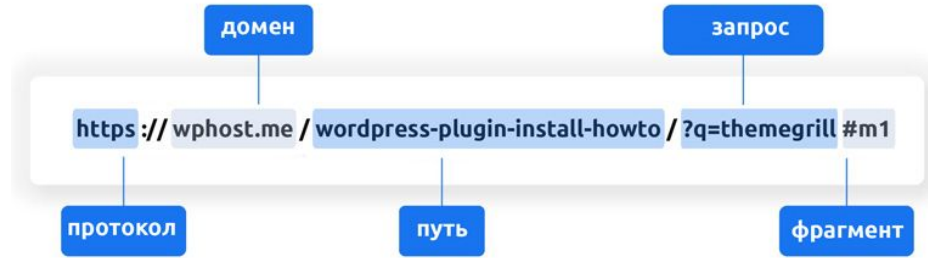
Метод – описывает, какое именно действие нужно совершить со страницей.

Самые популярные:

- **GET** (получение данных)
- **POST** (отправка данных)
- **PUT**(отправка данных)
- **DELETE** (удаление)



URL (Uniform Resource Locator) – единообразный идентификатор ресурса, идентифицирует ресурс и определяет его точное местоположение. Именно с помощью URL записаны ссылки в интернете.



Версия определяет, в соответствии с какой версией стандарта HTTP составлен запрос. Указывается как два числа, разделённых точкой (например 1.1).

headers (заголовки)

Заголовки HTTP позволяют клиенту и серверу отправлять дополнительную информацию с HTTP запросом или ответом

POST / HTTP/1.1

Host: example.com

User-Agent: Mozilla/5.0 (X11;...) Firefox/91.0

Accept: text/html, application/json

Accept-Language: ru-RU

Accept-Encoding: gzip, deflate

Connection: keep-alive

Upgrade-Insecure-Requests: 1

Content-Type: multipart/form-data; boundary=b4e4fbd93540

Content-Length: 345

Заголовки
запроса

Заголовки общего
назначения

Заголовки
представления

body (тело)

Тело сообщения опционально, оно содержит данные, связанные с запросом, либо документ (например HTML-страницу), передаваемый в ответе. Некоторые виды запросов могут отправлять данные на сервер в теле запроса

```
POST /?id=1 HTTP/1.1
```

Request line

```
Host: www.swingvy.com
Content-Type: application/json; charset=utf-8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:53.0)
Gecko/20100101 Firefox/53.0
Connection: close
Content-Length: 136
```

Header

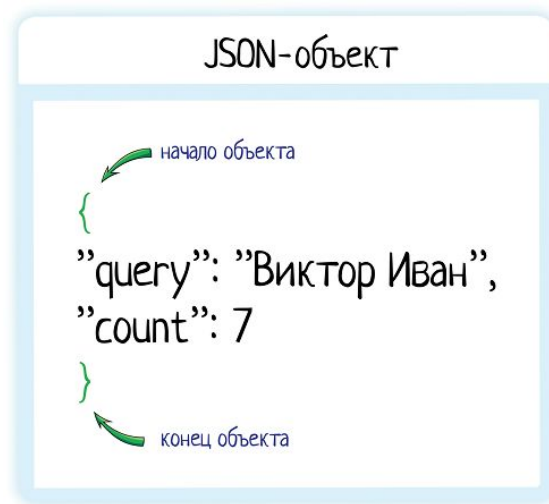
```
{
  "status": "ok",
  "extended": true,
  "results": [
    {"value": 0, "type": "int64"},
    {"value": 1.0e+3, "type": "decimal"}
  ]
}
```

Body message

Категория	Описание
200 OK	Возвращается в случае успешной обработки запроса, при этом тело ответа обычно содержит запрошенный ресурс.
302 Found	Перенаправляет клиента на другой URL. Например, данный код может прийти, если клиент успешно прошел процедуру аутентификации и теперь может перейти на страницу своей учетной записи.
400 Bad Request	Данный код можно увидеть, если запрос был сформирован с ошибками. Например, в нем отсутствовали символы завершения строки.
403 Forbidden	Означает, что клиент не обладает достаточными правами доступа к запрошенному ресурсу. Также данный код можно встретить, если сервер обнаружил вредоносные данные, отправленные клиентом в запросе.
404 Not Found	Каждый из нас, так или иначе, сталкивался с этим кодом ошибки. Данный код можно увидеть, если запросить у сервера ресурс, которого не существует на сервере.
500 Internal Error	Данный код возвращается сервером, когда он не может по определенным причинам обработать запрос.



