React

useEffect

참고링크 :
https://dmitripavlutin.com/react-useeffect-explanation/

컴포넌트 렌더링 (Mount): 컴포넌트가 처음으로 브라우저에 그려집니다.

useEffect 실행?:

- 처음 마운트될 때는 무조건 Yes로 진행됩니다.
- 리렌더링될 때는 의존성 배열을 확인합니다.

콜백 함수 (Side Effect) 실행: useEffect의 첫 번째 인자로 전달된 함수가 실행됩니다.

State/Props 변경?: 콜백 함수 내에서 상태나 props를 변경했을 수 있습니다.

- Yes라면 다시 컴포넌트가 리렌더링됩니다 (A로 돌아감).
- No라면 컴포넌트 업데이트 단계로 넘어갑니다.

컴포넌트 업데이트 : 변경된 내용이 있다면 브라우저에 반영됩니다.

리렌더링 ?: 상태나 props 변경 등으로 인해 컴포넌트가 다시 렌더링될 필요가 있는지 확인합니다.

- Yes라면 **의존성 배열 확인** 단계로 넘어갑니다.
- No (Unmount)라면 컴포넌트가 화면에서 사라지는 **언마운트** 단계로 넘어갑니다.

의존성 배열 확인 : useEffect의 두 번째 인자인 의존성 배열을 이전 렌더링 값과 비교합니다.

- **변경됨 :** 배열 내의 값 중 하나라도 변경되었다면 콜백 함수를 다시 실행합니다 (C로 돌아감).
- **변경 없음 :** 콜백 함수를 실행하지 않고 컴포넌트 업데이트 단계로 넘어갑니다 (E로 이동).

정리(Cleanup) 함수 실행 (반환된 경우): 컴포넌트가 언마운트되기 직전이나 다음 Effect 실행 전에 정리 함수가 있다면 실행됩니다.

컴포넌트 언마운트 : 컴포넌트가 브라우저 화면에서 제거됩니다.

종료 : 컴포넌트의 생명 주기가 종료됩니다.

1. **rendering 된 후 실행**
2. ***useEffect*(()=>{}, [Dependancy Array])**
3. ***useEffect*(()=>{}, []): 처음 렌더링될 때만 실행, 업데이트될 때에는 실행되지 않음**
4. **컴포넌트가 마운트 된 이후**
5. **의존성 배열에 있는 변수들 중 하나라도 값이 변경되었을 때 실행됨**
6. **의존성 배열 생략 시 컴포넌트 업데이트 시마다 실행됨**
7. **return () => {}: 컴포넌트가 언마운트되기 전이나 업데인트되기 직전에 작업을 수행, 언마운트될 때만 뒷정리 함수를 호출하고 싶으면 두 번째 파라미터에 빈배열을 넣는다.**

의존성 배열(Dependancy Array)에 따라 콜백함수가 실행되는 경우

1. 의존성 배열이 주어지지 않았을 때: 매 렌더링마다 실행
2. 의존성 배열이 빈 배열 [ ]일 때: 최초 렌더링 이후 1번만 실행
3. 의존성 배열에 값이 있을 때: 값이 변경되면 실행
4. 의존성 배열에 값이 2개 이상 있을 때: 어느 하나의 값이라도 변경되면 실행

clean up Effect

```
useEffect(() => {
  return () => {
    // Clean up Effect
  };
 }, []);
```

모든 useEffect 함수는 mount(마운트)될 때 실행되고, unmount(언마운트)될 때 모든 clean up 함수가 실행된다.

