

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА
ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Лабораторна робота №3
з дисципліни « **Технології програмування**»

Варіант №21

Виконав:

студент гр. ІР-21

Анна Шмідт

Перевірив:

Пирог Микола
Володимирович

Зараховано від

____.____.____

(підпис
викладача)

Київ-2025

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.

21	Написати функцію, яка перевіряє, чи число парне. Якщо аргумент не число виводиться повідомлення про помилку.	Згенерувати масив із 10 випадкових чисел. Перетворити їх у логічні значення (парне - true, непарне - false) і підрахувати кількість парних.
----	--	---

Завдання 1

```

1
2 // Підключаємо модуль для читання вводу
3 const readline = require('readline');
4
5 // Налаштовуємо інтерфейс для вводу/виводу
6 const rl = readline.createInterface({
7   input: process.stdin,
8   output: process.stdout
9 });
10
11 /**
12  * Стрілкова функція перевіряє, чи є число парним.
13  */
14 const isEven = (num) => num % 2 === 0;
15
16 // Функція для запиту вводу користувача
17 const runTask21_1 = () => {
18   rl.question("Введіть ціле число для перевірки: ", (input) => {
19     // Перетворюємо рядок в число
20     const testNumber = parseInt(input.trim());
21
22     // Перевірка
23     if (isNaN(testNumber)) {
24       console.log("Помилка: Ви ввели не коректне число. Спробуйте ще раз.");
25     } else {
26       console.log(`Число для перевірки: ${testNumber}`);
27       console.log(`Результат (парне?): ${isEven(testNumber)}`);
28     }
29     // Закриваємо інтерфейс вводу
30     rl.close();
31   });
32 };
33
34 // Запуск завдання
35 runTask21_1();

```

Введіть ціле число для перевірки: 22

Число для перевірки: 22

Результат (парне?): true

=== Code Execution Successful ===

Введіть ціле число для перевірки: 33

Число для перевірки: 33

Результат (парне?): false

=== Code Execution Successful ===




Введіть ціле число для перевірки: к

Помилка: Ви ввели не коректне число. Спробуйте ще раз.

=== Code Execution Successful ===

Завдання 2

main.js

 Share

Run

```
1 // 1. Створення масиву з 10 випадкових чисел (0-99)
2 const randomNumbers = Array.from({ length: 10 }, () => Math.floor(Math.random() *
   100));
3
4 // 2. Перетворення у логічні значення та одночасний підрахунок парних
5 // Використовуємо метод масиву `reduce` для ефективності
6 const result = randomNumbers.reduce((acc, num) => {
7   const isNumEven = num % 2 === 0;
8
9   // Додаємо логічне значення та оновлюємо лічильник
10  acc.bools.push(isNumEven);
11  if (isNumEven) {
12    acc.count++;
13  }
14  return acc;
15 }, { bools: [], count: 0 }); // Початкове значення акумулятора
16
17
18 // Виведення результатів
19 console.log(`Згенерований масив чисел: ${randomNumbers.join(', ')}`);
20 console.log(`Масив логічних значень (парне: true): ${result.bools.join(', ')}`);
21 console.log(`Кількість парних чисел: ${result.count}`);
```

Згенерований масив чисел: [90, 22, 48, 84, 57, 1, 0, 93, 35, 14]
Масив логічних значень (парне: true): [true, true, true, true, false, false, true, false, false, true]
Кількість парних чисел: 6

=== Code Execution Successful ===

Контрольні питання

1. **Що таке функція в JavaScript?** Це **блок коду**, призначений для виконання певного завдання, який є **об'єктом першого класу** (може зберігатися, передаватися та повертатися).
2. **Способи оголошення функцій. Різниця між Function Declaration та Function Expression.**
 - a. **Function Declaration:** `function name() { ... }`. Підлягає **Hoisting** (підняттю), тобто може бути викликана до оголошення.
 - b. **Function Expression:** `const name = function() { ... }`. Не підлягає Hoisting, викликається **тільки після** присвоєння змінній.
 - c. Також є **Arrow Function**.
3. **Що таке стрілкова функція? Переваги й обмеження.**
 - a. **Стрілкова функція** (`=>`) – це лаконічний синтаксис ES6.
 - b. **Перевага:** Має **лексичний this** (успадковує контекст від батьківської області видимості), що вирішує багато проблем із контекстом у колбеках.
 - c. **Обмеження:** Не має власного `this`, не має об'єкта `arguments`, не може використовуватися як конструктор (`new`).
4. **Що означає поняття «функції – об'єкти першого класу»?** Означає, що функції можна розглядати як будь-яку іншу сутність чи значення: **присвоювати** змінним, **передавати** як аргументи та **повертати** з інших функцій.
5. **Для чого потрібні параметри та аргументи? Як перевірити типи?**
 - a. **Параметри** (у визначенні) та **Аргументи** (при виклику) потрібні для забезпечення гнучкості та багаторазового використання коду з різними вхідними даними.
 - b. Перевірка типів здійснюється головним чином за допомогою оператора **`typeof`** (для примітивних типів) та **`instanceof`** (для об'єктів/класів), а також **`isNaN()`** для чисел.
6. **Що таке область видимості? Які типи областей існують?**
 - a. **Область видимості (Scope)** — це правило, яке визначає, де в коді доступна змінна.
 - b. **Типи:**
 - i. **Глобальна:** Доступна всюди.
 - ii. **Функціональна:** Створюється функцією (для змінних `var`).
 - iii. **Блочна:** Створюється блоком `{ }` (для змінних `let` і `const`).
7. **Що таке замикання? Наведіть приклад.**
 - a. **Замикання (Closure)** — це здатність функції **пам'ятати та отримувати доступ** до змінних із зовнішньої функції **навіть після** завершення її виконання.
 - b. *Приклад:*

JavaScript

```
function makeCounter() { let count = 0; return () => ++count; }  
const counter = makeCounter();  
counter(); // 1
```

8. Що таке рекурсія? Переваги й недоліки.

- a. **Рекурсія** — це техніка, коли функція **викликає сама себе**.
- b. **Переваги**: Чистий код для природно рекурсивних завдань (наприклад, обхід дерева).
- c. **Недоліки**: Ризик **Stack Overflow** (переповнення стека) при глибоких викликах, часто повільніше за ітерацію.

9. Методи роботи з масивами. Різниця між `forEach`, `map`, `filter`, `reduce`.

- a. **`forEach`**: Ітерує масив, виконуючи дію. **Нічого не повертає** (`undefined`).
- b. **`map`**: **Перетворює** кожен елемент, повертаючи **новий масив** тієї ж довжини.
- c. **`filter`**: **Відбирає** елементи за умовою, повертаючи **новий масив** меншої або рівної довжини.
- d. **`reduce`**: **Згортає** масив до **єдиного значення** (сума, об'єкт, зведення).

10. Різниця між методами `call`, `apply` та `bind`. Усі вони явно встановлюють контекст `this` для функції.

- a. **`call`**: **Викликає** функцію негайно. Аргументи передаються **списком**.
- b. **`apply`**: **Викликає** функцію негайно. Аргументи передаються **масивом**.
- c. **`bind`**: **Повертає нову функцію** з фіксованим `this`, яку можна викликати пізніше.