JSON - текстовый формат данных, следующий за синтаксисом объекта JavaScript, который был популяризирован Дугласом Крокфордом. Несмотря на то, что он очень похож на буквенный синтаксис объекта JavaScript, его можно использовать независимо от JavaScript, и многие среды программирования имеют возможность читать (анализировать) и генерировать JSON.



JSON представляет собой коллекцию пар ключ-значение. Ключи и значения разделяются двоеточием, а пары ключ-значение - запятой. Объект начинается с { и заканчивается }.

```
1 {  
2   "orderID": 12345,  
3   "shopperName": "Ivan Ivanov",  
4   "shopperEmail": "ivanov@example.com",  
5   "contents": [  
6     {  
7       "productID": 34,  
8       "productName": "Super product",  
9       "quantity": 1  
10    },  
11    {  
12      "productID": 56,  
13      "productName": "Wonderful product",  
14      "quantity": 3  
15    }  
16  ],  
17  "orderCompleted": true  
18 }
```

Примечания

- JSON - это чисто формат данных - он содержит только свойства, без методов.
- JSON требует двойных кавычек, которые будут использоваться вокруг строк и имён свойств. Одиночные кавычки недействительны.
- Даже одна неуместная запятая или двоеточие могут привести к сбою JSON-файла и не работать.

Методы работы с JSON

1. Преобразует JavaScript объект в JSON-строку - `JSON.stringify(obj)`

```
const obj = { name: "John", age: 30 };  
const jsonString = JSON.stringify(obj);
```

1. Преобразует JSON-строку в JavaScript объект - `JSON.parse(jsonString)`

```
const jsonString = '{"name":"John","age":30}';  
const obj = JSON.parse(jsonString);
```

- Простой и легко читаемый синтаксис, который понятен и компьютеру, и человеку.
- Лёгкость и компактность. JSON-файлы весят меньше, чем файлы других форматов, и загружаются быстрее.
- Отсутствие зависимости от конкретного языка программирования.
- Универсальность в типах данных. JSON можно использовать для хранения массивов, упорядоченных списков и коллекций пар «ключ — значение».
- Широкая поддержка в браузерах. Все современные браузеры поддерживают работу с JSON.
- Самостоятельное документирование.



XML

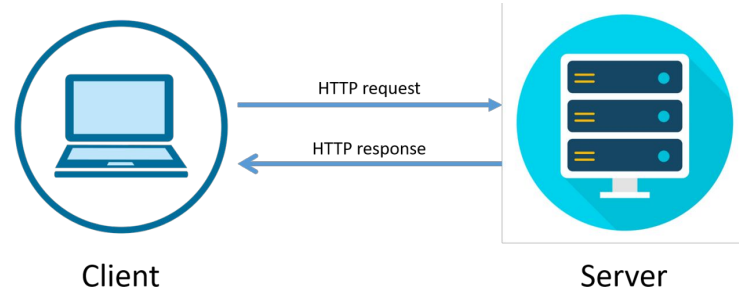
```
<?xml version="1.0" encoding="UTF-8" ?>
<person>
  <name>Иван</name>
  <age>37</age>
  <mother>
    <name>Ольга</name>
    <age>58</age>
  </mother>
  <children>
    <child>Маша</child>
    <child>Игорь</child>
    <child>Таня</child>
  </children>
  <married>true</married>
  <dog null="true" />
</person>
```

