

Mulai menggunakan redux pada ReactJS

🌐 April 18, 2021 • Code



Redux

Apa sih redux?

Redux adalah salah satu state management yang sangat hype pada waktunya dan masih relevan sampai sekarang. Redux juga menawarkan tools untuk masing-masing browser contoh [chrome redux devtools](#) untuk memonitor keadaan state kita saat ini. Package middleware-nya juga sudah banyak di kembangkan gratis dan siap digunakan untuk memudahkan kita mengembangkan aplikasi yang kita sedang kerjakan.

Kenapa redux?

Kenapa enggak? Haha sebenarnya semua state management itu cuman tools, kita lah sebagai operator yang baik harus bisa memaksimalkan potensinya, tapi jujur aja redux didukung package middleware yang bertebaran di npm yang mana memudahkan kita untuk melakukan development seperti redux-thunk, redux-saga, redux-persist dan sebagainya. Kalau kita memilih context API dibanding redux bisa-bisa saja tapi untuk logic seperti thunk nanti kita butuh melakukan extra karena harus buat dari scratch.

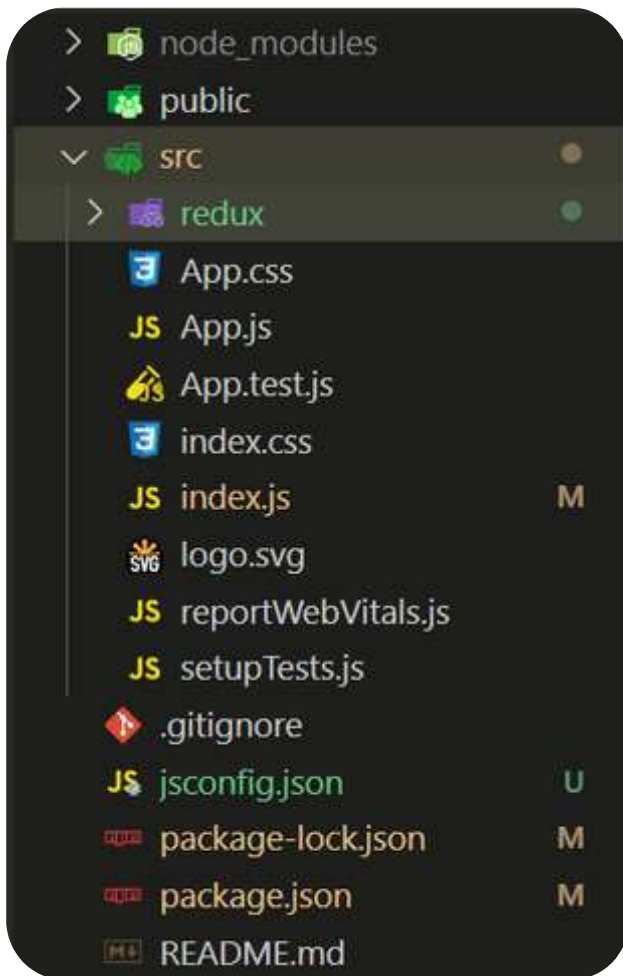
1. Setup Redux

Ada banyak contoh penerapan redux atau state management pada aplikasi nantinya, harapan saya kalian tidak terpaku pada contoh ini tapi berikut adalah penerapan best practice menurut saya, pada aplikasi ReactJS anda, mulai dengan menginstall:

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

Yang pasti kita akan menginstall redux dan react-redux sebagai jembatan untuk mengkomunikasikan react dengan redux state. Dua package ini saja sudah cukup untuk menjalankan redux di aplikasi ReactJS kita.

Siapkan folder redux di dalam src/



Folder redux pada /src/

2. File Store

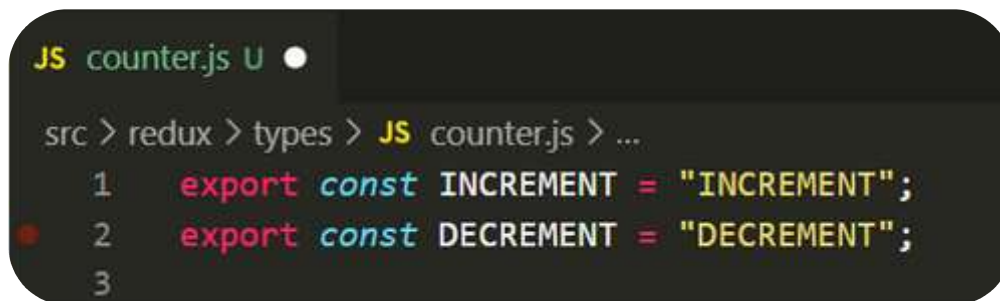
Buat file store.js pada folder redux [/src/redux/store.js] dan sematkan kode berikut:



Pada contoh kali ini kita akan membahas redux sederhana yang hanya menyimpan state **counter**

3. Types

Ada yang menyebut sebagai constants atau types dimana kita mendeklarasi setiap kegiatan yang akan terjadi pada reducer **counter** agar tidak ada salah sebut nama variable ketika digunakan. Maka sebaiknya kita buat 1 folder types pada redux [/src/redux/types] dan sebuah file baru dengan nama **counter.js** dengan kode berikut:



```
JS counter.js U ●  
src > redux > types > JS counter.js > ...  
1 export const INCREMENT = "INCREMENT";  
2 export const DECREMENT = "DECREMENT";  
3
```

Tidak ada yang terlalu istimewa dari kode diatas, kita hanya mengexport constanta dengan nama kegiatan tertentu.

