

REACT-REDUX-MIDDLEWARE

```
client > src > store > middlewares > JS logger.js > [o] logger > ⓘ <function> > ⓘ <
  1 const logger = (store) => (next) => (action) => {
  2   console.log("dispatching", action);
  3   let result = next(action);
  4   console.log("next state", store.getState());
  5   return result;
  6 };
  7
  8 export default logger;
  9

  ...
  JS index.js ●

client > src > store > JS index.js > [o] store
  1 import { legacy_createStore as createStore, applyMiddleware } from "redux";
  2 import rootReducer from "./reducers/rootReducers";
  3 import logger from "./middlewares/logger"
  4
  5 let store = createStore(rootReducers, applyMiddleware(logger));
  6
  7 export default store;
  8
```

Hasil Logger

Currying dalam pemrograman x | Invitation: WID3 - Redux-Thunk x | Middleware | Redux x | Middleware | Redux x | React App x +

localhost:3000/login

Elements Console Sources Network > 1 Issue: 1 Default levels 1 Issue: 1

Download the React DevTools for a better development experience: <https://reactjs.org/link/react-devtools>

```
dispatching > Object logger.js:2
next state > Object logger.js:4
dispatching > Object logger.js:2
next state > Object logger.js:4
dispatching > Object logger.js:2
next state > Object logger.js:4
dispatching > (type: 'counter/incremented', payload: undefined) logger.js:2
next state > (value: 1, users: Array(10), categories: Array(0)) logger.js:4
dispatching > (type: 'counter/incremented', payload: undefined) logger.js:2
next state > (value: 2, users: Array(10), categories: Array(0)) logger.js:4
dispatching > (type: 'counter/incremented', payload: undefined) logger.js:2
next state > (value: 3, users: Array(10), categories: Array(0)) logger.js:4
dispatching > (type: 'counter/incremented', payload: undefined) logger.js:2
next state > (value: 4, users: Array(10), categories: Array(0)) logger.js:4
  > categories: []
  > users: (10) [{}]
    > value: 4
      > [[Prototype]]: Object
dispatching > (type: 'counter/incremented', payload: undefined) logger.js:2
next state > (value: 5, users: Array(10), categories: Array(0)) logger.js:4
  > categories: []
  > users: (10) [{}]
    > value: 5
      > [[Prototype]]: Object
dispatching > (type: 'counter/incremented', payload: undefined) logger.js:2
next state > (value: 6, users: Array(10), categories: Array(0)) logger.js:4
  > 
```

Atom logo

Home Login

Login Page

Increments

Higher order function

- Parameter nya adalah function
- Function yang mengembalikan sebuah function

```
13
14  const logger2 = function (store) {
15    return function (next) {
16      return function (action) {
17        console.log("dispatching", action);
18        let result = next(action);
19        console.log("next state", store.getState());
20        return result;
21      };
22    };
23  };
24
```

Middleware redux itu **Currying** - logger(store)(next)(action)

```
10 console.log(res); // 20
```

Hemm,, dua cara tadi hasilnya sama saja, jadi apa keuntungannya menggunakan teknik **currying** ????

Baik,, kita akan cari tau apa keuntungan menggunakan teknik ini.

Katakanlah kita ingin membuat fungsi untuk menghitung diskon pada harga jual sebuah barang.

```
1 function hitungDiskon(harga, diskon) {  
2     return harga * diskon  
3 }
```

Dengan menggunakan fungsi **hitungDiskon** di atas jika kita ingin menghitung diskon 10% untuk banyak barang maka kita akan menuliskannya seperti ini.

```
1 function hitungDiskon(harga, diskon) {  
2     return harga * diskon  
3 }  
4  
5 const diskon1 = hitungDiskon(1500,0.10); // 150  
6  
7 const diskon2 = hitungDiskon(2000,0.10); // 200  
8  
9 const diskon3 = hitungDiskon(5000,0.10); // 500  
10  
11  
12 console.log(diskon1);  
13 console.log(diskon2);  
14 console.log(diskon3);
```

Sekilas tidak ada yang salah dengan code di atas, dan memang tidak ada yang salah. Hanya saja ada banyak pengulangan parameter untuk diskonnya.