VS

JSON

```
{
  "person":{
    "name":"Иван",
    "age":37,
    "mother":{
      "name":"Ольга",
      "age":58
    },
    "children":[
      "Маша",
      "Игорь",
      "Таня"
    ],
    "married":true,
    "dog":null
  }
}
```

Fetch

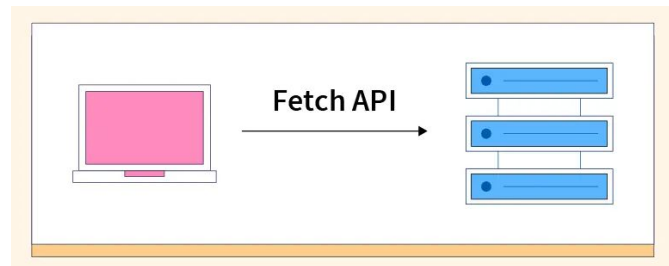


Метод `fetch()` — современный и очень мощный способ для создания сетевых запросов и получения информации с сервера

```
1 let promise = fetch(url, [options])
```

url – URL для отправки запроса.

options – дополнительные параметры: метод, заголовки и так далее.



fetch() запускает запрос и возвращает **promise**.

Когда запрос удовлетворяется, promise разрешается(resolved) объектом ответа (Response object)

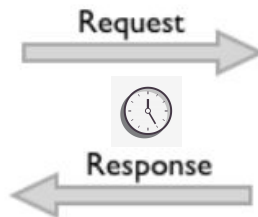
Если ответ не получается получить из-за, например, неполадок сети, **promise** отклоняется (promise is rejected).



async/await синтаксис очень помогает при работе с **fetch()**

Для примера сделаем запрос, чтобы получить информацию о фильмах:

```
async function fetchMovies() {  
  const response = await fetch('/movies');  
  // waits until the request completes...  
  console.log(response);  
}
```



Объект **Response** имеет некоторые свойства, с помощью которых можно обращаться к некоторым данным ответа

Response.status– возвращает статус код ответа

Response.statusText – возвращает текст для статус кода

Response.ok – булевское значение, которое указывает, выполнен ли запрос успешно или нет

Response.headers – объект, который описывает заголовок ответа.



Существует метод, который помогает извлечь из объекта **Response** информацию ответа в **JSON** формате:

Response.json()

Этот метод тоже возвращает **promise**, поэтому его необходимо дожидаться при помощи **await**:

```
async function f() {  
    let url = 'https://api.github.com/repos/javascript-tutorial/commits';  
    let response = await fetch(url);  
    let commits = await response.json(); // читаем ответ в формате JSON  
    alert(commits[0].author);  
}
```

Работа с ошибками



Промис завершается с ошибкой, если `fetch` не смог выполнить HTTP-запрос, например при ошибке сети или если нет такого сайта. HTTP-статусы 404 и 500 (и другие) не являются ошибкой.

```
1 let response = await fetch(url);
2
3 if (response.ok) { // если HTTP-статус в диапазоне 200-299
4   // получаем тело ответа (см. про этот метод ниже)
5   let json = await response.json();
6 } else {
7   alert("Ошибка HTTP: " + response.status);
8 }
```

Метод GET

Метод GET используется для получения данных с сервера.

```
const response = await fetch('https://api.example.com/data', {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json',
    // Дополнительные заголовки, если необходимо
  },
  // Дополнительные параметры запроса, например, параметры URL
  // Также можно использовать URLSearchParams для удобной работы с параметрами
});

const data = await response.json();
console.log('GET data:', data);
```

