📆 Hari 2 (Selasa) - Komponen, Props, dan State

🎯 Tujuan Pembelajaran

- Memahami konsep dasar komponen dalam React dan perannya dalam membangun UI.
- Mampu membuat dan menggunakan komponen fungsional.
- Mampu menggunakan props untuk mengirim data dari komponen induk ke komponen anak.
- Mampu mengelola state lokal dalam komponen fungsional menggunakan hook `useState`.
- Memahami perbedaan antara props dan state serta kapan menggunakan keduanya.
- Memahami konsep "lifting state up" untuk berbagi state antar komponen.

🧠 Materi Inti (2 Jam)

1. Konsep Komponen di React (30 Menit)

- **Apa itu Komponen?**
  - Komponen adalah blok bangunan dasar (building blocks) dari aplikasi React.
  - Mereka adalah unit UI yang independen, dapat digunakan kembali, dan mengelola logika serta tampilannya sendiri.
  - Mirip dengan fungsi JavaScript, komponen menerima input (props) dan mengembalikan elemen React yang menjelaskan apa yang seharusnya muncul di layar.
  - **Analogi:** Bayangkan komponen seperti Lego bricks. Anda bisa membuat berbagai bentuk dan struktur (UI) dengan menggabungkan Lego bricks yang berbeda (komponen).
- **Jenis Komponen:**
  - **Komponen Fungsional (Functional Components):**
    - Ditulis sebagai fungsi JavaScript biasa.
    - Menerima `props` sebagai argumen pertama.
    - Mengembalikan elemen React (JSX).
    - Ini adalah cara modern dan direkomendasikan untuk menulis komponen di React, terutama sejak diperkenalkannya Hooks.
    - **Contoh:**

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

  - **Komponen Kelas (Class Components):**
    - Ditulis sebagai kelas JavaScript yang mewarisi dari `React.Component`.
    - Memiliki metode `render()` yang mengembalikan elemen React (JSX).
    - Mengakses props melalui `this.props` dan state melalui `this.state`.
    - Kurang umum digunakan dalam pengembangan baru karena komponen fungsional dengan Hooks menawarkan cara yang lebih sederhana dan ringkas untuk mengelola state dan lifecycle.
    - **Contoh:**

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

- **Membuat dan Menggunakan Komponen:**
  - Komponen didefinisikan sebagai fungsi atau kelas.
  - Digunakan dalam JSX seperti tag HTML kustom, dengan nama komponen diawali huruf kapital (PascalCase).
  - **Contoh:**

```
function App() {
  return (
    <div>
      <Welcome name="Alice" />
      <Welcome name="Bob" />
    </div>
  );
}
```

2. Menggunakan Props untuk Komunikasi Antar Komponen (45 Menit)

- **Apa itu Props?**
  - Singkatan dari "properties".
  - Objek yang berisi data yang diteruskan dari komponen induk (parent) ke komponen anak (child).
  - Props memungkinkan komponen untuk menerima data dari luar, membuatnya dinamis dan dapat digunakan kembali.
- **Sifat Props:**
  - **Read-only:** Komponen anak tidak boleh mengubah props yang diterimanya. Props bersifat immutable dari sudut pandang komponen anak.
  - **Analogi:** Props seperti argumen yang Anda berikan ke sebuah fungsi. Fungsi tersebut menggunakan argumen tersebut, tetapi tidak mengubah nilai asli argumen di luar fungsi.
- **Meneruskan Props:**
  - Data diteruskan ke komponen anak sebagai atribut dalam JSX.
  - Nilai props bisa berupa string, angka, boolean, array, objek, fungsi, atau bahkan elemen React lainnya.
  - **Contoh:**

```
function ParentComponent() {
  const greetingText = "Selamat Pagi";
  const user = { name: "Budi", age: 25 };

  return (
    <ChildComponent
```

```
        text={greetingText}
        userData={user}
        isLoggedIn={true}
        items={[1, 2, 3]}
        onClick={() => console.log('Clicked!')}
      />
    );
}
```

- **Menerima Props:**
  - Komponen fungsional menerima props sebagai argumen pertama fungsi.
  - Komponen kelas mengakses props melalui `this.props`.
  - **Contoh (Komponen Fungsional):**

```
function ChildComponent(props) {
  return (
    <div>
      <p>{props.text}</p>
      <p>Nama: {props.userData.name}, Usia: {props.userData.age}
</p>
      {props.isLoggedIn && <p>User is logged in</p>}
      <button onClick={props.onClick}>Click Me</button>
    </div>
  );
}
```

- **Destructuring Props:** Cara umum dan rapi untuk mengakses props dalam komponen fungsional.
  - **Contoh:**

```
function ChildComponent({ text, userData, isLoggedIn, onClick })
{
  return (
    <div>
      <p>{text}</p>
      <p>Nama: {userData.name}, Usia: {userData.age}</p>
      {isLoggedIn && <p>User is logged in</p>}
      <button onClick={onClick}>Click Me</button>
    </div>
  );
}
```

3. Mengelola State dalam Komponen Fungsional (45 Menit)

- **Apa itu State?**
  - Data yang dikelola di dalam komponen itu sendiri.
  - State bersifat mutable (dapat diubah).

    - Perubahan pada state akan memicu React untuk me-render ulang komponen tersebut dan anak-anaknya.
    - State digunakan untuk data yang dapat berubah seiring waktu karena interaksi pengguna atau event lainnya.
    - **Analogi:** State seperti memori jangka pendek komponen. Komponen mengingat informasi ini dan bereaksi ketika informasi itu berubah.
- **State dalam Komponen Fungsional Menggunakan `useState` Hook:**
    - `useState` adalah Hook yang memungkinkan Anda menambahkan state ke komponen fungsional.
    - Dipanggil di dalam komponen fungsional.
    - Menerima nilai awal state sebagai argumen (opsional).
    - Mengembalikan array dengan dua elemen:
        1. Nilai state saat ini.
        2. Fungsi untuk memperbarui state tersebut.
    - **Contoh:**

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

function Counter() {
  // Mendeklarasikan state variable 'count' dengan nilai awal 0
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

- **Memperbarui State:**
    - **Penting:** Jangan pernah mengubah state secara langsung (misalnya, `count = count + 1`).
    - Selalu gunakan fungsi updater yang dikembalikan oleh `useState` (misalnya, `setCount`).
    - Pembaruan state bersifat asynchronous. Jika Anda perlu memperbarui state berdasarkan nilai state sebelumnya, gunakan bentuk fungsi dari updater.
    - **Contoh Pembaruan Berdasarkan Nilai Sebelumnya:**

```
setCount(prevCount => prevCount + 1);
```

      Ini penting jika pembaruan state terjadi secara berurutan atau bergantung pada nilai state yang mungkin sudah berubah.
- **Perbedaan Props vs State (Ringkasan):**
    - **Props:** Data yang diteruskan dari induk ke anak. Read-only di komponen anak. Membuat komponen dapat digunakan kembali dengan data yang berbeda.

- **State:** Data yang dikelola di dalam komponen itu sendiri. Dapat diubah oleh komponen. Digunakan untuk data yang berubah seiring waktu dan memengaruhi rendering komponen.

📝 Praktik Mandiri (8 Jam)

Praktik ini akan memandu Anda membuat aplikasi sederhana yang menggunakan komponen, props, dan state.

1. **Setup Project React:** (Jika belum dilakukan dari Hari 1)
   - Inisialisasi project React baru menggunakan Vite (direkomendasikan karena lebih cepat).

     ```
     npm create vite@latest my-react-app --template react
     cd my-react-app
     npm install
     npm run dev
     ```

   - **Tujuan:** Memastikan lingkungan pengembangan React siap.
2. **Buat Komponen Fungsional Sederhana (`Header.jsx`):**
   - Buat file baru `src/components/Header.jsx`.
   - Buat komponen fungsional yang mengembalikan elemen `<h1>` dengan judul aplikasi.
   - **Contoh:**

     ```jsx
     // src/components/Header.jsx
     import React from 'react';

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

     export default Header;
     ```

   - **Tujuan:** Memahami struktur dasar komponen fungsional.
3. **Buat Komponen dengan Props (`Greeting.jsx`):**
   - Buat file baru `src/components/Greeting.jsx`.
   - Buat komponen fungsional `Greeting` yang menerima `name` sebagai props.
   - Tampilkan pesan "Hello, [name]!" menggunakan nilai props.
   - **Contoh:**

     ```

     ```

```jsx
// src/components/Greeting.jsx
import React from 'react';

function Greeting(props) {
  return (
    <p>Hello, {props.name}!</p>
  );
}

export default Greeting;
```

- **Tujuan:** Memahami cara menerima dan menggunakan props.

4. **Buat Komponen dengan State (`Counter.jsx`):**
   - Buat file baru `src/components/Counter.jsx`.
   - Import hook `useState` dari 'react'.
   - Buat state `count` dengan nilai awal 0 menggunakan `useState`.
   - Tampilkan nilai `count` dan tambahkan tombol.
   - Tambahkan event handler `onClick` pada tombol untuk memanggil fungsi updater `setCount` dan menambah nilai `count`.
   - **Contoh:**

```jsx
// src/components/Counter.jsx
import React, { useState } from 'react';

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

  return (
    <div>
      <p>Anda mengklik sebanyak {count} kali</p>
      <button onClick={() => setCount(count + 1)}>
        Klik untuk menambah
      </button>
    </div>
  );
}

export default Counter;
```

   - **Tujuan:** Memahami cara mendeklarasikan, membaca, dan memperbarui state lokal menggunakan `useState`.

5. **Integrasi Komponen dalam `App.jsx`:**
   - Buka file `src/App.jsx`.
   - Hapus konten default jika ada.
   - Import komponen `Header`, `Greeting`, dan `Counter`.

- Gunakan komponen-komponen tersebut dalam JSX yang dikembalikan oleh komponen `App`.
- Teruskan props `name` ke komponen `Greeting` dengan nilai yang berbeda.
- **Contoh:**

```
// src/App.jsx
import React from 'react';
import Header from './components/Header';
import Greeting from './components/Greeting';
import Counter from './components/Counter';

function App() {
  return (
    <div>
      <Header />
      <Greeting name="Alice" />
      <Greeting name="Bob" />
      <Counter />
    </div>
  );
}

export default App;
```

- **Tujuan:** Memahami cara menyusun UI dari beberapa komponen dan meneruskan props dari induk ke anak.

6. **Uji Aplikasi dan Amati Perubahan:**

- Jalankan aplikasi (`npm run dev`).
- Buka di browser.
- Amati tampilan aplikasi yang merupakan gabungan dari komponen-komponen yang Anda buat.
- Klik tombol pada komponen `Counter`. Amati bagaimana hanya angka di komponen `Counter` yang berubah, bukan seluruh halaman atau komponen lain.
- Gunakan React Developer Tools (install extension browser jika belum) untuk memeriksa hierarki komponen, props yang diterima oleh `Greeting`, dan state `count` di komponen `Counter`.
- **Tujuan:** Memverifikasi integrasi komponen, memahami bagaimana state lokal memicu re-render, dan menggunakan Dev Tools untuk debugging.

7. **Latihan Lifting State Up (Opsional tapi Direkomendasikan):**

- Modifikasi aplikasi sehingga state `count` dikelola di komponen `App.jsx`.
- Teruskan nilai `count` dan fungsi `setCount` sebagai props ke komponen `Counter`.
- Ubah komponen `Counter` untuk menerima `count` dan `setCount` sebagai props dan menggunakan props tersebut.
- **Tujuan:** Memahami konsep "lifting state up" untuk berbagi state antar komponen yang tidak memiliki hubungan induk-anak langsung (dalam contoh ini, `App` menjadi induk dari `Counter` yang mengelola state).

💡 Tips untuk Pemula

- **Pahami Alur Data Satu Arah:** Data di React mengalir ke bawah, dari komponen induk ke anak melalui props. Ini adalah prinsip penting yang membuat state management lebih mudah diprediksi.
- **State Lokal vs State Global:** `useState` digunakan untuk state lokal komponen. Untuk state yang perlu diakses oleh banyak komponen di berbagai level, pertimbangkan state management library (akan dibahas di minggu selanjutnya) atau Context API.
- **Immutability:** Jangan pernah mengubah state atau props secara langsung. Selalu gunakan fungsi updater state atau buat objek/array baru saat memperbarui state yang berisi objek atau array.
- **React Developer Tools:** Ini adalah alat yang sangat berharga. Gunakan untuk memeriksa hierarki komponen, melihat props dan state, serta memahami kapan dan mengapa komponen di-render ulang.
- **Latihan, Latihan, Latihan:** Konsep komponen, props, dan state adalah inti dari React. Semakin sering Anda mempraktikkannya, semakin baik pemahaman Anda.
- **Baca Dokumentasi Resmi:** Dokumentasi React sangat komprehensif dan selalu up-to-date. Jadikan kebiasaan untuk merujuk ke sana.

📚 Referensi

- React Official Docs - Your First Component
- React Official Docs - Passing Props to a Component
- React Official Docs - State: A Component's Memory
- React Official Docs - Sharing State Between Components (Lifting State Up)
- Functional vs Class Components in React
- Understanding State and Props in React