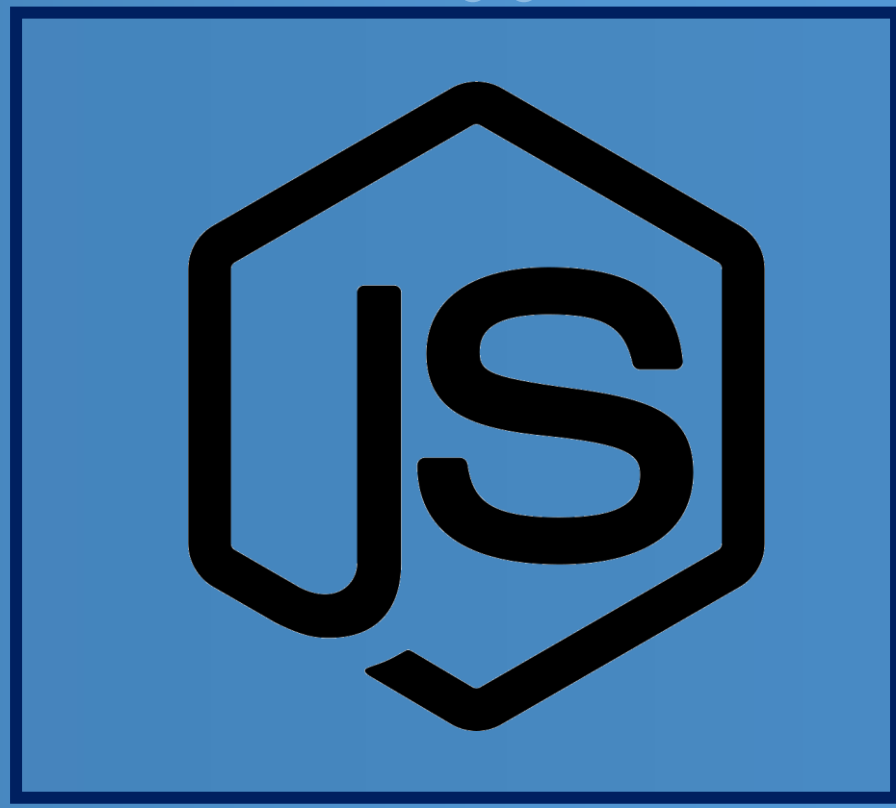
STEP IT ACADEMY

Занятие №9

JSON, Ajax



Передача данных между серверами

Отправка запросов

POSTMAN

Postman — это платформа API, позволяющая разработчикам проектировать, создавать, тестировать и повторять свои API

AJAX

Для отправки запросов используется технология Ajax.

Ajax — технология для взаимодействия с сервером без перезагрузки страницы. Технология Ajax основана на JavaScript и XML и поддерживает асинхронность, что видно из ее аббревиатуры **Asynchronous JavaScript And XML**. В конце указан формат данных XML, но, несмотря на это, также можно использовать JSON