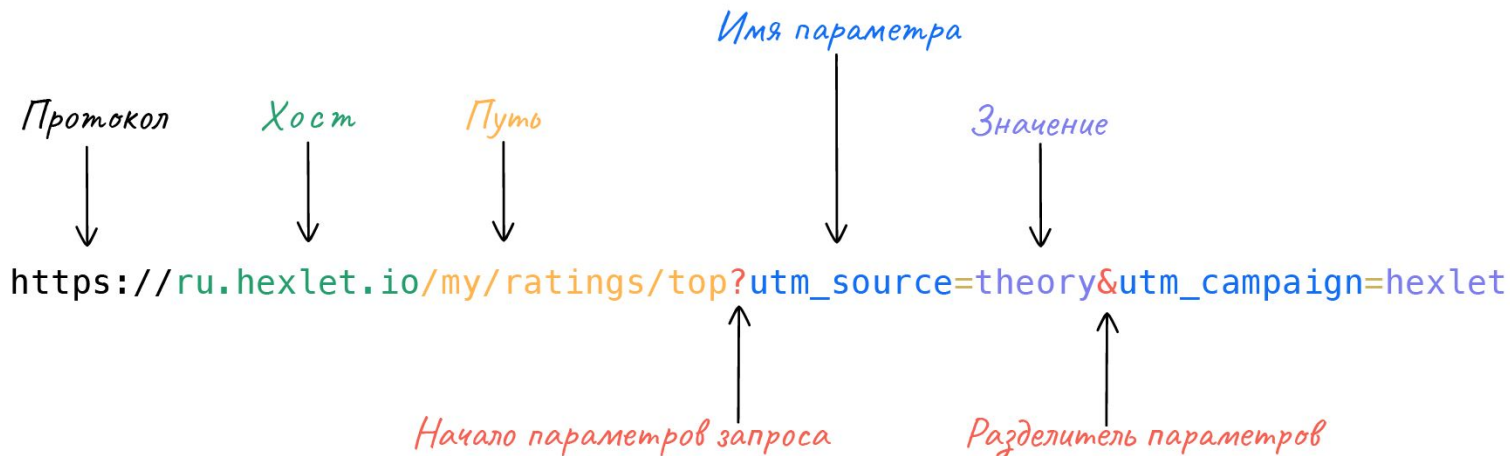
Для метода GET можно не указывать options

```
const response = await fetch('https://api.example.com/data');
```

Метод GET

GET-запросы, по определению, не предназначены для отправки данных на сервер.

Однако, иногда может потребоваться отправить данные в URL в виде параметров запроса.



Метод POST

Метод **POST** отправляет данные на сервер и создает новый ресурс

```
const response = await fetch('https://api.example.com/data', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    // Дополнительные заголовки, если необходимо
  },
  body: JSON.stringify({
    key1: 'value1',
    key2: 'value2',
    // Данные для отправки на сервер
  }),
});

const data = await response.json();
console.log('POST data:', data);
```

Метод PUT

Метод **PUT** чаще всего используется для обновления существующего ресурса. Для этого необходим URL ресурса и новая его версия.

```
const response = await fetch('https://api.example.com/data/123', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
    // Дополнительные заголовки, если необходимо
  },
  body: JSON.stringify({
    key1: 'updatedValue1',
    key2: 'updatedValue2',
    // Обновленные данные для отправки на сервер
  }),
});

const data = await response.json();
console.log('PUT data:', data);
```