Nah code di atas coba kita sederhanakan menggunakan teknik *currying*.

```
1 function hitungDiskon(diskon) {  
2     return (harga) => {  
3         return diskon * harga;  
4     }  
5 }  
6  
7 const diskon10Persen = hitungDiskon(0.10)
```

Sekilas tidak ada yang salah dengan code di atas, dan memang tidak ada yang salah. Hanya saja ada banyak pengulangan parameter untuk diskonnya.

Nah code di atas coba kita sederhanakan menggunakan teknik *currying*.

```
1 function hitungDiskon(diskon) {  
2     return (harga) => {  
3         return diskon * harga;  
4     }  
5 }  
6  
7 const diskon10Persen = hitungDiskon(0.10)  
8  
9 const diskon1 = diskon10Persen(1500); // 150  
10  
11 const diskon2 = diskon10Persen(2000); // 200  
12  
13 const diskon3 = diskon10Persen(5000); // 500  
14  
15  
16 console.log(diskon1);  
17 console.log(diskon2);  
18 console.log(diskon3);
```

Hasilnya sama, hanya saja sekarang tidak ada pengulangan untuk parameter diskonnya. Hal ini menghindari kesalahan jika kita mengulangi penulisan parameter untuk diskonnya. Dengan currying untuk fungsi **hitungDiskon** di atas, kita juga bisa menggunakan fungsi **hitungDiskon** untuk diskon lain. Misalnya untuk diskon 20%

Kode di atas kita sedemikian menggunakan teknik currying.

```
1 function hitungDiskon(diskon) {  
2     return (harga) => {  
3         return diskon * harga;  
4     }  
5 }  
6  
7 const diskon10Persen = hitungDiskon(0.10)  
8  
9 const diskon1 = diskon10Persen(1500); // 150  
10  
11 const diskon2 = diskon10Persen(2000); // 200  
12  
13 const diskon3 = diskon10Persen(5000); // 500  
14  
15  
16 console.log(diskon1);  
17 console.log(diskon2);  
18 console.log(diskon3);
```

Hasilnya sama, hanya saja sekarang tidak ada pengulangan untuk parameter diskonnya. Hal ini menghindari kesalahan jika kita mengulangi penulisan parameter untuk diskonnya. Dengan currying untuk fungsi **hitungDiskon** di atas, kita juga bisa menggunakan fungsi **hitungDiskon** untuk diskon lain. Misalnya untuk diskon 20%

```
1 function hitungDiskon(diskon) {  
2     return (harga) => {  
3         return diskon * harga;  
4     }  
5 }  
6  
7 const diskon20Persen = hitungDiskon(0.20)
```

Memahami konsep dan teknik di atas sangat penting ketika kalian ingin mempelajari bahasa pemrograman fungsional murni seperti Elixir, Erlang, Clojure, dan Haskell.



55



4



4



3



...

REDUX THUNK

Why ? redux berasa hanya synchronous update, Thunk bisa melakukan interaksi async

package:

```
npm install redux-thunk  
yarn add redux-thunk
```

The thunk middleware is the default export.

► More Details: Importing the thunk middleware

Then, to enable Redux Thunk, use `applyMiddleware()`:

```
import { createStore, applyMiddleware } from 'redux'  
import thunk from 'redux-thunk'  
import rootReducer from './reducers/index'  
  
const store = createStore(rootReducer, applyMiddleware(thunk))
```

Injecting a Custom Argument

Since 2.1.0, Redux Thunk supports injecting a custom argument into the thunk middleware. This is typically useful for cases like using an API service layer that could be swapped out for a mock service in tests.

For Redux Toolkit, the `getDefaultMiddleware` callback inside of `configureStore` lets you pass in a custom `extraArgument`:

```
import { configureStore } from '@reduxjs/toolkit'  
import rootReducer from './reducer'  
import { myCustom ApiService } from './api'  
  
const store = configureStore({  
  reducer: rootReducer,  
  middleware: getDefaultMiddleware =>  
    getDefaultMiddleware({  
      thunk: {  
        extraArgument: myCustom ApiService  
      }  
    })  
})
```

JS Home.js M JS logger.js U JS index.js M X ... JS index.js M X

```

client > src > store > action > JS index.js > [e]fetchUser
7   type: COUNTER_INCREMENT,
8   payload,
9 };
10};

12 export const fetchSuccess = function (payload) {
13   return {
14     type: USERS_FETCH_SUCCESS,
15     payload,
16   };
17};

19 export const fetchUser = function () {
20   return function (dispatch) {
21     fetch("http://localhost:3001/users")
22       .then((res) => {
23         if (!res.ok) throw new Error("error ngab");
24         return res.json();
25       })
26       .then((data) => {
27         dispatch(fetchSuccess(data));
28       })
29       .catch(console.log);
30   };
31};

32

