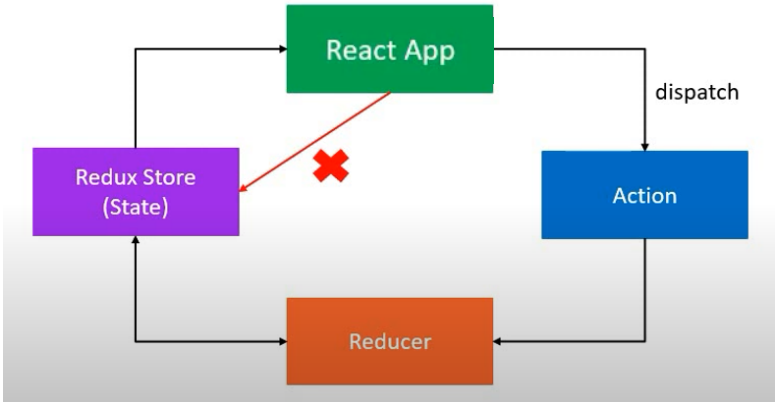
React w Redux



React app

npx create-react-app react-redux-tutorial-demo

npm install redux react-redux

create component folder, and container => rfce => functional component CakeContainer.js

```
import './App.css';
import CakeContainer from './components/CakeContainer';


function App() {
  return (
    <div className="App">
      <CakeContainer />
    </div>
  );
}

export default App;
```

```
import React from 'react'

function CakeContainer() {
  return (
    <div>
        <h2>Number of cakes</h2>
        <button>buy cake</button>

    </div>
  )
}

export default CakeContainer
```

Action

Creating action-creators

New folder redux, new files cakeActions.js and cakeTypes.js

It is a **convention to have the action types separate from action creators** hence we create the cakeTypes.js and export it to cakeActions.js

```
ntainer.js   JS App.js M   JS cakeTypes.js U ×   JS cakeActions.js U
x-tutorial-demo > src > components > redux > cake > JS cakeTypes.js > [e] BUY_CAKE
export const BUY_CAKE = "BUY_CAKE"
```

```
JS CakeContainer.js   JS App.js M   JS cakeTypes.js U   JS cakeActions.js U ×
React-redux-tutorial-demo > src > components > redux > cake > JS cakeActions.js > [e] buyCake
1  import { BUY_CAKE } from "./cakeTypes";
2
3  const buyCake = () => {
4      return {
5          type: BUY_CAKE
6      }
7  }
```

Reducers

Creating reducer JS file, defy initial value, creating reducer, which accepts a state and an action, import BUY_CAKE

JS CakeContainer.js | JS App.js | JS cakeTypes.js | JS cakeReducer.js U × | JS cakeActions.js

React-redux-tutorial-demo > src > components > redux > cake > JS cakeReducer.js > default

```javascript
import { BUY_CAKE } from "./cakeTypes";

const initalState = {
  numOfCakes: 10,
};

const cakeReducer = (state = initalState, action) => {
  switch (action.type) {
    case BUY_CAKE:
      return {
        ...state,
        numOfCakes: state.numOfCakes - 1,
      };
    default:
      return state;
  }
};

export default cakeReducer;
```

Creating store

JS store.js × | JS CakeContainer.js | JS index.js | JS cakeTypes.js | JS cakeR

React-redux-tutorial-demo > src > redux > JS store.js > default

```javascript
import { ~~createStore~~ } from "redux";
import cakeReducer from "./cake/cakeReducer";

const store = ~~createStore~~(cakeReducer)

export default store
```

Connect it all together via react-redux function

```
React-redux-tutorial-demo
  node_modules
  public
    favicon.ico
    index.html
  src
    components
      CakeContainer.js
    redux
      cake
        cakeActions.js
        cakeReducer.js
        cakeTypes.js
      index.js
      store.js
    App.css
    App.js
    App.test.js
    index.css
    index.js
    reportWebVitals.js
  .gitattributes
  .gitignore
  package-lock.json
  package.json
  README.md
```

```
App.js   CakeContainer.js   store.js   index.js   cakeTypes.js   cakeR
React-redux-tutorial-demo > src > App.js > App
import "./App.css";
import { Provider } from "react-redux";
import store from "./redux/store";
import CakeContainer from "./components/CakeContainer";

function App() {
  return (
    <Provider store={store}>
      <div className="App">
        <CakeContainer />
      </div>
    </Provider>
  );
}

export default App;
```

