Bài tập thực hành Tuần 6 – Môn Công nghệ Web

Các bài tập sau đây giúp sinh viên thực hành tổng hợp kiến thức từ tuần 2 đến tuần 5 (HTML cơ bản, CSS, JavaScript cơ bản – DOM, sự kiện, async/await) bằng cách xây dựng một website giới thiệu sản phẩm cho một cửa hàng nhỏ. Chủ đề trang web có thể do sinh viên tùy chọn (ví dụ: cửa hàng sách, thời trang, hoặc cà phê), nhưng các yêu cầu kỹ thuật thì thống nhất như dưới đây. Tổng thời gian làm bài khoảng **3 giờ**.

Bài tập 1: Tạo trang HTML cơ bản với semantic tags và form đơn giản

Mô tả

- Xây dựng một trang HTML giới thiệu sản phẩm cho cửa hàng của bạn với cấu trúc **semantic** rõ ràng.
- Trang gồm các phần chính: **header**, **main** (danh sách sản phẩm), và **footer**. Sử dụng các thẻ semantic thích hợp như `<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<footer>`, v.v.
- Trong phần nội dung, tạo danh sách một số **sản phẩm mẫu** (khoảng 3 sản phẩm) gồm tên, mô tả ngắn và giá. *(Gợi ý: Mỗi sản phẩm có thể đặt trong một thẻ `<article>` riêng.)*
- Thêm một **form đơn giản** trên trang (ví dụ: form liên hệ hoặc đăng ký nhận tin). Form bao gồm một vài trường như tên, email và thông điệp, cùng nút gửi.

Hướng dẫn từng bước

1. **Tạo tệp HTML:** Tạo file `index.html` và khai báo cấu trúc HTML5 cơ bản (doctype, `<html>`, `<head>`, `<body>`). Đặt tiêu đề trang (trong `<title>`) phù hợp với cửa hàng, ví dụ: "Cửa hàng Sách ABC".
2. **Thẻ meta viewport:** Trong phần `<head>`, thêm thẻ meta khai báo viewport để hỗ trợ thiết kế responsive:

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

3. **Phần Header:** Sử dụng thẻ `<header>` bao gồm:
4. Tên cửa hàng (vd: đặt trong thẻ `<h1>`).
5. (Tuỳ chọn) Thanh điều hướng `<nav>` với các liên kết (vd: Trang chủ, Sản phẩm, Liên hệ… – có thể sử dụng # làm placeholder cho href).
6. **Phần Main - Giới thiệu sản phẩm:** Dùng thẻ `<main>` chứa nội dung chính:
7. Viết đoạn mô tả ngắn về cửa hàng (vd: slogan hoặc giới thiệu chung) trong một thẻ `<section>`.
8. Tạo một thẻ `<section>` (hoặc `<div>`) khác để chứa danh sách sản phẩm. Bên trong, tạo khoảng 3 sản phẩm mẫu, mỗi sản phẩm dùng một thẻ `<article>`:

    - Mỗi `<article>` bao gồm tên sản phẩm (thẻ tiêu đề `<h3>` hoặc `<h2>`), mô tả ngắn (đoạn văn `<p>`), và giá (ví dụ dùng `<p>` hoặc `<span>` kèm đơn vị tiền tệ).
    - (Tuỳ chọn) Thêm một hình ảnh sản phẩm dùng thẻ `<img>` (có thể dùng link ảnh placeholder hoặc ảnh minh hoạ có sẵn).
9. **Phần Footer:** Trong thẻ `<footer>`, thêm thông tin footer đơn giản, ví dụ: tên cửa hàng, địa chỉ, hoặc bản quyền.
10. **Form đơn giản:** Quyết định vị trí đặt form (có thể đặt trong `<main>` hoặc trong thanh bên nếu có). Ví dụ tạo form liên hệ ở cuối trang:
11. Dùng thẻ `<form>` với các trường: Tên (`<input type="text">`), Email (`<input type="email">`), Nội dung (`<textarea>`), và nút gửi (`<button type="submit">Gửi</button>`).
12. Mỗi trường nên có nhãn (`<label>`) mô tả. Sử dụng các thuộc tính phù hợp (như `required` cho các trường bắt buộc).
13. **Kiểm tra:** Lưu file và mở trong trình duyệt để kiểm tra giao diện cơ bản. Đảm bảo các nội dung (sản phẩm, form) hiển thị đầy đủ, cấu trúc hợp lý.