```

JS logger.js U JS index.js M X ... JS index.js M JS Home.js M X

```

client > src > store > action > JS index.js > [e]fetchSuccess
7   type: COUNTER_INCREMENT,
8   payload,
9 };
10};

12 export const fetchSuccess = function (payload) {
13   return {
14     type: USERS_FETCH_SUCCESS,
15     payload,
16   };
17};

19 export const fetchUser = function () {
20   return function (dispatch) {
21     fetch("http://localhost:3001/users")
22       .then((res) => {
23         if (!res.ok) throw new Error("error ngab");
24         return res.json();
25       })
26       .then((data) => {
27         dispatch(fetchSuccess(data));
28       })
29       .catch(console.log);
30   };
31};

32

```

client > src > pages > JS Home.js > [e]useEffect() callback

```

4 import { fetchSuccess, fetchUser } from "../store/action";
5
6 export default function Home() {
7   const { users, value } = useSelector((state) => state); // re
8   const dispatch = useDispatch();
9   const navigate = useNavigate();
10
11   useEffect(() => [
12     dispatch(fetchUser()),
13   ], []);
14
15   function handleClick(id) {
16     console.log(id);
17     navigate(`/${id}`);
18   }
19
20   return (
21     <div>
22       <h1>Home Page</h1>
23       <span>Counter: {value}</span>
24       <ul>
25         {users.map((user) => {
26           return (
27             <li key={user.id}>
28               {user.name} |{" "}
29               <button onClick={() => handleClick(user.id)}>
30                 Detail
31               </button>
32             </li>
33           );
34         });
35       </ul>
36     </div>
37   );
38 }

39

```

TANPA THUNK

The screenshot shows a browser developer tools console with several error messages from the redux-thunk middleware. The errors are related to actions being plain objects instead of functions. The stack traces point to various Redux and React-DOM components, including `fetchUser`, `useEffect`, and `useNavigate`.

```
react-dom.development.js:29840
redux.js.org/link/react-devtools
dispatching f (dispatch) {
  fetch("http://localhost:3001/users").then(res => {
    if (!res.ok) throw new Error("error ngab");
    return res.json();
  }).then(data => {
    dispatch(fetchSuccess(data));
  })
}

Uncaught Error: Actions must be plain objects. Instead, redux.js:275
the actual type was: 'function'. You may need to add middleware to your
store setup to handle dispatching other values, such as 'redux-thunk' to
handle dispatching functions. See https://redux.js.org/tutorials/fundamentals/part-6-async-logic#using-the-redux-thunk-middleware for examples.
at dispatch (redux.js:275:1)
at logger.js:3:1
at Home.js:12:1
at commitHookEffectListMount (react-dom.development.js:23150:1)
at commitPassiveMountOnFiber (react-dom.development.js:24926:1)
at commitPassiveMountEffectsComplete (react-dom.development.js:2489
11:1)
at commitPassiveMountEffectsBegin (react-dom.development.js:24978:1)
at commitPassiveMountEffects (react-dom.development.js:24866:1)
at flushPassiveEffectsImpl (react-dom.development.js:27039:1)
at flushPassiveEffects (react-dom.development.js:26984:1)
at dispatch(flushPassiveEffects)
  dispatching f (dispatch) {
  fetch("http://localhost:3001/users").then(res => {
    if (!res.ok) throw new Error("error ngab");
    return res.json();
  }).then(data => {
    dispatch(fetchSuccess(data));
  })
}

Uncaught Error: Actions must be plain objects. Instead, redux.js:275
the actual type was: 'function'. You may need to add middleware to your
store setup to handle dispatching other values, such as 'redux-thunk' to
handle dispatching functions. See https://redux.js.org/tutorials/fundamentals/part-6-async-logic#using-the-redux-thunk-middleware for examples.
at dispatch (redux.js:275:1)
at logger.js:3:1
at Home.js:12:1
at commitHookEffectListMount (react-dom.development.js:23150:1)
at invokePassiveEffectMountInDEV (react-dom.development.js:25154:1)
at invokeEffectsInDev (react-dom.development.js:27351:1)
at commitPassiveMountEffects (react-dom.development.js:27056:1)
at flushPassiveEffectsImpl (react-dom.development.js:27056:1)
at flushPassiveEffects (react-dom.development.js:26984:1)
at react-dom.development.js:26769:1
  The above error occurred in the <Home> component:
  at Home (http://localhost:3000/static/js/bundle.js:709:63)
```

