PERTEMUAN 2

REACTJS

1. Tujuan Pembelajaran

1. Memahami konsep dasar React dan Virtual DOM.
2. Mampu membuat komponen React menggunakan JSX.
3. Memahami penggunaan props dan state dalam React.
4. Mampu membangun aplikasi web sederhana menggunakan React.

2. Alat dan Bahan

1. Komputer dengan sistem operasi Windows, macOS, atau Linux.
2. Node.js dan npm (Node Package Manager) terinstal.
3. Text editor seperti Visual Studio Code.
4. Browser modern (Chrome, Firefox, Edge, dll.)

3. ReactJS

A. Apa itu React?

React adalah **library JavaScript** yang digunakan untuk membangun antarmuka pengguna (UI) berbasis komponen. React dirancang untuk merespons perubahan data dan memperbarui UI secara efisien. React menggunakan konsep **Virtual DOM** untuk memastikan pembaruan UI yang cepat dan optimal, bahkan pada aplikasi yang kompleks.

Sejarah Singkat React

React dikembangkan oleh **Jordan Walke**, seorang software engineer di Facebook, pada tahun 2011. Awalnya, React dibuat untuk mengatasi tantangan dalam mengelola pembaruan yang konstan pada fitur **News Feed** Facebook. React memperkenalkan pendekatan baru dalam membangun UI dengan menggunakan **Virtual DOM**, yang memungkinkan pembaruan UI yang lebih efisien dibandingkan dengan manipulasi langsung pada DOM.

B. Virtual DOM

Apa itu DOM?

DOM (Document Object Model) adalah representasi struktur HTML dalam bentuk pohon (tree structure). Setiap perubahan pada UI (seperti interaksi pengguna) akan memicu pembaruan pada DOM. Namun, pembaruan DOM bisa menjadi operasi yang lambat dan mahal, terutama pada aplikasi web yang besar dan kompleks.

Apa itu Virtual DOM?

Virtual DOM adalah **representasi dalam memori** dari DOM yang sebenarnya. Virtual DOM adalah objek JavaScript ringan yang menyimpan properti dan atribut elemen-elemen pada halaman web. Ketika terjadi perubahan pada UI, React akan:

1. Memperbarui Virtual DOM terlebih dahulu (yang lebih cepat daripada memperbarui DOM langsung).
2. Membandingkan Virtual DOM yang baru dengan yang lama.
3. Menghitung perubahan minimal yang diperlukan untuk memperbarui DOM yang sebenarnya.
4. Menerapkan perubahan tersebut ke DOM dalam satu batch update.

Keuntungan Virtual DOM

- **Efisiensi**: React hanya memperbarui bagian yang berubah, bukan seluruh DOM.
- **Performansi**: Aplikasi menjadi lebih cepat dan responsif, bahkan pada UI yang kompleks.

C. Komponen (Components)

Komponen adalah **blok pembangun** dalam React. Setiap komponen adalah modul independen yang dapat digunakan kembali untuk membangun UI. Komponen memungkinkan pengembang untuk memecah UI menjadi bagian-bagian kecil yang dapat dikelola dengan mudah.

Contoh Komponen Sederhana

```
import React from 'react';

// Komponen Header
function Header() {
    return (
        <header>
            <h1>Aplikasi React Saya</h1>
        </header>
    );
}

// Komponen Main
function Main() {
    return (
        <main>
            <h2>Selamat datang di Aplikasi React Saya!</h2>
            <p>Ini adalah area konten utama.</p>
        </main>
    );
}
```

```jsx
// Komponen Footer
function Footer() {
    return (
        <footer>
            <p>&copy; 2023 Aplikasi React Saya</p>
        </footer>
    );
}

// Komponen App yang menggabungkan Header, Main, dan Footer
function App() {
    return (
        <div>
            <Header />
            <Main />
            <Footer />
        </div>
    );
}

export default App;
```

Dalam contoh di atas, kita memiliki tiga komponen: Header, Main, dan Footer. Komponen-komponen ini digabungkan dalam komponen App untuk membentuk struktur UI yang lengkap.

D. JSX (JavaScript XML)

JSX adalah sintaks yang digunakan React untuk mendefinisikan struktur UI. JSX terlihat seperti HTML, tetapi memungkinkan pengembang untuk menyisipkan kode JavaScript di dalamnya. Ini membuat komponen menjadi dinamis dan interaktif.

Contoh Penggunaan JSX

```jsx
import React, { useState } from 'react';

function Counter() {
    const [count, setCount] = useState(0);

    function handleClick() {
        setCount(count + 1);
    }

    return (
        <div>
            <h1>Hitungan: {count}</h1>
            <button onClick={handleClick}>Tambah</button>
        </div>
    );
}

export default Counter;
```

Dalam contoh ini, komponen Counter menggunakan **state** untuk menyimpan nilai hitungan. Setiap kali tombol diklik, nilai count akan bertambah, dan UI akan diperbarui secara otomatis.

E. Props (Properties)

Props adalah cara untuk mengirim data dari komponen induk (parent) ke komponen anak (child). Props bisa berupa string, number, boolean, object, atau bahkan fungsi.

Contoh Penggunaan Props

```jsx
function Greeting(props) {
    return <h1>Halo, {props.name}!</h1>;
}

// Penggunaan komponen Greeting dengan props
<Greeting name="John" />
```

Dalam contoh ini, komponen Greeting menerima props name dan menampilkan pesan "Halo, John!".

F. State

State adalah objek yang menyimpan data yang menentukan perilaku dan rendering sebuah komponen. Ketika state berubah, React akan secara otomatis me-render ulang komponen tersebut.

Contoh Penggunaan State

```jsx
import React, { useState } from 'react';

function Example() {
    const [name, setName] = useState('');
    const [age, setAge] = useState(0);
    const [email, setEmail] = useState('');

    const handleNameChange = (e) => {
        setName(e.target.value);
    };

    const handleAgeChange = (e) => {
        setAge(e.target.value);
    };

    const handleEmailChange = (e) => {
        setEmail(e.target.value);
    };

    return (
        <div>
            <input type="text" placeholder="Nama" value={name} onChange={handleNameChange} />
            <input type="number" placeholder="Umur" value={age} onChange={handleAgeChange} />
            <input type="email" placeholder="Email" value={email} onChange={handleEmailChange} /
>
            <p>{name} berumur {age} tahun dan emailnya adalah {email}.</p>
        </div>
    );
}

export default Example;
```

Dalam contoh ini, komponen Example menggunakan state untuk mengelola input pengguna dan menampilkan data yang dimasukkan.

G. Kesimpulan

React adalah library yang powerful untuk membangun antarmuka pengguna yang dinamis dan responsif. Dengan konsep-konsep seperti **Virtual DOM**, **Komponen**, **JSX**, **Props**, dan **State**, React memungkinkan pengembang untuk membuat aplikasi web yang efisien dan mudah dikelola.

Kelebihan React

- **Reusability**: Komponen dapat digunakan kembali di seluruh aplikasi.
- **Efisiensi**: Virtual DOM memastikan pembaruan UI yang cepat.
- **Fleksibilitas**: JSX memungkinkan penggabungan HTML dan JavaScript.

Keterbatasan React

- React hanya fokus pada **front-end** (UI), sehingga untuk fitur seperti server-side rendering dan routing, diperlukan framework tambahan seperti **Next.js**.

4. Langkah-langkah Praktikum

1. Persiapan Lingkungan

1. Pastikan Node.js dan npm sudah terinstal di komputer Anda. Anda dapat memeriksanya dengan menjalankan perintah berikut di terminal atau command prompt:

```
node -v
npm -v
```

2. Buat direktori baru untuk proyek React Anda:

```
mkdir praktikum-react
cd praktikum-react
```

3. Inisialisasi proyek React dengan menjalankan perintah berikut:

```
npx create-react-app my-react-app
cd my-react-app
```

4. Jalankan aplikasi React dengan perintah:

```
npm start
```

Aplikasi akan terbuka di browser pada alamat http://localhost:3000.

2. Membuat Komponen React

1. Buka file src/App.js di text editor Anda.
2. Ganti kode di dalamnya dengan kode berikut untuk membuat komponen sederhana:

```jsx
import React from 'react';

// Komponen Header
function Header() {
    return (
        <header>
            <h1>Aplikasi React Saya</h1>
        </header>
    );
}

// Komponen Main
function Main() {
    return (
        <main>
            <h2>Selamat datang di Aplikasi React Saya!</h2>
            <p>Ini adalah area konten utama.</p>
        </main>
    );
}

// Komponen Footer
function Footer() {
    return (
        <footer>
            <p>&copy; 2023 Aplikasi React Saya</p>
        </footer>
    );
}
```

```jsx
// Komponen App yang menggunakan Header, Main, dan Footer
function App() {
    return (
        <div>
            <Header />
            <Main />
            <Footer />
        </div>
    );
}

export default App;
```

3. Simpan file dan lihat perubahan di browser. Anda akan melihat tampilan sederhana dengan header, konten utama, dan footer.

3. Menggunakan JSX untuk Membuat Komponen Dinamis

1. Buat file baru di direktori src dengan nama Counter.js.
2. Tambahkan kode berikut untuk membuat komponen Counter yang dinamis:

```
import React, { useState } from 'react';

function Counter() {
    const [count, setCount] = useState(0);

    function handleClick() {
        setCount(count + 1);
    }

    return (
        <div>
            <h1>Hitungan: {count}</h1>
            <button onClick={handleClick}>Tambah</button>
        </div>
    );
}

export default Counter;
```

3. Buka file src/App.js dan impor komponen Counter:

```
import Counter from './Counter';
```

4. Tambahkan komponen Counter ke dalam komponen App:

```
function App() {
    return (
        <div>
            <Header />
            <Main />
            <Counter />
            <Footer />
        </div>
    );
}
```

5. Simpan file dan lihat perubahan di browser. Anda akan melihat tombol "Tambah" yang dapat meningkatkan hitungan saat diklik.

4. Menggunakan Props untuk Mengirim Data

1. Buat file baru di direktori src dengan nama Greeting.js.
2. Tambahkan kode berikut untuk membuat komponen Greeting yang menerima props:

```
function Greeting(props) {
    return <h1>Halo, {props.name}!</h1>;
}

export default Greeting;
```

3. Buka file src/App.js dan impor komponen Greeting:

```
import Greeting from './Greeting';
```

4. Tambahkan komponen Greeting ke dalam komponen App dan kirim props name:

```
function App() {
    return (
        <div>
            <Header />
            <Main />
            <Greeting name="John" />
            <Counter />
            <Footer />
        </div>
    );
}
```

5. Simpan file dan lihat perubahan di browser. Anda akan melihat pesan "Halo, John!" yang ditampilkan oleh komponen Greeting.

5. Menggunakan State untuk Mengelola Data

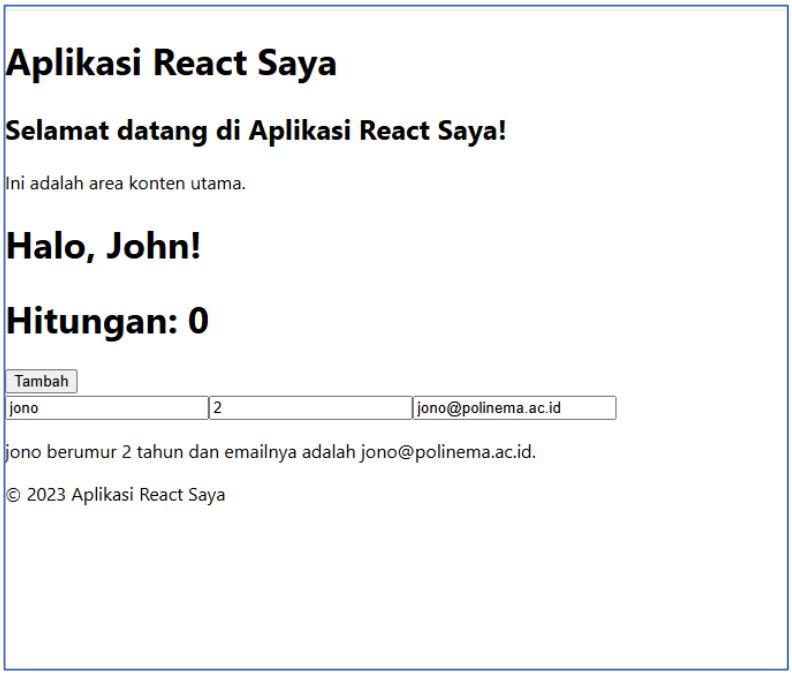
1. Buka file src/App.js dan tambahkan kode berikut untuk membuat komponen yang mengelola state:

```jsx
import React, { useState } from 'react';

function Example() {
    const [name, setName] = useState('');
    const [age, setAge] = useState(0);
    const [email, setEmail] = useState('');

    const handleNameChange = (e) => {
        setName(e.target.value);
    };

    const handleAgeChange = (e) => {
        setAge(e.target.value);
    };

    const handleEmailChange = (e) => {
        setEmail(e.target.value);
    };

    return (
        <div>
            <input type="text" placeholder="Nama" value={name} onChange={handleNameChange} /
>
            <input type="number" placeholder="Umur" value={age} onChange={handleAgeChange} /
>
            <input type="email" placeholder="Email" value={email} onChange={handleEmailChang
e} />
            <p>{name} berumur {age} tahun dan emailnya adalah {email}.</p>
        </div>
    );
}
```

2. Tambahkan komponen Example ke dalam komponen App:

```jsx
function App() {
    return (
        <div>
            <Header />
            <Main />
            <Greeting name="John" />
            <Counter />
            <Example />
            <Footer />
        </div>
    );
}
```

3. Simpan file dan lihat perubahan di browser. Anda akan melihat form input yang dapat mengupdate state dan menampilkan data yang dimasukkan.



5. Tugas

1. Buat komponen baru bernama TodoList yang menampilkan daftar tugas (todo list). Gunakan state untuk mengelola daftar tugas dan props untuk mengirim data tugas ke komponen anak.
2. Tambahkan fitur untuk menambahkan tugas baru ke dalam daftar menggunakan form input.
3. Implementasikan fitur untuk menghapus tugas dari daftar.

6. Kesimpulan

Dalam praktikum ini, Anda telah mempelajari dasar-dasar React, termasuk pembuatan komponen, penggunaan JSX, props, dan state. Anda juga telah membangun aplikasi web sederhana yang menggunakan konsep-konsep tersebut. React adalah alat yang powerful untuk membangun antarmuka pengguna yang dinamis dan responsif, dan dengan memahami konsep-konsep dasar ini, Anda dapat mulai mengembangkan aplikasi web yang lebih kompleks.