Метод DELETE

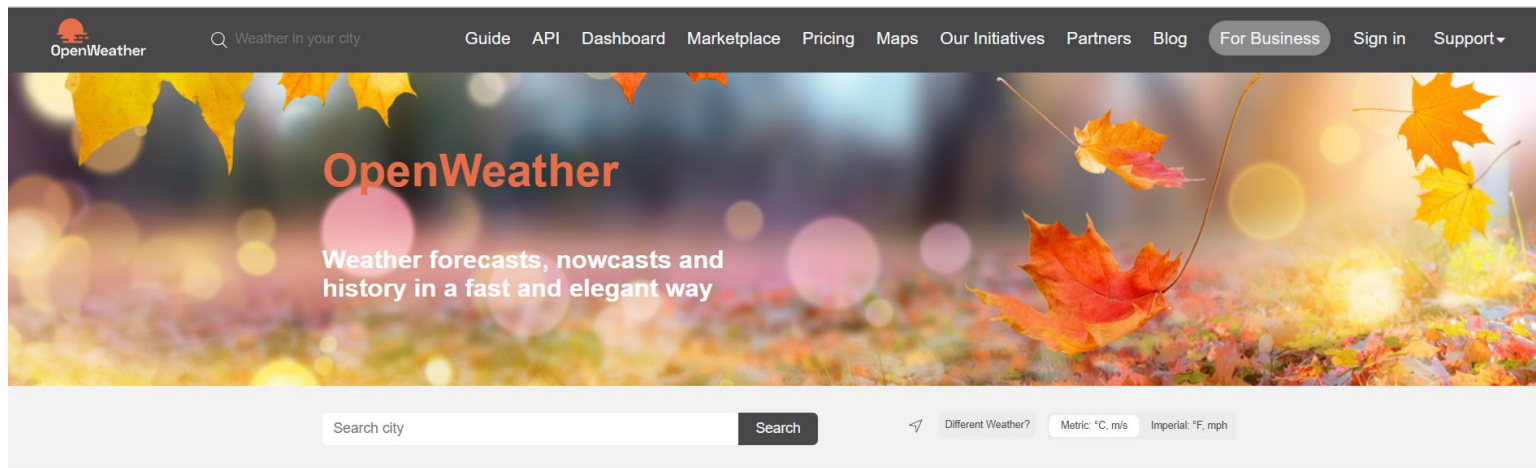
Метод **DELETE**

используется для удаления ресурса, который указывается с помощью его URL.

```
const response = await fetch('https://api.example.com/data/123', {  
  method: 'DELETE',  
  headers: {  
    'Content-Type': 'application/json',  
    // Дополнительные заголовки, если необходимо  
  },  
});  
  
const data = await response.json();  
console.log('DELETE data:', data);
```

