📆 Hari 5 (Jum'at) - Pengantar State Management & Tugas Mingguan

🎯 Tujuan Pembelajaran Hari Ini

- Memahami masalah "Prop Drilling" dan mengapa state management diperlukan dalam aplikasi React yang kompleks.
- Mengenal konsep dasar State Management.
- Mendapatkan pengantar tentang beberapa solusi State Management di ekosistem React (Context API, Redux, Zustand).
- Memahami persyaratan tugas mingguan untuk Minggu 7.
- Mempersiapkan diri untuk sesi presentasi dan review.

📚 Materi Inti (2 Jam)

1. Mengapa State Management?

- **Masalah "Prop Drilling":**
  - Dalam aplikasi React, data (state) seringkali perlu dibagikan antar komponen.
  - Jika state berada di komponen induk yang jauh dari komponen anak yang membutuhkannya, kita harus meneruskan state tersebut melalui props melewati komponen-komponen perantara yang sebenarnya tidak membutuhkan state tersebut.
  - Proses meneruskan props melalui banyak level komponen ini disebut "Prop Drilling".
  - Prop drilling membuat kode sulit dibaca, dipelihara, dan di-refactor. Perubahan pada struktur komponen bisa memerlukan perubahan pada banyak komponen perantara hanya untuk meneruskan props.
- **Kapan State Management Diperlukan?**
  - Saat state perlu diakses oleh banyak komponen yang tidak memiliki hubungan parent-child langsung atau hubungannya sangat jauh.
  - Saat state sering diperbarui dan memengaruhi banyak bagian UI.
  - Saat logika bisnis terkait state menjadi kompleks.
- **Konsep Dasar State Management:**
  - State global atau state bersama yang bisa diakses oleh komponen manapun dalam aplikasi.
  - Mekanisme terpusat untuk memperbarui state secara predictable.
  - Komponen "mendengarkan" perubahan state dan me-render ulang saat state yang relevan berubah.

2. Solusi State Management di React

- **Context API (Bawaan React):**
  - **Apa itu Context API?** Fitur bawaan React yang memungkinkan Anda meneruskan data melalui pohon komponen tanpa harus meneruskan props secara manual di setiap level.
  - **Kapan Menggunakan Context API?** Cocok untuk state yang jarang diperbarui atau state yang dibutuhkan oleh banyak komponen di berbagai level (misalnya, tema UI, preferensi bahasa, informasi user yang sedang login).

- **Keterbatasan:** Tidak dioptimalkan untuk state yang sering diperbarui karena setiap perubahan context akan me-render ulang semua komponen yang mengonsumsinya, bahkan jika hanya sebagian kecil state yang berubah. Tidak menyediakan solusi bawaan untuk manajemen logika kompleks (seperti Redux).
- **Komponen Utama:** `createContext`, `Provider`, `useContext`.

```
// Contoh sederhana Context API
import React, { createContext, useContext, useState } from 'react';

// 1. Buat Context
const ThemeContext = createContext();

// 2. Buat Provider
function ThemeProvider({ children }) {
  const [theme, setTheme] = useState('light');
  const toggleTheme = () => {
    setTheme(prevTheme => (prevTheme === 'light' ? 'dark' : 'light'));
  };

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
}

// 3. Gunakan Context di Komponen Anak
function ThemedButton() {
  const { theme, toggleTheme } = useContext(ThemeContext);
  return (
    <button onClick={toggleTheme} style={{ background: theme ===
'light' ? '#fff' : '#333', color: theme === 'light' ? '#000' : '#fff'
}}>
      Toggle Theme ({theme})
    </button>
  );
}

// Di App.js atau komponen root:
// <ThemeProvider><ThemedButton /></ThemeProvider>
```

- **Redux (Library Eksternal):**
  - **Apa itu Redux?** Library state management yang menyediakan "single source of truth" untuk state aplikasi dalam sebuah "store". Menggunakan prinsip unidirectional data flow.
  - **Konsep Utama:** Store, Actions, Reducers, Dispatch.
  - **Kapan Menggunakan Redux?** Untuk aplikasi besar dengan state yang kompleks, sering diperbarui, dan logika bisnis yang rumit. Memudahkan debugging dan memiliki ekosistem middleware yang kaya.
  - **Kelebihan:** Predictable state updates, powerful tooling (Redux DevTools), ekosistem besar.

    - **Kekurangan:** Boilerplate code yang cukup banyak, kurva belajar yang lebih curam dibandingkan Context API atau Zustand.
    - **Integrasi dengan React:** Menggunakan library `react-redux` (`Provider`, `useSelector`, `useDispatch`).
- **Zustand (Library Eksternal):**
    - **Apa itu Zustand?** Library state management yang kecil, cepat, dan scalable, menggunakan hook-based API.
    - **Kapan Menggunakan Zustand?** Alternatif yang lebih sederhana dari Redux, cocok untuk berbagai ukuran aplikasi. Boilerplate minimal.
    - **Kelebihan:** Sangat mudah digunakan, performa baik, tidak memerlukan Provider wrapper (state bisa diakses di mana saja).
    - **Kekurangan:** Ekosistem middleware tidak sebesar Redux.
    - **Penggunaan:** Membuat store menggunakan fungsi `create` dan mengonsumsinya menggunakan hook.

```
// Contoh sederhana Zustand
import { create } from 'zustand';

// Buat store
const useCounterStore = create((set) => ({
  count: 0,
  increment: () => set((state) => ({ count: state.count + 1 })),
  decrement: () => set((state) => ({ count: state.count - 1 })),
}));

// Gunakan store di komponen
function Counter() {
  const count = useCounterStore((state) => state.count);
  const increment = useCounterStore((state) => state.increment);
  const decrement = useCounterStore((state) => state.decrement);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
  }
```

- **Memilih Solusi:**
    - Untuk state sederhana atau state yang hanya dibutuhkan oleh sub-tree komponen, mulai dengan `useState` dan prop drilling.
    - Jika prop drilling mulai terasa merepotkan untuk state yang jarang berubah, pertimbangkan Context API.
    - Untuk aplikasi yang lebih besar, state yang kompleks, atau jika Anda membutuhkan fitur seperti middleware dan devtools yang kuat, pertimbangkan Redux atau Zustand.