- cakeTypes, which is a convention to have it stored as a string
  export const BUY_CAKE = "BUY_CAKE"
- cakeAction.js, which is an action creator with type property describing what will be performed
- cakeReducer.js, explaining how things will be carried out in response to the action.
- store.js, created the store in separate js file
  const store = createStore(cakeReducer)
  export default store
  and provided to app.js as <Provider>
  import { Provider } from "react-redux";
- Using react-redux {connect} function we map the state and send action creator as props

```
CakeContainer.js   store.js   index.js   cakeTypes.js   cakeReducer.js   cakeActions.js
React-redux-tutorial-demo > src > components > CakeContainer.js > ...
import React from "react";
import { connect } from "react-redux";
import { buyCake } from "../redux";

function CakeContainer(props) {
  return (
    <div>
      <h2>Number of cakes - {props.numOfCakes}</h2>
      <button onClick={props.buyCake}>buy cake</button>
    </div>
  );
}

const MapStateToProps = (state) => {
  return {
    numOfCakes: state.numOfCakes,
  };
};

const mapDispathToProps = (dispatch) => {
  return {
    buyCake: () => dispatch(buyCake()),
  };
};

export default connect(MapStateToProps, mapDispathToProps)(CakeContainer);
```

React-Redux using hooks

We can use hooks instead of connect, mapToState and mapDispathToProps

There can be warnings using hooks in with react-redux

React-Redux useSelector

It is a hook from react-redux makes us able to access the redux state. It’s very similar to the connect method and the mapToState function

```
const MapStateToProps = (state) => {
  return {
    numOfCakes: state.numOfCakes,
  };
};
```

HooksCakeContainer.js U ✕ | store.js | index.js | App.js M | CakeContainer.js | cakeType

act-redux-tutorial-demo > src > components > HooksCakeContainer.js > default

```
1  import React from "react";
2  import { useSelector } from "react-redux";
3
4  function HooksCakeContainer() {
5    const numOfCakes = useSelector((state) => state.numOfCakes);
6    return (
7      <div>
8        <h2>Num of cakes - {numOfCakes}</h2>
9        <button>Buy cake</button>
10     </div>
11   );
12 }
13
14 export default HooksCakeContainer;
```

React-Redux useDispatch

We can use the useDispatch hook from react-redux to achieve what we did in the CakeContainer.js

```
const mapDispathToProps = (dispatch) => {
  return {
    buyCake: () => dispatch(buyCake()),
  };
};
```

HooksCakeContainer.js U ✕ | store.js | index.js | App.js M | CakeContainer.js | cakeTypes.js

eact-redux-tutorial-demo > src > components > HooksCakeContainer.js > ...

```
1  import React from "react";
2  import { useSelector, useDispatch } from "react-redux";
3  import { buyCake } from "../redux";
4
5  function HooksCakeContainer() {
6    const numOfCakes = useSelector((state) => state.numOfCakes);
7    const dispatch = useDispatch()
8    return (
9      <div>
10       <h2>Num of cakes - {numOfCakes}</h2>
11       <button onClick={() => dispatch(buyCake())}>Buy cake</button>
12     </div>
13   );
14 }
15
16 export default HooksCakeContainer;
```

Combining reducers

Creating new action type, action creator and reducer (iceCream)

```
React-redux-tutorial-demo
  node_modules
  public
    favicon.ico
    index.html
  src
    components
      CakeContainer.js
      HooksCakeContainer.js
      IceCreamContainer.js
    redux
      cake
        cakeActions.js
        cakeReducer.js
        cakeTypes.js
      icecream
        iceCreamActions.js
        iceCreamReducer.js
        iceCreamTypes.js
      index.js
      rootReducer.js
      store.js
    App.css
    App.js
    App.test.js
    index.css
    index.js
    reportWebVitals.js
  .gitattributes
  .gitignore
  package-lock.json
  package.json
```

