Bài tập: XHR, Callback, Promise, Async/Await

Yêu cầu chung

- Tạo repo `f8-fullstack-day28` để làm bài tập. Sử dụng UI cho trước tại đây: https://codesandbox.io/p/devbox/holy-glitter-pvk74p (không bắt buộc).
- Deploy bài tập lên Github Pages
- Code phải **clean, dễ đọc, dễ hiểu**, có comment giải thích ở những chỗ quan trọng
- Giao diện **đơn giản nhưng rõ ràng**, tự làm CSS
- **Mục tiêu chính**: Nắm vững cách xử lý bất đồng bộ với Callback, Promise, Async/Await và hiểu rõ sự khác biệt giữa chúng

Cấu trúc thư mục

```
f8-fullstack-day28/
├── index.html (Trang chủ với links tới 3 bài)
├── bai1-xhr-callback/
│   ├── index.html
│   ├── style.css
│   └── app.js
├── bai2-promise/
│   ├── index.html
│   ├── style.css
│   └── app.js
└── bai3-async-await/
    ├── index.html
    ├── style.css
    └── app.js
```

Bài 1: XHR với Callback Pattern

Yêu cầu chi tiết

1.1. Tạo utility function để gọi API với XHR

Trong file `bai1-xhr-callback/app.js`, tạo function `sendRequest` tái sử dụng:

```
// Function này sẽ nhận vào method, url và callback
// Callback sẽ được gọi với (error, data) theo pattern phổ biến
function sendRequest(method, url, callback) {
    // Code XHR ở đây
    // Khi thành công: callback(null, data)
    // Khi lỗi: callback(error, null)
}
```

1.2. Implement 3 chức năng sử dụng JSONPlaceholder API

Chức năng 1: User Profile Card

- Có một input nhập User ID (từ 1-10) và button "Tìm User"
- Khi click button, gọi API: `https://jsonplaceholder.typicode.com/users/{id}`
- Hiển thị thông tin: Name, Email, Phone, Website, Company name, Address (street, city)
- Xử lý trường hợp user không tồn tại (hiển thị thông báo lỗi)

Chức năng 2: Posts với Comments

- Tự động load 5 posts đầu tiên khi vào trang
- API Posts: `https://jsonplaceholder.typicode.com/posts?_limit=5`
- Mỗi post có button "Xem comments"
- Khi click, gọi API: `https://jsonplaceholder.typicode.com/posts/{postId}/comments`
- Hiển thị comments bên dưới post (name, email, body)
- **Callback Hell Challenge**: Khi click "Xem comments", phải gọi 2 API nối tiếp:
    1. Lấy thông tin user của post: `/users/{userId}`
    2. Lấy comments của post: `/posts/{postId}/comments`
    3. Hiển thị cả author info và comments

Chức năng 3: Todo List với Filter

- Load todos của một user cụ thể
- API: `https://jsonplaceholder.typicode.com/users/{userId}/todos`
- Có 3 buttons filter: "Tất cả", "Đã hoàn thành", "Chưa hoàn thành"
- Hiển thị số lượng todo ở mỗi trạng thái
- Style khác nhau cho todo completed và incomplete

1.3. Loading State & Error Handling

- Mỗi khi gọi API, hiển thị text "Đang tải..." hoặc spinner
- Xử lý các lỗi: Network error, 404, 500...

- Hiển thị thông báo lỗi rõ ràng cho user

Bài 2: Refactor với Promise

Yêu cầu chi tiết

2.1. Tạo Promise wrapper cho XHR

Trong file `bai2-promise/app.js`, tạo function trả về Promise:

```
function sendRequest(method, url) {
    return new Promise((resolve, reject) => {
        // Wrap XHR logic trong Promise
        // resolve(data) khi thành công
        // reject(error) khi thất bại
    });
}
```

2.2. Refactor lại 3 chức năng từ Bài 1

Cải thiện Callback Hell từ chức năng 2:

- Sử dụng Promise chain với `.then()` để xử lý chuỗi API calls
- Code phải dễ đọc hơn so với callback hell ở Bài 1
- Thêm chức năng: Load thêm 5 posts nữa khi click "Xem thêm"

Giữ nguyên các chức năng:

- User Profile Card: Chuyển từ callback sang Promise
- Posts với Comments: Dùng Promise chain thay vì nested callbacks
- Todo List với Filter: Implement với Promise pattern

2.3. Promise Error Handling

- Sử dụng `.catch()` để bắt lỗi
- Implement retry mechanism: Nếu API fail, tự động retry 1 lần nữa sau 2 giây
- Hiển thị số lần retry cho user biết

Bài 3: Modern Approach với Async/Await

Yêu cầu chi tiết

3.1. Tạo async function wrapper

Trong file `bai3-async-await/app.js`:

```
// Có thể dùng fetch() API thay vì XHR để đơn giản hơn
async function sendRequest(method, url) {
    try {
        // Implementation với fetch hoặc Promise wrapper của XHR
    } catch (error) {
        // Error handling
    }
}
```

3.2. Refactor với async/await syntax

Cải thiện code từ Bài 2:

- Tất cả Promise chains chuyển thành async/await
- Code phải trông giống synchronous code, dễ đọc nhất
- Giữ nguyên tất cả chức năng

Implement lại 3 chức năng với async/await:

- User Profile Card: Dùng async/await cho API call
- Posts với Comments: Sequential API calls với async/await thay vì Promise chain
- Todo List với Filter: Clean code với async/await pattern

3.3. Error Handling với try/catch

- Wrap async code trong try/catch blocks
- Xử lý lỗi cụ thể cho từng loại (404, 500, network error)
- Giữ nguyên retry mechanism nhưng implement với async/await

> **Tip**: Hãy bắt đầu với Bài 1 để hiểu rõ callback pattern và vấn đề callback hell. Khi làm Bài 2, bạn sẽ thấy rõ Promise giải quyết vấn đề này như thế nào. Cuối cùng, Bài 3 với async/await sẽ cho thấy cách viết code bất đồng bộ modern và dễ đọc nhất. Deploy sớm và test kỹ trên Github Pages để đảm bảo mọi thứ hoạt động tốt.