React Hook 강의(2)

useCallback, custom hook(axios, modal)

~ by tunalee

React useCallback

- `useCallback`은 메모이제이션된 콜백 함수를 반환
- 의존성 배열에 있는 값이 변경되지 않으면 함수는 다시 생성되지 않음
- 주로 불필요한 렌더링을 방지하고, 함수가 자주 재생성되는 문제를 해결하는 데 사용

언제 사용하는가?

- 함수가 자식 컴포넌트에 props로 전달될 때
- 컴포넌트가 리렌더링될 때마다 새로운 함수가 생성되지 않도록 할 때
- 테마 적용할 때

useMemo

```
useMemo(()=> {
    return value;
}, [item])
```

useCallback

```
useMemo(()=>{
    return value;
}, [item])
```

렌더링마다 callback 함수가 실행 됨

```
function App() {
    const [inputValue, setInputValue] = useState('')
    const handleClick = () => {
        console.log(inputValue)
    }
    useEffect(() => {
        console.log('handleClick changed');
    }, [handleClick])
    return (
        <div>
            <input type="text"
                   onChange={e => setInputValue(e.target.value)}
            />
            <button onClick={handleClick}>button</button>
        </div>
    )
}

export default App
```

useCallback

```
function App() {
    const [inputValue, setInputValue] = useState('')
    const handleClick = useCallback(() => {
        console.log(inputValue)
    }, [inputValue])
    useEffect(() => {
        console.log('handleClick changed');
    }, [handleClick])
    return (
        <div>
            <input type="text"
                   onChange={e => setInputValue(e.target.value)}
            />
            <button onClick={handleClick}>button</button>
        </div>
    )
}

export default App
```

렌더링마다 callback 함수가 실행 됨

```
function App() {
    const [size, setSize] = useState(100);

    const createSize = () => {
        return {
            width: `${size}px`,
            height: `${size}px`,
            backgroundColor: 'orange'
        }
    }
    useEffect(() => {
        console.log('렌더링이 되었습니다.');
    }, [])

    return (
        <div>
            <input type="number"
                   onChange={e => setSize(e.target.value)}
                   value={size}
            />
            <Box createBoxStyle={createSize}/>
        </div>
    )
}

export default App
```

useCallback

```
function App() {
    const [size, setSize] = useState(100);

    const createSize = useCallback (() => {
        return {
            width: `${size}px`,
            height: `${size}px`,
            backgroundColor: 'orange'
        }
    }, [size]);
    useEffect(() => {
        console.log('렌더링이 되었습니다.');
    }, [])

    return (
        <div>
            <input type="number"
                   onChange={e => setSize(e.target.value)}
                   value={size}
            />
            <Box createBoxStyle={createSize}/>
        </div>
    )
}

export default App
```

Costom Hook

- 복잡한 로직을 단순화하고 재사용 가능하게 만듦
- 상태 및 효과를 쉽게 관리
- 컴포넌트 간의 로직 공유

useModal

```
function App() {
    const { isOpen, openModal, closeModal } = useModal();

    return (
        <div>
            <button onClick={openModal}>모달 열기</button>
            {isOpen &&
                <Modal>
                    <h2>모달입니당~</h2>
                    <button onClick={closeModal}>모달 닫기</button>
                </Modal>
            }
        </div>
    )
}

export default App;
```

useModal (useModal.js)

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

export default function useModal() {
    const [isOpen, setIsOpen] = useState(false);
    const openModal = useCallback(() => setIsOpen(true), []);
    const closeModal = useCallback(() => setIsOpen(false), []);

    return { isOpen, openModal, closeModal };
}
```

useModal (Modal.jsx)

```jsx
function Modal({ children }) {

    return (
        <div style={modalStyles.overlay}>
            <div style={modalStyles.modal}>
                {children}
            </div>
        </div>
    );
}

const modalStyles = {
    overlay: {
        position: "fixed",
        top: 0,
        left: 0,
        right: 0,
        bottom: 0,
        backgroundColor: "rgba(0, 0, 0, 0.5)",
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
    },
    modal: {
        backgroundColor: "white",
        padding: "20px",
        borderRadius: "8px",
        maxWidth: "500px",
        width: "100%",
    }
};