Bài tập 2: Thêm CSS để tạo layout và thiết kế responsive

Mô tả

- Viết file CSS để **trang trí giao diện** cho trang web cửa hàng đã tạo ở bài 1. Đảm bảo trang web có bố cục rõ ràng và **responsive** (hiển thị tốt trên cả màn hình lớn và điện thoại).
- **Layout:** Sử dụng CSS để chia bố cục trang: ví dụ, header cố định trên cùng, thanh menu ngang, danh sách sản phẩm chia cột trên màn hình lớn và xếp hàng dọc trên màn hình nhỏ.
- **Thiết kế:** Thêm màu sắc, khoảng cách, font chữ phù hợp để trang web nhìn chuyên nghiệp hơn.
- **Responsive design:** Sử dụng **media queries** để điều chỉnh kiểu dáng cho màn hình nhỏ (mobile) – ví dụ: chuyển layout sản phẩm thành 1 cột, ẩn bớt nội dung không cần thiết trên mobile,...

Hướng dẫn

1. **Tạo file CSS:** Tạo file `style.css` và liên kết trong trang HTML (thêm vào `<head>`: `<link rel="stylesheet" href="style.css">`).
2. **Đặt style chung:** Thiết lập các style chung cho trang, ví dụ font chữ, màu nền trang. Bạn có thể đặt `body { font-family: sans-serif; line-height: 1.6; }` để dễ đọc hơn.
3. **Styling Header:**
4. Đặt header có nền (ví dụ màu tối) và chữ trắng, căn các mục menu ngang hàng.
5. Sử dụng `display: flex;` cho `<nav> ul` để tạo menu ngang. Loại bỏ style mặc định của list (`list-style: none; margin: 0; padding: 0;`).

6. Thêm khoảng cách giữa các mục menu (ví dụ `nav ul li { margin-right: 20px; }`).
7. **Danh sách sản phẩm (Product list layout):**
8. Sử dụng lớp hoặc ID đã có (như `#product-list` hoặc `.product-item`) để định kiểu.
9. Với màn hình rộng (desktop), có thể dùng flexbox hoặc CSS Grid để chia các sản phẩm thành nhiều cột. Ví dụ:

```
#product-list {
  display: flex;
  flex-wrap: wrap;
  gap: 20px; /* khoảng cách giữa các sản phẩm */
}
.product-item {
  flex: 1 1 30%; /* mỗi sản phẩm chiếm ~30% chiều ngang, co giãn linh
hoạt */
  border: 1px solid #ccc;
  padding: 10px;
  border-radius: 4px;
  background: #fafafa;
}
```

   Đoạn trên sẽ giúp hiển thị 3 sản phẩm một hàng (vì mỗi cái ~30% và có khoảng cách).
10. Định dạng bên trong mỗi sản phẩm: tên sản phẩm dùng font lớn hơn, **in đậm**; mô tả và giá dùng font thường, có thể khác màu (vd: giá màu nổi bật).
11. **Form và Footer:**
12. Đặt form trong một khung hoặc canh lề cho đẹp. Ví dụ: `form { max-width: 400px; margin: 20px auto; }` để form nằm giữa trang với độ rộng tối đa 400px.
13. Style cho `<input>` và `<textarea>`: thêm `padding`, `margin-bottom` để các ô nhập liệu rõ ràng, có viền mờ.
14. Style cho nút gửi (`button`): màu nền nổi bật, chữ trắng, bo tròn nhẹ các góc. Thêm hiệu ứng hover (vd: sáng màu hơn khi hover).
15. Footer: căn giữa nội dung, chữ nhỏ hơn một chút và màu xám. Có thể thêm `padding: 10px 0` và nền sẫm màu tương tự header.
16. **Thiết kế Responsive:** Sử dụng media query để điều chỉnh layout khi màn hình nhỏ:
17. Ví dụ, khi chiều rộng dưới 600px, chuyển danh sách sản phẩm sang hiển thị một cột:

```
@media (max-width: 600px) {
  #product-list {
    flex-direction: column;
  }
  .product-item {
    flex: 1 1 100%;
  }
  nav ul {
```

```
    flex-direction: column; /* menu dọc trên mobile */
  }
  nav ul li {
    margin: 5px 0; /* khoảng cách các mục menu dọc */
  }
}
```

18. Đảm bảo các phần tử như hình ảnh, bảng (nếu có) co dãn vừa màn hình nhỏ (`max-width: 100%` cho ảnh để ảnh không tràn khỏi màn hình).
19. **Kiểm tra trên nhiều kích thước màn hình:** Mở công cụ DevTools hoặc trình duyệt di động để kiểm tra trang ở các độ phân giải khác nhau. Đảm bảo bố cục và chữ đọc tốt trên mobile.