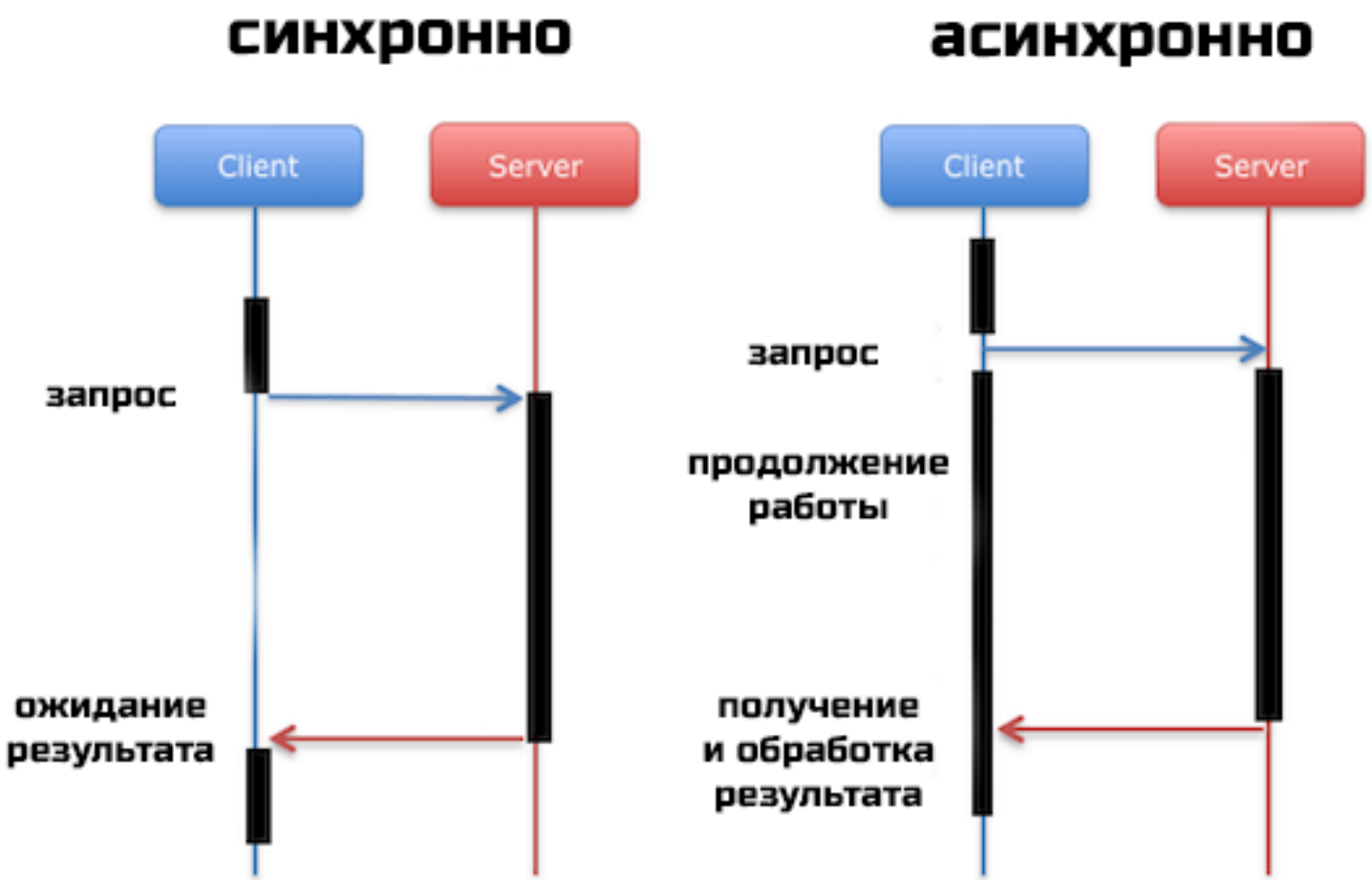
Синхронные и асинхронные запросы

Что означает **синхронность**?

Скажем, что у нас есть 2 строчки кода. Первая идет за второй. Синхронность означает то, что строка 2 не может запуститься до тех пор, пока строка 1 не закончит своё выполнение.

Что означает **асинхронность**?

В отличие от синхронности, асинхронность это модель поведения. Предположим, что у нас есть две строчки кода, первая за второй. Первая строка это код которому нужно время. Итак, первая строка начинает запуск в фоновом режиме, позволяя второй строке запуститься без ожидания завершения первой строки.


синхронно
Client
Server
запрос
ожидание
результата
асинхронно
Client
Server
запрос
продолжение
работы
получение
и обработка
результата

Всего существует **пять типов запросов:**

GET – Если вы производите этот запрос, сервер ищет информацию и отправляет ее вам назад. По сути, он производит операцию чтения на сервере. **Дефолтный тип запросов**.

POST – нужен для создания определенного ресурса на сервере. Сервер создает в базе данных новую сущность и оповещает вас, был ли процесс создания успешным.