Effect 실습1

Info.jsx

```
import React, {useState, useEffect} from 'react';
const Info = () => {
const [name, setName] = useState('')
const [nickname, setNickname] = useState('');
useEffect(()=>{
    console.log('effect');
    console.log(name);
    return()=>{
        console.log('cleanup');
        console.log(name);
    }
}, [name])
```

```
return (
 <div>
   <div>
     <input
         name="name"
         value={name}
         onChange
             ={e=>setName(e.target.value)} />
     <input
         name="nickname"
         value={nickname}
         onChange
             ={e=>setNickname(e.target.value)} />
   </div>
<div>
      <div>
        <b>이름:</b> {name}
      </div>
      <div>
        <b>닉네임: </b> {nickname}
      </div>
    </div>
  </div>
);
};
export default Info;
```

App.jsx

```
import React, { useState } from 'react';
import Info from './Info';
const App = () =>{
const [visible, setVisible] = useState(false);
return(
    <div>
      <button onClick={()=>{setVisible(!visible)}}>
        {visible ? "숨기기": "보이기"}
      </button>
      <hr/>
      {visible && <Info />}
    </div>
);
}
export default App;
```

Effect 실습2(api 받기)

json data server

```
npm install -g json-server
# 또는
yarn global add json-server
```

db.json

```
{
   "users": [
     { "id": 1, "name": "Alice", "age": 30, "email": "alice@example.com", "city": "New
York" },
     { "id": 2, "name": "Bob", "age": 25, "email": "bob@example.com", "city": "Los Angeles"
},
     { "id": 3, "name": "Charlie", "age": 35, "email": "charlie@example.com", "city":
"Chicago" },
     { "id": 4, "name": "David", "age": 28, "email": "david@example.com", "city": "Houston"
},
     { "id": 5, "name": "Eve", "age": 22, "email": "eve@example.com", "city": "Phoenix" },
     { "id": 6, "name": "Jona", "age": 40, "email": "jona@example.com", "city":
"Philadelphia" },
     { "id": 7, "name": "Kim", "age": 31, "email": "kim@example.com", "city": "San Antonio"
}
   ],
```

```json
"products": [
     { "id": "A1", "name": "Laptop", "price": 1200, "category": "Electronics", "inStock":
true },
     { "id": "B2", "name": "Mouse", "price": 25, "category": "Electronics", "inStock":
true },
     { "id": "C3", "name": "Keyboard", "price": 75, "category": "Electronics", "inStock":
false },
     { "id": "D4", "name": "Monitor", "price": 300, "category": "Electronics", "inStock":
true },
     { "id": "E5", "name": "Tablet", "price": 450, "category": "Electronics", "inStock":
true },
     { "id": "F6", "name": "Smartphone", "price": 900, "category": "Electronics",
"inStock": true },
     { "id": "G7", "name": "Headphones", "price": 150, "category": "Electronics",
"inStock": false },
     { "id": "H8", "name": "T-Shirt", "price": 20, "category": "Apparel", "inStock": true
},
     { "id": "I9", "name": "Jeans", "price": 60, "category": "Apparel", "inStock": true
},
     { "id": "J10", "name": "Book", "price": 15, "category": "Books", "inStock": true }
   ]
 }
```

package.json

```
{
  "name": "my-json-server-project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "json-server --watch db.json --port 3001"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "json-server": "^0.17.4" // 예시 버전1
  }
}
```

실행 명령어

```
json-server --watch db.json --port 3001
```

ProductList.jsx

```
import React, { useEffect, useState } from 'react';

const ProductList = () => {
 const [products, setProducts] = useState();

 useEffect(() => {
   const fetchProducts = async () => {
     try {
       const response = await fetch('http://localhost:3001/products');
       if (!response.ok) {
         throw new Error(`HTTP error! status: ${response.status}`);
       }
       const data = await response.json();
       setProducts(data);
```

```
    } catch (error) {
      console.error('상품 데이터를 불러오는 중 오류 발생했습니다.', error);
      setProducts([]);
    }
  };
  fetchProducts();
}, []);

return (
  <div>
    <h1>상품 목록</h1>
    {Array.isArray(products) && products.length > 0 ? (
      <ul>
        {products.map(product => (
           <li key={product.id}>
             <strong>{product.name}</strong> - ${product.price} ({product.category})
             {product.inStock ? ' (재고 있음)' : ' (재고 없음)'}
           </li>
         ))}
       </ul>
     ) : (
       <p>
         {Array.isArray(products) && products.length === 0
           ? '상품 데이터가 없습니다.'
           : '상품 데이터를 불러오는 중...'}
       </p>
     )}
   </div>
 );
};
export default ProductList;
```