- Zustand seringkali menjadi pilihan yang baik sebagai alternatif Redux yang lebih ringan dan mudah.

---

3. Tugas Mingguan (Waktu Pengerjaan: Sisa Minggu + Akhir Pekan)

Tema Tugas: Membangun Aplikasi E-commerce Sederhana dengan React, React Router, Tailwind CSS, Konsumsi API, dan Autentikasi.

Persyaratan Fungsional:

1. **Setup Proyek:** Buat proyek React baru (Create React App atau Vite).
2. **Routing:** Implementasikan routing menggunakan `react-router-dom` untuk halaman-halaman berikut:
   - Halaman Utama (Home) - menampilkan daftar produk.
   - Halaman Detail Produk (`/products/:productId`).
   - Halaman Login (`/login`).
   - Halaman Register (`/register`).
   - (Opsional) Halaman Dashboard/Profil User (`/dashboard` atau `/profile`) - halaman yang dilindungi.
3. **Styling:** Gunakan Tailwind CSS untuk styling seluruh aplikasi.
4. **Konsumsi API:**
   - Gunakan `axios` untuk mengambil data.
   - Ambil daftar produk dari dummy API (misalnya, `https://fakestoreapi.com/products`) dan tampilkan di halaman utama.
   - Ambil data detail produk dari dummy API berdasarkan ID produk dari URL dan tampilkan di halaman detail produk.
   - Implementasikan fetching data untuk proses login dan register ke dummy API (misalnya, `https://reqres.in/api/login` dan `https://reqres.in/api/register`) atau simulasikan respons di frontend.
5. **Autentikasi Sederhana:**
   - Implementasikan form login dan register.
   - Simpan token autentikasi (simulasi) di `localStorage` setelah login berhasil.
   - (Opsional) Gunakan `axios` interceptor untuk menambahkan header `Authorization: Bearer <token>` ke permintaan ke endpoint yang dilindungi (simulasi).
   - (Opsional) Buat Protected Route untuk halaman Dashboard/Profil yang hanya bisa diakses jika user sudah login.
6. **Komponen Reusable:** Pecah UI menjadi komponen-komponen yang lebih kecil dan reusable (misalnya, `ProductItem`, `Header`, `Footer`, `InputField`, `Button`).
7. **Penanganan State:** Kelola state loading dan error untuk setiap permintaan API.
8. **Struktur Proyek:** Atur file dan folder proyek secara logis (misalnya, folder `components`, `pages`, `api`, `hooks`).
9. **Dokumentasi:** Buat file `README.md` yang menjelaskan:
   - Cara menjalankan proyek.
   - Daftar halaman dan rutenya.
   - Penjelasan singkat tentang struktur proyek.
   - (Opsional) Penjelasan tentang bagaimana autentikasi disimulasikan.

Output Tugas:

- Kode sumber aplikasi React Anda.
- File `README.md`.
- Semua di-push ke repositori GitHub pribadi Anda.

🗣️ Sesi Presentasi (Waktu disesuaikan)

Format Presentasi:

- Setiap santri atau kelompok mempresentasikan aplikasi e-commerce sederhana yang sudah dibuat.
- Durasi presentasi disesuaikan.

Materi Presentasi:

1. **Demo Aplikasi:** Tunjukkan fungsionalitas aplikasi (menampilkan daftar produk, melihat detail produk, login, register).
2. **Penjelasan Kode:** Jelaskan struktur proyek, komponen utama, bagaimana routing diimplementasikan, bagaimana data diambil dari API (`axios`, `useEffect`), dan bagaimana autentikasi disimulasikan.
3. **Tantangan dan Solusi:** Ceritakan tantangan teknis yang dihadapi selama pengerjaan tugas dan bagaimana Anda mengatasinya.
4. **Pembelajaran:** Apa pembelajaran utama dari pengerjaan tugas ini?

Sesi Review dan Feedback:

- Tanya jawab dan feedback konstruktif dari rekan dan mentor.

💡 Tips Belajar Tambahan

- **Refactor Secara Bertahap:** Jangan ragu untuk me-refactor kode Anda saat menemukan pola yang berulang atau struktur yang mulai sulit dikelola.
- **Gunakan React Developer Tools:** Extension browser ini sangat membantu untuk memeriksa hirarki komponen, state, dan props.
- **Simulasikan Backend:** Jika backend asli belum siap, gunakan dummy API atau simulasikan respons API di frontend untuk memungkinkan Anda fokus pada pengembangan frontend.
- **Fokus pada Konsep Inti:** Pastikan Anda benar-benar memahami konsep hooks (`useState`, `useEffect`, `useParams`, `useNavigate`), fetching data (`axios`), dan routing (`react-router-dom`).

🔗 Referensi Tambahan

- React Documentation - Context
- Redux Toolkit Documentation
- Zustand Documentation
- React Router DOM Documentation
- Tailwind CSS Documentation

Minggu ini kita sudah mempelajari banyak hal fundamental tentang React: fetching data, routing, dan dasar autentikasi. Tugas mingguan ini adalah kesempatan emas untuk menggabungkan semua konsep tersebut

menjadi satu aplikasi utuh. Fokus pada pemahaman alur data dan interaksi antar komponen. Selamat mengerjakan dan sampai jumpa di sesi presentasi!