PUT и **PATCH** – используются для обновления определенной информации на сервере. Он просто изменяет информацию существующих сущностей в базе данных и оповещает об успехе выполнения операции.

DELETE – как и следует из названия, удаляет указанную сущность из базы или сигнализирует об ошибке, если такой сущности в базе не было.

Сам же API позволяет указать, какой метод должен быть использован в определенных контекстных ситуациях.

Свойство	GET	POST
Способ передачи данных	Через URL	В теле HTTP запроса
Защита данных	Данные видны всем в адресной строке браузера, истории браузера и т.п.	Данные можно увидеть только с помощью инструментов разработчика, расширений браузера, специализированных программ.
Длина запроса	Не более 2048 символов	Не ограничена *Примечание: ограничения могут быть установлены сервером.*
Сохранение в закладки	Страница с параметрами может быть добавлена в закладки	Страница с параметрами не может быть добавлена в закладки.
Кэширование	Страница с параметрами может быть кэширована	Страница с параметрами не может быть кэширована
Индексирование поисковыми системами	Страница с параметрами может быть индексирована	Страница с параметрами не может быть индексирована
Возможность отправки файлов	Не поддерживается	Поддерживается
Поддерживаемые типы кодирования	application/x-www-form-urlencoded	application/x-www-form-urlencoded multipart/form-data text/plain
Использование в гиперссылках <a>	Да	Нет
Использование в HTML формах	Да	Да
Использование в AJAX запросах	Да	Да

API

В мире программирования есть такое понятие как
API (Application programming interface).

Если говорить простыми словами, то API - это серверы, предоставляемые разными компаниями для работы с их серверами.

Например: GitHub. Сейчас разберем на примере работы с ним.

Есть API, которая возвращает данные о пользователе по его логину.

