Hooks

Tujuan

Program yang membimbing siswa dalam memperdalam terkait pemahaman tentang hooks yang ada pada React.
React sendiri merupakan framework JavaScript yang biasa digunakan untuk pengembangan web dari sisi frontend

Hooks

Hooks adalah fitur dalam React yang diperkenalkan pada React 16.8. Hooks memungkinkan Anda menggunakan fitur React seperti state dan lifecycle methods di dalam functional components, tanpa perlu menggunakan class components.

Mengapa Hooks ?

Sebelum Hooks, React memiliki class components untuk menangani state dan lifecycle methods. Hooks memberikan cara yang lebih sederhana dan efisien untuk mengelola state, *use effect*, dan fitur React lainnya menggunakan functional components.

Manfaat Hooks

1. Kode lebih sederhana: Mengurangi boilerplate yang diperlukan untuk membuat class components.
2. Reuse Logic : Hooks seperti custom hooks memungkinkan logika dapat digunakan ulang.
3. Stateful Logic : Memudahkan penggunaan stateful logic di functional components.
4. Improved Readability : Komponen menjadi lebih kecil dan mudah dibaca.

Kapan Menggunakan Hooks

1. Ketika Anda ingin menggunakan state atau efek samping di dalam functional components.
2. Ketika Anda ingin menghindari kerumitan class components.
3. Ketika logika yang kompleks membutuhkan pemisahan dan penggunaan ulang.

Jenis-jenis Hooks

1. useState
2. useEffect
3. useContext
4. useRef
5. useReducer
6. useMemo
7. useCallback

useState

Digunakan untuk mengelola state dalam functional components.

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

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

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
      <button onClick={() => setCount(count - 1)}>Decrement</button>
    </div>
  );
}
export default Counter;
```

useEffect

Digunakan untuk menangani efek samping seperti fetching data, manipulasi DOM, atau melakukan subscription.

```
import React, { useState, useEffect } from "react";
function Timer() {
  const [seconds, setSeconds] = useState(0);
  useEffect(() => {
    const interval = setInterval(() => {
      setSeconds((prev) => prev + 1);
    }, 1000);
    return () => clearInterval(interval); // Cleanup
  }, []);
  return <p>Seconds: {seconds}</p>;
}
export default Timer;
```

useContext

Digunakan untuk mengakses data dari React Context tanpa harus menggunakan props drilling.

```
import React, { createContext, useContext } from "react";
const UserContext = createContext();
function App() {
  return (
    <UserContext.Provider value={{ name: "John Doe" }}>
      <UserProfile />
    </UserContext.Provider>
  );
}
function UserProfile() {
  const user = useContext(UserContext);
  return <p>Welcome, {user.name}!</p>;
}
export default App;
```

useRef

Digunakan untuk mengakses elemen DOM atau menyimpan nilai yang tidak memicu re-render ketika berubah.

```
import React, { useRef } from "react";
function TextInput() {
  const inputRef = useRef();

  const focusInput = () => {
    inputRef.current.focus();
  };
  return (
    <div>
      <input ref={inputRef} type="text" placeholder="Type something..." />
      <button onClick={focusInput}>Focus Input</button>
    </div>
  );
}
export default TextInput;
```

useReducer

Digunakan untuk mengelola state yang kompleks menggunakan reducer (alternatif dari useState)

```
import React, { useReducer } from "react";
const initialState = { count: 0 };
function reducer(state, action) {
  switch (action.type) {
    case "increment":
      return { count: state.count + 1 };
    case "decrement":
      return { count: state.count - 1 };
    default:
      throw new Error();
  } }
```

```
function Counter() {
  const [state, dispatch] = useReducer(reducer,
initialState);
  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type:
"increment" })}>Increment</button>
      <button onClick={() => dispatch({ type:
"decrement" })}>Decrement</button>
    </div>
  ); }
export default Counter;
```

useMemo

Digunakan untuk mengoptimalkan performa dengan menghitung nilai yang memakan waktu hanya ketika dependensi berubah.

```
import React, { useState, useMemo } from "react";

function ExpensiveCalculation({ num }) {
  const calculate = (num) => {
    console.log("Calculating...");
    return num * 2;
  };
  const result = useMemo(() => calculate(num), [num]);

  return <p>Result: {result}</p>;
}

export default ExpensiveCalculation;
```

useCallback

Digunakan untuk mem-memoisasi fungsi sehingga tidak dibuat ulang kecuali dependensi berubah

```
import React, { useState, useCallback } from "react";
function Button({ handleClick }) {
  console.log("Button rendered");
  return <button onClick={handleClick}>Click Me</button>;
}
function App() {
  const [count, setCount] = useState(0);
  const increment = useCallback(() => {
    setCount((prev) => prev + 1);
  }, []);
  return (
    <div>
      <p>Count: {count}</p>
      <Button handleClick={increment} />
    </div>
  );
}
export default App;
```

Refactor

Refactor (useState)

```
import React useEffect } from "react";
import produkData from "../utils/produkData";
import styles from "../styles/Produk.module.css";

function Produk() {
  const [produkList, setProdukList] = useState([...produkData]);
// State untuk menyimpan daftar produk
  const handleClick = () => {
    const newProduk = {
      id: produkList.length + 1,
      nama: "Printer Epson",
      tahun: 2023,
      harga: "Rp 3.000.000",
      gambar: "https://via.placeholder.com/150",
    };
    // Menambahkan produk baru ke state produkList
    setProdukList((prevList) => [...prevList, newProduk]);
    alert("Produk baru berhasil ditambahkan!");
  };
```

Refactor (useState)

```
  return (
    <div  className={styles.produkContainer}>
      <h1  className={styles.title}>Daftar  Produk</h1>
      <div  className={styles.cardContainer}>
        {produkList.map((item)  =>  (
           <div  key={item.id}  className={styles.card}>
            <img  src={item.gambar}  alt={item.nama}  />
            <h3>{item.nama}</h3>
            <p>Tahun:  {item.tahun}</p>
            <p>Harga:  {item.harga}</p>
          </div>
        ))}
      </div>
      <button  onClick={handleClick}  className={styles.addButton}>
        Tambah  Produk  Baru
      </button>
    </div>
  );
}
export  default  Produk;
```

NF ACADEMY
DIGITAL TRAINING SOLUTIONS

Refactor (useEffect)

```
  // useEffect untuk inisialisasi data produk (misalnya mengambil data
dari API)
  useEffect(() => {
    // Jika produkData diambil dari API, bisa menambahkan kode di sini.
    // Simulasi dengan setTimeout atau fetch data API
    console.log("Komponen Produk telah dimuat!");
  }, []); // Menggunakan [] untuk menandakan bahwa ini hanya dijalankan
sekali saat pertama kali komponen dimuat

  // useEffect untuk memantau perubahan pada produkList
  useEffect(() => {
    console.log("Daftar produk diperbarui:", produkList);
    // Menyimpan perubahan produk ke localStorage atau database bisa
dilakukan di sini
  }, [produkList]); // Efek ini hanya dijalankan ketika produkList berubah
  //cek pada console
```

Referensi

https://react.dev/learn/describing-the-ui

https://nextjs.org/learn/react-foundations