Лабораторна робота № 1

Загальні відомості. Поняття адаптивного web-інтерфейсу. Twitter bootstrap.

Одне із питань, що виникають на початку розробки мобільного застосунку: створювати мобільний (адаптивний) веб-сайт, нативний або гібридний застосунок. Від прийнятого рішення буде залежати подальший процес розробки.

Адаптивний веб-дизайн - дизайн веб-сторінок, що забезпечує правильне відображення сайту на різних пристроях, підключених до інтернету, і динамічно підстроюється під задані розміри вікна браузера. Метою адаптивного веб-дизайну є універсальність відображення вмісту веб-сайту для різних пристроїв. Для того, щоб веб-сайт був зручно переглядається з пристроїв форматів і з екранами різної розділювальної здатності , за технологією адаптивного веб-дизайну не потрібно створювати окремі версії веб-сайту для окремих видів пристроїв.

Нативні (рідні) застосунки написані на мові програмування, специфічній для платформи, для якої вони розробляються. Зазвичай це Objective-C або Swift для iOS та Java або Kotlin для Android. Нативні застосунки зазвичай мають кращу продуктивність при рендерінгу і анімації, ніж гібридні програми. Але якщо вам необхідна реалізація як для Android, так і для iOS – необхідно буде створити два застосунки.

Гібридний застосунок - це мобільний застосунок, який містить веб-представлення (по суті, ізольований екземпляр браузера) для запуску веб-застосунку із використанням вбудованої оболонки, яка може взаємодіяти із платформою пристрою та веб-представленням. Це означає, що веб-застосунки можуть працювати на мобільному пристрої, маючи доступ до, наприклад, камери або функцій GPS.

Гібридні застосунки можливі завдяки створеним інструментам, які полегшують зв'язок між веб-представленням і платформою. Ці інструменти не є частиною

офіційних платформ iOS або Android, та є сторонніми інструментами, такими як Apache Cordova. Коли гібридний застосунок буде створено, він буде скомпільований, перетворивши ваш веб-застосунок в нативний застосунок.

Підхід із створенням web-інтерфейсу, адаптованого для відображення на мобільному пристрої є найбільш швидким рішенням та вимагає мінімальну кількість ресурсів для реалізації. Тому, у разі відсутності необхідності у використанні нативних функцій мобільного пристрою даний підхід є досить привабливим. Крім того, у даному випадку немає необхідності встановлення ПЗ та мобільний пристрій, достатньо наявності інтернет-браузера. Також плюсом буде незалежність від операційної системи мобільного пристрою.

Розробка адаптивного веб-інтерфейсу у переважній більшості базується на використанні гнучких макетів на основі сіток (flexible, grid-based layout). Одною з найпопулярніших бібліотек, що реалізують даний підхід, є twitter bootstrap. Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-застосунків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-застосунків.

Далі розглянемо приклад реалізації мобільного web-інтерфейсу для застосунку роботи із списком студентських груп на базі twitter bootstrap. Створимо index.html із стандартною html5 розміткою (рис.1).

index.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6  </head>
7  <body>
8
9  </body>
10 </html>
```

Рис. 1. Створення index.html.

Потім виконаємо підключення бібліотеки bootstrap. Зробити це можна декількома способами, один із яких полягає у завантаженні необхідних файлів бібліотек локально у директорію assets (рис.2). Нагадаємо, що для коректної роботи скриптів bootstrap.js необхідно попередньо підключити також бібліотеки jquery.js та popper.js. Необхідні файли можуть бути завантажені на офіційних сайтах jquery, popper та bootstrap.