https://api.github.com/users/**dan**

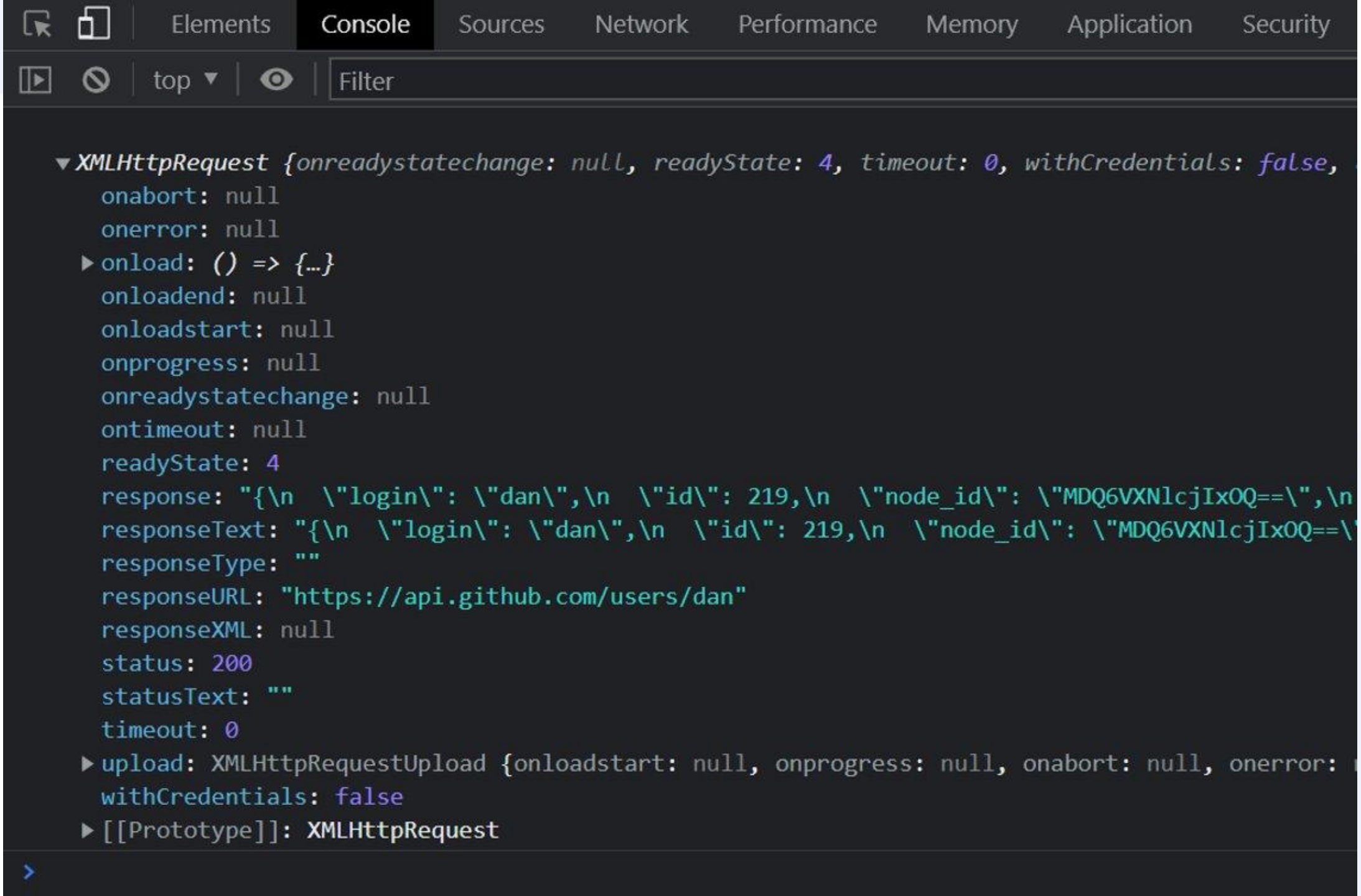
Документация:
https://docs.github.com/ru/rest/users/users?apiVersion=2022-11-28

```
<script>
    const xhr = new XMLHttpRequest(); // объект для работы с запросами

    // GET-запрос к ресурсу
    xhr.open("GET", "https://api.github.com/users/dan"); // метод open для соединения

    // обработчик получения ответа сервера
    xhr.onload = () => {
        console.log(xhr);// внутри мы можем работать с ответом когда он вернется (async)
    };

    xhr.send();     // отправляем запрос
</script>
```


Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
▼XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 0, withCredentials: false,
onabort: null
onerror: null
▶onload: () => {…}
onloadend: null
onloadstart: null
onprogress: null
onreadystatechange: null
ontimeout: null
readyState: 4
response: "{\n \"login\": \"dan\",\n \"id\": 219,\n \"node_id\": \"MDQ6VXNlcjIxOQ==\",\n
responseText: "{\n \"login\": \"dan\",\n \"id\": 219,\n \"node_id\": \"MDQ6VXNlcjIxOQ==\
responseType: ""
responseURL: "https://api.github.com/users/dan"
responseXML: null
status: 200
statusText: ""
timeout: 0
▶upload: XMLHttpRequestUpload {onloadstart: null, onprogress: null, onabort: null, onerror:
withCredentials: false
▶[[Prototype]]: XMLHttpRequest
>

Данные возвращаются в формате JSON

JSON (англ. *JavaScript Object Notation*) — текстовый формат обмена данными, основанный на *JavaScript*.

Но при этом формат **независим от JS** и может использоваться в любом языке программирования. В то же время важно их различать: JavaScript является языком программирования, а **JSON является форматом данных.**

Простота JSON привела к тому, что в настоящий момент он является наиболее популярным форматом передачи данных в среде web, вытеснив другой некогда популярный формат xml.

Объект человека в JAVASCRIPT:

```
let person = {
    firstName: "Andrey",
    lastName: "Ivanov",
    birthDate : "04.05.2000"

}
```

Представление объекта в JSON формате:

```
"{
    "firstName" : "Andrey",
    "lastName" : "Ivanov",
    "birthDate" : "04.05.2000"
}"
```

Несмотря на общее сходство, в то же время есть и различия: в JSON названия свойств заключаются в кавычки, как обычные строки.
Кроме того, **объекты JSON не могут хранить функции**, переменные, как объекты javascript.

