МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Навчально-науковий інститут електричної інженерії
та інформаційних технологій

КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ЗВІТ

З ЛАБОРАТОРНИХ РОБІТ

З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

«Frontend-розробка»

Виконав студент групи КН-23-1

Полинько Ігор Миколайович

Перевірив старший викладач кафедри АІС Притчин О. С.

КРЕМЕНЧУК 2025

Лабораторна робота № 3

Тема: Створення інтерактивного стейтфул компонента в React з обробкою подій.

Мета: Познайомитись і отримати головні навички роботи з інтерактивним стейтфул компонентом React з обробкою подій.

Виконання завдання лабораторної роботи:

Завдання 1. Додайте звичайну змінну в компонент App. Передайте в ваш кастомний компонент кілька пропсів. Це мають бути пропси різного типу, на кшталт створеної раніше змінної, простого значення та виразу.

Прийміть пропси в кастомному компоненті та виведіть їх на екран. Можна деякі з них вивести не напряму, а використати їх як частину якогось виразу.

Додайте в кастомний компонент кілька стейт-змінних та зробіть на їх основі, щоб програма давала можливість користувачу ввести якийсь текст і виводила його на екран.

Також додайте можливість змінювати якесь значення виведене на екран по натисканню на кнопку. Наприклад збільшення чи зменшення числового значення.

На додачу додайте можливість щоб програма питала користувача про підтвердження і, в разі позитивної відповіді, змінювала кілька стилів різних елементів або приховувала чи показувала якісь елементи. Можна імплементувати обидва під-завдання(і стилі і приховування/показування елементів) чи навіть зробити кілька кнопок, щоб цей функціонал можна було вмикати чи вимикати.

App.jsx:

```
import "./App.css";
import MyInfo from "./components/MyInfo";

function App() {
  // звичайна змінна
  const myName = "Ігор";
  const myAge = 20;

  return (
    <div className="App">
      <MyInfo
```

```
        name={myName}          // змінна
        age={myAge}            // змінна
        favNumber={7}          // просте значення
        expression={10 + 5}    // вираз
        isStudent={true}       // булеве
      />
    </div>
  );
}

export default App;
```

MyInfo.jsx:

```
import { useState } from "react";
import "./MyInfo.css";

function MyInfo({ name, age, favNumber, expression, isStudent }) {
  // стейт для введення тексту
  const [inputValue, setInputValue] = useState("");

  // стейт для числового значення
  const [count, setCount] = useState(0);

  // стейт для зміни стилю/приховування
  const [isVisible, setIsVisible] = useState(true);
  const [isStyled, setIsStyled] = useState(false);

  // підтвердження від користувача
  const handleToggle = () => {
    if (window.confirm("Змінити вигляд елементів?")) {
      setIsVisible(!isVisible);
      setIsStyled(!isStyled);
    }
  };

  return (
    <div className={`my-info ${isStyled ? "highlight" : ""}`}>
      <h1>Привіт, я {name}!</h1>
      <p>Мені {age} років.</p>
      <p>Моє улюблене число: {favNumber}</p>
      <p>Вираз 10+5 = {expression}</p>
      <p>{isStudent ? "Я студент" : "Я вже випускник"}</p>

      <hr />

      {/* введення тексту */}
      <h2>Ввести текст:</h2>
      <input
        type="text"
        placeholder="Написати текст..."
```

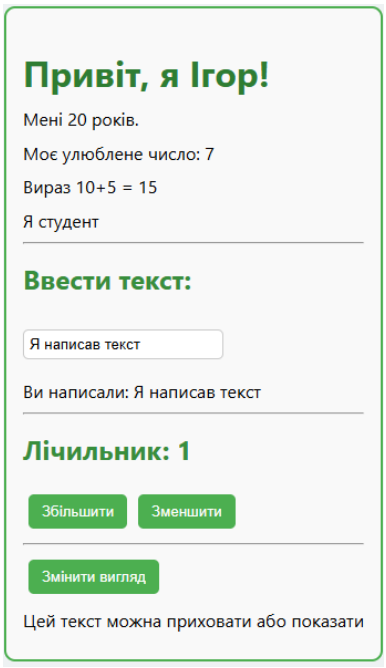
```
        value={inputValue}
        onChange={(e) => setInputValue(e.target.value)}
      />
      <p>Ви написали: {inputValue}</p>

      <hr />

      {/* зміна числового значення */}
      <h2>Лічильник: {count}</h2>
      <button onClick={() => setCount(count + 1)}>Збільшити</button>
      <button onClick={() => setCount(count - 1)}>Зменшити</button>

      <hr />

      {/* зміна стилів та видимості */}
      <button onClick={handleToggle}>Змінити вигляд</button>
      {isVisible && <p>Цей текст можна приховати або показати</p>}
    </div>
  );
}

export default MyInfo;
```