Настройка для практической работы

Перейдите на сайт - <https://openweathermap.org/>



Nov 13, 11:03pm

London, GB

12°C

Yellow thunderstorm warning

Feels like 11°C. Broken clouds. Moderate breeze

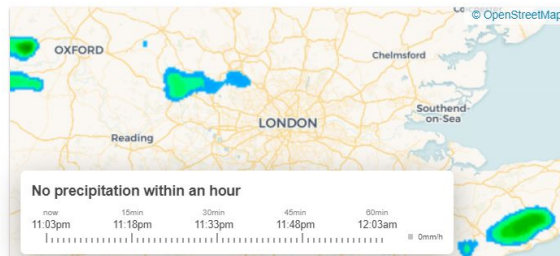
6.7m/s WSW

1001hPa

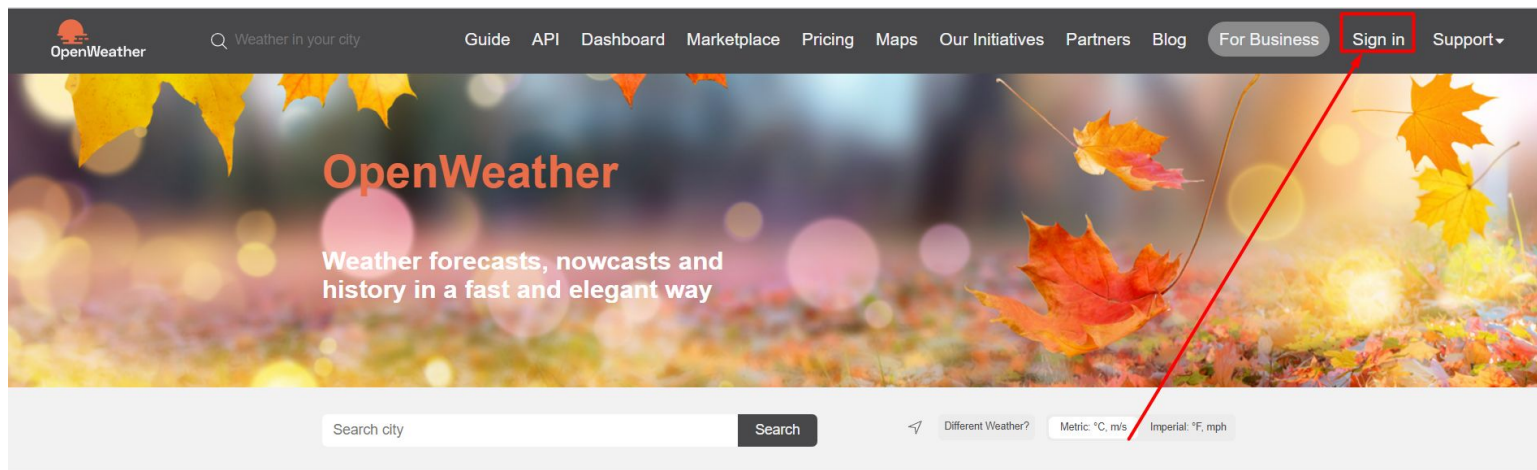
Humidity: 75%

Dew point: 8°C

Visibility: 10.0km



Перейдите во вкладку Sign in



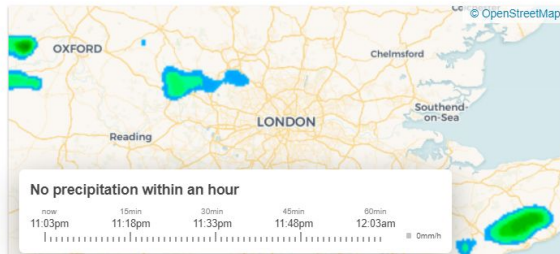
Nov 13, 11:03pm
London, GB

12°C

Yellow thunderstorm warning

Feels like 11°C. Broken clouds. Moderate breeze

➤ 6.7m/s WSW ☉ 1001hPa
Humidity: 75% Dew point: 8°C
Visibility: 10.0km



Нажмите Create an Account.

Sign In To Your Account

 Enter email

 Password

☐ Remember me

Submit

Not registered? [Create an Account.](#)

Lost your password? [Click here to recover.](#)

Product Collections

[Current and Forecast APIs](#)
[Historical Weather Data](#)
[Weather Maps](#)
[Weather Dashboard](#)
[Widnets](#)


Subscription

[How to start](#)
[Pricing](#)
[Subscribe for free](#)
[FAQ](#)

Company

OpenWeather is a team of IT experts and data scientists that has been practising deep weather data science since 2014. For each point on the globe, OpenWeather provides historical, current and forecasted weather data via light-speed APIs. Headquarters in London, UK.

Заполните необходимые поля и нажмите на кнопку Create Account



[Guide](#) [API](#) [Dashboard](#) [Marketplace](#) [Pricing](#) [Maps](#) [Our Initiatives](#) [Partners](#) [Blog](#) [For Business](#) [Sign In](#) [Support](#)


Create New Account

We will use information you provided for management and administration purposes, and for keeping you informed by mail, telephone, email and SMS of other products and services from us and our partners. You can proactively manage your preferences or opt-out of communications with us at any time using [Privacy Centre](#). You have the right to access your data held by us or to request your data to be deleted. For full details please see the [OpenWeather Privacy Policy](#).

☒ I am 16 years old and over
☒ I agree with [Privacy Policy](#), [Terms and conditions of sale](#) and [Websites terms and conditions of use](#)

I consent to receive communications from OpenWeather Group of Companies and their partners:

☐ System news (API usage alert, system update, temporary system shutdown, etc)
☐ Product news (change to price, new product features, etc)
☐ Corporate news (our life, the launch of a new service, etc)

☒ I'm not a robot 
[Privacy](#) [Terms](#)