Making store aware of the new reducer

Creating rootReducer.js
importing reducers and combining them with combineReducers (originally store only can accept one reducer)

```
import { combineReducers } from "redux";
import cakeReducer from "./cake/cakeReducer";
import iceCreamReducer from "./icecream/iceCreamReducer";

const rootReducer = combineReducers({
    cake: cakeReducer,
    iceCream: iceCreamReducer
})

export default rootReducer
```

Amend store.js

```
import { ~~createStore~~ } from "redux";
import rootReducer from "./rootReducer";

const store = ~~createStore~~(rootReducer)

export default store
```

Due to the root reducer now we broken down the global state into multiple reducers

In both cakeContainer and iceCreamcontainer we need to amend

From

```
const MapStateToProps = (state) => {
  return {
    numOfIceCreams: state.numOfIceCreams,
  };
};
```

To

```
const MapStateToProps = (state) => {
  return {
    numOfIceCreams: state.iceCream.numOfIceCreams,
  };
};
```

Middleware (logger)

Install npm install redux-logger
Import logger add applyMiddleware function
Add applymiddleware to the create store method
In the browiser console we have the logs

```
store.js M ×   cakeTypes.js   cakeActions.js   cakeReducer.js   CakeContainer.js
React-redux-tutorial-demo > src > redux > store.js > default
1  import { ~~createStore~~, applyMiddleware } from "redux";
2  import logger from "redux-logger";
3  import rootReducer from "./rootReducer";
4
5  const store = ~~createStore~~(rootReducer, applyMiddleware(logger))
6
7  export default store
```

```
▼ action BUY_ICECREAM @ 17:47:43.370
    prev state ▶ {cake: {…}, iceCream: {…}}
    action     ▶ {type: 'BUY_ICECREAM'}
    next state ▶ {cake: {…}, iceCream: {…}}
▼ action BUY_CAKE @ 17:47:46.377
    prev state ▶ {cake: {…}, iceCream: {…}}
    action     ▶ {type: 'BUY_CAKE'}
    next state ▶ {cake: {…}, iceCream: {…}}
▼ action BUY_CAKE @ 17:47:47.321
    prev state ▶ {cake: {…}, iceCream: {…}}
    action     ▶ {type: 'BUY_CAKE'}
    next state ▶ {cake: {…}, iceCream: {…}}
```

