React-query-1

태그	
날짜	@2023년 5월 15일

https://www.jeje01.me/react/about-react-query/

위 사이트를 참고 했다.

리액트 쿼리란?

TS/JS, React, Solid, Vue와 Svelte를 위한 강력한 비동기 상태 관리 도구이다.

리액트 쿼리의 특징

- Declarative & Automatic
  - 캐싱, 백그라운드 업데이트와 오래된 데이터를 설정없이 다룬다.
- Simple & Familiar
  - promises, async/await와 익숙하다면, TanStack Query 사용법을 알 수 있다. 관리 해야할 전역 상태, reducers, 정규화 시스템 또는 이해해야할 복잡할 설정이 없다.
- Extensible
  - TanStack Query는 구성 가능하다.

Queris

쿼리 기본

- 쿼리는 유일한 키에 연결된 비동기적인 데이터에 대한 선언적인 의존성이다. 쿼리는 서버에서 데이터를 불러오는 Promise를 기반으로 한 모든 메서드(get과 Post 메서드를 포함한)와 사용 가능하다. 만약 그 메서드가 서버에서 데이터를 수정한다면, Mutations를 사용하는 것을 추천한다.

- 컴포넌트나 커스텀 훅의 쿼리를 구독하려면, 최소한 다음을 만족하는 useQuery Hook을 호출하라.
  - 쿼리에 맞는 유일한 key
  - 데이터를 불러오거나 에러를 반환하는 함수

useQuery

- 데이터를 가져오기 위한 api이다. post, udate는 useMutation을 사용해야한다.
- useQuery는 비동기로 작동해서 한 컴포넌트에 여러개의 useQuery가 있다면 하나가 끝나고 다음 useQuery가 실행되는 것이 아닌 두개의 useQuery가 동시에 실행된다. 여러개의 비동기 query가 있다면 useQueries를 추천한다.
- enabled를 사용하면 useQuery를 동기적으로 사용 가능하다.

Mutations

- 쿼리와 다르게 , mutations는 전형적으로 데이터를 create/update/delete하거나 서버의 부작용을 수행하기 위해 사용된다. 이 목적으로, TanStack Query는 useMutation Hook을 제공한다.

```
const App = () => {
  const { isLoading, isSuccess, mutate } = useMutation({
    mutationFn: (newTodo) => {
      return axios.post('/todos', newTodo)
    },
  })

  return (
    <button
      onClick={() => {
        mutate({
          id: new Date(),
          title: "Do laundry",
        })
      }}
    >
      Create Todo
    </button>
```

```
  )
}
```

Query key

- TanStack Query는 쿼리 키를 기반으로 쿼리 캐싱을 관리한다.
- 쿼리 키는 불러오는 데이터를 고유하게 설명하기 때문에, 쿼리 함수에서 사용하는 변하는 변들을 다 포함해야 한다. 호출 시 id가 필요한 경우에는 id도 키에 추가해야 한다.

```
function Todos({ todoId }) {
  const result = useQuery({
    queryKey: ['todos', todoId],
    queryFn: () => fetchTodoById(todoId),
  })
}
```

Query fn

- 쿼리 함수는 Promise를 반환하는 모든 함수가 될 수 있다. 반환되는 promise는 데이터를 불러오거나 에러를 뱉어야 한다.
- 다음의 쿼리 함수 구성은 모두 유효하다.

```
useQuery({ queryKey: ['todos'], queryFn: fetchAllTodos })
useQuery({ queryKey: ['todos', todoId], queryFn: () => fetchTodoById(todoId) })
useQuery({
  queryKey: ['todos', todoId],
  queryFn: ({ queryKey }) => fetchTodoById(queryKey[1]),
})
```

장점

- 상태 관리를 편리하게 해주고 데이터가 변경될 때마다 자동으로 리렌더링 해주고, 직접 상태 관리를 할 필요가 없다.
- 코드 작성을 간단하게 해주어 쿼리를 작성하면 데이터를 가져오고 변환할 수 있고, 리액트 컴포넌트에서 쉽게 사용할 수 있다.
- 선언적 프로그래밍을 지원하는데, 코드 작성 시에 원하는 것을 설명하기 때문에 가독성이 높아지고 유지보수가 쉬워진다.
- 리액트 쿼리는 데이터를 캐싱해 중복된 API호출을 방지하고, 성능 향상에 도움을 준다.

단점

- 리액트 쿼리는 데이터에 종속적이라 데이터 구조에 변경이 생기면 코드를 수정해야할 수 있다.
- 새로운 개념의 도입으로 러닝 커브가 있을 수 있기 때문에, 초기 설정과 문제 해결에 시간이 걸릴 수 있다.
- 리액트 쿼리는 고급 사용법에 대한 이해가 필요한데, 이를 통해 성능을 최적화 하고 다양한 데이터를 처리할 수 있지만 이해하는 것이 어려울 수 있다.