Bài tập 3: Thêm JavaScript xử lý sự kiện và thao tác DOM

Mô tả

- Bổ sung JavaScript để thêm tính năng **tương tác** cho trang web:
- **Tìm kiếm/Lọc sản phẩm:** Cho phép người dùng nhập từ khóa để tìm sản phẩm theo tên. Khi thực hiện tìm kiếm, chỉ những sản phẩm có tên chứa từ khóa (không phân biệt hoa thường) được hiển thị, các sản phẩm khác ẩn đi.
- **Hiển thị form "Thêm sản phẩm":** Thêm một nút "Thêm sản phẩm" trên trang. Khi bấm nút này, hiện ra form cho phép nhập thông tin sản phẩm mới (form này đã tạo ở bài 1 hoặc sẽ tạo bổ sung). Việc ẩn/hiện form được thực hiện bằng JS (ví dụ: form ẩn mặc định, bấm nút thì hiện, bấm lần nữa thì ẩn lại).
- Yêu cầu sử dụng **DOM API** để lấy phần tử và cập nhật giao diện (ví dụ: `document.querySelector`, `.style.display` hoặc thêm/xóa class CSS).
- Sử dụng **sự kiện (event)** để bắt các hành động của người dùng: sự kiện click vào nút, sự kiện submit form, hoặc sự kiện nhập liệu (keyup) cho ô tìm kiếm.

Hướng dẫn

1. **Chuẩn bị tệp JS:** Tạo file `script.js` và nhúng vào cuối trang HTML (trước thẻ `</body>`):

   ```
   <script src="script.js"></script>
   ```

   Đảm bảo file JS được nhúng **sau** các phần HTML (để có thể truy cập các phần tử đã tạo).
2. **Thêm phần tử giao diện (HTML) nếu cần:**
3. Thêm một ô input và nút cho chức năng tìm kiếm, nếu chưa có từ bài 1. Ví dụ đặt ở trên danh sách sản phẩm:

   ```
   <input type="text" id="searchInput" placeholder="Tìm kiếm sản phẩm...">
   <button id="searchBtn">Tìm</button>
   ```

4. Thêm nút "Thêm sản phẩm" để mở form thêm mới. Ví dụ đặt bên trên hoặc dưới danh sách sản phẩm:

   ```
   <button id="addProductBtn">Thêm sản phẩm</button>
   ```

5. Đảm bảo form thêm sản phẩm (có thể dùng lại form liên hệ hoặc tạo form mới) có `id` để dễ truy cập. Ví dụ: `<form id="addProductForm" class="hidden"> ... </form>` và CSS `.hidden { display: none; }` để form ẩn mặc định.
6. **Xử lý sự kiện tìm kiếm:**
7. Sử dụng `document.getElementById` hoặc `querySelector` để lấy ô nhập tìm kiếm và nút tìm kiếm.
8. Gắn sự kiện cho nút tìm hoặc ô nhập (ví dụ `searchBtn.addEventListener('click', ...)`).
9. Trong hàm xử lý, lấy giá trị từ ô nhập (`searchInput.value`), chuyển về chữ thường để so sánh.
10. Duyệt qua danh sách các sản phẩm (có thể dùng `document.querySelectorAll('.product-item')` để lấy tất cả sản phẩm). Với mỗi sản phẩm, lấy tên (vd: phần tử có class `.product-name` bên trong) và kiểm tra xem tên có chứa từ khóa tìm kiếm không:
    - Nếu **có**, hiển thị sản phẩm đó (ví dụ `element.style.display = ""` hoặc gỡ lớp ẩn nếu dùng class).
    - Nếu **không**, ẩn sản phẩm (`element.style.display = "none"` hoặc thêm lớp ẩn).
11. **Xử lý sự kiện nút "Thêm sản phẩm":**
12. Lấy phần tử form thêm sản phẩm (vd: `addProductForm`) và nút `addProductBtn`.
13. Gắn sự kiện `click` cho `addProductBtn`. Trong hàm xử lý, kiểm tra nếu form đang ẩn thì hiện ra, nếu đang hiện thì ẩn đi (toggle):
    - Cách 1: Sử dụng class CSS: thêm hoặc loại bỏ class `"hidden"` cho form.
    - Cách 2: Thay đổi trực tiếp kiểu hiển thị:

      ```
      if (addProductForm.style.display === "none") {
        addProductForm.style.display = "block";
      } else {
        addProductForm.style.display = "none";
      }
      ```

    - Nếu form ẩn bằng class, có thể dùng `classList.toggle("hidden")`.
14. **Kiểm tra:**
15. Thử nhập các từ khóa khác nhau vào ô tìm kiếm và nhấn "Tìm", kiểm tra xem danh sách sản phẩm lọc đúng chưa. (Ví dụ: tìm "Sách A" chỉ còn hiện sản phẩm A).
16. Thử nhấn nút "Thêm sản phẩm" xem form có xuất hiện và ẩn đi khi nhấn lại không. Đảm bảo các thao tác không gây lỗi JavaScript (mở console kiểm tra nếu cần).

Bài tập 4: Tích hợp toàn bộ – Form đăng ký sản phẩm mới, hiển thị và cập nhật danh sách sản phẩm

Mô tả

- Hoàn thiện trang web bằng cách cho phép **thêm sản phẩm mới** thông qua form và cập nhật vào danh sách sản phẩm trên trang **mà không cần tải lại trang**.
- Chức năng cụ thể:
- Khi người dùng nhập thông tin sản phẩm mới vào form "Thêm sản phẩm" (ví dụ: tên, giá, mô tả) và nhấn Submit, sử dụng JavaScript để **validate** dữ liệu và sau đó thêm sản phẩm mới vào danh sách hiện có.
- **Validation:** Kiểm tra các trường dữ liệu trước khi thêm:
    - Tên sản phẩm không được rỗng.
    - Giá phải là số hợp lệ và lớn hơn 0 (có thể dùng input type="number" và kiểm tra thêm).
    - (Tuỳ chọn) Mô tả không quá ngắn, hoặc các trường khác tuỳ ý.
    - Nếu không đạt, hiển thị thông báo lỗi (có thể dùng `alert` hoặc tạo thẻ `<span>` thông báo dưới form).
- Nếu dữ liệu hợp lệ, tạo một phần tử HTML mới đại diện cho sản phẩm vừa nhập và chèn vào **đầu danh sách** (hoặc cuối danh sách) các sản phẩm. Xóa nội dung trong form (reset form) và ẩn form sau khi thêm.
- Đảm bảo rằng sản phẩm thêm mới cũng tương tác được với chức năng tìm kiếm/lọc (nghĩa là sau khi thêm, nếu tìm kiếm vẫn xét đến cả sản phẩm mới).

Hướng dẫn

1. **Thiết kế trường thông tin sản phẩm:** Đảm bảo form "Thêm sản phẩm" có các trường cần thiết. Ví dụ:

```
<form id="addProductForm" class="hidden">
  <h3>Thêm sản phẩm mới</h3>
  <label for="newName">Tên sản phẩm:</label>
  <input type="text" id="newName" required>

  <label for="newPrice">Giá:</label>
  <input type="number" id="newPrice" required>

  <label for="newDesc">Mô tả:</label>
  <textarea id="newDesc" rows="3"></textarea>

  <button type="submit">Thêm</button>
  <button type="button" id="cancelBtn">Hủy</button>
  <p id="errorMsg" style="color:red;"></p> <!-- Thông báo lỗi -->
</form>
```

Lưu ý: Thêm nút "Hủy" (type="button") để đóng form nếu người dùng không muốn thêm nữa, và một phần tử `<p id="errorMsg">` để hiển thị lỗi (ẩn khi không có lỗi).

2. **Bắt sự kiện submit của form:** Trong `script.js`, lấy phần tử form thêm sản phẩm (đã có `addProductForm`) và gắn sự kiện:

```
addProductForm.addEventListener('submit', function(event) {
    event.preventDefault();
    // ... xử lý thêm sản phẩm
});
```

   Dùng `event.preventDefault()` để tránh form gửi yêu cầu HTTP và reload trang.

