Лабораторная работа № 13

Тема: Асинхронные операции в реальности: HTTP-запросы.

Цель:

🔷 Вариант 1

1. Запрос поста (JSONPlaceholder)
Получить данные о посте с ID=1 и вывести его title и body в консоль.

2. Список пользователей (JSONPlaceholder)
Загрузить список пользователей и вывести имена (name) всех пользователей.

3. Случайная картинка собаки (Dog API)
Сделать запрос к https://dog.ceo/api/breeds/image/random и вывести ссылку на картинку.

4. Курс валют (ExchangeRate API)
Получить курсы валют относительно USD и вывести курс доллара к евро и к йене.

5. Обработка ошибки (404)
Сделать запрос к несуществующему адресу (https://jsonplaceholder.typicode.com/unknown) и обработать ошибку: если response.ok === false, вывести "Ошибка: код <status>".

🔷 Вариант 2

1. Случайная цитата (Quotable API)
Сделать запрос к https://api.quotable.io/random и вывести цитату и её автора.

2. Информация о стране (REST Countries API)
Получить данные о Франции (https://restcountries.com/v3.1/name/france) и вывести: название, столицу и регион.

3. Случайный пользователь (Random User Generator)
Сделать запрос к https://randomuser.me/api/ и вывести имя, email и страну пользователя.

4. Картинка кота (The Cat API)
Сделать запрос к https://api.thecatapi.com/v1/images/search и вывести ссылку на картинку.

5. Сетевые ошибки (try/catch)
Вызвать fetch с заведомо неверным доменом (https://wrong-domain.example.com/data) и обработать ошибку в try/catch, выведя "Сетевая ошибка".

⚡ **Отчет должен содержать (см. образец):**
- номер и тему лабораторной работы;
- фамилию, номер группы студента и вариант задания;
- скриншоты окна VSC с исходным кодом программ;
- скриншоты с результатами выполнения программ;
- пояснения, если необходимо;
- выводы.

Отчеты в формате **pdf** отправлять на email:
colledge20education23@gmail.com

Шпаргалка: fetch, async/await и статус ответа
1. Простейший запрос с выводом статуса

```
async function demo() {
  let response = await
fetch("https://jsonplaceholder.typicode.com/todos/1");
  console.log("Статус ответа:", response.status);

  let data = await response.json();
  console.log("Данные:", data);
}

demo();
```

📌 Здесь видно:
response.status → числовой код (200, 404 и т. д.),
данные преобразуются в JSON.

2. Проверка статуса через ok

```
async function checkStatus() {
  let response = await
fetch("https://jsonplaceholder.typicode.com/unknown");
```

```
  console.log("Статус ответа:", response.status);

  if (!response.ok) {
    console.error("Ошибка HTTP:", response.status);
    return;
  }

  let data = await response.json();
  console.log("Данные:", data);
}

checkStatus();
```

📌 Если сервер вернёт 404, мы это заметим через ok.

3. Сетевая ошибка (try/catch)

```
async function networkFail() {
  try {
    let response = await fetch("https://wrong-
domain.example.com/data");
    console.log("Статус ответа:", response.status); // сюда
не дойдёт
    let data = await response.json();
    console.log("Данные:", data);
  } catch (error) {
    console.error("Сетевая ошибка:", error.message);
  }
}

networkFail();
```

📌 Ошибки сети (например, нет интернета) ловятся только через catch.

4. Работа с текстом, а не JSON

```
async function getText() {
  let response = await fetch("https://httpbin.org/html");
  console.log("Статус ответа:", response.status);

  let text = await response.text();
  console.log("HTML длиной:", text.length, "символов");
}

getText();
```

📌 У объекта Response можно вызывать не только .json(), но и .text(), .blob(), .arrayBuffer().

5. Универсальная функция для запросов

```
async function fetchData(url) {
  try {
    let response = await fetch(url);
    console.log("Статус:", response.status);

    if (!response.ok) {
      throw new Error("Ошибка HTTP " + response.status);
    }

    let data = await response.json();
    return data;
  } catch (error) {
    console.error("Ошибка запроса:", error.message);
  }
}
```

// Пример использования:

```
fetchData("https://jsonplaceholder.typicode.com/todos/2")
  .then(data => console.log("Задача:", data));
```

📌 Такая функция помогает не дублировать проверку статуса в каждом запросе.

⚡ **Мини-таблица полезных статусов HTTP**

200 — OK (успешно).
201 — Created (создано, обычно для POST).
400 — Bad Request (ошибка клиента).
401 — Unauthorized (нужна авторизация).
403 — Forbidden (доступ запрещён).
404 — Not Found (ресурс не найден).
500 — Internal Server Error (ошибка сервера).

Используйте response.status для проверки кода.

response.ok помогает быстро понять, успешный ли запрос.
Сетевые ошибки ловятся только в try/catch.
Помимо .json() есть и другие методы для чтения ответа.
Универсальная функция fetchData экономит время и уменьшает ошибки.