```
3   <head>
4       <meta charset="UTF-8">
5       <title>Document</title>
6       <link rel="stylesheet" href="assets/bootstrap.min.css">
7       <script src="assets/jquery-3.4.1.min.js"></script>
8       <script src="assets/popper.min.js"></script>
9       <script src="assets/bootstrap.min.js"></script>
```

Рис. 2. Підключення завантажених бібліотек.

Також можна скористатись cdn-посиланнями (рис.3), але зазначимо, що даний підхід вимагатиме наявності постійного інтернет-з'єднання для роботи із бібліотекою. Звернімо увагу, що cdn-посилання можуть бути іншими.

```
3   <head>
4       <meta charset="UTF-8">
5       <title>Document</title>
6       <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
        bootstrap/4.4.1/css/bootstrap.min.css">
7       <script
8               src="https://code.jquery.com/jquery-3.4.1.min.js"
9               integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlB
                w8HfCJo="
10              crossorigin="anonymous"></script>
11      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/
        1.15.0/esm/popper.min.js"></script>
12      <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/
        js/bootstrap.min.js"></script>
```

Рис. 3. Підключення бібліотек через cdn-посилання.

Далі виконаємо реалізацію обробки натискання на кнопку для перевірки правильності роботи бібліотеки. Додаємо на сторінку кнопку, по натисканню на яку буде виводитись повідомлення alert (рис. 4). Обробку кода натискання на кнопку розмістимо у файлі code/my.js (рис. 5). Також у файлі index.html виконаємо підключення даного скрипта (рис. 4).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="stylesheet" href="assets/bootstrap.min.css">
    <script src="assets/jquery-3.4.1.min.js"></script>
    <script src="assets/popper.min.js"></script>
    <script src="assets/bootstrap.min.js"></script>

    <script src="code/my.js"></script>
</head>
<body>
    <div class="text-center mt-5" >
        <button type="button" class="btn btn-primary center"
        id="click-me">Натисни мене !</button>
    </div>
    <div class="m-5" id="alert">
    </div>
</body>
</html>
```

Рис. 4. Додавання кнопки та підключення my.js.

```
$("document").ready(function() {
    $('#click-me').click(function() {
        var html = `<div class="text-left alert alert-success" role="alert">
            <h4 class="alert-heading">Привіт !
                <button type="button" class="close" data-dismiss="alert"
                aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
```

```
8               </h4>
9               <p>Вітаємо із успішним підключенням twitter bootstrap !</p>
10              <hr>
11              <p>Тепер можна переходити до реалізації проекту.</p>
12          </div>`;
13          $('#alert').html(html);
14      });
15  });
```

Рис. 5. Файл my.js.

Перевіримо результат, завантажимо файл у браузері, та для перегляду зовнішнього вигляду сторінки так, як вина виглядатиме на мобільному пристрої, запустимо режим розробника та перейдемо у режим адаптивного дизайну (рис. 6-7).

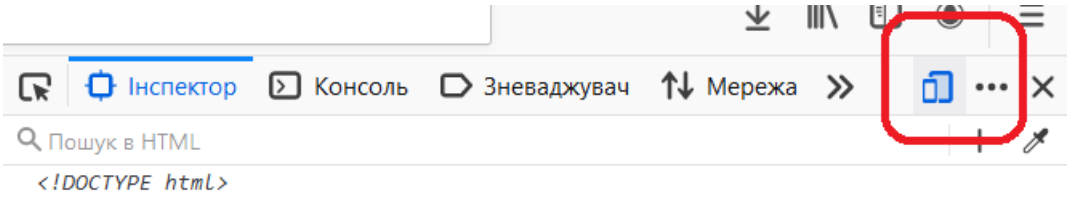

Рис. 6. Перехід до режиму адаптивного дизайну.