Redux Devtool Extension

Add extenstion to chrome

https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd

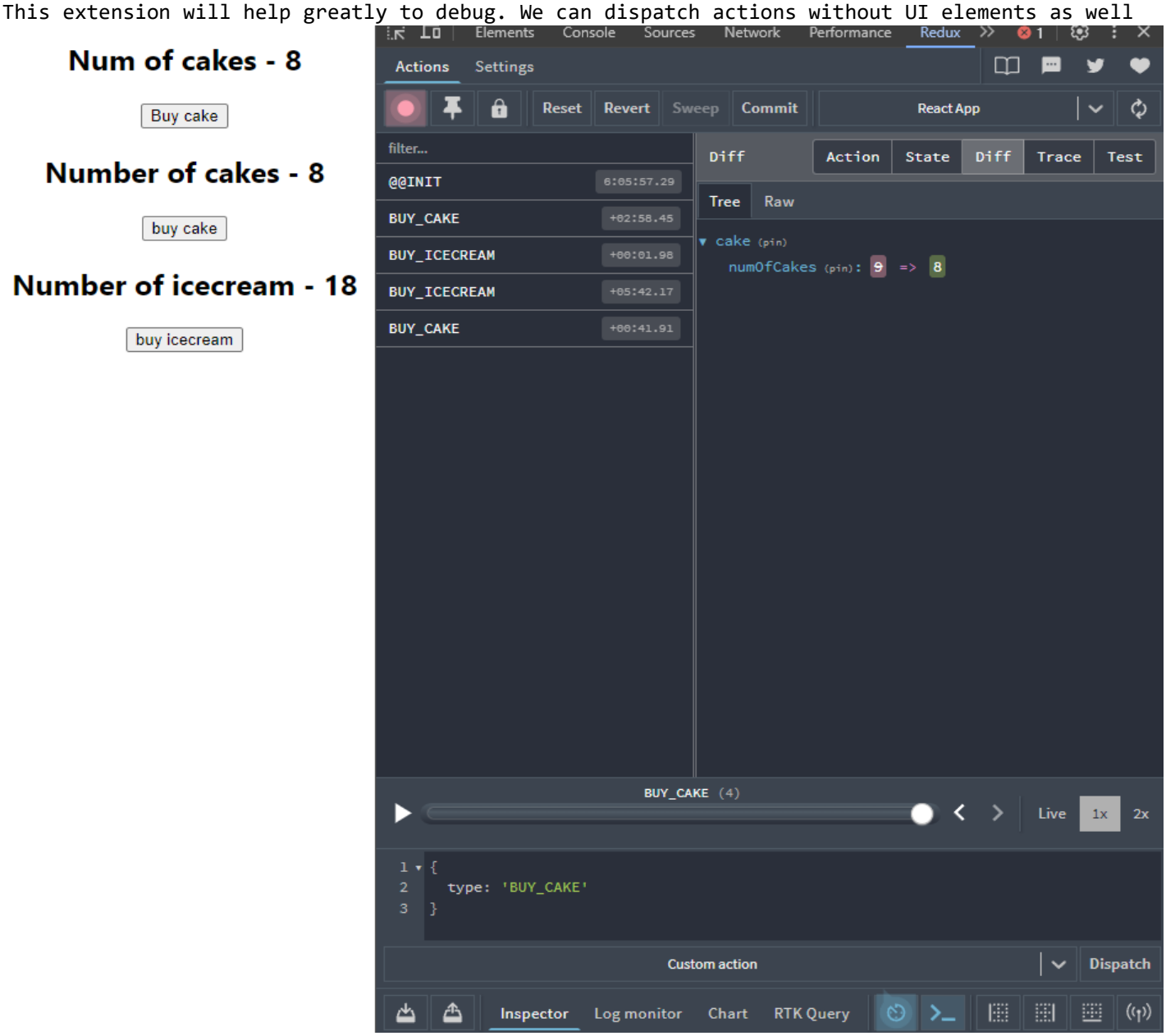
Check documentation in GitHub

https://github.com/reduxjs/redux-devtools

```
npm i @redux-devtools/extension
```

```
store.js M   cakeTypes.js   cakeActions.js   cakeReducer.js   CakeContainer.js
React-redux-tutorial-demo > src > redux > store.js > ...
1  import { createStore, applyMiddleware } from "redux";
2  import logger from "redux-logger";
3  import rootReducer from "./rootReducer";
4  import { composeWithDevTools } from "@redux-devtools/extension"
5
6  const store = createStore(
7    rootReducer,
8    composeWithDevTools(applyMiddleware(logger))
9  );
10
11 export default store;
12
```

This extension will help greatly to debug. We can dispatch actions without UI elements as well



React-redux Payload

At the moment we canm only buy one cake or icecream. To be able to buy multiple, we need to add an input method and update the app.

Adding input, setting state of input pass it with onlick, dispatch action with the number.

```
CakeContainer.js ×   NewCakeContainer.js U   cakeTypes.js   cakeActions.js M   cakeReducer.js M
ux-tutorial-demo > src > components > CakeContainer.js > CakeContainer
import React from "react";
import { connect } from "react-redux";
import { buyCake } from "../redux";

function CakeContainer(props) {
  return (
    <div>
      <h2>Number of cakes - {props.numOfCakes}</h2>
      <button onClick={props.buyCake}>buy cake</button>
    </div>
  );
}

const MapStateToProps = (state) => {
  return {
    numOfCakes: state.cake.numOfCakes,
  };
};

const mapDispathToProps = (dispatch) => {
  return {
    buyCake: () => dispatch(buyCake()),
  };
};

export default connect(MapStateToProps, mapDispathToProps)(CakeContainer);
```

```
NewCakeContainer.js U ×   CakeContainer.js   cakeTypes.js   cakeActions.js M   cakeReducer.js M
-tutorial-demo > src > components > NewCakeContainer.js > ...
import React, { useState } from "react";
import { connect } from "react-redux";
import { buyCake } from "../redux";

function NewCakeContainer(props) {
  const [number, setNumber] = useState(1);

  return (
    <div>
      <h2>Number of cakes - {props.numOfCakes}</h2>
      <input
        type="text"
        value={number}
        onChange={(e) => setNumber(e.target.value)}
      ></input>
      <button onClick={() => props.buyCake(number)}>buy {number} cakes</butto
    </div>
  );
}

const MapStateToProps = (state) => {
  return {
    numOfCakes: state.cake.numOfCakes,
  };
};

const mapDispathToProps = (dispatch) => {
  return {
    buyCake: number => dispatch(buyCake(number)),
  };
};

export default connect(MapStateToProps, mapDispathToProps)(NewCakeContainer)
```