Синтаксис JSON

Данные в нем представляются в виде пар ключ-значение в фигурных скобках.

Пары ключ-значение разделены двоеточием.

Ключи всегда должны быть заключены в двойные кавычки,
а одинарные или обратные кавычки приводят к ошибке.

Значения могут быть:

- Строкой, обязательно в двойных скобках;
- Числом;
- Логическим значением;
- Объектом, заключенным в { };
- Массивом ,заключенным в [ ];
- null

Объект JSON

Для работы с JSON форматом в JavaScript есть объект JSON.

Он предоставляет методы для конвертации JSON-строки в объект и наоборот. Кроме того, можно преобразовывать не только объекты, но и примитивные значения и массивы.

Объект JSON предоставляет две функции:

- **stringify**() — из JS в JSON (**сериализации объекта**)
- **parse**() — из JSON в JS (**десериализация или парсинг**)

Здесь мы используем метод PARSE, чтоб ответ, который мы приняли, перевести в объект JS

```
<script>
    const xhr = new XMLHttpRequest();

    xhr.open("GET", "https://api.github.com/users/dan");

    xhr.onload = () => {
        console.log(xhr.response);
        var response = JSON.parse(xhr.response); // десериализуем
        console.log(response);
    };

    xhr.send();
</script>
```

https://api.github.com/users/**dan**

В этом запросе вы можете передать любой логин (вместо dan) и вам придет ответ.

Через prompt запросить у пользователя логин и передавать его в запрос. Выводить ответ с сервера в консоль.

Создать сайт, где можно указать логин пользователя.

Отправить запрос к API и отобразить информацию на странице:

- Фото (указать, полученную ссылку, как src у img)
- Имя
- Логин
- Город
- Почта

Если какой-то информации о пользователе нет, необходимо указать, что таких данных нет.

Апи, который позволяет получить текущую погоду по координатам

https://openweathermap.org/current

На сайте необходимо зарегистрироваться, получить Апи ключ, который вы будете передавать в запрос.

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

Вместо параметров, необходимо передавать значения.
Пример:

https://api.openweathermap.org/data/2.5/weather?lat=**43.15**&lon=**76.54**&appid=**68b1479a45657cfb8272443b09a2ec9**

В ответе вы получите тело в формате JSON


```
{"coord":{"lon":76.54,"lat":43.15},"weather":
[{"id":804,"main":"Clouds","description":"overcast
clouds","icon":"04d"}],"base":"stations","main":
{"temp":271.04,"feels_like":271.04,"temp_min":271.04,"temp_max":
271.04,"pressure":1025,"humidity":81,"sea_level":1025,"grnd_leve
l":889},"visibility":10000,"wind":
{"speed":1.18,"deg":256,"gust":1.34},"clouds":
{"all":99},"dt":1672222319,"sys":
{"type":1,"id":8818,"country":"KZ","sunrise":1672194295,"sunset"
:1672226715},"timezone":21600,"id":1519293,"name":"Chemolgan","c
od":200}
```

Так же в теле ответа есть название иконки


```
{"coord":{"lon":76.54,"lat":43.15},"weather":
[{"id":804,"main":"Clouds","description":"overcast
clouds","icon":"04d"}],"base":"stations","main":
{"temp":271.04,"feels_like":271.04,"temp_min":271.04
```


Ссылка для иконки:

https://openweathermap.org/img/wn/**03d**@2x.png

Надо заменять название!

Задание

Пользователь вводит через prompt() долготу и широту. Вы отправляете запрос в openweatherapi и ответ отображает на сайте. Он должен иметь такую информацию:

- Город;
- Дата (сами создаете через JS);
- Описание погоды;
- Иконка;
- Текущая температура;
- Мин температура;
- Макс температура;
- Скорость ветра.

ВОПРОСЫ

Давайте подведем итоги.