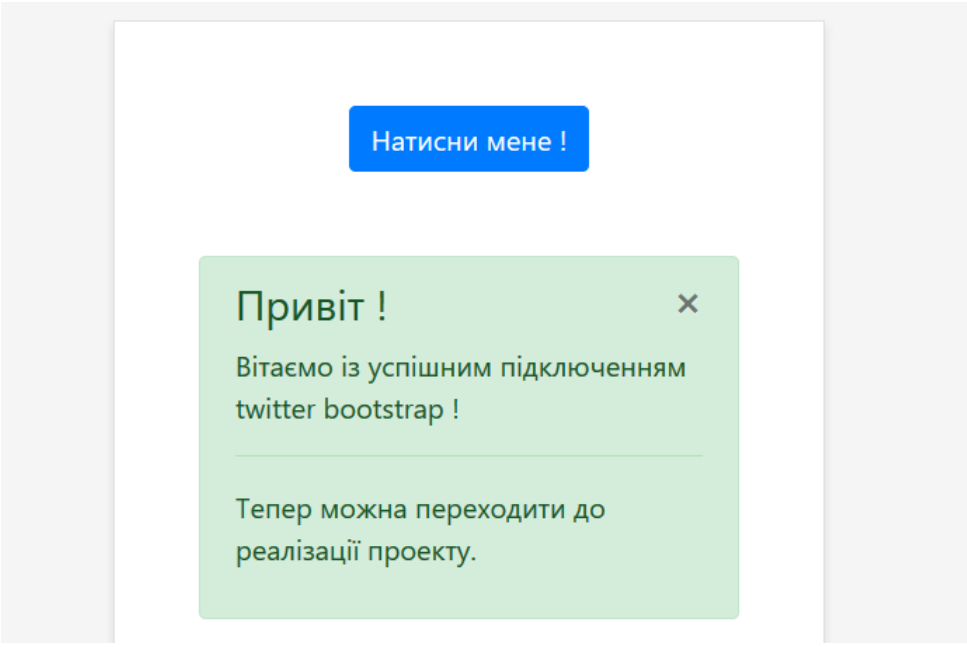

Рис. 7. Перегляд зовнішнього вигляду сторінки.

Реалізуємо сторінку для виведення даних щодо списку студентських груп. Сам список будемо зберігати у LocalStorage. Почнімо із скрипту, що буде отримувати із

LocalStorage списку груп, але якщо там нічого немає – записувати туди 2 групи. Для цього у папці code створимо data.js (рис. 8).

```
var groups = localStorage.getItem("groups_data");
if (localStorage.getItem("groups_data") === null) {
    groups = [
        {
            number: 301,
            faculty: 'Computer science'
        },

        {
            number: 302,
            faculty: 'Computer science'
        }
    ];
    localStorage.setItem("groups_data", JSON.stringify(groups));
} else {
    groups = JSON.parse(groups);
}
```

Рис. 8. Файл data.js.

Тепер підключимо даний скрипт до index.html та розмістимо у елементі body nav-панель із заголовком вікна, кнопку для додавання нової групи та таблицю для виведення існуючих груп (рис. 9). Також підключимо font-awesome для можливості виведенні іконок на кнопках.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="stylesheet" href="assets/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/
    4.7.0/css/font-awesome.min.css">

    <script src="assets/jquery-3.4.1.min.js"></script>
    <script src="assets/popper.min.js"></script>
    <script src="assets/bootstrap.min.js"></script>
    <script src="code/data.js"></script>
    <script src="code/my.js"></script>
</head>
```

```
<body>
    <nav class="navbar navbar-dark bg-dark text-white">
        <h3>Студентські групи</h3>
    </nav>
    <div class="container mt-3">
        <div class="row">
            <div class="col">
                <a class="btn btn-outline-success" href="#">
                    <i class="fa fa-fw fa-plus"></i>
                </a>
            </div>
        </div>

        <div class="row mt-2">
            <div class="col">
                <table class="table">
                    <thead>
                        <tr>
                            <th scope="col" style="width: 60%">Група</th>
                            <th scope="col" style="width: 40%"></th>
                        </tr>
                    </thead>

                    <tbody id="groups">
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</body>
</html>
```

Рис. 9. Зміни файлу index.html.

У файлі code/my.js реалізуємо отримання даних списку студентських груп із LocalStorage та виведення їх до таблиці (рис. 10).