Only (number) would have been enough, but without declaring a value would break the previous cakecontainer

Adding poayload to action creator

```
import { BUY_CAKE } from "./cakeTypes";

export const buyCake = () => {
    return {
        type: BUY_CAKE
    }
}
```

```
import { BUY_CAKE } from "./cakeTypes";

export const buyCake = (number = 1) => {
    return {
        type: BUY_CAKE,
        payload: number
    }
}
```

Add payload to reducer

```
import { BUY_CAKE } from "./cakeTypes";

const initalState = {
  numOfCakes: 10,
};

const cakeReducer = (state = initalState, action) => {
  switch (action.type) {
    case BUY_CAKE:
      return {
        ...state,
        numOfCakes: state.numOfCakes - 1,
      };
    default:
      return state;
  }
};

export default cakeReducer;
```

```
cakeReducer.js M ×   NewCakeContainer.js U   CakeContainer.js   cakeT
dux-tutorial-demo > src > redux > cake > cakeReducer.js > ...
import { BUY_CAKE } from "./cakeTypes";

const initalState = {
  numOfCakes: 10,
};

const cakeReducer = (state = initalState, action) => {
  switch (action.type) {
    case BUY_CAKE:
      return {
        ...state,
        numOfCakes: state.numOfCakes - action.payload,
      };
    default:
      return state;
  }
};

export default cakeReducer;
```

mapStateToProps function

we mapping redux state

```
const mapStateToProps = (state) => {
  return {
    numOfCakes: state.cake.numOfCakes,
  };
};
```

but now we have a second parameter, the props of the component itself called ownProps

```
const mapStateToProps = (state, ownProps) => {

}
```

we can conditiannaly assign the Redux state
if cake props was passed in, we access numberOfCakes, if not numberOfIcecreams

```
const mapStateToProps = (state, ownProps) => {
   const itemState = ownProps.cake ? state.cake.numberOfCakes : state.iceCream.numberOfIceCreams

   return {
    item: itemState
   }
}

export default ItemContainer
```

this to work we need to connect to Redux store
and export it accordinly

```
import React from 'react'
import { connect } from 'react-redux'

function ItemContainer(props) {
  return (
    <div>
        <h2>Item - {props.item} </h2>
    </div>
  )
}

const mapStateToProps = (state, ownProps) => {
   const itemState = ownProps.cake ? state.cake.numberOfCakes : state.iceCream.numberOfIceCreams

   return {
    item: itemState
   }
}

export default connect(mapStateToProps)(ItemContainer)
```

adding props on App.js if prop exist will get the
num of cakes otherwise the icecream

```
function App() {
  return (
    <Provider store={store}>
      <div className="App">
        <ItemContainer cake/>
        <ItemContainer />
        <HooksCakeContainer />
        <CakeContainer />
        <IceCreamContainer />
        <NewCakeContainer />
      </div>
    </Provider>
  );
}
```

Item - 10

Item - 20

Num of cakes - 10

Buy cake

Number of cakes - 10

buy cake

Number of icecream - 20

buy icecream

Number of cakes - 10

1 buy 1 cakes

mapDispatchToProps function

mapDispatchToProps also can have a second parameter called ownProps, like this we also can conditionally dispatch and action

```
const mapDispatchToProps = (dispatch, ownProps) => {
  const dispatchFunction = ownProps.cake
    ? () => dispatch(buyCake())
    : () => dispatch(buyIceCream());

    return {
      buyItem: dispatchFunction
    }
};

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

If the ownprop is cake the buy items button buy cake otherwise will buy icecream

