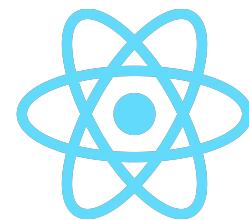




Pemrograman Sisi Cleint

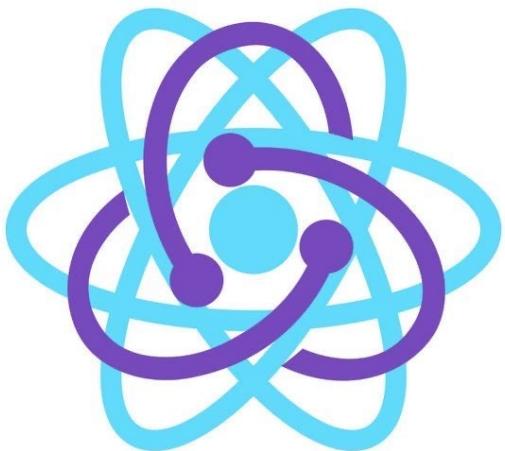
ReactJS Redux

Muhammad Syaifur Rohman, S.Kom, M.CS





Definisi



FYI

Redux adalah pustaka manajemen state yang populer untuk aplikasi JavaScript, khususnya React. Redux membantu mengelola state aplikasi secara global, memungkinkan komponen untuk berinteraksi dan berbagi state tanpa harus meneruskan props secara manual melalui setiap level komponen.