Create Account

После регистрации появиться окно, его можно закрыть

How and where will you use our API? X

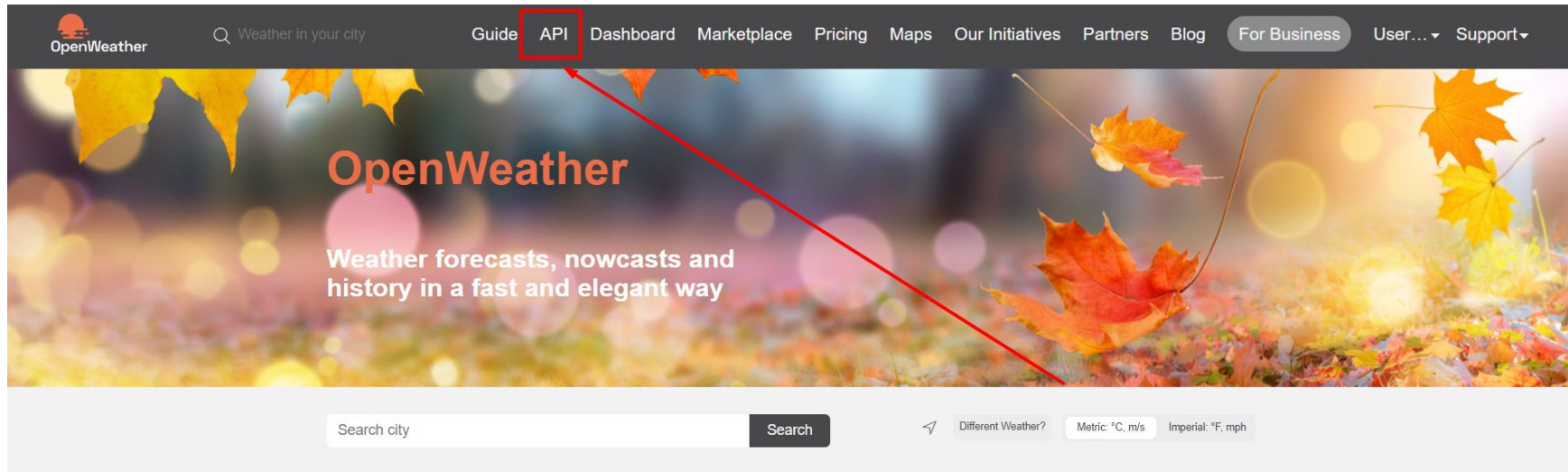
Hi! We are doing some housekeeping around thousands of our customers. Your impact will be much appreciated. All you need to do is to choose in which exact area you use our services.

Company

* Purpose

Cancel Save

Переходим на вкладку API



Nov 13, 11:52pm

London, GB

12°C

Yellow thunderstorm warning

Feels like 11°C Broken clouds Moderate breeze



Листаем вниз до раздела указанного на слайде и нажимаем на кнопку API doc

You can read the [How to Start](#) guide and enjoy using our powerful weather APIs right now.

Current & Forecast weather data collection

Current Weather Data

[API doc](#)[Subscribe](#)

- Access current weather data for any location
- We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations
- JSON, XML, and HTML formats
- Included in both free and paid subscriptions

Hourly Forecast 4 days

[API doc](#)[Subscribe](#)

- Hourly forecast is available for 4 days
- Forecast weather data for 96 timestamps
- JSON and XML formats
- Included in the Developer, Professional and Enterprise subscription plans

Daily Forecast 16 days

[API doc](#)[Subscribe](#)

- 16 days forecast is available for any location on the globe
- 1-day step for 16 days
- JSON and XML formats
- Included in all paid subscription plans

Climatic Forecast 30 days

[API doc](#)[Subscribe](#)

- Forecast weather data for 30 days
- JSON format
- Included in the Developer, Professional and Enterprise subscription plans

Bulk Download

[API doc](#)[Subscribe](#)

- Current weather, a variety of weather forecasts and their 7-day archive via regularly updated files
- Weather bulks are grouped by types of weather data and location lists (global city lists or ZIP code lists of EU, UK,

Global Weather Alerts Push notifications

[Doc](#)[Get access](#)

- Get all the **warnings from national weather agencies**
- Weather alerts are pushed to your endpoint as soon as they occur

В открывшемся окне нажимаем на раздел указанный на слайде

Current weather data

[Home](#) / [API](#) / [Current weather](#)

Product concept

Access current weather data for any location on Earth! We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations. Data is available in JSON, XML, or HTML format.