```
function App() {
  return (
    <Provider store={store}>
      <div className="App">
        <ItemContainer cake/>
        <ItemContainer />
        <HooksCakeContainer />
        <CakeContainer />
        <IceCreamContainer />
        <NewCakeContainer />
      </div>
    </Provider>
  );
}
```

Item - 9

Buy items

Item - 19

Buy items

There will be times we don't want to subscibe to mapStateToProps, but only mapDispatchToProps, we pass the first params as null

```
export default connect(null, mapDispatchToProps)(ItemContainer);
```

The buttoons will still work but won`t display or update the state changes

Item -

Buy items

Item -

Buy items

Async actions

Actions

Synchronous Actions

As soon as an action was dispatched, the state was immediately updated.

If you dispatch the BUY_CAKE action, the numOfCakes was right away decremented by 1.

Same with BUY_ICECREAM action as well.

Async Actions

Asynchronous API calls to fetch data from an end point and use that data in your application.

Our Application

Fetches a list of users from an API end point and stores it in the redux store.

State ?

Actions ?

Reducer ?

State

```
state = {
  loading: true,
  data: [ ],
  error: ' '
}
```

loading - Display a loading spinner in your component
data - List of users
error – Display error to the user

Actions

FETCH_USERS_REQUEST – Fetch list of users

FETCH_USERS_SUCCESS – Fetched successfully

FETCH_USERS_FAILURE – Error fetching the data

Reducers

case: **FETCH_USERS_REQUEST**
loading: true

case: **FETCH_USERS_SUCCESS**
loading: false
users: data (from API)

case: **FETCH_USERS_FAILURE**
loading: false
error: error (from API)

userTypes.js

```
JS userTypes.js ×  JS userActions.js  JS userReducer.js  JS index.js M  JS rootReducer.js M
React-redux-tutorial-demo > src > redux > user > JS userTypes.js > [⊙] FETCH_USERS_FAILURE
1  export const FETCH_USERS_REQUEST = "FETCH_USERS_REQUEST"
2  export const FETCH_USERS_SUCCESS = "FETCH_USERS_SUCCESS"
3  export const FETCH_USERS_FAILURE = "FETCH_USERS_FAILURE"
```

userReducer.js

```
JS userTypes.js  JS userActions.js M  JS userReducer.js M ×  JS UserContainer.js  JS App.
React-redux-tutorial-demo > src > redux > user > JS userReducer.js > [⊙] reducer
1  import {
2    FETCH_USERS_REQUEST,
3    FETCH_USERS_SUCCESS,
4    FETCH_USERS_FAILURE,
5  } from "./userTypes";
6
7  const initalState = {
8    loading: false,
9    users: [],
10   error: "",
11 };
12
13 const reducer = (state = initalState, action) => {
14   switch (action.type) {
15     case FETCH_USERS_REQUEST:
16       return {
17         ...state,
18         loading: true,
19       };
20     case FETCH_USERS_SUCCESS:
21       return {
22         loading: false,
23         users: action.payload,
24         error: "",
25       };
26     case FETCH_USERS_FAILURE:
27       return {
28         loading: false,
29         users: [],
30         error: action.payload,
31       };
32     default: return state;
33   }
34 };
35
36 export default reducer;
```

userActions.js

```
JS userTypes.js  JS userActions.js ×  JS userReducer.js  JS index.js M  JS rootR
React-redux-tutorial-demo > src > redux > user > JS userActions.js > ...
1  import {
2    FETCH_USERS_REQUEST,
3    FETCH_USERS_SUCCESS,
4    FETCH_USERS_FAILURE,
5  } from "./userTypes";
6
7  export const fetchUsersRequest = () => {
8    return {
9      type: FETCH_USERS_REQUEST,
10   };
11 };
12
13 export const fetchUsersSuccess = (users) => {
14   return {
15     type: FETCH_USERS_SUCCESS,
16     payload: users,
17   };
18 };
19
20 export const fetchUsersFailure = (error) => {
21   return {
22     type: FETCH_USERS_FAILURE,
23     payload: error,
24   };
25 };
```

index.js