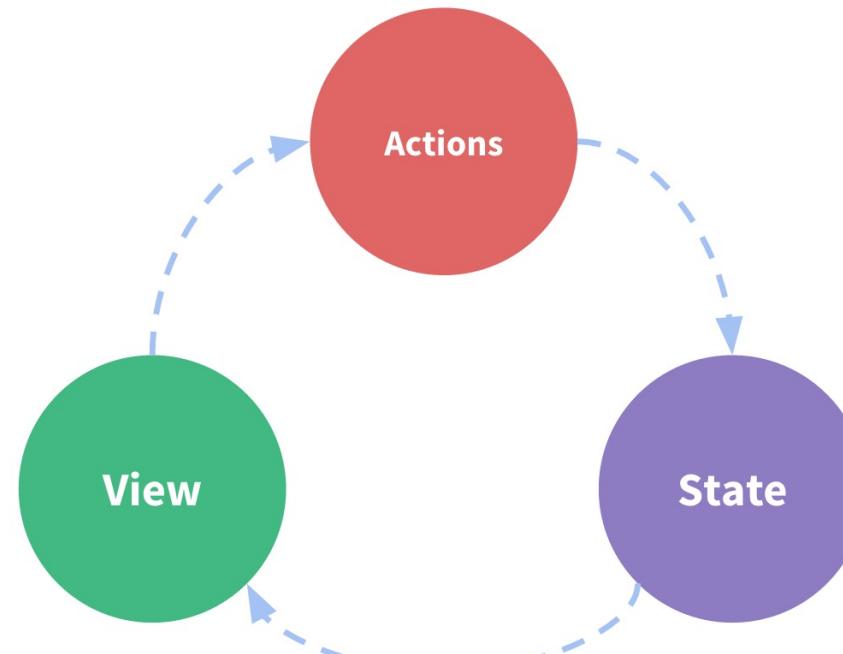


Konsep

Penggunaan

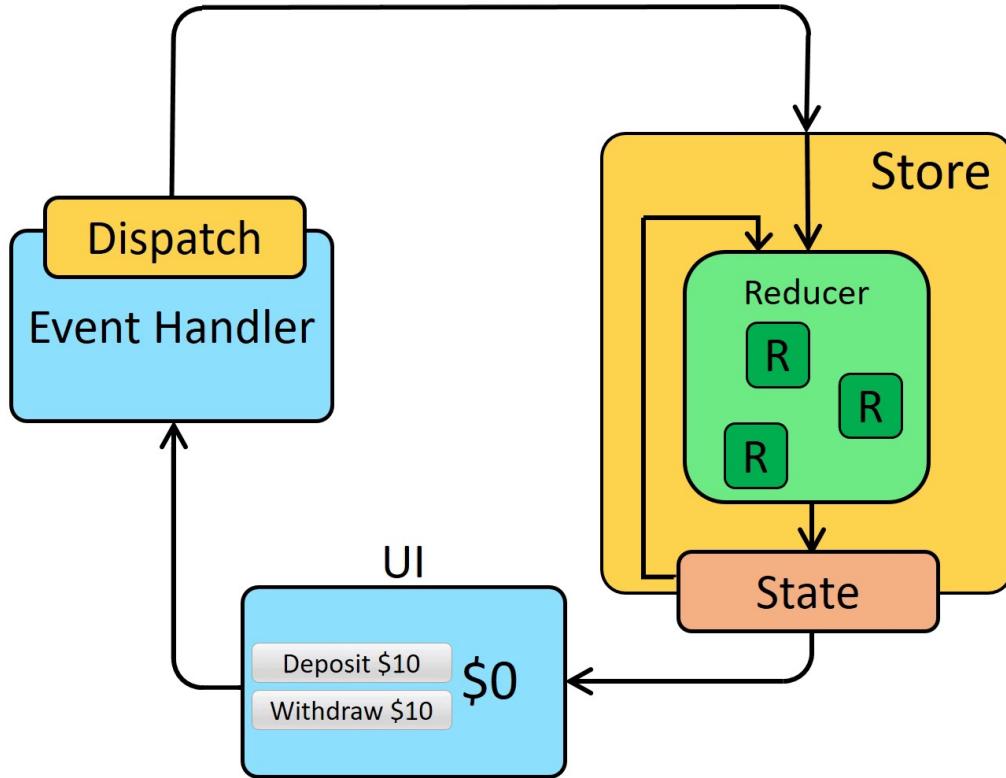
Tanpa Redux, manajemen state pada aplikasi React sering kali mengandalkan lifting state up dan prop drilling, di mana state diangkat ke komponen induk dan kemudian diteruskan ke komponen anak sebagai props. Ini bisa menjadi rumit dan sulit diatur pada aplikasi besar dengan banyak komponen yang membutuhkan akses ke state yang sama.

Konsep Reactjs State



Konsep sebelumnya adalah dengan **State** yang menerima data lalu pada **View** dilakukan deskripsi deklaratif UI berdasarkan keadaan pada saat interaksi lalu **Actions** adalah interaksi yang terjadi di aplikasi berdasarkan masukan pengguna, dan memicu pembaruan di status. Ini disebut juga dengan aliran data satu arah

Konsep Redux



Saat tombol "Deposit \$10" ditekan, event bernama deposit akan terpicu dan ditangani oleh event handler.

Event handler tersebut akan menjalankan fungsi dispatch dan mengirimkan sebuah action yang memiliki tipe: deposit dan payload: 10, di mana payload merupakan nilai yang terkandung dalam action tersebut.

Selanjutnya, store akan memproses fungsi reducer yang relevan dengan action yang diberikan.

Reducer ini bertugas untuk mengupdate state dari \$0 menjadi \$10, yang akan kemudian ditunjukkan pada antarmuka pengguna.

Konsep Redux

Pada "aliran data satu arah" yang menggambarkan urutan langkah-langkah untuk memperbarui aplikasi sebagai berikut:

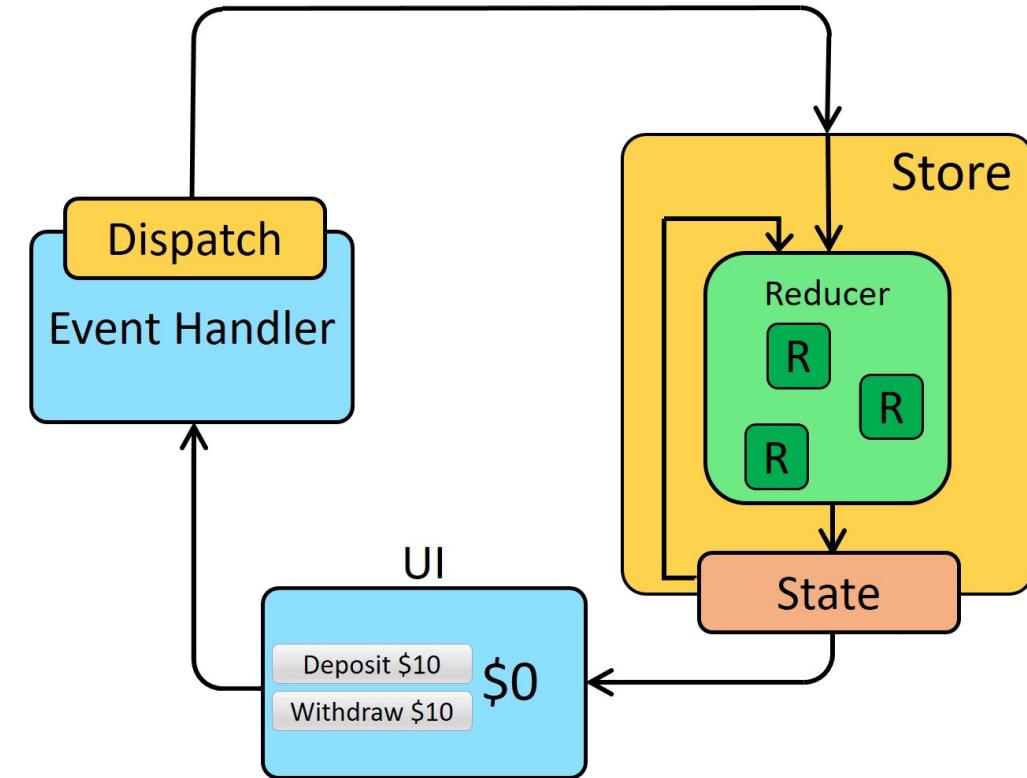
1. State menggambarkan kondisi aplikasi pada titik waktu tertentu.
2. UI dirender berdasarkan state tersebut.
3. Ketika sesuatu terjadi (seperti pengguna mengklik tombol), state diperbarui berdasarkan apa yang terjadi.
4. UI dirender ulang berdasarkan state yang baru.

1. Sebuah store Redux dibuat menggunakan fungsi root reducer.
2. Store memanggil root reducer sekali dan menyimpan nilai kembalian sebagai state awalnya.
3. Ketika UI pertama kali dirender, komponen UI mengakses state saat ini dari store Redux dan menggunakan data tersebut untuk memutuskan apa yang akan dirender. Mereka juga berlangganan ke pembaruan store di masa depan sehingga mereka dapat mengetahui jika state telah berubah.

Konsep Redux

Redux

1. Sesuatu terjadi di aplikasi, seperti pengguna mengklik tombol.
2. Kode aplikasi mendispatch sebuah aksi ke store Redux, seperti `dispatch({type: 'counter/incremented' })`.
3. Store menjalankan fungsi reducer lagi dengan state sebelumnya dan aksi saat ini, dan menyimpan nilai kembalian sebagai state baru.
4. Store memberi tahu semua bagian UI yang berlangganan bahwa store telah diperbarui.
5. Setiap komponen UI yang membutuhkan data dari store memeriksa apakah bagian state yang mereka butuhkan telah berubah.
6. Setiap komponen yang melihat datanya telah berubah memaksa render ulang dengan data baru, sehingga bisa memperbarui apa yang ditampilkan di layar.



Tanpa Redux

State diangkat ke komponen induk: State counter dikelola di App dan diteruskan sebagai props ke Counter.

Prop drilling: Fungsi increment dan decrement juga diteruskan ke Counter melalui props.

```
// App.js
import React, { useState } from 'react';
import Counter from './Counter';

function App() {
  const [counter, setCounter] = useState(0);

  const increment = () => setCounter(counter + 1);
  const decrement = () => setCounter(counter - 1);

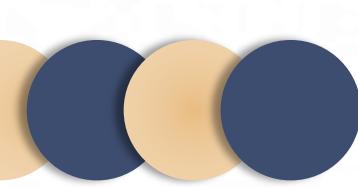
  return (
    <div>
      <h1>Counter App</h1>
      <Counter counter={counter} increment={increment} decrement={decrement} />
    </div>
  );
}

export default App;
```

```
// Counter.js
import React from 'react';

function Counter({ counter, increment, decrement }) {
  return (
    <div>
      <p>Counter: {counter}</p>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
}

export default Counter;
```



Instalasi

```
npm install redux react-redux
```

Redux Step 1

Dengan Redux, state diangkat ke store global yang dikelola oleh Redux, dan komponen dapat mengakses state atau dispatch actions ke store tanpa harus meneruskan props melalui banyak level komponen.

initialState: Mendefinisikan state awal.

