

## LAB 3: ĐỒNG BỘ VÀ BẤT ĐỒNG BỘ

### Bài 1: Tạo form nhập thông tin

## Form Đăng Ký

Tên:

Email:

Gửi

Gửi thành công!

#### Yêu cầu:

- Tạo một form với các trường: Tên, Email.
- Khi người dùng nhấn nút "Gửi", hiển thị thông báo "Đang xử lý..." và sau 2 giây hiển thị thông báo "Gửi thành công!" hoặc "Gửi thất bại!" dựa trên kết quả của Promise.

#### Hướng dẫn

File.js

```
<body>
  <div class="container">
    <h1>Form Đăng Ký</h1>
    <form id="registrationForm">
      <div class="form-group">
        <label for="name">Tên:</label>
        <input type="text" id="name" required>
      </div>
      <div class="form-group">
        <label for="email">Email:</label>
        <input type="email" id="email" required>
      </div>
      <button type="submit">Gửi</button>
    </form>
    <div id="message"></div>
  </div>
  <script src="Bai1.js"></script>
</body>
```

File.js

```

document.getElementById('registrationForm').addEventListener('submit', function(event) {
    event.preventDefault();

    const name = document.getElementById('name').value;
    const email = document.getElementById('email').value;
    const message = document.getElementById('message');

    message.textContent = 'Đang xử lý...';

    simulateServerRequest(name, email)
        .then(response => {
            message.textContent = response;
            message.style.color = 'green';
        })
        .catch(error => {
            message.textContent = error;
            message.style.color = 'red';
        });
});

function simulateServerRequest(name, email) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            // Giả lập việc gửi dữ liệu lên server và nhận kết quả
            const success = Math.random() > 0.5;
            if (success) {
                resolve('Gửi thành công!');
            } else {
                reject('Gửi thất bại!');
            }
        }, 2000);
    });
}

```

- Lắng nghe sự kiện **submit** của form.
- Khi form được submit, ngăn không cho form gửi dữ liệu đi theo cách mặc định.
- Hiển thị thông báo "Đang xử lý..."
- Gọi hàm `simulateServerRequest` để giả lập việc gửi dữ liệu lên server.
- Hàm `simulateServerRequest` trả về một Promise. Sau 2 giây, Promise sẽ được giải quyết ngẫu nhiên với kết quả thành công hoặc thất bại.
- Dựa trên kết quả của Promise, hiển thị thông báo tương ứng cho người dùng.

## Bài 2: Quản lý người dùng

Tạo một ứng dụng quản lý người dùng với các tính năng:

1. Thêm người dùng mới.
2. Lấy danh sách người dùng từ server.
3. Xóa người dùng.

### Yêu cầu

1. Tạo một form với các trường: Tên, Email và một nút "Thêm người dùng".
2. Khi người dùng nhấn nút "Thêm người dùng", gửi thông tin lên server giả lập và cập nhật danh sách người dùng.

3. Hiển thị danh sách người dùng từ server.
4. Cho phép xóa người dùng từ danh sách.

## Quản lý người dùng

Tên:

Email:

Thêm người dùng

Thêm người dùng thành công!

Trần Ngọc Lan Anh (anh@gmail.com)

Xóa

Nguyễn Văn Bền (ban@gmail.com)

Xóa

Hướng dẫn:

File .html

```
<body>
  <div class="container">
    <h1>Quản lý người dùng</h1>
    <form id="userForm">
      <div class="form-group">
        <label for="name">Tên:</label>
        <input type="text" id="name" required>
      </div>
      <div class="form-group">
        <label for="email">Email:</label>
        <input type="email" id="email" required>
      </div>
      <button type="submit">Thêm người dùng</button>
    </form>
    <div id="message"></div>
    <ul id="userList"></ul>
  </div>
  <script src="Bai2.js"></script>
</body>
```

File .js

- Sử dụng sự kiện **DOMContentLoaded** để đảm bảo mã JavaScript chỉ chạy sau khi toàn bộ nội dung của trang đã được tải.
- Lắng nghe sự kiện submit của form để thêm người dùng mới.
- Sử dụng Promise để giả lập việc thêm người dùng, lấy danh sách người dùng, và xóa người dùng từ server.
- Cập nhật danh sách người dùng sau mỗi thao tác thêm hoặc xóa.

## Bài 3: Quản lý công việc

## Quản lý công việc

Học Vuejs

Thêm công việc

Thêm công việc thành công!

Học JavaScript

Hoàn thành

Xóa

Làm bài tập về nhà

Hoàn thành

Xóa

## Quản lý công việc

Học Vuejs

Thêm công việc

Thêm công việc thất bại!