4. Actions

Pada folder ini kita harus setuju bahwa semua action atau kegiatan atau juga bisa disebut kejadian akan kita letakkan pada folder ini.

Buat folder baru **actions** di dalam folder redux [/src/redux/actions/] dan buat sebuah file dengan nama **counter.js**

```
JS counter.js U ●
src > redux > actions > JS counter.js > ...
1  import { INCREMENT, DECREMENT } from "../types/counter";
2
3  export const increment = () => ({
4    type: INCREMENT,
5  });
6
7  export const decrement = () => ({
8    type: DECREMENT,
9  });
10
```

- Pada baris pertama kita import semua kegiatan yang kita miliki pada tipe kegiatan yang akan terjadi pada reducer **counter**
- Pada baris 3-5 adalah sebuah arrow function dengan nama increment yang hanya mengirim sebuah object (dispatch) ke reducer kita dengan isian object seperti tertera pada line 4.

Jika action kita cukup kompleks dan memiliki data yang akan dikirim, kita bisa sematkan sebagai kode snippet berikut:

```
export const someAction = (data) => ({
  type: INCREMENT,
  payload: data
});
```

Abaikan langkah berikut jika kalian ingin melanjutkan reducer **counter**

- Pada baris 7-9 sama dengan baris function sebelumnya hanya beda kegiatan.

5. Reducer

Pada folder ini dimana semua **states** kita tinggal, kita perlu membuat folder reducer [/src/redux/reducers/] dan file pertama yaitu **index.js** yang akan kita gunakan untuk indexing setiap reducer kita dan menggabungkan semua state yang kita miliki.

JS index.js U ●

src > redux > reducers > JS index.js > ...

```
1  import { combineReducers } from "redux";
2
3  import counter from "../counter";
4
5  export default combineReducers({ counter });
6
7
```

- Line pertama kita import **combineReducers** sebuah function dari package redux untuk menggabungkan object-object state yang kita miliki.
- Line 3 kita import counter dari file counter.js
- Line 5 kita export secara default object yang sudah digabung menggunakan combineReducers.

Jika kalian memiliki banyak reducers, inilah tempat yang pas untuk menggabungkannya seperti contoh berikut:

```
3  import counter from "../counter";
4  import people from "../people";
5  import todo from "../todo";
6  import settings from "../settings";
7
8  export default combineReducers({ counter, people, todo, settings });
9
```

Abaikan jika kalian hanya ingin melanjutkan redux **counter**

Nah sekarang kita buat reducer pertama kita dengan nama counter.js pada folder reducers [/src/redux/reducers/counter.js]


```
JS counter.js U ●
src > redux > reducers > JS counter.js > ...
1  import { INCREMENT, DECREMENT } from "../types/counter";
2
3  const initialState = {
4    value: 1,
5  };
6
7  function reducer(state = initialState, action) {
8    switch (action.type) {
9      case INCREMENT:
10       return { value: state.value + 1 };
11      case DECREMENT:
12       return { value: state.value - 1 };
13      default:
14       return state;
15    }
16  }
17
18  export default reducer;
19
```

- Pada line pertama sama seperti pada file actions/counter.js dimana kita import semua kegiatan yang akan terjadi pada reducer **counter**
- Berikutnya pada line 3 kita buat sebuah variable dengan nama initialState yang mana isinya adalah sebuah object untuk menampung keadaan state dari reducer counter pada saat aplikasi pertama kali di load
- Pada line 7 kita buat function reducer kita yang mana function ini menerima arguments state yang akan di isi dengan initialState ketika pertama di load, dan argument kedua yaitu action, argument action ini akan berisi sesuai dengan object yang dikirim dari file [/src/redux/actions/counter.js]

```
export const increment = () => ({
  type: INCREMENT,
});
```

Contoh jika kita nanti panggil function increment ini pada logic aplikasi kita nanti, argument action nanti akan berisi sebuah object seperti berikut:

```
{ type: INCREMENT }
```

- Selanjutnya di line 8-15 kita buat logic pemisah setiap kegiatan dengan code switch dan di cek berdasarkan action.type
- Pada line 9-10 kita deklarasikan setiap case/kejadian/kegiatan kita, kita bisa taruh logic penyimpanan state kita pada baris tersebut seperti melakukan mapping data, memfilter data, data penjumlahan dan sebagainya. Sangat saya merekomendasikan kalian untuk benar-benar paham dengan object pada javascript untuk melakukan pengolahan pada reducer ini. Saya sendiri sangat tersesat ketika belajar redux tanpa memahami basic-basic object sendiri.
- Pada line berikutnya kita bisa sisipkan case yang lain dan di akhiri dengan line 18 dimana kita export reducer kita agar bisa digunakan pada file [/src/redux/reducers/index.js]