Reducer: Menentukan bagaimana state berubah berdasarkan action yang diterima. Di sini, reducer menangani INCREMENT dan DECREMENT.

Setup Reducer

```
// reducers/counterReducer.js
const initialState = {
  counter: 0,
};

function counterReducer(state = initialState, action) {
  switch (action.type) {
    case 'INCREMENT':
      return { counter: state.counter + 1 };
    case 'DECREMENT':
      return { counter: state.counter - 1 };
    default:
      return state;
  }
}

export default counterReducer;
```

Redux Step 2

createStore: Membuat store Redux menggunakan reducer counterReducer.

Membuat Store

```
// store.js
import { createStore } from 'redux';
import counterReducer from './reducers/counterReducer';

const store = createStore(counterReducer);

export default store;
```

Menyambungkan Redux dengan React

```
// App.js
import React from 'react';
import { Provider } from 'react-redux';
import store from './store';
import Counter from './Counter';

function App() {
  return (
    <Provider store={store}>
      <Counter />
    </Provider>
  );
}

export default App;
```

Provider: Membungkus aplikasi dengan Provider dari react-redux dan menyambungkan store Redux ke aplikasi.

Redux Step 3

Menghubungkan Komponen dengan Store

```
// Counter.js
import React from 'react';
import { connect } from 'react-redux';

function Counter({ counter, increment, decrement }) {
  return (
    <div>
      <p>Counter: {counter}</p>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
}

const mapStateToProps = (state) => ({
  counter: state.counter,
});

const mapDispatchToProps = (dispatch) => ({
  increment: () => dispatch({ type: 'INCREMENT' }),
  decrement: () => dispatch({ type: 'DECREMENT' }),
});

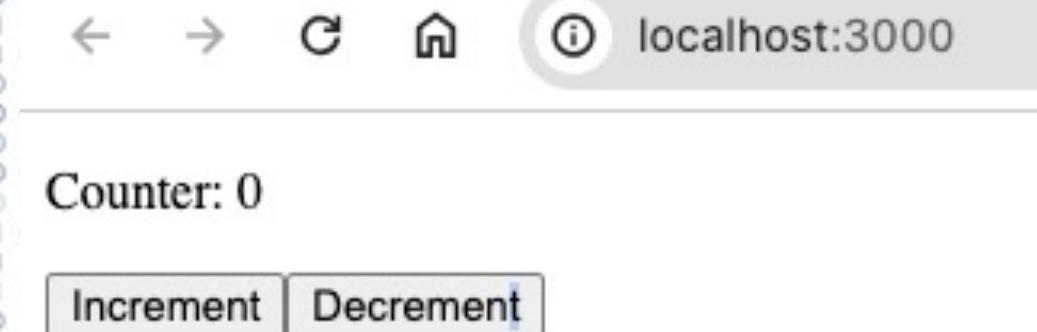
export default connect(mapStateToProps, mapDispatchToProps)(Counter);
```

connect: Fungsi dari react-redux untuk menghubungkan komponen dengan store Redux.

mapStateToProps: Fungsi untuk mengambil state dari store dan meneruskannya sebagai props ke komponen.

mapDispatchToProps: Fungsi untuk membuat fungsi dispatch action yang diteruskan sebagai props ke komponen.