Misal mau login

The screenshot shows two code editor tabs: `index.js` and `Home.js`. The `index.js` file contains a `fetchUser` function using the `redux-thunk` middleware. The `Home.js` file uses `useEffect` and `useNavigate` hooks.

```
client > src > store > action > JS index.js > [e] fetchUser
15 |     payload,
16 |
17 |};

18
19 export const fetchUser = function (payload) {
20   console.log();
21   return function (dispatch) {
22     fetch("http://localhost:3001/users")
23       .then((res) => {
24         if (!res.ok) throw new Error("error ngab");
25         return res.json();
26       })
27       .then((data) => {
28         dispatch(fetchSuccess(data));
29         dispatch({type: "users/loading", payload: false});
30       })
31       .catch(console.log);
32   };
33 }

34
35 export const fetchCategories = function () {
36   return function (dispatch) {
37     fetch("http://localhost:3001/categories")
38       .then((res) => {
39         if (!res.ok) throw new Error("error ngab");
40         return res.json();
41       })
42       .then((data) => {
43         dispatch(fetchSuccessCategory(data));
44         dispatch({type: "category/loading", payload: false});
45       })
46     );
47   };
48 }

client > src > pages > JS Home.js > Home > useEffect() callback > password
1 port { useSelector, useDispatch } from "react-redux";
2 port { fetchSuccess, fetchUser } from "../store/action";
3
4 port default function Home() {
5   const { users, value } = useSelector((state) => state); // redux
6   const dispatch = useDispatch();
7   const navigate = useNavigate();
8
9   useEffect(() => {
10     dispatch(fetchUser({ email: "yohanes@mail.com", password: "1234" }));
11   }, []);
12
13   function handleClick(id) {
14     console.log(id);
15     navigate(`/${id}`);
16   }
17
18   return (
19     <div>
20       <h1>Home Page</h1>
21       <span>Counter: {value}</span>
22       <ul>
23         {users.map((user) => {
24           return (
25             <li key={user.id}>
26               {user.name} {" "}
27               <button onClick={(e) => handleClick(user.id)}>
28                 Detail
29               </button>
30             </li>
31           );
32         });
33       </ul>
34     </div>
35   );
36 }
```

```

JS logger.js U JS index.js M X JS rootReducer.js ... JS index.js M JS rootReducer.js JS Home.js M X ...
client > src > store > action > JS index.js > [e] fetchUser > <function>
15     payload,
16   };
17 }
18
19 export const fetchUser = function () {
20   return function (dispatch) {
21     return fetch("http://localhost:3001/users")
22       .then((res) => {
23         if (!res.ok) throw new Error("error ngab");
24         return res.json();
25       })
26       .then((data) => {
27         dispatch(fetchSuccess(data));
28         dispatch({ type: "users/loading", payload: true });
29       });
30   };
31 }
32

```

```

client > src > pages > JS Home.js > <function>
< import { useNavigate } from "react-router-dom";
3 import { useSelector, useDispatch } from "react-redux";
4 import { fetchSuccess, fetchUser } from "../store/action";
5
6 export default function Home() {
7   const { users, value } = useSelector((state) => state); // redux
8   const dispatch = useDispatch();
9   const navigate = useNavigate();
10
11   useEffect(() => {
12     dispatch(fetchUser())
13       .then(() => {
14         console.log("berhasil ambil data");
15       })
16       .catch(console.log);
17   }, []);
18
19   function handleClick(id) {
20     console.log(id);
21     navigate(`/${id}`);
22   }
23
24   return (
25     <div>
26       <h1>Home Page</h1>
27       <span>Counter: {value}</span>
28       <ul>
29         {users.map((user) => {
30           return (
31             <li key={user.id}>
32               {user.name}
33             </li>
34           );
35         });
36       </ul>
37     </div>
38   );
39 }
40

```

GABISA PAKE HOOKS KARENA BUKAN REACT COMPONENTS, USE (HELPER)