UserList.jsx

```
import React, { useEffect, useState } from 'react';
const UserList = () => {
 const [users, setUsers] = useState([]); // 빈 배열로 초기화

 useEffect(() => {
   const fetchUsers = async () => {
     try {
       const response = await fetch('http://localhost:3001/users');
       if (!response.ok) {
         throw new Error(`HTTP error! status: ${response.status}`);
       }
       const data = await response.json();
       setUsers(data); // prev 불필요
```

```
      } catch (error) {
        console.error('사용자 데이터를 불러오는 중 오류가 발생했습니다:',
error);
        setUsers([]);
      }
    };
    fetchUsers();
  }, []); // apiUrl은 상수이므로 의존성 제외

  return (
    <div>
      <h1>사용자 목록</h1>
      {users.length > 0 ? (
        <ul>
          {users.map((user) => (
           <li key={user.id}>
             <strong>{user.name}</strong> ({user.age}세, {user.city}) - {user.email}
           </li>
         ))}
       </ul>
     ) : (
       <p>
         {users.length === 0
           ? '사용자 데이터가 없습니다.'
           : '사용자 데이터를 불러오는 중...'}
       </p>
     )}
   </div>
 );
};
export default UserList;
```

App.js

```
npm install react-router-dom
# 또는
yarn add react-router-dom
```

```
import React from "react";
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";
import UserList from "./pages/UserList"; // UserList 컴포넌트 import
import ProductList from "./pages/ProductList"; // ProductList 컴포넌트 import

function App() {
 return (
   <Router>
     <div>
       <header>
         <h1>우리 쇼핑몰</h1>
         <nav>
```

```
    <ul>
      <li>
        <Link to="/users">사용자 목록</Link>
      </li>
      <li>
        <Link to="/products">상품 목록</Link>
      </li>
    </ul>
  </nav>
</header>
```

```
        <main>
          <Routes>
            <Route path="/users" element={<UserList />} />
            <Route path="/products" element={<ProductList />} />
            <Route path="/" element={<div>메인 페이지입니다.</div>} />{"
"}
            {/* 기본 경로 */}
          </Routes>
        </main>
        <footer>
          <p>&copy; 2025 우리 쇼핑몰</p>
        </footer>
      </div>
    </Router>
 );
}
export default App;
```

useReducer

dataReducer.js

```
export const initialState = {
   loading: true,
   data: [],
   error: null,
 };
  export function dataReducer(state, action) {
   switch (action.type) {
     case 'FETCH_INIT':
       return { ...state, loading: true, error: null };
     case 'FETCH_SUCCESS':
       return { loading: false, data: action.payload, error: null };
     case 'FETCH_ERROR':
       return { loading: false, data: [], error: action.error };
     default:
       return state;
   }
 }
```

ProductList.jsx

```
import React, { useEffect, useReducer } from 'react';
import { dataReducer, initialState } from '../reducers/dataReducer';
const ProductList = () => {
 const [state, dispatch] = useReducer(dataReducer, initialState);
 const { loading, data: products, error } = state;
 useEffect(() => {
   const fetchProducts = async () => {
     dispatch({ type: 'FETCH_INIT' });
     try {
       const response = await fetch('http://localhost:3001/products');
       if (!response.ok) throw new Error(`HTTP error! status: ${response.status}`);
       const data = await response.json();
       dispatch({ type: 'FETCH_SUCCESS', payload: data });
```

```
      } catch (err) {
        dispatch({ type: 'FETCH_ERROR', error: err.message });
      }
    };
    fetchProducts();
  }, []);
  return (
    <div>
      <h1>상품 목록</h1>
      {loading && <p>상품 데이터를 불러오는 중...</p>}
      {error && <p>에러 발생: {error}</p>}
      {!loading && products.length === 0 && <p>상품 데이터가 없습니다.</p>}
      <ul>
        {products.map((product) => (
          <li key={product.id}>
            <strong>{product.name}</strong> - ${product.price} ({product.category})
            {product.inStock ? ' (재고 있음)' : ' (재고 없음)'}
          </li>
        ))}
      </ul>
    </div>
  );
};
export default ProductList;
```