Hasil





Pentingnya Redux

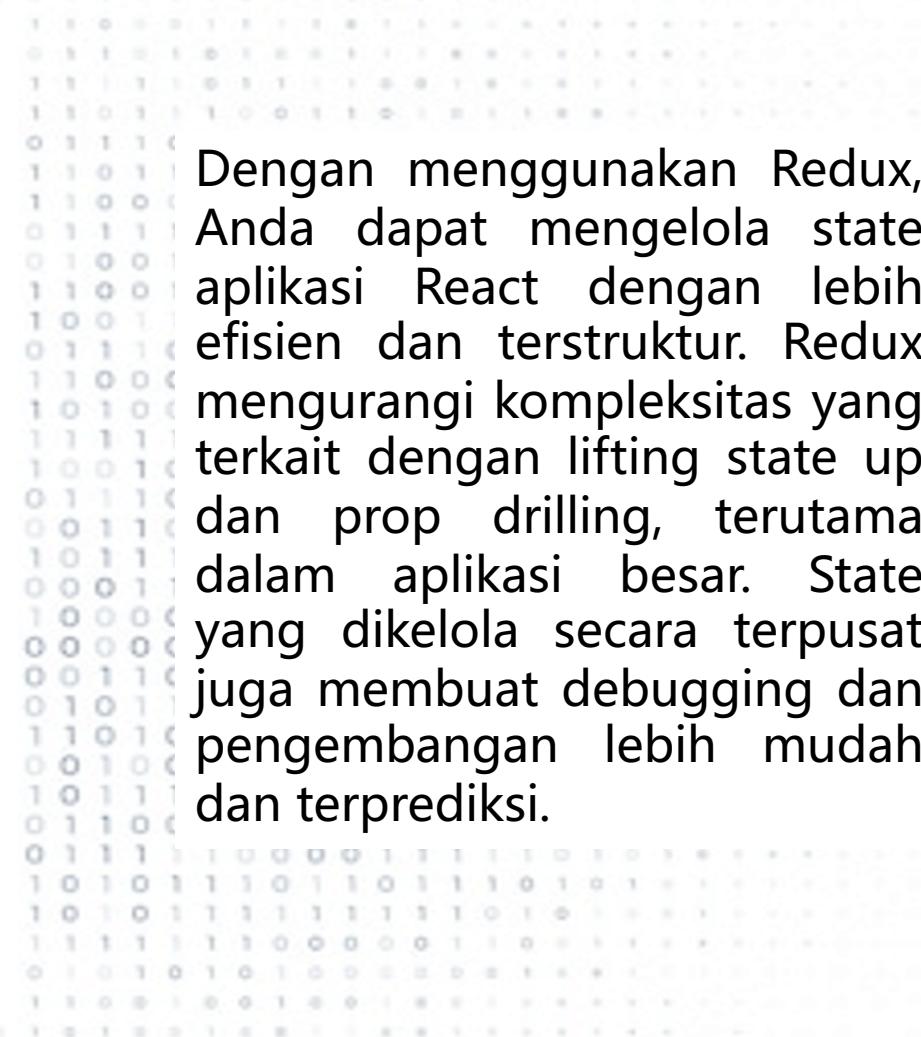
Manajemen State Terpusat: Redux menyediakan store tunggal untuk seluruh aplikasi, membuat manajemen state lebih terpusat dan terorganisir.

Prop Drilling: Mengurangi kebutuhan untuk meneruskan props melalui banyak level komponen.

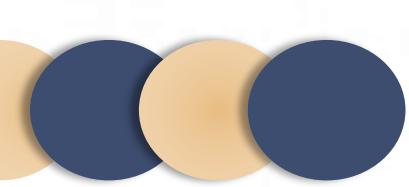
Debugging dan Logging: Redux DevTools memudahkan debugging dan memantau perubahan state.

Prediktabilitas: State di Redux hanya dapat diubah dengan action yang dikirimkan, membuat perubahan state lebih prediktif dan mudah dipahami.

Middleware: Mendukung middleware seperti redux-thunk untuk menangani side effects dan permintaan asinkron.



Dengan menggunakan Redux, Anda dapat mengelola state aplikasi React dengan lebih efisien dan terstruktur. Redux mengurangi kompleksitas yang terkait dengan lifting state up dan prop drilling, terutama dalam aplikasi besar. State yang dikelola secara terpusat juga membuat debugging dan pengembangan lebih mudah dan terprediksi.



Referensi

1. Eran Krinsbruner, **A Frontend Web Developer's Guide to Testing**, 2022
2. Adam Boduch , Roy Derks , Mikhail Sakhniuk, **React and React Native - Fourth Edition**, 2022
3. Maya Shavin , Raymond Camden, **Frontend Development Projects with Vue.js 3 - Second Edition**, 2023
4. Waweru Mwaura, **End-to-End Web Testing with Cypress**, 2021
5. <https://roadmap.sh/frontend>
6. <https://jsonplaceholder.typicode.com/users>
7. <https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow>



Thanks

Muhammad Syaifur Rohman, S.Kom, M.CS