3. **Lấy giá trị và validate:** Bên trong hàm xử lý submit:
4. Lấy giá trị từ các trường input:

```
const name = document.getElementById('newName').value.trim();
const price = document.getElementById('newPrice').value.trim();
const desc = document.getElementById('newDesc').value.trim();
```

5. Thực hiện kiểm tra hợp lệ:
   - Nếu tên hoặc giá bị trống, hoặc giá không phải số dương, đặt thông báo lỗi (ví dụ: `errorMsg.textContent = "Vui lòng nhập tên và giá hợp lệ!"`) và không tiếp tục thêm.
   - Có thể dùng `isNaN()` hoặc chuyển `price` sang Number để kiểm tra.
   - Nếu tất cả hợp lệ, xóa thông báo lỗi (`errorMsg.textContent = ""` hoặc ẩn nó đi).
6. **Tạo phần tử sản phẩm mới:** Sử dụng `document.createElement` để tạo các phần tử cần thiết (article, h3, p, ...). Điền nội dung từ giá trị form:

```
const newItem = document.createElement('article');
newItem.className = 'product-item';
// Tạo các phần tử con và thêm vào newItem
const title = document.createElement('h3');
title.className = 'product-name';
title.textContent = name;
...
newItem.appendChild(title);
// (Tương tự tạo phần tử chứa mô tả, giá rồi append)
```

   Hoặc cách khác: dùng template string và `innerHTML` để tạo nhanh nội dung HTML cho sản phẩm mới, ví dụ:

```
newItem.innerHTML = `<h3 class="product-name">${name}</h3>
                     <p class="product-desc">${desc}</p>
                     <p class="product-price">Giá: ${price}đ</p>`;
```

   (Chọn phương pháp mà sinh viên thấy thuận tiện, nhưng chú ý tránh nhập trực tiếp HTML nếu có dữ liệu nhạy cảm để phòng XSS – trong bài này có thể tạm chấp nhận.)

7. **Chèn sản phẩm mới vào danh sách:** Lấy phần tử chứa danh sách sản phẩm (ví dụ sử dụng `document.getElementById('product-list')`). Thêm phần tử mới vào:

8. Nếu muốn thêm lên đầu: dùng `parentElement.insertBefore(newItem, parentElement.firstChild)`.
9. Nếu thêm cuối: dùng `parentElement.appendChild(newItem)`.
10. Ví dụ:

```
const productList = document.getElementById('product-list');
productList.prepend(newItem); // thêm vào đầu danh sách
```

11. **Cập nhật danh sách sản phẩm cho chức năng tìm kiếm:** Nếu ở Bài 3, ta đã lưu `productItems = document.querySelectorAll('.product-item')` lúc đầu, thì khi thêm sản phẩm mới, biến `productItems` (NodeList) sẽ **không tự động cập nhật** phần tử mới. Có vài cách xử lý:
12. Cách đơn giản: Sau khi thêm sản phẩm, gọi lại `document.querySelectorAll` để lấy lại danh sách, hoặc chuyển `productItems` thành mảng và push phần tử mới. Hoặc không lưu `productItems` cố định, mà mỗi lần tìm kiếm thì gọi `querySelectorAll` để luôn bao gồm cả sản phẩm mới.
13. Ví dụ, sau khi `productList.prepend(newItem)`, có thể cập nhật:

```
productItems = document.querySelectorAll('.product-item');
```

(Nếu khai báo `let productItems` thay vì `const` ban đầu, có thể gán lại.)

14. **Đóng form và reset:** Sau khi thêm sản phẩm thành công:
15. Đặt lại giá trị form: `addProductForm.reset();` (hoặc gán `""` cho từng field).
16. Ẩn form đi (tương tự như khi nhấn nút "Thêm sản phẩm"):
    `addProductForm.style.display = "none";`
17. Người dùng lần sau có thể nhấn "Thêm sản phẩm" để mở form trống và nhập tiếp.
18. Nếu có nút "Hủy", gắn sự kiện `click` cho nút đó để đơn giản là ẩn form và (tuỳ chọn) reset nội dung.
19. **Thử nghiệm:**
20. Nhấn "Thêm sản phẩm", nhập các giá trị khác nhau để kiểm tra validate (bỏ trống tên hoặc giá, nhập giá = 0, ...) xem thông báo lỗi có hiện.
21. Khi nhập đúng và nhấn "Thêm", kiểm tra sản phẩm mới xuất hiện đầu danh sách, với định dạng giống các sản phẩm cũ.
22. Thử dùng chức năng tìm kiếm để chắc chắn sản phẩm mới có thể được tìm thấy.

