React & React Native Basics
sesi 6

Statefull dan Stateless Component

Stateful & Stateless Component - Sesi 6

Introduction

Di bagian sebelumnya kita sudah mempelajari tentang bagaimana membuat komponen dengan sintaks class yang diperkenalkan oleh ES6. Berikut beberapa cara yang bisa kita gunakan untuk membuat sebuah komponen React:

Komponen functional yang bersifat stateless. Sesuai namanya, kita bisa mendefinisikan komponen dengan fungsi yang me-return JSX. Isinya mirip dengan method render yang kita buat saat mendefinisikan komponen react dengan class. Bedanya functional komponen ini sifatnya stateless. Artinya komponen yang dibuat dengan fungsi ini tidak menyimpan state. Dia hanya dapat menerima props. Komponen functional ini dapat digunakan untuk menampilkan data tanpa butuh banyak logika.

Komponen class. Kita sudah menggunakan jenis komponen ini di chapter sebelumnya. Kita mendefinisikan komponen dengan sintaks class yang meng-extend React.Component. Dengan sintaks class ini kita dapat membuat komponen yang kompleks dengan kemampuan akses ke state dan seluruh siklus hidup komponen dapat digunakan di tipe komponen ini.

React.createClass. Mendeklarasikan komponen dengan createClass sudah jarang atau . tidak pernah digunakan lagi di React yang terbaru. Sebelum sintaks class ES6 tersedia, kita harus menggunakan method createClass ini untuk membuat komponen. Kita tidak akan menggunakan method createClass di course ini, hanya sekedar pengetahuan umum saja jikalau teman-teman melihat method ini digunakan, mungkin ketika melihat kode React versi terdahulu.

Introduction

Statefull Component

Ketika mendengar istilah "stateful", artinya sebuah komponen atau app akan menyimpan informasi terkait komponen tersebut. Dan komponen itu juga dapat mengubah informasi tersebut atau dengan kata lain, state

```
import React from 'react';

export default class Users extends React.Component {
  constructor() {
    super()
    this.state = {
      username: 'user01'
    }
  }

  render() {
    return (
      <h1>{this.state.username}</h1>
    )
  }
}

```

Stateless Component

Dan ketika mendengar istilah "stateless", artinya sebuah komponen tidak memiliki dan tidak bertanggungjawab terhadap state. Tugasnya hanya menampilkan data yang dikirim oleh parent component melalui props. Pada kasus tertentu, stateless component ini lebih dianjurkan daripada statefull component

```
import React from 'react';

export default const Movie = (props) => {
  return (
    <div>
      <h1>{props.movie.title}</h1>
      <h2>{props.movie.year} - {props.movie.rating}</h2>
      <p>{props.movie.synopsis}</p>
    </div>
  )
}

```

Introduction

Masih ingat dengan aplikasi React pertama kita ? Sebelum kita lanjut, mari kita ubah aplikasi tersebut menjadi menggunakan stateless component, atau yang lebih lanjut akan kita sebut dengan functional component. Temen-temen boleh merubah kode aplikasi yang terdahulu, atau membuat sebuah React app baru.

Langkah 1 :
Bukalah file src/App.js, lalu ubah isinya menjadi seperti di samping ini

```
1   import './App.css';
2
3   function Header() {
4   }
5
6   function Content() {
7   }
8
9   function Footer() {
10  }
11
12  function App() {
13    return (
14      <div className="container">
15      </div>
16    );
17  }
18
19  export default App;
20
```

Introduction

Langkah 2 :
Isikan functional component Header() dengan kode seperti di bawah ini

```
3   function Header() {
4     return (
5       <header className="header">
6         <h1>My First React app</h1>
7       </header>
8     )
9   }
10
```

Langkah 3 :
Isikan functional component Content() dengan kode seperti di bawah ini

```
11  function Content() {
12    return (
13      <div className="content">
14        <p>
15          Silakan isikan dengan
16          paragraf yang kamu kehendaki
17        </p>
18      </div>
19    )
20  }
21
```

Introduction

Langkah 4 :
Isikan functional component Footer() dengan kode seperti di bawah ini

```
19  function Footer() {
20    return (
21      <div className="footer">
22        <p>&copy; My self - 2021</p>
23      </div>
24    )
25  }
26
```

Langkah 5 :
Mari kita panggil semua functional component kita di App component kita, dengan kode seperti di bawah ini :

```
27  function App() {
28    return (
29      <div className="container">
30        <Header />
31        <hr />
32        <Content />
33        <hr />
34        <Footer />
35      </div>
36    );
37  }
38
```

Introduction

Langkah 6 :
Jalankan aplikasi dengan menjalankan perintah berikut pada folder root aplikasi kita :

> npm start

Dan silakan perhatikan hasilnya pada browser kita. Jika sudah muncul seperti di samping, maka kita telah berhasil membuat aplikasi React pertama kita. Tentu saja, bentuk ini masih sangat dasar, masih diperlukan banyak pengembangan setelah ini

Introduction

[Class component vs Function component]

Temen-temen mungkin bingung kenapa dicontohkan pembuatan aplikasi dengan 2 jenis component, class dan functional. Apa sih bedanya ? Dan yang mana sih yang merupakan best practice.