UserList.jsx

```
import React, { useEffect, useReducer } from 'react';
import { dataReducer, initialState } from '../reducers/dataReducer';

const UserList = () => {
 const [state, dispatch] = useReducer(dataReducer, initialState);
 const { loading, data: users, error } = state;
 useEffect(() => {
   const fetchUsers = async () => {
     dispatch({ type: 'FETCH_INIT' });
     try {
       const response = await fetch("http://localhost:3001/users");
       if (!response.ok) throw new Error(`HTTP error! status: ${response.status}`);
       const data = await response.json();
       dispatch({ type: 'FETCH_SUCCESS', payload: data });
} catch (err) {
    dispatch({ type: 'FETCH_ERROR', error: err.message });
  }
 };
 fetchUsers();
}, []);
return (
 <div>
   <h1>사용자 목록</h1>
   {loading && <p>사용자 데이터를 불러오는 중...</p>}
   {error && <p>에러 발생: {error}</p>}
   {!loading && users.length === 0 && <p>사용자 데이터가 없습니다.</p>}
```

```
     <ul>
       {users.map((user) => (
         <li key={user.id}>
           <strong>{user.name}</strong> ({user.age}세, {user.city}) - {user.email}
         </li>
       ))}
     </ul>
   </div>
 );
};
export default UserList;
```

redux thunk

npm install redux react-redux redux-thunk
or
yarn add redux react-redux redux-thunk

redux/actions/userActions.js

```
export const fetchUsers = () => async (dispatch) => {
   dispatch({ type: 'FETCH_USERS_REQUEST' });
    try {
     const res = await fetch('http://localhost:3001/users');
     if (!res.ok) throw new Error(`HTTP error! status: ${res.status}`);
     const data = await res.json();
      dispatch({ type: 'FETCH_USERS_SUCCESS', payload: data });
   } catch (error) {
     dispatch({ type: 'FETCH_USERS_FAILURE', error: error.message });
   }
 };
```

redux/actions/productActions.js

```
export const fetchProducts = () => async (dispatch) => {
  dispatch({ type: 'FETCH_PRODUCTS_REQUEST' });
  try {
    const res = await fetch('http://localhost:3001/products');
    if (!res.ok) throw new Error(`HTTP error! status: ${res.status}`);
    const data = await res.json();
    dispatch({ type: 'FETCH_PRODUCTS_SUCCESS', payload: data });
  } catch (error) {
    dispatch({ type: 'FETCH_PRODUCTS_FAILURE', error: error.message });
  }
};
```

redux/reducers/userReducer.js

```
const initialState = {
  loading: false,
  data: [],
  error: null,
};
export const userReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'FETCH_USERS_REQUEST':
      return { ...state, loading: true, error: null };
    case 'FETCH_USERS_SUCCESS':
      return { ...state, loading: false, data: action.payload };
    case 'FETCH_USERS_FAILURE':
      return { ...state, loading: false, error: action.error };
    default:
      return state;
  }
};
```

redux/reducers/productReducer.js

```
const initialState = {
  loading: false,
  data: [],
  error: null,
};
export const productReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'FETCH_PRODUCTS_REQUEST':
      return { ...state, loading: true, error: null };
    case 'FETCH_PRODUCTS_SUCCESS':
      return { ...state, loading: false, data: action.payload };
    case 'FETCH_PRODUCTS_FAILURE':
      return { ...state, loading: false, error: action.error };
    default:
      return state;
  }
};
```

redux/reducers/index.js

```
import { combineReducers } from 'redux';
import { productReducer } from './productReducer';
import { userReducer } from './userReducer';

const rootReducer = combineReducers({
 products: productReducer,
 users: userReducer,
});

export default rootReducer;
```

redux/store.js

```
import { createStore, applyMiddleware } from 'redux';
import {thunk} from 'redux-thunk';
import rootReducer from './reducers'; // index.js는 자동으로 인식됨

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

export default store;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { Provider } from 'react-redux';
import store from './redux/store';

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

UserList.jsx