Học JavaScript

Hoàn thành

Xóa

Làm bài tập về nhà

Hoàn thành

Xóa

Đánh dấu hoàn thành

## Quản lý công việc

Học Vuejs

Thêm công việc

Công việc "Học JavaScript" đã hoàn thành!

Học JavaScript

Hoàn thành

Xóa

Làm bài tập về nhà

Hoàn thành

Xóa

Xóa công việc

**Quản lý công việc**

Học Vuejs Thêm công việc

Xóa công việc "Làm bài tập về nhà" thành công!

Học JavaScript Hoàn thành Xóa

Làm bài tập về nhà Hoàn thành Xóa

Tạo một ứng dụng quản lý công việc (To-Do List) với các tính năng:

1. Thêm công việc mới.
2. Lấy danh sách công việc từ server.
3. Đánh dấu công việc là đã hoàn thành.
4. Xóa công việc.

### Yêu cầu

1. Tạo một form với trường nhập công việc và một nút "Thêm công việc".
2. Khi người dùng nhấn nút "Thêm công việc", gửi thông tin lên server giả lập và cập nhật danh sách công việc.
3. Hiển thị danh sách công việc từ server.
4. Cho phép đánh dấu công việc là đã hoàn thành và xóa công việc.

Hướng dẫn:

File .html

```
<body>
  <div class="container">
    <h1>Quản lý công việc</h1>
    <form id="todoForm">
      <input type="text" id="task" placeholder="Nhập công việc mới" required>
      <button type="submit">Thêm công việc</button>
    </form>
    <div id="message"></div>
    <ul id="todoList"></ul>
  </div>
  <script src="Bai3.js"></script>
</body>
```

File .js

- Sử dụng sự kiện DOMContentLoaded để đảm bảo mã JavaScript chỉ chạy sau khi toàn bộ nội dung của trang đã được tải.
- Lắng nghe sự kiện submit của form để thêm công việc mới.

- Sử dụng Promise để giả lập việc thêm công việc, lấy danh sách công việc, đánh dấu hoàn thành công việc, và xóa công việc từ server.
- Cập nhật danh sách công việc sau mỗi thao tác thêm, hoàn thành hoặc xóa.

#### **Bài 4: Hệ thống quản lý người dùng**

Từ bài tập 2, tạo một ứng dụng quản lý người dùng với các tính năng:

1. Thêm người dùng mới.
2. Lấy danh sách người dùng từ server giả lập.
3. Xóa người dùng.

#### **Yêu cầu**

1. Tạo một form với trường nhập tên người dùng và một nút "Thêm người dùng".
2. Khi người dùng nhấn nút "Thêm người dùng", gửi thông tin lên server giả lập và cập nhật danh sách người dùng.
3. Hiển thị danh sách người dùng từ server.
4. Cho phép xóa người dùng.
5. Sử dụng async và await thực hiện bài tập trên

Hướng dẫn:

File .js

- Sử dụng async và await để làm việc với các tác vụ bất đồng bộ một cách dễ dàng và trực quan.
- Sử dụng sự kiện DOMContentLoaded để đảm bảo mã JavaScript chỉ chạy sau khi toàn bộ nội dung của trang đã được tải.
- Lắng nghe sự kiện submit của form để thêm người dùng mới.
- Sử dụng các hàm async để giả lập việc thêm người dùng, lấy danh sách người dùng, và xóa người dùng từ server.
- Cập nhật danh sách người dùng sau mỗi thao tác thêm hoặc xóa.

## Bài 5: Lấy thông tin thời tiết từ API

# Weather App



## Ho Chi Minh City

Temperature: 27.26°C

Humidity: 83%

Weather: few clouds




### Yêu cầu:

- Tạo một form để nhập tên thành phố.
- Gửi yêu cầu tới API thời tiết và nhận dữ liệu thời tiết của thành phố.
- Hiển thị thông tin thời tiết trên giao diện.

*Lưu ý: nên nhập tên tiếng việt không dấu*

### Hướng dẫn

- B1: Vào trang <https://home.openweathermap.org/> để đăng ký tài khoản
- B2: Sau khi đăng ký xong vào mail xác nhận
- B3: Vào menu API lấy key

Key	Name	Status	Actions	Create key
e6a75bb2d155d76c5f783a0d3dfe536a	Default	Active	  	<input type="text" value="API key name"/> <input type="button" value="Generate"/>

### File .html

```
<body>
  <div class="container">
    <h1>Weather App</h1>
    <form id="weatherForm">
      <input type="text" id="city" placeholder="Enter city name" required>
      <button type="submit">Get Weather</button>
    </form>
    <div id="weatherResult"></div>
  </div>
  <script src="Bai5.js"></script>
</body>
```