```
$('document').ready(function() {
  var rowText;
  var content = $('#groups');
  for(var row of groups) {
      rowText =
      `<tr>
          <td>${row.number}</td>
```

```
            <td class="text-right">
                <a class="btn btn-outline-secondary" href="group.html?
                number=${row.number}">
                    <i class="fa fa-fw fa-edit"></i>
                </a>
                <button type="button" class="btn btn-outline-danger rem-row"
                rowid="${row.number}">
                    <i class="fa fa-fw fa-trash"></i>
                </button>
            </td>
        </tr>`;
        content.append(rowText);
    }
});
```

Рис. 10. Скрипт my.js.

Завантажимо відкориговану сторінку та переглянемо результат (рис. 11).

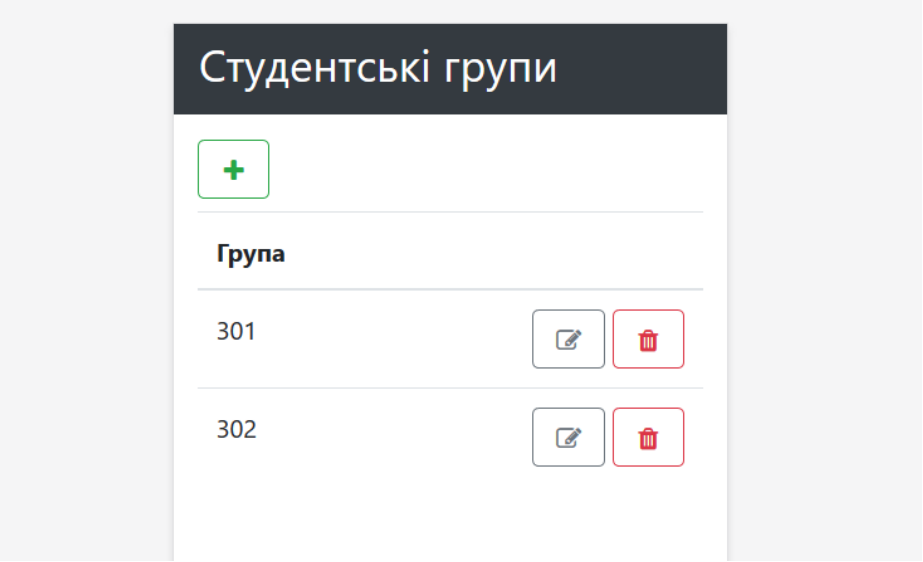

Рис. 11. Сторінка виведення студентських груп.

Реалізуємо функціонал редагування даного списку. Для початку трохи змінимо файл code/data.js – виділимо функцію saveGroups для її подальшого використання при додаванні, зміні та видаленні елементів списку (рис. 12).

```
function saveGroups(groups) {
    localStorage.setItem("groups_data", JSON.stringify(groups));
}
```

Рис. 12. Функція saveGroups в data.js.

Тепер реалізуємо сторінку для додавання та редагування елементу списку group.html (рис. 13). Тут розмістимо html-форму для виведення елементів студентської групи та підключимо скрипт code/group.js, вміст якого буде наведено пізніше.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Студентські групи</title>
    <link rel="stylesheet" href="assets/bootstrap.min.css">
    <script src="assets/jquery-3.4.1.min.js"></script>
    <script src="assets/popper.min.js"></script>
    <script src="assets/bootstrap.min.js"></script>
    <script src="code/data.js"></script>
    <script src="code/group.js"></script>
</head>
<body>
    <nav class="navbar navbar-dark bg-dark text-white">
        <h3>Студентська група</h3>
    </nav>
    <div class="container mt-3">
        <div class="row">
            <div class="col">
                <form>
                    <div class="form-group">
                        <label for="number">Номер групи</label>
                        <input type="text" class="form-control" id="number"
                        placeholder="Номер групи">
                    </div>
                    <div class="form-group">
                        <label for="faculty">Факультет</label>
                        <input type="text" class="form-control" id="faculty"
                        placeholder="Факультет">
                    </div>
                    <button type="button" id="save" class="btn
                    btn-secondary">Зберегти</button>
```

```
            </form>
        </div>
    </div>
</div>
</body>
</html>
```

Рис. 13. Сторінка редагування даних group.html.

