Міністерство освіти і науки України
Львівський національний університет імені Івана Франка

Звіт
про виконання лабораторної роботи №6
на тему:
**«Асинхронні запити з використанням fetch та xhr»**

Виконала:
Студентка групи ФеС-32
Філь Дарина

Львів - 2025

***Мета роботи:***

Зрозуміти, як виконуються HTTP-запити в JavaScript різними способами.

***Теоретичні відомості:***

**Що таке XHR (XMLHttpRequest)?**

XMLHttpRequest (або XHR) – це стара технологія для виконання HTTP-запитів у JavaScript, яка з'явилася ще у 1999 році. Вона дозволяє отримувати дані з сервера без перезавантаження сторінки.

До появи fetch та axios, XHR був основним способом виконання AJAX-запитів.

**Що таке AJAX у JavaScript?**

AJAX (Asynchronous JavaScript and XML) — це технологія, яка дозволяє веб-сторінкам отримувати та надсилати дані з сервера без перезавантаження.

Вона використовується для створення динамічних веб-додатків, де контент оновлюється в реальному часі(наприклад, повідомлення в чаті, оновлення лайків, фільтрація товарів на сайті).

**Як працює AJAX?**

1. Користувач виконує дію (натискає кнопку, вводить текст тощо).

2. JavaScript надсилає запит на сервер.

3. Сервер обробляє запит і повертає відповідь (JSON, HTML, XML тощо).

4. JavaScript оновлює сторінку без перезавантаження.

**Технології, що використовуються в AJAX:**

- XMLHttpRequest (старий метод, XHR)
- fetch (сучасний метод)
- axios (бібліотека для HTTP-запитів)
- WebSockets (для двостороннього зв'язку в реальному часі)

**Що таке Promise в JavaScript?**

Promise – це об'єкт, який представляє результат асинхронної операції. Він може перебувати в одному з трьох станів:

● Pending (Очікування) – операція ще не завершена.

● Fulfilled (Виконано) – операція успішно завершена.

● Rejected (Відхилено) – сталася помилка.

***Хід роботи:***

1. Створити html сторінку. Використовуючи javascript та ajax зробити запити до api та написати код який відображає 10 елементів кожні 5 секунд використовуючи різні підходи:

● Xhr

● Fetch

● async/await

Створення html сторінки:

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Цікаві факти 💡</title>
</head>
<body>
  <h2>Цікаві факти 💡</h2>
  <div id="main"></div>

  <script src="main.js"></script>
</body>
</html>
```

Функція додавання даних на екран:

```
let page = 1;

function add_data(factsArray) {
    const main = document.getElementById('main');
    factsArray.forEach(text => {
        const p = document.createElement('p');
        p.textContent = '💡 ' + text;
        main.appendChild(p);
    });
}
```

Реалізація підходу XHR:

```
function fetchDatabyXHR() {
    let count = 0;
    const results = [];

    function requestOne() {
        const xhr = new XMLHttpRequest();
        xhr.open('GET', 'https://uselessfacts.jsph.pl/api/v2/facts/random?language=en', true);

        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                const data = JSON.parse(xhr.responseText);
                results.push(data.text);
                count++;
                if (count < 10) {
                    requestOne();
                } else {
                    add_data(results);
                    page++;
                    setTimeout(fetchDatabyXHR, 5000);
                }
            }
        };

        xhr.send();
    }
```

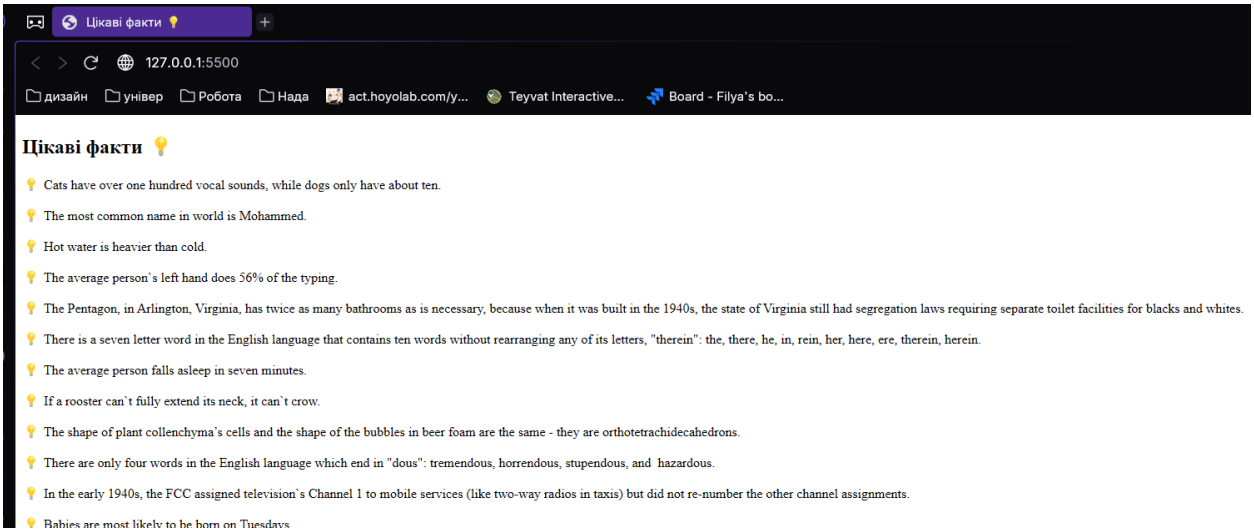
Реалізація підходу Fetch:

```
function fetchDatabyfetch() {
    const urls = Array.from({ length: 10 }, () =>
        fetch('https://uselessfacts.jsph.pl/api/v2/facts/random?language=en')
            .then(res => res.json())
            .then(data => data.text)
    );

    Promise.all(urls)
        .then(facts => {
            add_data(facts);
            page++;
            setTimeout(fetchDatabyfetch, 5000);
        })
        .catch(console.error);
}
```

Реалізація підходу Async:

```
async function fetchDatabyasync() {
    try {
        const promises = Array.from({ length: 10 }, async () => {
            const res = await fetch('https://uselessfacts.jsph.pl/api/v2/facts/random?language=en');
            const data = await res.json();
            return data.text;
        });

        const facts = await Promise.all(promises);
        add_data(facts);
        page++;
        setTimeout(fetchDatabyasync, 5000);
    } catch (error) {
        console.error(error);
    }
}
```

Вивід результату:




***Висновок:***

У цій лабораторній роботі я розглянула три підходи до виконання HTTP-запитів у JavaScript: XMLHttpRequest, fetch та async/await. Метод XMLHttpRequest є класичним, проте дещо застарілим — він вимагає більше коду, працює на основі станів і є менш зручним у підтримці. Fetch API забезпечує сучасний підхід, заснований на промісах, і дозволяє писати значно чистіший та лаконічніший код.
Async/await працює поверх fetch, даючи змогу ще більше спростити логіку за рахунок послідовного синтаксису.

Особисто для мене найбільш зручним виявився підхід fetch, оскільки він має просту структуру та читається інтуїтивно. Він ідеально підходить для виконання циклічних або однотипних запитів, як-от отримання фактів з API. У складніших ситуаціях, де необхідно обробляти кілька залежних запитів або помилки — async/await може бути ще кращим вибором.

## Додаток

**Код програми:**

**Index.html**

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Цікаві факти 💡</title>
</head>
<body>
  <h2>Цікаві факти 💡</h2>
  <div id="main"></div>

  <script src="main.js"></script>
</body>
</html>
```

**main.js**

```
let page = 1;

function add_data(factsArray) {
    const main = document.getElementById('main');
    factsArray.forEach(text => {
        const p = document.createElement('p');
        p.textContent = '💡 ' + text;
        main.appendChild(p);
    });
}

function fetchDatabyfetch() {
    const urls = Array.from({ length: 10 }, () =>
```

```
        fetch('https://uselessfacts.jsph.pl/api/v2/facts/random?language=en')
            .then(res => res.json())
            .then(data => data.text)
    );

    Promise.all(urls)
        .then(facts => {
            add_data(facts);
            page++;
            setTimeout(fetchDatabyfetch, 5000);
        })
        .catch(console.error);
}

async function fetchDatabyasync() {
    try {
        const promises = Array.from({ length: 10 }, async () => {
            const res = await
fetch('https://uselessfacts.jsph.pl/api/v2/facts/random?language=en');
            const data = await res.json();
            return data.text;
        });

        const facts = await Promise.all(promises);
        add_data(facts);
        page++;
        setTimeout(fetchDatabyasync, 5000);
    } catch (error) {
        console.error(error);
    }
}

function fetchDatabyXHR() {
    let count = 0;
    const results = [];

    function requestOne() {
        const xhr = new XMLHttpRequest();
        xhr.open('GET',
'https://uselessfacts.jsph.pl/api/v2/facts/random?language=en', true);
```

```
        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                const data = JSON.parse(xhr.responseText);
                results.push(data.text);
                count++;
                if (count < 10) {
                    requestOne();
                } else {
                    add_data(results);
                    page++;
                    setTimeout(fetchDatabyXHR, 5000);
                }
            }
        };

        xhr.send();
    }

    requestOne();
}
fetchDatabyfetch();
```