Bài tập nâng cao (không bắt buộc, không có đáp án mẫu)

Bài tập 5 (Nâng cao): Lưu trữ dữ liệu sản phẩm với LocalStorage
Nâng cấp trang web để các sản phẩm thêm mới không bị mất khi tải lại trang, bằng cách sử dụng **LocalStorage**:

- Mỗi khi thêm sản phẩm (Bài tập 4), sau khi cập nhật giao diện, lưu danh sách sản phẩm vào `localStorage` (có thể lưu dưới dạng chuỗi JSON). Gợi ý: tạo một mảng chứa các đối tượng sản phẩm `{ name, price, desc }`, mỗi lần thêm thì `push` vào mảng rồi `localStorage.setItem('products', JSON.stringify(mảng))`.

- Khi trang web được tải (load) lần tiếp theo, sử dụng JavaScript để kiểm tra `localStorage`: - Nếu có dữ liệu sản phẩm (key `'products'`), thì parse JSON và tạo giao diện danh sách sản phẩm tương ứng thay vì dùng các sản phẩm mẫu cố định trong HTML.

- Nếu chưa có, có thể khởi tạo `localStorage` với các sản phẩm mẫu ban đầu.

- Với cách làm này, dữ liệu sẽ được lưu trên trình duyệt người dùng. Thử nghiệm bằng cách thêm vài sản phẩm, sau đó nhấn F5 (tải lại) và kiểm tra xem sản phẩm đã thêm có được hiển thị hay không.

Bài tập 6 (Nâng cao): Hiệu ứng nâng cao với JavaScript
Chọn một trong các hướng sau để làm cho trang web sinh động và chuyên nghiệp hơn:

- **Hiệu ứng trực quan:** Thêm hiệu ứng chuyển cảnh (transition) khi ẩn/hiện form hoặc khi lọc sản phẩm. Ví dụ: khi bấm "Thêm sản phẩm", thay vì form xuất hiện ngay lập tức, có thể thêm lớp CSS để form trượt xuống mượt mà (sử dụng `transition` và thuộc tính như `max-height` hoặc `opacity`). Gợi ý: đặt `addProductForm.style.maxHeight = addProductForm.scrollHeight + "px"` để mở rộng, và `maxHeight = "0"` để thu gọn, kèm CSS `transition: max-height 0.5s`.

- **Tìm kiếm nâng cao:** Thay vì chỉ có một ô tìm kiếm đơn giản, bổ sung tính năng lọc nâng cao, ví dụ: thêm ô chọn (dropdown) để chọn khoảng giá, hoặc lọc theo danh mục sản phẩm. Khi đó, hàm lọc cần xét thêm điều kiện trên giá hoặc danh mục.

- **Sử dụng Fetch API (async/await):** Giả sử dữ liệu sản phẩm được lấy từ máy chủ, viết mã sử dụng `fetch` để lấy dữ liệu sản phẩm. Có thể tạo một file JSON (vd: `products.json`) chứa mảng sản phẩm mẫu, sau đó dùng `fetch('./products.json')` và `.then()` hoặc `async/await` để lấy dữ liệu rồi hiển thị ra giao diện. (Lưu ý: Nếu mở file HTML trực tiếp, fetch local file có thể bị chặn do CORS, nên cần chạy trên máy chủ cục bộ hoặc dùng Live Server).

- **Tính năng khác:** Sinh viên có thể sáng tạo thêm tính năng như **sắp xếp sản phẩm** theo tên hoặc giá, **xóa sản phẩm** khỏi danh sách, hoặc thậm chí mở popup chi tiết sản phẩm khi click vào một sản phẩm nào đó. Tất cả đều sử dụng kiến thức DOM, sự kiện và có thể kết hợp async/await (ví dụ: xác nhận xóa bằng một thao tác async giả lập).

Gợi ý chung: Bài tập nâng cao nhằm khuyến khích sinh viên tự tìm tòi mở rộng. Hãy chọn một hướng mà bạn hứng thú, phác thảo giải pháp và thử nghiệm. Giảng viên sẽ hỗ trợ nếu cần thiết trong quá trình thực hiện. Chúc các bạn học tốt!