МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

**Лабораторна робота №3**

з дисципліни «Технології програмування»

Підгрупа 1

**Виконав:**

студент гр. ІР-21

Петро Кравченко

**Перевірив:**

Бондаренко О.С.

Зараховано від

\_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                      (підпис викладача)

Київ-2024

**ЛАБОРАТОРНА РОБОТА №3**

**Функції та методи функцій**

**Мета:** закріпити знання про створення та використання функцій у JavaScript,

навчитися застосовувати різні підходи до виклику та організації функцій (звичайні,

стрілкові, рекурсія, замикання), а також відпрацювати практику обробки та

перетворення даних у масивах засобами мови.

**ЗАВДАННЯ**

У всіх завданнях студенти мають:

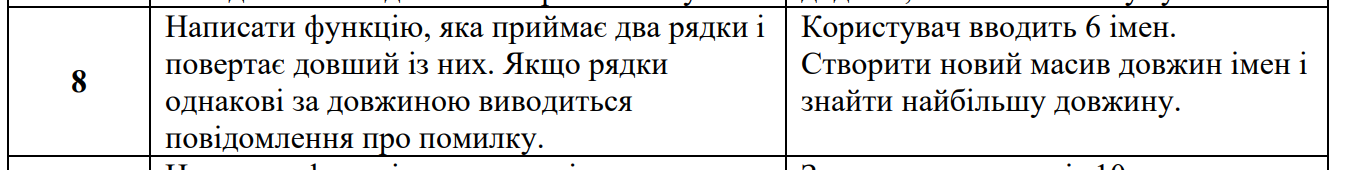
• реалізовувати власні функції (звичайні чи стрілкові) для обробки даних;

• використовувати методи масивів (map, filter, reduce, find, some, every, sort

тощо) там, де це доцільно;

• перевіряти коректність вхідних даних і виводити результати у зручному

для сприйняття форматі.



Index.html

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>LB\_3</title>

</head>

<body>

    <h1>Task 1</h1>

    <label for="sen1">Enter the first sentence:</label>

    <input type="text" id="sen1" name="sentence">

    <p id="result1"></p>

    <label for="sen2">Enter the second sentence:</label>

    <input type="text" id="sen2" name="sentence">

    <p id="result2"></p>

    <button onclick="compareSentences()">Check</button>

    <h1>Task 2</h1>

    <label for="six\_name">Enter six names (через пробіл):</label>

    <input type="text" id="six\_name" name="sentence">

    <button onclick="findLongestName()">Find longest</button>

    <script src="task1.js"></script>

    <script src="task2.js"></script>

</body>

</html>

task1.js

function compareSentences() {

    const sen1 = document.getElementById("sen1").value.trim();

    const sen2 = document.getElementById("sen2").value.trim();

    if (!sen1 || !sen2) {

        alert("Enter both sentence");

        return;

    }

    const len\_sen\_one = sen1.length;

    const len\_sen\_sec = sen2.length;

    if (len\_sen\_one > len\_sen\_sec) {

        document.getElementById("result1").innerText = "First sentence longer!";

        document.getElementById("result2").innerText = "";

    } else if (len\_sen\_sec > len\_sen\_one) {

        document.getElementById("result2").innerText = "Second sentence longer!";

        document.getElementById("result1").innerText = "";

    } else {

        alert("Sentences are equal in length!");

    }

}

task2.js

function findLongestName() {

    const input = document.getElementById("six\_name").value.trim();

const arr=input.split(/\s+/).filter(n=>n!=="");

    if (arr.length !== 6) {

        alert("Enter exactly six names!");

        return;

