SHỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 7

Tìm hiểu về Node.js và Redux

Giảng viên hướng dẫn	**: TS. Kim Ngọc Bách**
Họ và tên sinh viên	**: Phạm Trung Kiên**
Mã sinh viên	**: B22DCVT263**
Lớp	**: E22CQCN02-B**

Hà Nội – 2025

I. Tìm hiểu về Node.js

1. Giới thiệu về Node.js

Node.js là một nền tảng (runtime) mã nguồn mở cho phép thực thi mã JavaScript ở phía server. Nó dựa trên công nghệ V8 JavaScript Engine của Google Chrome, biến JavaScript từ ngôn ngữ chỉ chạy trên trình duyệt thành công cụ server-side mạnh mẽ.

Ứng dụng của Node.js:

- Xây dựng API (REST, GraphQL)
- Tạo ứng dụng backend cho web/mobile
- Viết các dịch vụ microservices
- Xử lý các thao tác I/O như đọc file, truy cập hệ thống

2. Node REPL và CLI

Node REPL

- REPL (Read-Eval-Print-Loop) là môi trường dòng lệnh tương tác cho phép thử nghiệm nhanh các đoạn mã JavaScript.
- **Các bước:**
  - **Read:** nhận đầu vào từ người dùng
  - **Eval:** thực thi đoạn mã đó
  - **Print:** in ra kết quả
  - **Loop:** tiếp tục chu trình

Node CLI

- Command Line Interface của Node.js cho phép chạy file JavaScript từ dòng lệnh (node filename.js) hoặc nhập lệnh trực tiếp.

3. Các biến toàn cục trong Node.js

Trong Node.js, một số biến/đối tượng luôn tồn tại và có thể dùng ở bất kỳ đâu:

Biến/Đối tượng	Ý nghĩa
global	Tương tự window trong trình duyệt

__dirname	Đường dẫn thư mục chứa file hiện tại
__filename	Đường dẫn tuyệt đối tới file hiện tại
process	Thông tin về tiến trình Node.js
module, exports	Cơ chế module của Node.js
setTimeout, setInterval	Các hàm hẹn giờ
console	In ra màn hình console

4. Khái niệm về Modules

Module trong Node.js là cách tổ chức mã thành các file riêng biệt, tái sử dụng được.

Các loại module:

- Core Modules: tích hợp sẵn, ví dụ http, fs, path
- Local Modules: do người dùng tự viết
- Third-Party Modules: cài từ npm, như express, axios

5. CommonJS vs ECMAScript Modules

CommonJS

- Hệ thống module truyền thống của Node.js

- Sử dụng `require()` để nhập và `module.exports` để xuất
- Mang tính đồng bộ

ES Modules (ESM)

- Chuẩn module hiện đại (dùng `import` và `export`)
- Hỗ trợ bất đồng bộ
- Tương thích với trình duyệt

Tiêu chí	CommonJS	ES Modules
Cú pháp import	require()	import
Cú pháp export	module.exports	export
Đồng bộ	Có	Không
Hỗ trợ trình duyệt	Không	Có

6. Các framework nổi bật từ Node.js

- Express.js: Framework phổ biến nhất cho backend web
- Nest.js: Framework theo hướng OOP, dựa trên TypeScript
- Electron.js: Xây dựng ứng dụng desktop dùng Node.js
- Hapi.js: Framework server mạnh mẽ khác

7. Express Framework

Tổng quan về Express

Express.js là một framework tối giản, nhanh, và linh hoạt giúp xây dựng các ứng dụng web và API hiệu quả trên nền Node.js.

Các tính năng chính:

- Routing mạnh mẽ
- Middleware xử lý request/response
- Hỗ trợ RESTful API
- Tích hợp dễ dàng với các cơ sở dữ liệu và thư viện khác

Các bước khởi tạo server Express:

1. Tạo thư mục dự án
2. Khởi tạo npm: `npm init -y`
3. Cài đặt Express: `npm install express`
4. Tạo file `index.js`:

Các phương thức HTTP phổ biến:

- `GET`: Lấy tài nguyên
- `POST`: Gửi dữ liệu mới
- `PUT`: Cập nhật toàn bộ tài nguyên
- `PATCH`: Cập nhật một phần tài nguyên
- `DELETE`: Xóa tài nguyên

Middleware trong Express

Middleware là các hàm được gọi giữa quá trình nhận request và gửi response.