Тепер виконаємо невеликі зміни у index.html та code/my.js – додамо посилання на кнопки «додати» та «змінити» і атрибут rowid для кнопки «видалити» (рис. 14-15). Також у code/my.js реалізуємо обробку натискання на кнопку «видалити».

```
        <div class="row">
            <div class="col">
                <a class="btn btn-outline-success" href="group.html">
                    <i class="fa fa-fw fa-plus"></i>
                </a>
            </div>
```

Рис. 14. Зміни у index.html.

```
$('document').ready(function() {
    var rowText;
    var content = $('#groups');
    for(var row of groups) {
        rowText =
        `<tr>
            <td>${row.number}</td>
            <td class="text-right">
                <a class="btn btn-outline-secondary" href="group.html?number=${row.number}">
                    <i class="fa fa-fw fa-edit"></i>
                </a>
                <button type="button" class="btn btn-outline-danger rem-row" rowid="${row.number}">
                    <i class="fa fa-fw fa-trash"></i>
                </button>
```

```
            </td>
        </tr>`;
        content.append(rowText);
    }
    $('.rem-row').click(function() {
        let number = $(this).attr('rowid');
        saveGroups(groups.filter((g)=>g.number !== number));
        location.reload();
    })
});
```

Рис. 15. Зміни у my.js.

На останок реалізуємо скрипт data/group.js, що виконуватиме додавання та редагування елементу списку (рис. 16). При редагуванні елементу у get-запиті передаватимемо значення number. У group.js виконуємо перевірку, якщо number передано, то виконуємо зміну елементу масиву, інакше додаємо новий елемент.

```
$('document').ready(function() {
    let searchParams = new URLSearchParams(window.location.search);
    let number = 0;
    if (searchParams.has('number')) {
        number = searchParams.get('number');
        let group = groups.find((g)=>g.number == number);
        $('#number').val(group.number);
        $('#faculty').val(group.faculty);
    }

    $('#save').click(function() {
        if (number === 0) {
            groups.push({
                number: $('#number').val(),
                faculty: $('#faculty').val()
            });
        } else {
            let group = groups.find((g)=>g.number == number);
            group.number = $('#number').val();
            group.faculty = $('#faculty').val();
        }
        saveGroups(groups);
        $(location).attr('href','index.html');
    });
});
```

Рис. 16. Файл data/group.js.

Перевіряємо роботу коду, пробуємо додати змінити та видалити елемент списку (рис. 17).