export default Modal;
```

API 요청이란?

- API(Application Programming Interface)는 서로 다른 소프트웨어 간의 상호작용을 위한 규칙
- 클라이언트는 API를 통해 서버와 데이터를 주고받음

API 요청

- 데이터 가져오기 (GET 요청)
- 데이터 생성 (POST 요청)
- 데이터 업데이트 (PUT/PATCH 요청)
- 데이터 삭제 (DELETE 요청)

API Response Status

- 2xx 성공
  - 200 ok
  - 201 created
  - 204 No Content
- 4xx 클라이언트 오류
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
- 5xx 서버 오류

Axios

- Axios는 Promise 기반의 HTTP 클라이언트로, API 요청을 간편하게 처리
- **특징**:
  - 브라우저와 Node.js 지원
  - 요청 및 응답 인터셉터
  - JSON 자동 변환

패키지 설치

```
$ npm install axios json-server concurrently
```

package.json

```
  "scripts": {
    "dev": "concurrently \"vite\" \"npm run json-server\"",
    "json-server": "json-server --watch db.json --port 3000"
  },
```

db.json 작성

```
{
  "posts": [
    { "id": 1, "title": "First Post", "body": "This is the first post." },
    { "id": 2, "title": "Second Post", "body": "This is the second post." }
  ]
}
```

react & json 서버 실행

```
$ npm run dev
```

```
[1] ♡ㅅ(〃> ᴗ <〃)ㅅ♡
[1]
[1] Index:
[1] http://localhost:3000/
[1]
[1] Static files:
[1] Serving ./public directory if it exists
[1]
[1] Endpoints:
[1] http://localhost:3000/posts
```

Axios 적용 (Get)

```
function Post() {
    const [posts, setPosts] = useState([]);
    const fetchPost = async () => {
        try{
            const post = await axios.get('http://localhost:3000/posts')
            setPosts(post.data);
        } catch (e) {
            console.log(e)
        }
    }
    useEffect(() => {
        fetchPost()
    })
    return(
        <div>
            {posts.map((post, index) => (
               <div key={index}>
                   <h1>{post.title}</h1>
                   <div>
                       {post.body}
                   </div>
               </div>
            ))}
        </div>
    )
}
export default Post
```

Axios 적용 (Post)

```
function PostForm() {
    const [title, setTitle] = useState('');
    const [body, setBody] = useState('');

    const createPost = async () => {
        const newPost = { title, body };
        try {
            const response = await axios.post('http://localhost:3000/posts', newPost);
            setTitle('');
            setBody('');
        } catch (error) {
            console.error('Error creating post:', error);
        }
    };

    return (
        <div>
            <h1>게시글 만들기</h1>
            <input
                type="text"
                placeholder="Title"
                value={title}
                onChange={e => setTitle(e.target.value)}
            />
            <input
                placeholder="Body"
                value={body}
                onChange={e => setBody(e.target.value)}
            />
            <button onClick={createPost}>Submit</button>
        </div>
    );
}
```

useAxios 적용

```
const customAxios = axios.create({
    baseURL: 'http://localhost:3000/',
});

customAxios.interceptors.response.use(
    response => {
        return response;
    },
    error => {
        // if (error.response.status === 401) {
        //     window.location.href = '/login';
        //} 에러 예외 처리
        return Promise.reject(error);
    }
);

const useAxios = () => {
    // axios 설정들을 할 수 있음
    // const [cookies] = useCookies(['token']);
    // customAxios.defaults.headers.common['Authorization'] = `Token ${cookies.token}`;
    return customAxios;
};

export default useAxios;
```

```
function app() {
    const [title, setTitle] = useState('');
    const [body, setBody] = useState('');
    const axios = useAxios();

    const createPost = async () => {
        const newPost = { title, body };
        try {
            const response = await axios.post('posts', newPost);
            console.log('Post created:', response.data);
            setTitle('');
            setBody('');
        } catch (error) {
            console.error('Error creating post:', error);
        }
    };

    return (
        <div>
            <h1>게시글 만들기</h1>
            <input
                type="text"
                placeholder="Title"
                value={title}
                onChange={e => setTitle(e.target.value)}
            />
            <input
                placeholder="Body"
                value={body}
                onChange={e => setBody(e.target.value)}
            />
            <button onClick={createPost}>Submit</button>
        </div>
    );
}

export default App;
```

실습

- react-router-dom 을 사용하여 post 목록, post detail, post add 페이지 만들어보기
- axios를 사용하여 json server에 Posts 테이블의 값을 post, get 해보기
- 그 외 최적화 할 수 있으면 해보기