Các loại Middleware:

- **Application-level**: dùng `app.use()`
- **Router-level**: dùng cho từng router
- **Error-handling**: xử lý lỗi
- **Built-in**: Express có sẵn như `express.json()`
- **Third-party**: cài thêm ví dụ như `morgan`, `cors`

Nguyên lý hoạt động:

- Yêu cầu đi qua từng middleware
- Middleware có thể:
    - Xử lý request
    - Kết thúc response
    - Hoặc gọi `next()` để đi tiếp

II. Tìm hiểu về Redux

1. Tổng quan về Redux

Redux là một thư viện JavaScript hỗ trợ quản lý trạng thái (state management) trong ứng dụng theo cách đơn giản và có thể dự đoán được. Nó được lấy cảm hứng từ mô hình kiến trúc Flux và thiết kế ngôn ngữ Elm.

Redux giúp toàn bộ dữ liệu (state) của ứng dụng được lưu trữ tập trung tại một nơi (store), dễ kiểm soát và debug, nhất là đối với những ứng dụng lớn nhiều component.

2. Nhu cầu ra đời Redux

Khi ứng dụng front-end ngày càng mở rộng, các luồng dữ liệu nội bộ giữa các thành phần trở nên phức tạp. Các framework như React cho phép từng component tự quản lý state, nhưng lại thiếu cơ chế quản lý tập trung. Redux giải quyết vấn đề đó bằng việc cung cấp một store duy nhất, nơi mọi sự thay đổi đều phải đi qua các quy trình chặt chẽ (dispatch -> reducer -> store update).

3. Khi nào cần công cụ quản lý state?

Ban đầu, ứng dụng đơn giản có thể chỉ cần quản lý state trong từng component riêng lẻ. Tuy nhiên khi:

- Dữ liệu cần chia sẻ qua nhiều component
- Các hành động phức tạp (ví dụ: đồng bộ nhiều API, cập nhật nhiều state cùng lúc)
- Quản lý sự kiện bất đồng bộ (async)

4. Lợi ích của Redux trong React Native

React Native vốn đã cung cấp cơ chế `props` để truyền dữ liệu, nhưng khi ứng dụng di động phát triển lớn hơn:

- Redux cho phép lưu trữ toàn bộ trạng thái tại một điểm duy nhất.

- Hỗ trợ debug dễ dàng nhờ Redux DevTools.
- Cho phép kiểm soát luồng dữ liệu qua từng action rõ ràng.
- Thân thiện khi làm việc nhóm do tính nhất quán cao.

=> Redux là giải pháp lý tưởng khi phát triển các ứng dụng di động React Native lớn.

5. Các thành phần chính trong Redux

Store

Store lưu toàn bộ trạng thái ứng dụng. Chỉ có một store duy nhất tồn tại trong một ứng dụng Redux.

Vai trò:

- Chứa state
- Cung cấp phương thức `dispatch`, `getState`
- Lắng nghe sự thay đổi của state qua `subscribe`

Action

Action là những object thuần (plain object) có thuộc tính type bắt buộc, mô tả một ý định thay đổi state.

Reducer

Reducer là những hàm nhận vào state hiện tại và action, trả về state mới.

Dispatch

Dispatch là phương thức dùng để gửi action vào store, yêu cầu state được cập nhật.

6. Redux Toolkit

Mục đích

Redux Toolkit (RTK) ra đời nhằm đơn giản hóa việc sử dụng Redux:

- Cấu hình store dễ dàng
- Giảm boilerplate code
- Tích hợp các công cụ async như createAsyncThunk

Các tính năng nổi bật

- configureStore(): Tự động tích hợp DevTools và Middleware.
- createSlice(): Tạo reducer + action chỉ với một lần khai báo.
- createAsyncThunk(): Hỗ trợ xử lý các tác vụ bất đồng bộ như fetch API.
- createEntityAdapter(): Tối ưu hóa việc xử lý tập dữ liệu dạng list.

RTK Query

RTK Query là module trong Redux Toolkit hỗ trợ:

- Fetch API tự động
- Cache và invalidation
- Auto-generate hook như useGetTodosQuery