6. Provider

Kini reducer **counter** kita siap digunakan, kita bisa implementasi store kita dengan cara berikut:


```
JS index.js M
src > JS index.js
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import "./index.css";
4 import App from "./App";
5 import reportWebVitals from "./reportWebVitals";
6
7 import { Provider } from "react-redux";
8 import store from "redux/store";
9
10 ReactDOM.render(
11   <React.StrictMode>
12     <Provider store={store}>
13       <App />
14     </Provider>
15   </React.StrictMode>,
16   document.getElementById("root")
17 );
18
19 // If you want to start measuring performance in your app, pass a function
20 // to log results (for example: reportWebVitals(console.log))
21 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
22 reportWebVitals();
23
```

Perhatikan hanya kode-kode yang saya blokade diatas

- Pada line 7-8 kita butuh import Provider dari react-redux dan store data kita pada folder [redux/store/index.js]
- Langkah terakhir di file ini kita hanya butuh membungkus aplikasi kita paling luar dengan seperti pada line 12-14 dan jangan lupa sertakan props store pada component Provider diisi dengan store yang kita sudah siapkan diatas.

7. The App

Kita akan membuat aplikasi yang sangat canggih dengan redux counter diatas yaitu aplikasi penjumlahan +1 dan pengurangan -1

```
JS App.js M ●
src > JS App.js > ...
1 import logo from "../logo.svg";
2 import "../App.css";
3 import { useSelector, useDispatch } from "react-redux";
4
5 import { increment, decrement } from "redux/actions/counter";
6
7 function App() {
8   const counter = useSelector((state) => state.counter);
9   const dispatch = useDispatch();
10  return (
11    <div className="App">
12      <header className="App-header">
13        <img src={logo} className="App-logo" alt="logo" />
14        <p>Aplikasi penjumlahan dan pengurangan super canggih!</p>
15        <button onClick={() => dispatch(increment())}>Tambah 1</button>
16        <button onClick={() => dispatch(decrement())}>Kurang 1</button>
17        nilah saat ini: {counter.value}
18      </header>
19    </div>
20  );
21 }
22
23 export default App;
24
```

Pada contoh artikel ini saya hanya pakai template create-react-app jadi tidak ada desain yang terlalu wah untuk layoutnya. Pada file App.js ini:

- Di line 3 kita import useSelector dan useDispatch dari react-redux
- Kita juga import tiap kegiatan yang kita butuhkan di line 5 yaitu increment dan decrement dari file [/src/redux/actions/counter.js]
- Pada line 8 kita buat variable baru dengan nama counter, lalu kita isi variable ini dengan cara panggil function useSelector dan kita sisipkan arrow function. Function callback tersebut menerima 1 argument yaitu state lalu pada bagian returnnya kita panggil state.counter atau state[namaReducer] jika memiliki reducer lainnya
- Line 9 kita kan siapkan function dispatch untuk melakukan action nantinya
- Perhatikan line 15-17, disana saya siapkan 2 button untuk menambah atau mengurangi state saat ini dan 1 text untuk menunjukkan keadaan state saat ini, pada setiap function kita beri event onClick dimana ketika kita click button tersebut akan menjalankan sebuah function dispatch dan kita

sisipkan action yang sudah kita buat sebelumnya pada file
[/src/redux/actions/counter.js]

Jika kalian mengikuti dari awal dengan baik dan benar, paling tidak tampilannya akan seperti gambar di bawah.



Akhir kata Redux sendiri hanya sebagai tools untuk mengelola state kita yang berada di layer berbeda, dan setiap perubahan harus dicatat pada actions. Pelajari lebih lanjut mengenai redux pada kelas **Full-stack Javascript** dengan stack MERN (Mongo, Express, ReactJS, Node).

Silahkan pelajari berbagai kelas [React JS](#) gratis di [BuildWithAngga](#) untuk meningkatkan skills kamu lebih mantap lagi sebagai front-end developer di tahun 2023.

Pelajari Selengkapnya



Building Design System with React JS & Storybook

★★★★★ (175)



React JavaScript

★★★★★ (17,404)

Career Preparation

Bangun Karir Impian. Dibantu Expert, Gratis 🌟

Stop buang waktu browsing sana-sini karena di BuildWithAngga sudah tersedia arahan pasti

Chat ke CS

Karir HandBook

Content Editor



Yein Narayana

Yein Narayana, an expert content writer at BuildWithAngga, continually acquires valuable knowledge in web technology, UI UX design, SEO, and business. This knowledge is shared freely with the extensive community in Indonesia.



BuildWithAngga

Explore More

Discover

Design

Code

Soft Skills

User-Interface

User-Experience

Front-End

Back-End

Graphic Design

Business

IoT (Embedded System)