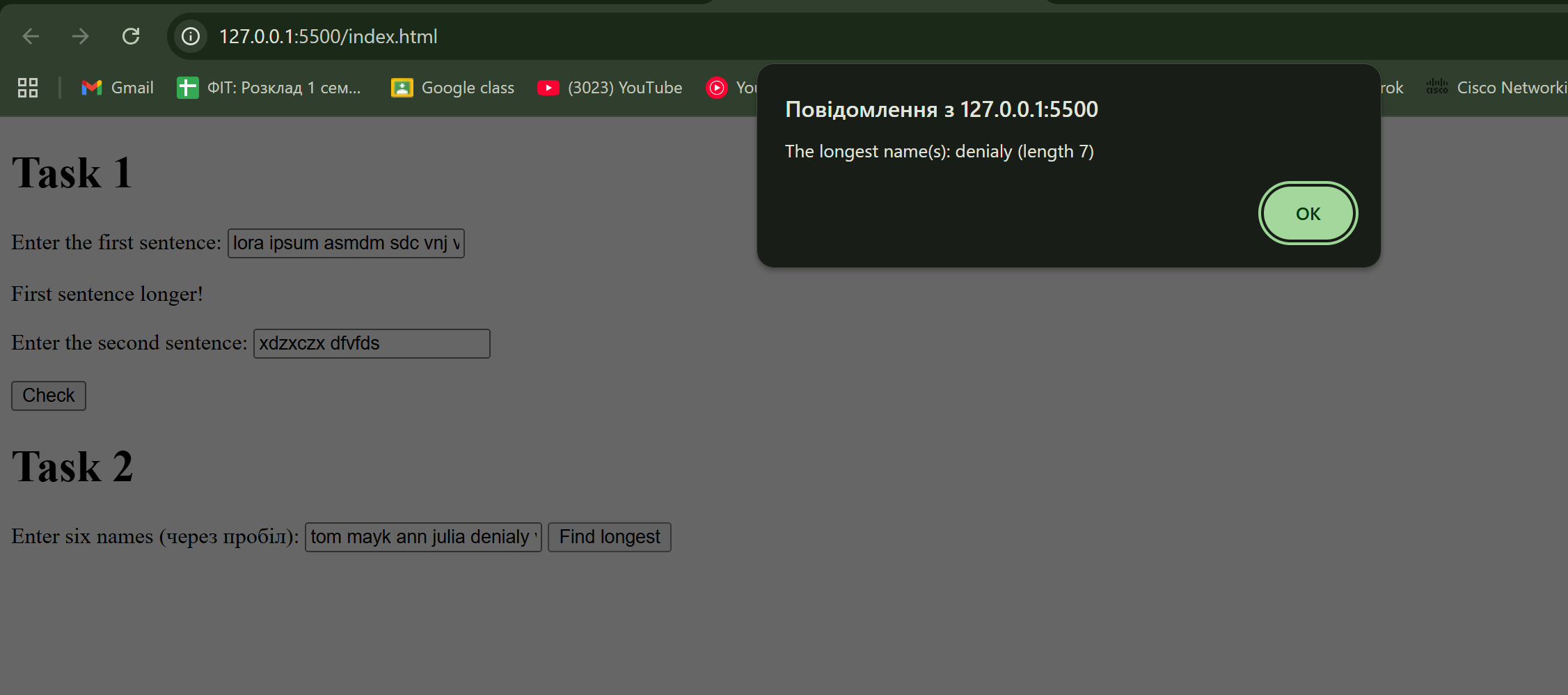
    }

    let max = Math.max(...arr.map(n => n.length));

    let nameMax = arr.filter(n => n.length === max);

    alert("The longest name(s): " + nameMax.join(", ") + " (length " + max + ")");

}



**Що значить /\s+/ у регулярному виразі**

Коротко: /\s+/ — це шаблон, який **збирає будь-яку послідовність пробільних символів (один або більше)**. Розберемо по частинах.

* /.../ — це літерал регулярного виразу в JavaScript.  
  Альтернатива: new RegExp("\\s+") (в рядку слеш потрібно екранувати).
* \s — клас **пробільних** символів. Це не тільки звичайний пробіл " ", а й:
  + табуляція \t
  + перенос рядка \n
  + повернення каретки \r
  + інші невидимі пробільні символи (форм-фід, вертикальна табуляція тощо).  
    Тобто \s = «будь-який пробіл».
* + — квантифікатор «**один або більше**».  
  Тому \s+ означає «один або більше пробільних символів підряд».

**Контрольні питання**

**1. Що таке функція в JavaScript?**  
Функція в JavaScript — це блок коду, який можна викликати за іменем. Вона приймає аргументи, виконує дії та може повертати значення.

**2. Які є способи оголошення функцій у JavaScript? Поясніть різницю між Function Declaration та Function Expression.**

* **Function Declaration**:

function add(a, b) {

return a + b;

}

Функція доступна у всьому коді завдяки підйому (hoisting).

* **Function Expression**:

const add = function(a, b) {

return a + b;

};

Функція створюється тільки тоді, коли виконання доходить до цього рядка.  
**Різниця**: Function Declaration піднімається вгору коду, а Function Expression — ні.

**3. Що таке стрілкова функція? У чому її переваги й обмеження порівняно зі звичайними функціями?**  
Стрілкова функція — це скорочений синтаксис функції:

const add = (a, b) => a + b;

**Переваги:**

* короткий запис
* не має власного this, зручно в колбеках  
  **Обмеження:**
* не можна викликати як конструктор (new)
* немає власного arguments
* не підходить для методів об’єктів, які працюють з this.

**4. Що означає поняття «функції – об’єкти першого класу» у JavaScript?**  
Функції можна:

* зберігати у змінних
* передавати як аргументи
* повертати з інших функцій
* додавати властивості  
  Тобто функції в JS — це такі ж об’єкти, як і все інше.

**5. Для чого потрібні параметри та аргументи функцій? Як можна перевірити правильність типів аргументів?**

* **Параметри** — це змінні, які функція очікує.
* **Аргументи** — це значення, які передаються під час виклику.  
  Перевірка типів:

function add(a, b) {

if (typeof a !== "number" || typeof b !== "number") {

throw new Error("Arguments must be numbers");

}

return a + b;

}

**6. Що таке область видимості змінних? Які типи областей існують у JS (глобальна, функціональна, блочна)?**  
Область видимості — це частина коду, де змінна доступна.

* **Глобальна** — змінна доступна всюди.
* **Функціональна** (var) — змінна видима лише всередині функції.
* **Блочна** (let, const) — змінна видима лише всередині блоку {}.

**7. Що таке замикання? Наведіть приклад його використання.**  
Замикання — це функція, яка «пам’ятає» змінні з області, де була створена.

function counter() {

let count = 0;

return function() {

count++;

return count;

}

}

const c = counter();

c(); // 1

c(); // 2

**8. Що таке рекурсія? Які є переваги й недоліки рекурсивних функцій?**  
Рекурсія — це коли функція викликає сама себе.  
**Переваги:** код коротший, зручний для дерев і математичних задач.  
**Недоліки:** може викликати переповнення стека, часто повільніше за цикли.

**9. Які методи роботи з масивами ви знаєте? У чому різниця між ітераційними методами (forEach, map, filter, reduce)?**

* forEach — перебирає елементи, нічого не повертає.
* map — створює новий масив з результатами.
* filter — створює новий масив з елементами, що відповідають умові.
* reduce — зводить масив до одного значення (наприклад, сума).

**10. У чому полягає різниця між методами call, apply та bind? Наведіть приклади використання.**

* call — викликає функцію з вказаним this і аргументами через кому.

sum.call(null, 1, 2);

* apply — те саме, але аргументи передаються масивом.

sum.apply(null, [1, 2]);

* bind — повертає нову функцію з прив’язаним this.

const newSum = sum.bind(null, 1);

newSum(2); // 3