EXPLORER

- P3-REACT-ROUTER
 - > node_modules
 - > public
 - src
 - > components
 - > pages
 - JS Detail.js**
 - JS Home.js M**
 - JS Login.js**
 - > store
 - > action
 - > middlewares
 - > reducers
 - JS counterReducer.js U**
 - JS rootReducer.js M**
 - JS userReducer.js U**
 - > index.js M
 - # App.css
 - JS App.js
 - JS App.test.js
 - # index.css
 - JS index.js
 - logo.svg
 - JS reportWebVitals.js
 - > OUTLINE
 - > TIMELINE

```

JS rootReducer.js M JS Home.js M X JS userReducer.js U JS counterReducer.js U ...
client > src > pages > JS Home.js > <function>
1 import { useEffect } from "react";
2 import { useNavigate } from "react-router-dom";
3 import { useSelector, useDispatch } from "react-redux";
4 import { fetchSuccess, fetchUser } from "../store/action";
5
6 export default function Home() {
7   const { users } = useSelector((state) => {
8     // console.log(state); // { counter: {value: 0}, users: []}
9     return state.users;
10  });
11
12  const { value } = useSelector((state) => state.counter);
13
14  const dispatch = useDispatch();
15  const navigate = useNavigate();
16
17  useEffect(() => {
18    dispatch(fetchUser())
19      .then(() => {
20        console.log("berhasil ambil data");
21      })
22      .catch(console.log);
23  }, []);
24
25  function handleClick(id) {
26    console.log(id);
27    navigate(`/${id}`);
28  }
29
30  return (
31    <div>
32      <h1>Home Page</h1>
33      <span>Counter: {value}</span>
34      <ul>
35        {users.map((user) => {
36          return (
37            <li key={user.id}>
38              {user.name}
39            </li>
40          );
41        });
42      </ul>
43    </div>
44  );
45 }
46

```

===== REVIEW =====

Install redux

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar:
 - P3-REACT-ROUTER project
 - client folder
 - src folder
 - components
 - pages
 - store folder
 - action
 - middlewares
 - reducers
 - JS index.js (selected)
 - # App.css
 - JS App.js
 - JS App.test.js
 - # index.css
 - JS index.js
 - logo.svg
 - JS reportWebVitals.js
 - JS setupTests.js
 - .gitignore
 - { package-lock.json
 - { package.json
 - README.md

- editor**: JS index.js .../action M JS index.js .../store M X
Content:

```
client > src > store > JS index.js > [e] store
1 import { legacy_createStore as createStore, applyMiddleware } from "redux";
2 import rootReducers from "./reducers/rootReducers";
3 import logger from "./middlewares/logger";
4 import thunk from "redux-thunk";
5
6 let store = createStore(rootReducers, applyMiddleware(logger, thunk));
7
8 export default store;
9
```

Ada fitur middleware di pasang di createStore, applyMiddleware logger dan thunk
karena thunk jadi action bisa kirim function

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar:
 - P3-REACT-ROUTER project
 - client folder
 - src folder
 - components
 - pages
 - store folder
 - action folder
 - JS actionTypes.js
 - JS index.js (M)
 - JS userAction.js (U selected)
 - JS index.js
 - # App.css
 - JS App.js
 - JS App.test.js
 - # index.css
 - JS index.js
 - logo.svg
 - JS reportWebVitals.js
 - JS setupTests.js
 - .gitignore- editor**: JS index.js M JS userAction.js U X
Content:

```
client > src > store > action > JS userAction.js > ...
1 import { USERS_FETCH_SUCCESS } from "./actionType";
2
3 export const fetchSuccess = function (payload) {
4   return {
5     type: USERS_FETCH_SUCCESS,
6     payload,
7   };
8 }
9
10 export const fetchUser = function () {
11   return function (dispatch) {
12     return fetch("http://localhost:3001/users")
13       .then((res) => {
14         if (!res.ok) throw new Error("error ngab");
15         return res.json();
16       })
17       .then((data) => {
18         dispatch(fetchSuccess(data));
19         dispatch({ type: "users/loading", payload: false });
20       });
21   };
22 }
23
```

Di home ada redux thunk, middleware dispatch didalamnya fetchuser untuk fetch async ke server