### File .js

- Sử dụng async và await để làm việc với API bất đồng bộ.
- Lắng nghe sự kiện submit của form để lấy tên thành phố và gọi hàm getWeather để lấy dữ liệu thời tiết.
- Hàm getWeather gửi yêu cầu tới API OpenWeatherMap và trả về dữ liệu thời tiết.
- Hàm displayWeather hiển thị thông tin thời tiết trên giao diện.

```

async function getWeather(city) {
  const apiKey = 'e6a75bb2d155d76c5f783ad3dfe536a'; // Thay YOUR_API_KEY bằng API key của bạn từ OpenWeatherMap
  const response = await fetch('https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey}&units=metric');
  if (!response.ok) {
    const message = `An error has occurred: ${response.statusText}`;
    throw new Error(message);
  }
  return await response.json();
}

function displayWeather(data) {
  const { name, main, weather } = data;
  weatherResult.innerHTML = `
    <h2>${name}</h2>
    <p>Temperature: ${main.temp}°C</p>
    <p>Humidity: ${main.humidity}%</p>
    <p>Weather: ${weather[0].description}</p>
  `;
}

```

## Bài 6: Quản lý sản phẩm

### Yêu cầu:

- Người dùng có thể thêm sản phẩm vào giỏ hàng, và khi nhấn vào nút "Mua hàng", thông tin giỏ hàng sẽ được gửi đến máy chủ giả lập (mock server).

### Chú ý

- Đây là ví dụ sử dụng jsonplaceholder.typicode.com như một máy chủ giả lập (mock server) để thực hiện yêu cầu POST. Trong thực tế, bạn cần thay thế URL bằng API endpoint thực tế

## Bài 7: Tạo Form phản hồi người dùng

### Người Dùng Feedback

Anh

anh@itc.edu.vn

Mẫu mã đẹp, chất lượng tốt

Submit

Thank you for your feedback, Anh! Your message: "Mẫu mã đẹp, chất lượng tốt" has been received.



**Yêu cầu:**

- Tạo một form cho phép người dùng gửi phản hồi.
- Sử dụng fetch API để gửi dữ liệu đến server giả lập và hiển thị thông báo phản hồi từ server.

**Hướng dẫn:**

File .html

```
<body>
  <div class="container">
    <h1>Người Dùng Feedback</h1>
    <form id="feedbackForm">
      <input type="text" id="name" placeholder="Your Name" required>
      <input type="email" id="email" placeholder="Your Email" required>
      <textarea id="message" rows="4" placeholder="Your Feedback" required></textarea>
      <button type="submit">Submit</button>
    </form>
    <div id="response" class="response"></div>
  </div>
  <script src="Bai7.js"></script>
</body>
```

File .js

- **async function:** Hàm xử lý sự kiện submit được định nghĩa là async để có thể sử dụng await bên trong nó.
- **await fetch:** Sử dụng await để đợi kết quả của lệnh fetch. Điều này giúp mã nguồn dễ đọc và xử lý lỗi dễ dàng hơn.
- **response.ok:** Kiểm tra xem phản hồi từ server có thành công không. Nếu không, ném ra lỗi.
- **await response.json():** Chờ cho đến khi dữ liệu JSON được parse xong trước khi tiếp tục.

**Bài 8: Ứng dụng quản lý tiền tệ**

## Currency Converter

1 EUR = 1.09 USD

**Yêu cầu:**

- **Mô tả:** Xây dựng ứng dụng theo dõi và chuyển đổi tỷ giá tiền tệ giữa các loại tiền tệ khác nhau.
- **API sử dụng:** Sử dụng API từ các dịch vụ như ExchangeRate-API hoặc Open Exchange Rates để lấy tỷ giá tiền tệ và thực hiện chuyển đổi.

Hướng dẫn:

File .html

```
<body>
  <div class="container">
    <h1>Currency Converter</h1>
    <input type="number" id="amount" placeholder="Amount" required>
    <select id="fromCurrency">
      <option value="USD">USD</option>
      <option value="EUR">EUR</option>
      <option value="GBP">GBP</option>
      <option value="VND">VND</option>
      <!-- Add more currencies as needed -->
    </select>
    <select id="toCurrency">
      <option value="USD">USD</option>
      <option value="EUR">EUR</option>
      <option value="GBP">GBP</option>
      <option value="VND">VND</option>
      <!-- Add more currencies as needed -->
    </select>
    <button onclick="convertCurrency()">Convert</button>
    <div id="result" class="result"></div>
  </div>
  <script src="currency.js"></script>
</body>
```