Pada awal kemunculannya, React memiliki Class component (yang merupakan Statefull component), dan functional component (yang merupakan Stateless component). Namun, sejak React versi 16.8, Hooks mulai diperkenalkan. Hooks ini membuat functional component dapat memiliki kemampuan yang sama dengan Class component, yaitu bisa memiliki state dan lifecycle. Hooks yang dimaksud di sini adalah useState dan useEffect

Maintainer dan contributor dari React nampaknya lebih cenderung untuk merekomendasikan penggunaan Functional component, terutama untuk menjawab beberapa hal yang terkait dengan *best practices,* diantaranya untuk mempertahankan simplicity dan clean code.

Mengikuti motivasi yang sama, untuk selanjutnya, kita akan lebih cenderung menggunakan Functional component

Untuk lebih lengkapnya, bisa dibaca di https://reactjs.org/docs/hooks-intro.html dan sebagai tambahan : https://www.twilio.com/blog/react-choose-functional-components

Introduction to Hooks Concept

Introduction

[Hooks]

Kita akan berkenalan dengan Hooks. Seperti arti kata nya, Hooks dapat diartikan sebagai hal yang “mengaitkan”. Apa yang dikaitkan ya ? Nah, yang “dikaitkan” oleh hooks adalah beberapa fitur yang terdapat di Class component, untuk bisa digunakan di Functional component.

Dikarenakan Functional component akan terus dikembangkan karena alasan best practice, maka perlu ada fitur unggulan Class component yang bisa di implementasikan juga di Functional component

Pada kesempatan kali ini, 2 hal yang paling sering dipakai di Class component, yang juga akan sering kita gunakan di Functional component adalah state dan lifecycle. 2 hal ini akan “dikaitkan” ke dalam Functional component oleh useState dan useEffect.

Mari kita pelajari lebih dalam...

Introduction

[Hooks - useState]

Dalam versi React sebelumnya, komponen fungsional sering disebut juga stateless component karena ga punya state.
Tapi istilah stateless sudah tidak lagi akurat karena ada useState.

Jadi sekarang semua komponen baik yang class-based ataupun yang functional bisa punya state.

```
const [state, setState] = useState(initialState);
```

useState adalah sebuah fungsi yang mengembalikan sebuah nilai stateful, dan sebuah fungsi untuk memperbaruinya.

Selama render awal, state yang dikembalikan (state) sama dengan nilai yang telah dioper pada argumen pertama (initialState).

```
setState(newState);
```

Fungsi setState digunakan untuk memperbarui state. Fungsi tersebut menerima sebuah nilai state yang baru dan meminta sebuah render ulang pada komponen tersebut.

Selama *render* ulang berikutnya, nilai pertama yang dikembalikan oleh useState akan selalu menjadi *state* yang paling terbaru setelah pembaruan diterapkan.

Introduction

[Hooks - useState]

Jika state baru dikomputasi menggunakan state sebelumnya, Anda dapat mengoper sebuah fungsi ke setState. Fungsi tersebut akan menerima nilai sebelumnya, dan mengembalikan sebuah nilai yang telah diperbarui. Berikut adalah contoh komponen penghitung yang menggunakan kedua bentuk setState:

```
function Counter({initialCount}) {
  const [count, setCount] = useState(initialCount);
  return (
    <>
      Count: {count}
      <button onClick={() => setCount(initialCount)}>Reset</button>
      <button onClick={() => setCount(prevCount => prevCount - 1)}>-</button>
      <button onClick={() => setCount(prevCount => prevCount + 1)}>+</button>
    </>
  );
}
```

Tombol ”+” dan ”-” menggunakan bentuk fungsional, karena nilai yang telah diperbarui didasari oleh nilai sebelumnya. Tetapi tombol “Reset” menggunakan bentuk normal, karena tombol tersebut selalu mengatur perhitungan kembali ke nilai awal.

Jika fungsi pembaruan Anda mengembalikan nilai yang sama dengan state saat ini, pe-render-an ulang selanjutnya akan dilewati.

Introduction

[Hooks - useEffect]

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

function Example() {
  // Deklarasi variabel state baru yang kita sebut "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Anda menekan sebanyak {count} kali</p>
      <button onClick={() => setCount(count + 1)}>
        Klik saya
      </button>
    </div>
  );
}
```

Untuk membahas useEffect, kita akan memulai dengan potongan kode yang menggunakan useState seperti di samping.

Baris paling pertama melakukan import untuk fungsi useState yang akan kita pakai. Dan baris-baris selanjutnya dapat kita pahami seperti slide sebelumnya.

Sekarang, kita membutuhkan useEffect untuk bisa “memantau” perubahan yang terjadi dan melakukan aksi sesuai dengan kebutuhan kita

Atau dengan kata lain, useEffect membuat kita dapat melakukan “efek samping” di dalam functional component.

Introduction

[Hooks - useEffect]

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

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

  // Mirip dengan componentDidMount dan componentDidUpdate:
  useEffect(() => {
    // Memperbarui judul dokumen menggunakan API browser
    document.title = `You clicked ${count} times`;
  });

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

Potongan code disamping berdasarkan pada contoh counter dari slide sebelumnya, tetapi kita menambahkan fitur baru didalamnya: kita akan mengisi judul dokumen dengan pesan kustom termasuk dari jumlah klik.

Pengambilan data, pengaturan berlangganan (subscription) , dan perubahan manual DOM di dalam komponen React adalah beberapa contoh dari efek samping.

LATIHAN

Nah, sebagai ajang latihan, gimana kalau temen-temen coba implementasi kode dari 2 slide terakhir ke dalam skeleton aplikasi React yang sudah temen-temen bikin sebelumnya. Happy coding ^_^