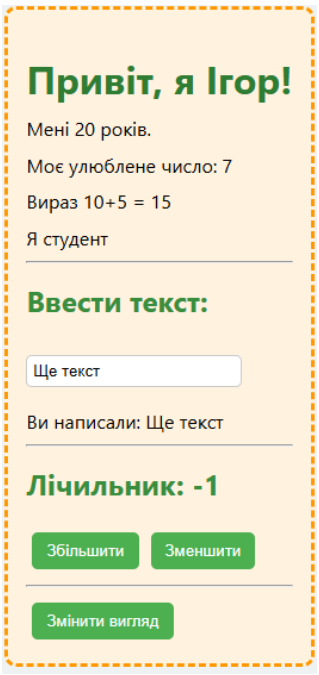
MyInfo.css:

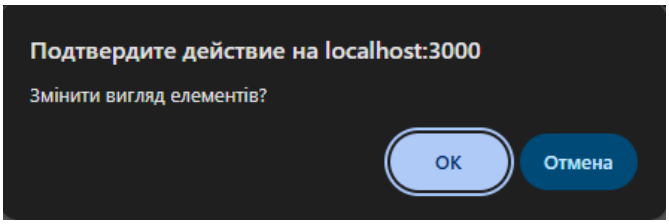
```
.my-info {
  margin: 20px;
  padding: 15px;
  border: 2px solid #4caf50;
  border-radius: 10px;
  background-color: #f9f9f9;
}

.my-info h1 {
  color: #2e7d32;
  margin-bottom: 10px;
}

.my-info h2 {
  color: #388e3c;
  margin-top: 15px;
}

.my-info p {
  margin: 8px 0;
  font-size: 16px;
  line-height: 1.4;
}

.my-info input {
  padding: 5px;
```

```
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.my-info button {
  margin: 5px;
  padding: 8px 12px;
  border: none;
  background-color: #4caf50;
  color: white;
  border-radius: 5px;
  cursor: pointer;
}

.my-info button:hover {
  background-color: #388e3c;
}

/* додатковий стиль */
.highlight {
  border: 3px dashed #ff9800;
  background-color: #fff3e0;
}
```

Результат:







Висновок: на цій лабораторній роботі ми познайомились і отримали головні навички роботи з інтерактивним стейтфул компонентом React з обробкою подій.

Контрольні питання:

1. Чим відрізняються обробники подій в HTML та в JSX?

- У HTML пишемо напряму в атрибут:

<button onclick="alert('hi')">Клік</button>

- У JSX передаємо функцію у camelCase-атрибут:

<button onClick={() => alert('hi')}>Клік</button>

2. Як React опрацьовує ре-рендеринг компонентів?

При зміні props або state React викликає функцію-компонент ще раз → створює новий Virtual DOM → порівнює його зі старим (diffing) → міняє тільки те, що реально змінилось у DOM.

3. Що таке props?

Props (properties) — це вхідні параметри, які компонент отримує від батьківського компонента. Вони роблять компонент "налаштовуваним".

4. Як передати пропси в компонент?

Як атрибути:

<User name="Ігор" age={20} />

і в компоненті:

function User(props) { return {props.name}, {props.age} ; }

5. Чи можна змінювати в компоненті його власні пропси?

Ні. Props — тільки для читання. Змінювати їх не можна.

6. Що таке state?

State — це внутрішні змінні компонента, які зберігають його стан і можуть змінюватися під час роботи. Зміна state викликає ре-рендер.

7. Чому звичайні змінні для виведення на екран змін не працюють?

Бо при зміні звичайної змінної React не знає, що треба оновити DOM. Лише state "повідомляє" React про зміни.

8. У чому є різниця між пропсами та стейтом?

- **Props** → приходять від батьківського компонента, незмінні.
- **State** → належить самому компоненту, можна змінювати.

9. Що таке хуки?

Hooks — це спеціальні функції React, які дозволяють використовувати state та інші можливості у функціональних компонентах (наприклад, useState, useEffect).

10. Для чого потрібен хук useState?

Для створення state-змінних у функціональних компонентах:

```
const [count, setCount] = useState(0);
```

11. Які способи виклику функції для зміни значення state-змінної ви знаєте?

- Передати нове значення напряму:

```
setCount(5);
```

- Використати функцію від попереднього значення:

```
setCount(prev => prev + 1);
```

12. Які існують правила використання хуків?

- Викликати тільки на верхньому рівні компонента (не в циклах, умовах, функціях).
- Використовувати тільки в React-компонентах або власних хуках.
- Завжди дотримуватись порядку виклику.