Call current weather data

How to make an API call

API call


```
https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}
```

Parameters

required Latitude. If you need the geocoder to automatic convert city names and zip-codes to geo coordinates and the other way around, please

- Product concept
- Call current weather data
 - How to make an API call
- API response
 - JSON format API response example
 - JSON format API response fields
 - XML format API response example
 - XML format API response fields
 - List of weather condition codes
 - Min/max temperature in current weather
 - API and forecast API
- Bulk downloading
- Other features
 - Geocoding API
 - Built-in geocoding
 - Built-in API request by city name**
 - Built-in API request by city ID
 - Built-in API request by ZIP code
- Format
 - Units of measurement
 - Multilingual support
 - Call back function for JavaScript code

Для нашего fetch запроса понадобится первый url с переменными {city name} и {API key}

Weather in your cityGuideAPIDashboardMarketplacePricingMapsOur InitiativesPartnersBlogFor BusinessUser...Support

Built-in API request by city name

You can call by city name or city name, state code and country code. Please note that searching by states available only for the USA locations.

API call

`https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}`

`https://api.openweathermap.org/data/2.5/weather?q={city name},{country code}&appid={API key}`

`https://api.openweathermap.org/data/2.5/weather?q={city name},{state code},{country code}&appid={API key}`

Parameters

q

required

City name, state code and country code divided by comma, Please refer to [ISO 3166](#) for the state codes or country codes. You can specify the parameter not only in English. In this case, the API response should be returned in the same language as the language of requested location name if the location is in our predefined list of more than 200,000 locations.

appid

required

Your unique API key (you can always find it on your account page)

Product concept

Call current weather data

How to make an API call

API response

JSON format API response example

JSON format API response fields

XML format API response example

XML format API response fields

List of weather condition codes

Min/max temperature in current weather

API and forecast API

Bulk downloading

Other features

Geocoding API

Built-in geocoding

Built-in API request by city name

Built-in API request by city ID

Built-in API request by ZIP code


Format

Units of measurement

Multilingual support

Call back function for JavaScript code

Чтобы получить переменную API key для вашего запроса необходимо нажать на неё в url

 Weather in your city

Guide API Dashboard Marketplace Pricing Maps Our Initiatives Partners Blog For Business User... Support

Built-in API request by city name

You can call by city name or city name, state code and country code. Please note that searching by states available only for the USA locations.

API call

```
https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
```

```
https://api.openweathermap.org/data/2.5/weather?q={city name},{country code}&appid={API key}
```

```
https://api.openweathermap.org/data/2.5/weather?q={city name},{state code},{country code}&appid={API key}
```

Parameters

q	required	City name, state code and country code divided by comma, Please refer to ISO 3166 for the state codes or country codes. You can specify the parameter not only in English. In this case, the API response should be returned in the same language as the language of requested location name if the location is in our predefined list of more than 200,000 locations.
appid	required	Your unique API key (you can always find it on your account page)

Product concept

Call current weather data

How to make an API call

API response

JSON format API response example

JSON format API response fields

XML format API response example

XML format API response fields

List of weather condition codes

Min/max temperature in current weather

API and forecast API

Bulk downloading

Other features

Geocoding API

Built-in geocoding

Built-in API request by city name

Built-in API request by city ID

Built-in API request by ZIP code

Format

Units of measurement

Multilingual support

Call back function for JavaScript code

В открывшемся окне вы можете скопировать имеющийся API key (обратите внимание на статус - он должен быть активным)


New Products Services **API keys** Billing plans Payments Block logs My orders My profile Ask a question

You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

Key	Name	Status	Actions
5c41ec20e551056f2ba6f4eaf6350ea	Default	Active	<input type="checkbox"/> <input type="checkbox"/>

Create key

API key name



Product Collections

Current and Forecast APIs
Historical Weather Data
Weather Maps
Weather Dashboard
Widgets

Subscription

How to start
Pricing
Subscribe for free
FAQ

Company

OpenWeather is a team of IT experts and data scientists that has been practising deep weather data science since 2014. For each point on the globe, OpenWeather provides historical, current and forecasted weather data via light-speed APIs. Headquarters in London, UK.



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH