TUGAS PENDAHULUAN

Praktikum Rekayasa Perangkat Lunak Berbasis Komponen

MODUL 3

REACT STATE AND LIFESCYCLE

Nama : Fadel Rizky Nurfitanto
NIM :21120120140098
Kelompok :17

1. Dalam komponen fungsional React, kita dapat menggunakan Hook seperti ‘`useState`’ dan ‘`useEffect`’ untuk menginisiasi dan memperbarui state.

   - Menginisiasi State : untuk menginisiasi state pada functional components kita dapat menggunakan Hook ‘useState’.

     Contoh : kita menginisiasi state ‘count’ dengan nilai awal 0 menggunakan ‘useState’.

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

     function MyComponent() {
       const [count, setCount] = useState(0);
       // ...
     }
     ```

   - Memperbarui State : Untuk memperbarui state, kita dapat menggunakan fungsi ‘setCount’. atau fungsi set yang sesuai dengan nama state yang kita gunakan.

     Contoh :

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

  const incrementCount = () => {
    setCount(count + 1); // Memperbarui
state count dengan nilai yang baru
  };

  // ...
}
```

2. 3 fase component lifecycle :
   - Mounting : fase ketika components di buat atau pertama kali di render ke DOM. Pada fase ini ada tiga methods yang akan di eksekusi yaitu *componentWillMount, render*, dan *componentDidMount*.
   - Updating : fase ketika sebuah component akan di render ulang, biasanya ini terjadi ketika ada perubahan pada state atau props yang mengakibatkan perubahan DOM. Pada fade ini terdapat 5 methods yang akan di eksekusi seperti *componentWillReceiveProps, shouldComponentUpdate*
   - Unmounting : fase ketika component di hapus dari DOM. Pada fase ini hanya ada satu method yang akan di eksekusi yaitu *componentWillUnmount*, yang di jalankan sebelum sebuah component di hapus dari DOM.

3. Cara membuat conditional rendering :
   - Menggunakan If-Else dalam Render: menggunakan pernyataan '*if'* untuk menentukan kondisi dan mengembalikan JSX yang sesuai.

```
function MyComponent(props) {
  if (props.condition) {
    return <p>Kondisi terpenuhi.</p>;
  } else {
    return <p>Kondisi tidak terpenuhi.</p>;
  }
}
```

- Menggunakan Operator Ternary :

  Operator ternary *('condition ? trueExpression : falseExpression'*) memungkinkan Anda untuk melakukan conditional rendering dalam satu baris.

```
function MyComponent(props) {
  return props.condition ? <p>Kondisi
terpenuhi.</p> : <p>Kondisi tidak
terpenuhi.</p>;
}
```

- Menggunakan Logical && Operator:

  Operator '&&' akan mengembalikan elemen JSX kedua hanya jika kondisi pertama benar.

```
function MyComponent(props) {
  return props.condition && <p>Kondisi
terpenuhi.</p>;
}
```

- Menggunakan Metode '.map' untuk Mengembalikan Elemen Bersyarat:

```
function MyComponent(props) {
  return (
    <div>
      {props.items.map((item, index) => (
        item.condition && <p
key={index}>{item.text}</p>
      ))}
    </div>
  );
}
```

  Di atas, kita menggunakan metode 'map' untuk melakukan conditional rendering pada elemen-elemen dalam larik.

- Menggunakan Variabel untuk Kondisi Lebih Rumit:

  Kita dapat menggunakan variabel untuk menyimpan kondisi

yang lebih rumit sebelum melakukan conditional rendering.

```
function MyComponent(props) {
  const shouldRender = props.conditionA &&
props.conditionB;

  return shouldRender ? <p>Kondisi
terpenuhi.</p> : null;
}
```

4. Cara event handling pada jsx :

- Menambahkan Handler ke elemen : Kita dapat menambahkan handler event langsung ke elemen JSX dengan menggunakan atribut seperti `'onClick', 'onChange'`, dll.

  Contoh :

  Di bawah, `'onClick'` adalah atribut yang menentukan handler untuk event klik pada elemen **`'<button>'`**.

```
function MyComponent() {
  const handleClick = () => {
    alert("Tombol diklik!");
  };

  return (
    <button onClick={handleClick}>
      Klik Saya
    </button>
  );
}
```

- Menggunakan Arrow function dalam Inline : Kita juga dapat menggunakan fungsi panah (arrow function) dalam deklarasi inline untuk menangani event.

```
function MyComponent() {
  return (
    <button onClick={() => alert("Tombol
diklik!")}>
      Klik Saya
    </button>
  );
}
```

- Menggunakan metode di komponen :

```
function MyComponent() {
  const handleClick = () => {
    alert("Tombol diklik!");
  };

  return (
    <button onClick={handleClick}>
      Klik Saya
    </button>
  );
}
```

Metode ‘`handleClick`’ didefinisikan dalam komponen dan digunakan sebagai handler event.

- Meneruskan Argumen ke Handler :

Kita juga dapat meneruskan argumen tambahan ke handler event dengan menggunakan fungsi panah (arrow function) atau ‘`.bind`’.

```
function MyComponent() {
  const handleClick = (message) => {
    alert(message);
  };

  return (
    <button onClick={() => handleClick("Tombol
diklik!")}>
      Klik Saya
    </button>
  );
}
```

Dalam contoh di atas, pesan "Tombol diklik!" diteruskan ke handler ‘`handleClick`’.

5. Sorry bang udah cape prak, ngga dengerin lagu Alan Walker tapi ‘Tak Segampang Itu’.

Fadel Rizky Nurfitanto
21120120140098