**ЛАБОРАТОРНА РОБОТА №7**

**Тема: WEB-СХОВИЩЕ. WEB STORAGE API. МОДУЛЬНОСТЬ КОДУ. СТВОРЕННЯ МОДАЛЬНОГО ВІКНА. ДЕЛЕГУВАННЯ ПОДІЙ.**

**Мета**: придбати практичні навички роботи створення модального вікна; повторити процес створення макету, зокрема створення макету галереї довільних зображень; закріпити навички роботи з делегуванням подій.

**ХІД ВИКОНАННЯ РОБОТИ**

**ЗАВДАННЯ №1**

1. Створити репозиторій у GitHub
2. Створити посилання на живу сторінку.
3. Склонувати репозиторій на свій комп’ ютер.
4. Створити файл index.html і дві папки: CSS та JS. У папці CSS створити файл style.css. У папці JS створити файл gallery.js
5. В Figma створити макет галереї довільних зображень. Зразок макету можна подивитись за посиланням <https://www.figma.com/design/m8k9NQV7qZrtYDCvxfD68B/%D0%94%D0%97-JavaScript?node-id=3-941&t=pjPWtBAEZqJYzmz0-0>
6. Виконай верстку власного макету. В HTML коді (index.html ) додай тег контейнера галереї — невпорядкований список із класом gallery, згідно макету.

<ul class="gallery"></ul>

1. У свій JavaScript файл додай масив зображень. Кожний об’єкт являє собою один елемент галереї.

* preview — посилання на маленьку версію зображення для картки галереї
* original — посилання на велику версію зображення для модального вікна
* description — текстовий опис зображення, для атрибута alt малого зображення та підпису великого зображення в модалці.

const images = [

    {

      preview:

        'https://…..',

      original:

        'https://….',

      description: ' текстовий опис зображення ',

    },

    {

      preview:

        'https://…..',

      original:

        'https://….',

      description: ' текстовий опис зображення ',

    },

Добавляєте потрібну кількість зображень,

];

1. Виконати в JavaScript коді **розмітку** елементів, після чого додати усю розмітку всередину ul.gallery.
2. Додати **стилізацію** галереї згідно макету.
3. Виконати **делегування**, а саме додати функціонал прослуховування кліка по елементах галереї та отримання посилання на велике зображення при кліку. Для цього використовуй прийом делегування на ul.gallery. Поки що при кліку на елемент галереї виводь у консоль посилання на велике зображення.
4. Підключення **бібліотеки**.

Бібліотека [basicLightbox](https://github.com/electerious/basicLightbox/tree/master) представляє повністю функціональне модальне вікно, яке відмінно підходить під поставлену задачу. Використовуйте [CDN сервіс jsdelivr](https://www.jsdelivr.com/package/npm/basiclightbox?path=dist) і додай в HTML файл посилання на мініфіковані (.min) JS та CSS файли бібліотеки.

[basicLightbox](https://github.com/electerious/basicLightbox/tree/master) - <https://github.com/electerious/basicLightbox/tree/master>

[CDN сервіс jsdelivr](https://www.jsdelivr.com/package/npm/basiclightbox?path=dist)  - <https://www.jsdelivr.com/package/npm/basiclightbox?path=dist>

1. **Модальне вікно**

Доповнити свій код так, щоб при кліку по елементу галереї відкривалось модальне вікно підключеної бібліотеки. Для того щоб дізнатися, як ініціалізувати модальне вікно у своєму коді і як його використовувати, ознайомся з [документацією](https://github.com/electerious/basicLightbox#readme) і [прикладами](https://basiclightbox.electerious.com/).

Документація <https://github.com/electerious/basicLightbox#readme>

Приклади - <https://basiclightbox.electerious.com/>

ВАРТО ЗВЕРНУТИ УВАГУ:

* На живій сторінці відображається галерея зображень із масиву даних images
* Галерея зображень стилізована згідно з макетом
* Дані для галереї створені динамічно в JS
* Під час прослуховування події натискання на елементи галереї використаний прийом делегування
* При кліку між елементами галереї нічого не відбувається
* Підключена бібліотека basicLightbox
* При кліку по елементу галереї відкривається модальне вікно підключеної бібліотеки, в якому міститься збільшена версія зображення, по якому клікнули

**ЗАВДАННЯ №2**

**Форма зворотного зв'язку**

1. На макеті добавити форми (Приклад за посиланням <https://www.figma.com/design/m8k9NQV7qZrtYDCvxfD68B/%D0%94%D0%97-JavaScript?node-id=3-969&t=ms4nQGevNBJzbAlx-0>
2. Виконуй це завдання у файлах 2-form.html і 2-form.js
3. Додати у HTML розмітку форми. У JS написати скрипт, який буде зберігати значення полів у локальне сховище, коли користувач щось друкує.

 form class="feedback-form" autocomplete="off">

  <label>

    Email

    <input type="email" name="email" autofocus />

  </label>

  <label>

    Message

    <textarea name="message" rows="8"></textarea>

  </label>

  <button type="submit">Submit</button>

</form>

1. Виконуй це завдання у файлах 2-form.html і 2-form.js. Розбий його на декілька підзавдань:

* Оголосити поза будь-якими функціями об’єкт **formData** з полями **email** та **message**, які спочатку мають порожні рядки як значення: **{ email: "", message: "" }**.
* Використовувати метод делегування для відстеження змін у формі через подію **input**. Зберігати актуальні дані з полів **email** та **message** у **formData** та записувати цей об’єкт у локальне сховище. Використовувати ключ **"feedback-form-state"** для зберігання даних у сховищі.
* При завантаженні сторінки перевірити, чи є дані у локальному сховищі. Якщо так, використовувати їх для заповнення форми та об'єкта **formData**. Якщо ні, залишити поля форми порожніми.
* Перед відправленням форми переконатися, що обидва поля форми заповнені. Якщо будь-яке з полів (властивостей об’єкта formData) порожнє, показати сповіщення з текстом «Fill please all fields». Якщо всі поля заповнені, вивести у консоль об’єкт **formData** з актуальними значеннями, очистити локальне сховище, об’єкт **formData** і поля форми.

ВАРТО ЗВЕРНУТИ УВАГУ:

* На живій сторінці відображається форма з двома елементами форми і кнопкою типу submit
* Форма стилізована згідно з макетом
* На формі прослуховуються події input і submit
* При введенні даних у будь-який елемент форми вони записуються у локальне сховище під ключем "feedback-form-state", збережені дані не містять пробіли по краях
* Введення даних в одне поле форми не видаляє дані в сховищі для іншого
* При оновленні сторінки дані з локального сховища підставляються в елементи форми, у полях форми відсутні undefined
* При сабміті форми є перевірка, щоб обидва елементи форми були заповнені
* Під час сабміту форми, якщо обидва елементи форми заповнені, виводиться у консоль об'єкт з полями email, message та їхніми поточними значеннями, а також очищаються сховище і поля форми
* Якщо після сабміту форми ввести в будь-який елемент форми дані, то в локальному сховищі не з’являються дані від попереднього сабміта