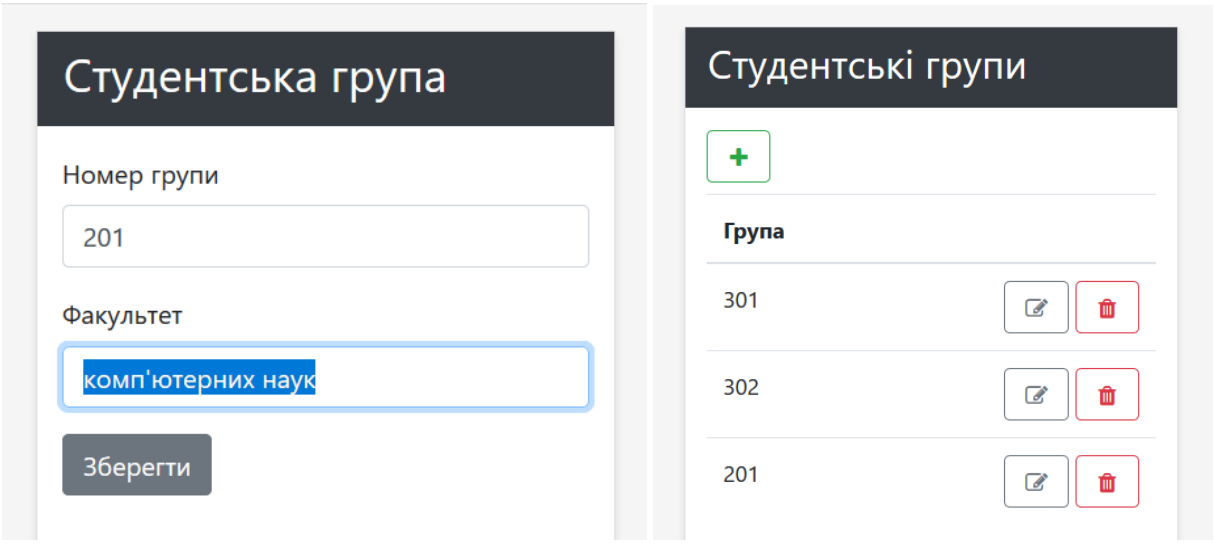

Рис. 17. Додавання студентської групи.

Останній штрих – додавання метатегу initial-scale=1 для відключення масштабування сторінки на мобільному пристрої (рис.18). Вигляд сторінки до та після включення тегу наведено на рис. 19.

```
4     <meta charset="UTF-8">
5     <meta name="viewport" content="initial-scale=1">
6     <title>Студентські групи</title>
```

Рис. 18. Додавання тегу viewport.

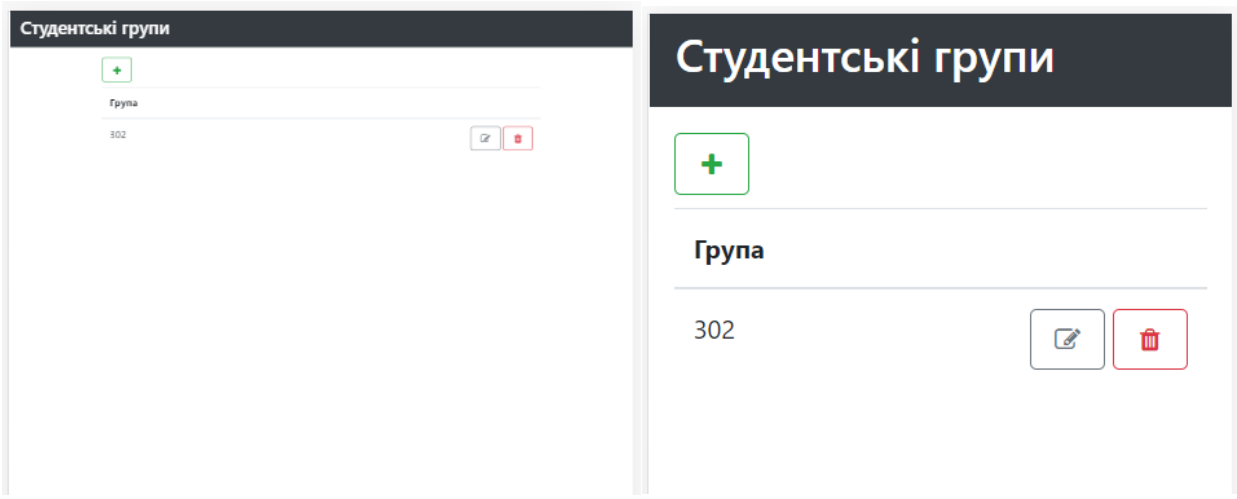

Рис. 19. Зовнішній вигляд сторінки на мобільному пристрої.