```
import React, { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchUsers } from '../redux/actions/userActions';

const UserList = () => {
 const dispatch = useDispatch();
 const { loading, data: users, error } = useSelector((state) => state.users);

 useEffect(() => {
   dispatch(fetchUsers());
 }, [dispatch]);
return (
   <div>
     <h1>사용자 목록</h1>
     {loading && <p>사용자 데이터를 불러오는 중...</p>}
     {error && <p>에러 발생: {error}</p>}
     {!loading && users.length === 0 && <p>사용자 데이터가 없습니다.</p>}
     <ul>
       {users.map((user) => (
         <li key={user.id}>
           <strong>{user.name}</strong> ({user.age}세, {user.city}) - {user.email}
         </li>
       ))}
     </ul>
   </div>
 );
};
export default UserList;
```

ProductList.jsx

```
import React, { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchProducts } from '../redux/actions/productActions';

const ProductList = () => {
 const dispatch = useDispatch();
 const { loading, data: products, error } = useSelector((state) => state.products);

 useEffect(() => {
   dispatch(fetchProducts());
 }, [dispatch]);
```

```
  return (
    <div>
      <h1>상품 목록</h1>
      {loading && <p>상품 데이터를 불러오는 중...</p>}
      {error && <p>에러 발생: {error}</p>}
      {!loading && products.length === 0 && <p>상품 데이터가 없습니다.</p>}
      <ul>
        {products.map((product) => (
          <li key={product.id}>
            <strong>{product.name}</strong> - ${product.price} ({product.category})
            {product.inStock ? ' (재고 있음)' : ' (재고 없음)'}
          </li>
        ))}
      </ul>
    </div>
  );
};
export default ProductList;
```

useReducer 실습

실습1(Count)

```
import React, {useReducer} from 'react'
const initialState = {
   value: 0,
}
const countReducer = (state, action) =>{

   switch(action.type){
       case "INCREASE":
           return {value:state.value + 1};
       case "DECREASE":
           return {value: state.value -1};
       default:
           return state;
   }
}
```

```
const Count = () => {
   const [state, dispatch] = useReducer(countReducer, initialState)
 return (
   <>
       <p>
           Count: <b>{state.value}</b>
       </p>
       <button onClick={()=>dispatch({type:"INCREASE"})}>
           +
       </button >
       <button onClick={()=>dispatch({type:"DECREASE"})}>
           -
       </button>
   </>)}
export default Count
```

redux thunk(Count)

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

redux/actions.js

```
export const increase = () => (dispatch) => dispatch({type:
"INCREASE"})
export const decrease = () => (dispatch) => dispatch({type:
"DECREASE"})
```

redux/reducers.js

```
const initialState = {
   value: 0
}

export const countReducer = (state=initialState, action) => {
   switch(action.type){
       case "INCREASE":
           return {...state, value: state.value+1}
       case "DECREASE":
           return {...state, value: state.value-1}
       default:
           return state;
   }
}
```

redux/index.js

```
import { combineReducers } from "redux";
import { countReducer } from "./reducers";

const rootReducers = combineReducers({
  countReducer,
})

export default rootReducers;
```

redux/store.js

```
import { applyMiddleware, legacy_createStore } from "redux";
import rootReducers from ".";
import { thunk } from "redux-thunk";

const store = legacy_createStore(rootReducers, applyMiddleware(thunk));

export default store;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { Provider } from 'react-redux';
import store from './redux/store';

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

components/Count.jsx

```
import React from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { increase, decrease } from '../redux/actions'

const Count = () => {
   const state = useSelector((state) => state.countReducer)
   const dispatch = useDispatch();
 return (
   <>
       <p>
           Count: <b>{state.value}</b>
       </p>
       <button onClick={()=>dispatch(increase())}>+</button >
       <button onClick={()=>dispatch(decrease())}>-</button>
   </>)}
export default Count
```