EXPLORER

JS index.js M JS userAction.js U ... JS Home.js M X

```

client > src > store > action > JS userAction.js > ...
1 import { USERS_FETCH_SUCCESS } from "./actionType"
2
3 export const fetchSuccess = function (payload) {
4   return {
5     type: USERS_FETCH_SUCCESS,
6     payload,
7   };
8 }
9
10 export const fetchUser = function () {
11   return function (dispatch) {
12     return fetch("http://localhost:3001/users")
13       .then((res) => {
14         if (!res.ok) throw new Error("error");
15         return res.json();
16       })
17       .then((data) => {
18         dispatch(fetchSuccess(data));
19         dispatch({ type: "users/loading", });
20       });
21   };
22 }
23

```

JS Home.js M X

```

client > src > pages > JS Home.js > ⚡ Home > ⚡ useEffect() ca
2 import { useNavigate } from "react-router-dom";
3 import { useSelector, useDispatch } from "react-
4 import { fetchUser } from "../store/action/userA
5
6 export default function Home() {
7   const { users } = useSelector((state) => {
8     // console.log(state); // { counter: { va
9     return state.users;
10   }); // redux
11
12   const { value } = useSelector((state) => sta
13
14   const dispatch = useDispatch();
15   const navigate = useNavigate();
16
17   useEffect(() => {
18     dispatch(fetchUser())
19       .then(() => {
20         console.log("berhasil ambil data");
21       })
22       .catch(console.log);
23   }, []);
24
25   function handleClick(id) {
26     console.log(id);
27     navigate(`/${id}`);
28   }
29