```
uTypes.js  JS userActions.js  JS userReducer.js  JS index.js M ×  JS rootReducer.js M
act-redux-tutorial-demo > src > redux > JS index.js
1  export { buyCake } from "./cake/cakeActions"
2  export { buyIceCream } from "./icecream/iceCreamActions"
3  export * from "./user/userActions"
```

rootReducer.js

```
.js  JS userActions.js  JS userReducer.js  JS index.js M  JS rootReducer.js M ×
tutorial-demo > src > redux > JS rootReducer.js > [⊙] default
import { combineReducers } from "redux";
import cakeReducer from "./cake/cakeReducer";
import iceCreamReducer from "./icecream/iceCreamReducer";
import userReducer from "./user/userReducer"

const rootReducer = combineReducers({
    cake: cakeReducer,
    iceCream: iceCreamReducer,
    user: userReducer
})

export default rootReducer
```

Axios, redux-thunk

npm install axios redux-thunk

store.js

```
pes.js ×  JS userActions.js  JS userReducer.js  JS index.js M  JS rootReducer.js M  JS store.js M ×
lux-tutorial-demo > src > redux > JS store.js > ...
import { createStore, applyMiddleware } from "redux";
import thunk from "redux-thunk";
import logger from "redux-logger";
import rootReducer from "./rootReducer";
import { composeWithDevTools } from "@redux-devtools/extension"

const store = createStore(
  rootReducer,
  composeWithDevTools(applyMiddleware(logger, thunk))
);

export default store;
```

with thunk middleware we are not returning and action, we are able to return a function

```
JS userTypes.js  JS userActions.js M ×  JS userReducer.js  JS index.js M  JS rootReducer.js M  JS store.js M
React-redux-tutorial-demo > src > redux > user > JS userActions.js > [ ] fetchUsers
import axios from "axios";
import {
  FETCH_USERS_REQUEST,
  FETCH_USERS_SUCCESS,
  FETCH_USERS_FAILURE,
} from "./userTypes";

export const fetchUsersRequest = () => {
  return {
    type: FETCH_USERS_REQUEST,
  };
};

export const fetchUsersSuccess = (users) => {
  return {
    type: FETCH_USERS_SUCCESS,
    payload: users,
  };
};

export const fetchUsersFailure = (error) => {
  return {
    type: FETCH_USERS_FAILURE,
    payload: error,
  };
};

export const fetchUsers = () => {
    return (dispatch) => {
        dispatch(fetchUsersRequest) // this sets loading to be true
        axios.get("https://jsonplaceholder.typicode.com/users")
        .then(response => {
            const users = response.data
            dispatch(fetchUsersSuccess(users))
        })
        .catch(error => {
            const errorMsg = error.message
            dispatch(fetchUsersFailure(errorMsg))
        })
    }
}
```

Subscribe to UserContainer.js

Useffect dispatch fetchUsers, results are conditionally rendered

```
import React, { useEffect } from "react";
import { connect } from "react-redux";
import { fetchUsers } from "../redux";

function UserContainer({ userData, fetchUsers }) {
  useEffect(() => {
    fetchUsers();
  }, []);

  return userData.loading ? (
    <h2>loading</h2>
  ) : userData.error ? (
    <h2>{userData.error}</h2>
  ) : (
    <div>
      <h2>user list:</h2>
      <div>
        {userData &&
          userData.users &&
          userData.users.map(user => <p>{user.name}</p>)}
      </div>
    </div>
  );
}

const mapStateToProps = state => {
  return {
    userData: state.user,
  };
};

const mapDispatchToProps = dispatch => {
  return {
    fetchUsers: () => dispatch(fetchUsers()),
  };
};

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

user list:

Leanne Graham

Ervin Howell

Clementine Bauch

Patricia Lebsack

Chelsey Dietrich

Mrs. Dennis Schulist

Kurtis Weissnat

Nicholas Runolfsdottir V

Glenna Reichert

Clementina DuBuque

Item -

Buy items

Item -

Buy items

Num of cakes - 10

Buy cake

Number of cakes - 10

buy cake

Number of icecream - 20

buy icecream

FIN

playlist