App.js

```
import React from 'react';
import Count from './components/Count';
import './App.css';

function App() {
 return (
   <div className="App">
     <header>
       <h1>useReducer 실습</h1>
     </header>
     <main><Count/></main>
     <footer>
       <p>&copy: 2025 React 연습</p>
     </footer>
   </div>
 );}
export default App;
```

실습2(name, nickname)

Nickname.jsx

```
import { useCallback, useReducer } from "react";

const initialState = {
   name: null,
   nickname: null,
}

const nicknameReducer = (state, action) =>{
   switch(action.type){
       case "CHANGE_INPUT":
           return {
               ...state,
               [action.name]: action.value
           }
       default:return state;
   }}
```

```
const Nickname = () => {
   const [state, dispatch] = useReducer(nicknameReducer, initialState);
   const handleOnChange = useCallback((e) => {
       const { name , value } = e.target;
       dispatch({type: "CHANGE_INPUT", name, value})
   }, [dispatch])

  return (
   <>
       <div>
          <input type="text" name="name" value={state.name} onChange={handleOnChange} />
       </div>
       <div>
          <input type="text" name="nickname" value={state.nickname} onChange={handleOnChange} />
       </div>
       <div>
           <b>이름: </b> {state.name}
       </div>
       <div>
           <b>닉네임: </b> {state.nickname}
       </div>
   </>
 )
}

export default Nickname
```

App.js

```
import React from 'react';
import './App.css';
import Nickname from './components/Nickname';

function App() {
 return (
   <div className="App">
     <header>
       <h1>useReducer 실습</h1>
     </header>
     <main><Nickname/></main>
     <footer>
       <p>&copy: 2025 React 연습</p>
     </footer>
   </div>);}
export default App;
```

redux thunk (name, nickname)

redux/actions.js

```
export const changeInput = (name, value) => {
 return (dispatch) => {
   dispatch({
     type: 'CHANGE_INPUT',
     payload: { name, value }
   });
 };
};
```

redux/reducers.js

```
// reducers/nicknameReducer.js
const nicknameInitialState = {
 name: '',
 nickname: ''
};

export const nicknameReducer = (state = nicknameInitialState,
action) => {
 switch (action.type) {
   case 'CHANGE_INPUT':
     return {...state,
       [action.payload.name]: action.payload.value
     };
   default:
     return state;}};
```

redux/index.js

```
import { combineReducers } from "redux";
import { countReducer, nicknameReducer } from "./reducers";

const rootReducers = combineReducers({
   countReducer,
   nicknameReducer,
})
export default rootReducers;
```

redux/store.js

```
import { applyMiddleware, legacy_createStore } from "redux";
import rootReducers from ".";
import { thunk } from "redux-thunk";

const store = legacy_createStore(rootReducers, applyMiddleware(thunk));

export default store;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { Provider } from 'react-redux';
import store from './redux/store';

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

components/Nickname.jsx

```
// components/Nickname.js
import React, { useCallback } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { changeInput } from '../redux/actions';

const Nickname = () => {
 const { name, nickname } = useSelector((state) => state.nickname);
 const dispatch = useDispatch();

 const handleOnChange = useCallback((e) => {
   const { name, value } = e.target;
   dispatch(changeInput(name, value));
 }, [dispatch]);
 return (
   <>
     <div>
       <input type="text" name="name" value={name} onChange={handleOnChange} />
     </div>
     <div>
       <input type="text" name="nickname" value={nickname} onChange={handleOnChange} />
     </div>
     <div>
       <b>이름: </b> {name}
     </div>
     <div>
       <b>닉네임: </b> {nickname}
     </div>
   </> );};
export default Nickname;
```

App.js

```
import React from 'react';
import './App.css';
import Nickname from './components/Nickname';

function App() {
 return (
   <div className="App">
     <header>
       <h1>useReducer 실습</h1>
     </header>
     <main><Nickname/></main>
     <footer>
       <p>&copy: 2025 React 연습</p>
     </footer>
   </div>);}
export default App;
```