30   return (
31     <div>

```

Kalau dispatch dipanggil bakal jalankan reducer

EXPLORER

JS userAction.js U JS userReducer.js U ... JS Home.js M X

```

client > src > store > reducers > JS userReducer.js > ⚡ userReduc
1 const { USERS_FETCH_SUCCESS } = require("../actionType");
2
3 const initialState = {
4   users: [],
5 }; // default value
6
7 function userReducer(state = initialState, action) {
8   switch (action.type) {
9     case USERS_FETCH_SUCCESS:
10       return {
11         ...state,
12         users: action.payload,
13       };
14     default:
15       return state;
16   }
17
18 export default userReducer;
19

```

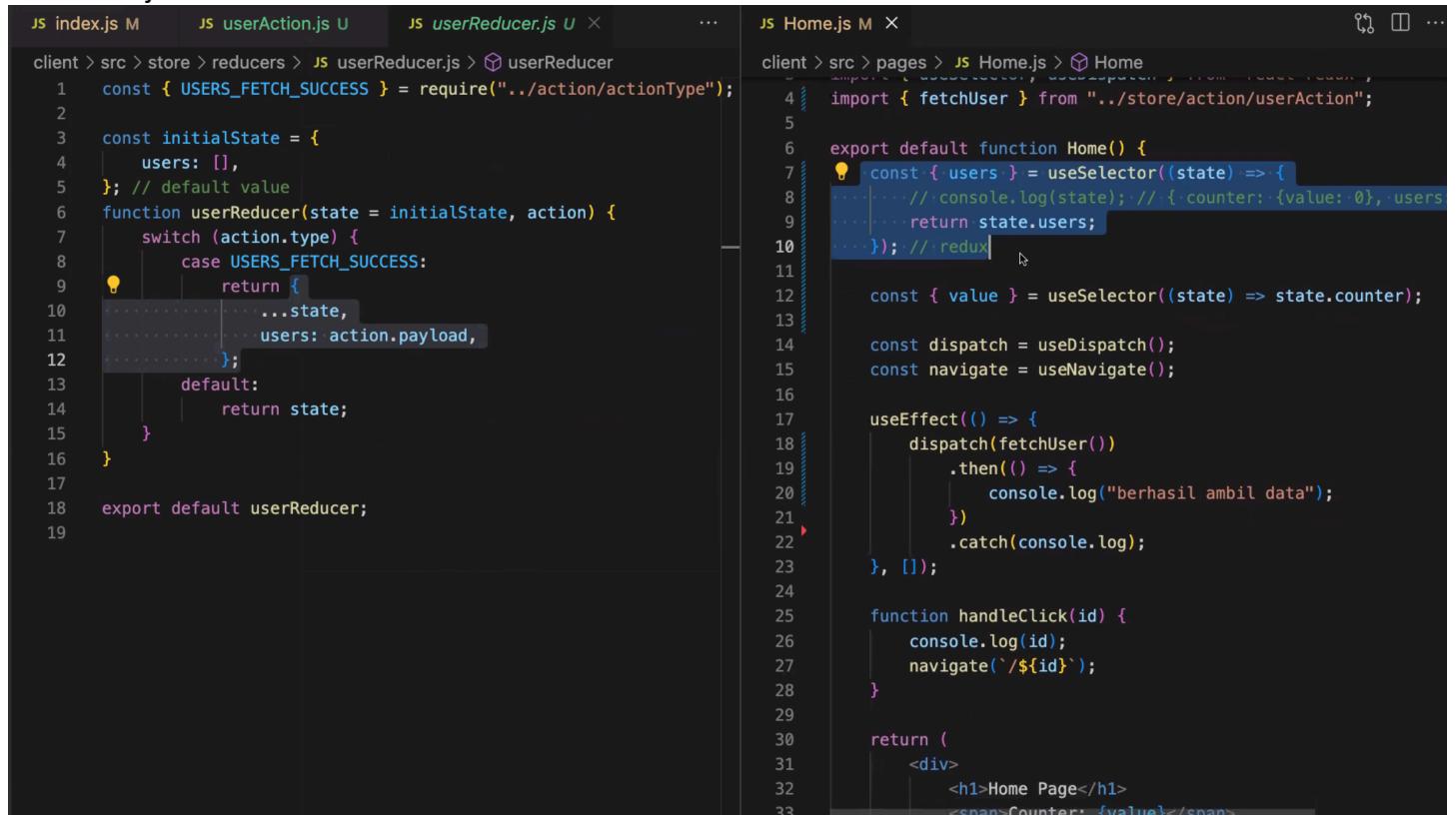
JS Home.js M X

```

client > src > pages > JS Home.js > ⚡ Home > ⚡ useEffect() ca
2 import { useNavigate } from "react-router-dom";
3 import { useSelector, useDispatch } from "react-
4 import { fetchUser } from "../store/action/userA
5
6 export default function Home() {
7   const { users } = useSelector((state) => {
8     // console.log(state); // { counter: { va
9     return state.users;
10   }); // redux
11
12   const { value } = useSelector((state) => sta
13
14   const dispatch = useDispatch();
15   const navigate = useNavigate();
16
17   useEffect(() => {
18     dispatch(fetchUser())
19       .then(() => {
20         console.log("berhasil ambil data");
21       })
22       .catch(console.log);
23   }, []);
24
25   function handleClick(id) {
26     console.log(id);
27     navigate(`/${id}`);
28   }
29
30   return (
31     <div>

