

01

useReducer

01. useReducer

- 상태와 상태 업데이트 로직을 보다 구조적으로 관리할 수 있게 도와주는 React의 내장 훅(HOOK)
- 상태 업데이트 로직이 복잡하거나 여러 타입의 상태 변경이 필요한 경우 사용

02. useReducer의 기본 구조

```
const [state, dispatch] = useReducer(reducer, initialState);
```

- **state**: 현재 상태 값
- **dispatch**: 상태를 업데이트하는 함수
 - 특정 액션을 reducer에 전달
 - 사용자에게 상태 값 변경을 위해 제공되는 함수
- **reducer**: 현재 상태와 액션을 받아서 새로운 상태를 반환하는 함수.
- **initialState**: 초기 상태 값.

03. 예제(Counter)

```
import { useReducer } from 'react';

const reducer = (state, action) => {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 };
    case 'decrement':
      return { count: state.count - 1 };
    default:
      return state;
  }
};

export default function Counter() {
  const initialState = { count: 0 };
  const [state, dispatch] = useReducer(reducer, initialState);
  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type: 'increment' })}>+</button>
      <button onClick={() => dispatch({ type: 'decrement' })}>-</button>
    </div>
  );
}
```

02

Redux

01. Redux

- 상태 관리를 돕는 라이브러리
- 애플리케이션의 전체 상태를 하나의 전역 상태로 관리
- 상태를 예측 가능한 방식으로 변경.
- 상태 관리가 복잡해지는 대규모 애플리케이션에서도 상태 흐름을 일관성 있게 유지

02. Redux의 개념

■ Store

- 전역 상태를 저장하는 하나의 중앙 저장소(스토어)

■ Action

- 상태를 변경하려면 액션(action)이라는 객체가 필요
- 액션은 반드시 type 속성을 포함해야 하며, 이 속성은 어떤 종류의 상태 변화를 일으키는지 설명
- 추가 데이터가 필요할 경우 payload라는 속성을 통해 전달
- `const incrementAction = { type: 'INCREMENT', payload: 1 };`

■ Reducer

- 스토어의 상태를 업데이트하는 함수
- 이전 상태와 액션을 인수로 받아 새로운 상태를 반환
- 상태를 변경하지 않고 새로운 상태 객체를 반환

02. Redux의 개념

■ Dispatch

- 액션을 스토어에 전달하는 함수
- 컴포넌트에서 dispatch를 호출하여 액션을 발생시키면, 스토어는 해당 액션을 리듀서로 전달해 상태를 업데이트
- `dispatch({ type: 'INCREMENT', payload: 1 });`

■ Selector

- 셀렉터는 스토어에서 필요한 데이터를 추출하여 컴포넌트에 전달하는 함수
- 컴포넌트는 스토어에 직접 접근하지 않고 셀렉터를 통해 필요한 상태를 접근

02. Redux의 개념

■ Redux의 데이터 흐름

1. Action 생성
2. Dispatch
3. Reducer
4. Store 업데이트
5. UI 업데이트

03. Redux 사용법

■ Redux 설치

- `npm install @reduxjs/toolkit react-redux`
- Redux 파일 구조 설정
- Redux 관련 코드를 별도의 파일로 분리하여 관리
- Ex) `store.js`, `counterSlice.js`..
- `counterSlice.js` - Slice 생성

03. Redux 사용법

```
import { createSlice } from '@reduxjs/toolkit';

const counterSlice = createSlice({
  name: 'counter',
  initialState: {
    count: 0,
  },
  reducers: {
    increment: (state) => {
      state.count += 1;
    },
    decrement: (state) => {
      state.count -= 1;
    },
    incrementByAmount: (state, action) => {
      state.count += action.payload;
    },
  },
});
// 액션생성자와 리듀서를 추출
export const { increment, decrement, incrementByAmount } = counterSlice.actions;
export default counterSlice.reducer;
```

03. Redux 사용법

■ store.js

```
import { configureStore } from '@reduxjs/toolkit';
import counterReducer from './counterSlice';
const store = configureStore({
  reducer: {
    counter: counterReducer,
  },
});
export default store;
```

■ index.js

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

const root=ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Provider store={store}>
    <App />
  </Provider>,
);
```

03. Redux 사용법

■ App.js

```
import React from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { increment, decrement, incrementByAmount } from './counterSlice';

function App() {
  const count = useSelector((state) => state.counter.count);
  // Redux 상태에서 count 값 가져오기
  const dispatch = useDispatch();
  // 액션 디스패치를 위한 함수

  return (
    <div>
      <h1>Counter: {count}</h1>
      <button onClick={() => dispatch(increment())}>Increment</button>
      <button onClick={() => dispatch(decrement())}>Decrement</button>
      <button onClick={() => dispatch(incrementByAmount(5))}>Increment by 5
      </button>
    </div>
  );
}
export default App;
```

03. Redux 사용법

■ 액션 생성자

```
import { useDispatch } from 'react-redux';
import { increment, decrement, incrementByAmount } from './counterSlice';

const dispatch = useDispatch();
dispatch(increment());
// { type: 'counter/increment' } 액션 객체 생성 후 디스패치
dispatch(decrement());
// { type: 'counter/decrement' } 액션 객체 생성 후 디스패치
dispatch(incrementByAmount(5));
// { type: 'counter/incrementByAmount', payload: 5 }
```