```

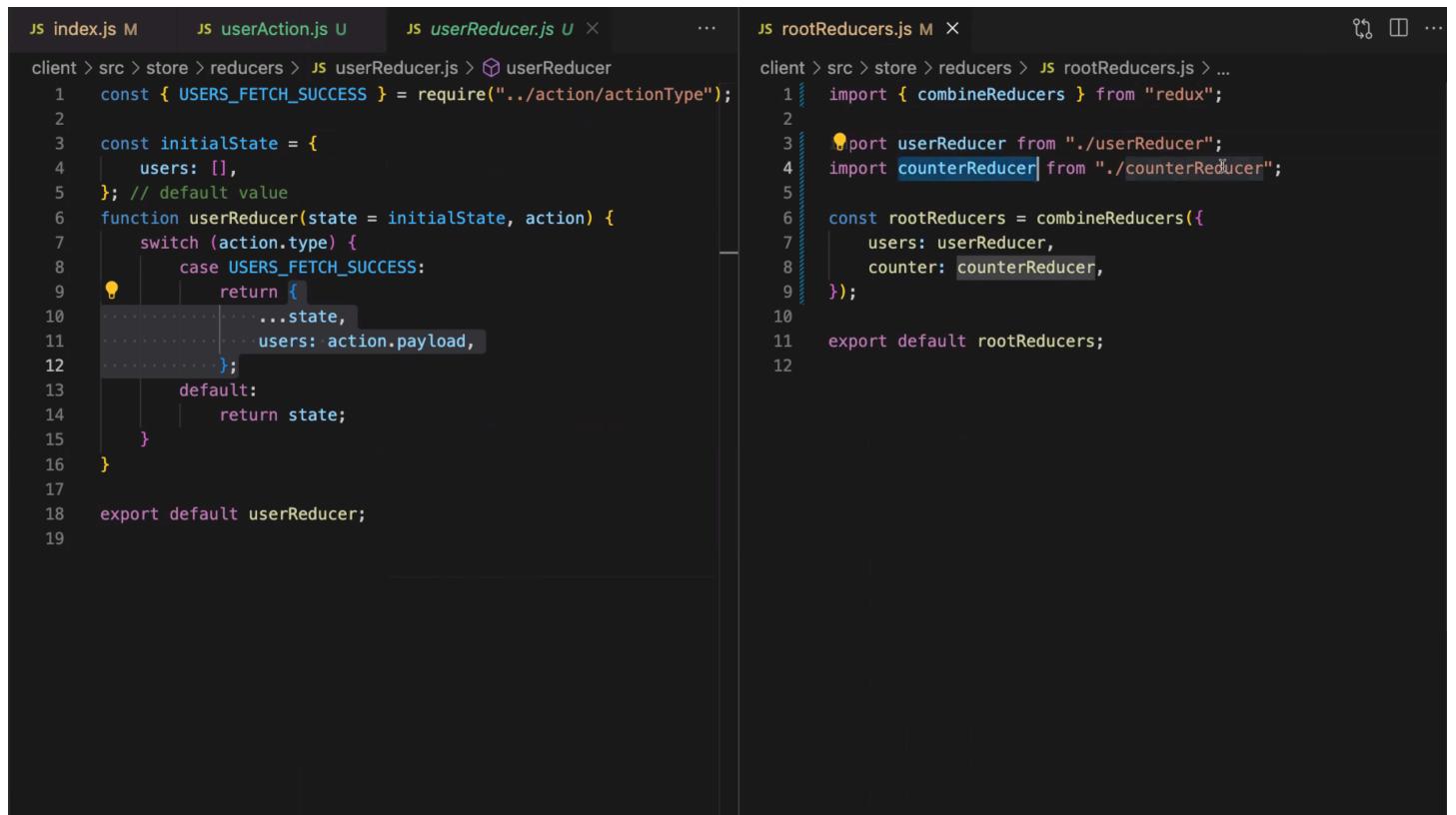
State user jadi reaktif



```
JS index.js M JS userAction.js U JS userReducer.js U X ...
client > src > store > reducers > JS userReducer.js > ↗ userReducer
1 const { USERS_FETCH_SUCCESS } = require("../action/actionType");
2
3 const initialState = {
4   users: [],
5 }; // default value
6 function userReducer(state = initialState, action) {
7   switch (action.type) {
8     case USERS_FETCH_SUCCESS:
9       return {
10       ...state,
11       users: action.payload,
12     };
13     default:
14       return state;
15   }
16 }
17
18 export default userReducer;
19

JS Home.js M X
client > src > pages > JS Home.js > ↗ Home
1 import { useState, useEffect } from "react";
2 import { fetchUser } from "../store/action/userAction";
3
4 export default function Home() {
5   const [users] = useState((state) => {
6     // console.log(state); // { counter: { value: 0 }, users: [] };
7     return state.users;
8   });
9   // redux
10
11   const [value] = useState((state) => state.counter);
12
13   const dispatch = useDispatch();
14   const navigate = useNavigate();
15
16   useEffect(() => {
17     dispatch(fetchUser())
18       .then(() => {
19         console.log("berhasil ambil data");
20       })
21       .catch(console.log);
22   }, []);
23
24   function handleClick(id) {
25     console.log(id);
26     navigate(`/${id}`);
27   }
28
29   return (
30     <div>
31       <h1>Home Page</h1>
32       <span>Counter: <span>{value}</span>
33     </div>
34   );
35 }
```

MATERI KE-2 ITU COMBINED REDUCER JADI MODULAR



```
JS index.js M JS userAction.js U JS userReducer.js U X ...
client > src > store > reducers > JS userReducer.js > ↗ userReducer
1 const { USERS_FETCH_SUCCESS } = require("../action/actionType");
2
3 const initialState = {
4   users: [],
5 }; // default value
6 function userReducer(state = initialState, action) {
7   switch (action.type) {
8     case USERS_FETCH_SUCCESS:
9       return {
10       ...state,
11       users: action.payload,
12     };
13     default:
14       return state;
15   }
16 }
17
18 export default userReducer;
19

JS rootReducer.js M X
client > src > store > reducers > JS rootReducer.js > ...
1 import { combineReducers } from "redux";
2
3 import userReducer from "./userReducer";
4 import counterReducer from "./counterReducer";
5
6 const rootReducer = combineReducers({
7   users: userReducer,
8   counter: counterReducer,
9 });
10
11 export default rootReducer;
12
```