

React 상태 관리와 라우팅

효율적인 상태 관리와 라우팅 설계로 React 앱의 성능과 유지보수성을 향상시키는 방법을 알아봅니다.

상태 공유 문제 해결을 위한 도구를 비교하고, React Router의 작동 원리 및 실전 구조 설계를 다룹니다.



by Hank Kim

상태 관리란 무엇인가?

로컬 상태

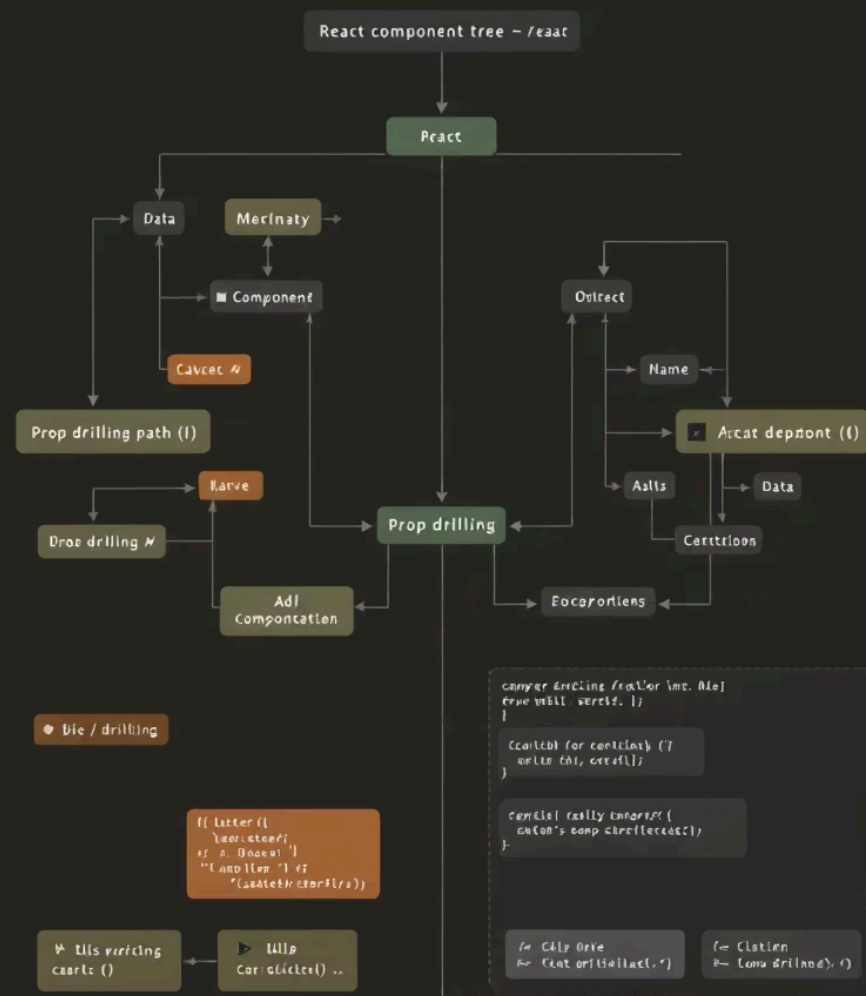
각 컴포넌트 내부에서 useState, useReducer를 사용하여 UI 상태를 관리합니다. 예를 들어, 입력 폼의 값이나 토글 버튼의 활성화 상태를 관리합니다.

전역 상태

애플리케이션의 여러 컴포넌트에서 공통으로 사용하는 로그인 정보, 테마 설정, 사용자 권한과 같은 데이터를 중앙 집중식으로 관리합니다.



















문제점

props drilling 현상으로 인해 하위 컴포넌트에 여러 단계를 거쳐 상태를 전달해야 하며, 중첩된 상태 관리가 복잡해져 유지보수가 어려워집니다.



HOW COMPARON TO STATE WATH STAE MANAGEMETTIN REACT

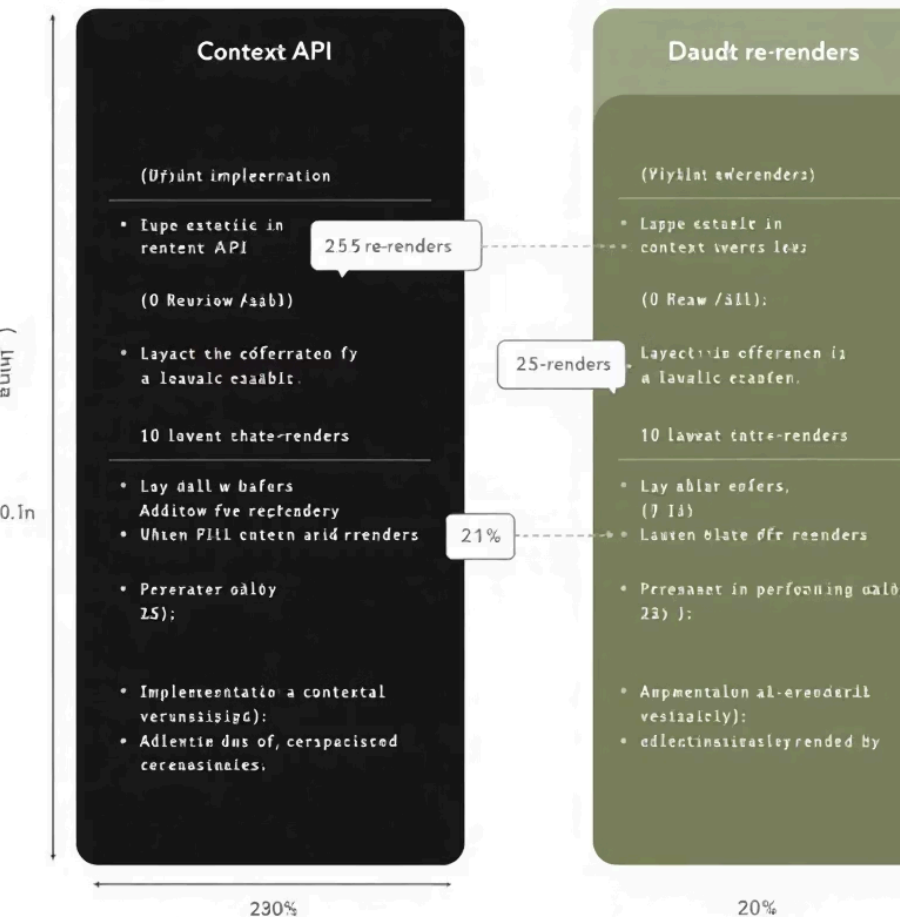
Please the counign of in regat ford egher land lnhicks leas goue ler curt state
managemt bulds and reants and on your with state management

REDUX		CONTECT API	
	Context API		Look the row is clesercing alefection attes the preputtiens df cleact all ome the sure of ther reetlis abusu un in resictly state.
	How lis theact eary laves		Sow l'ichensgat in iconsection
	Low if the ring lranind inglon ges and anaagrsent sheo very firen.		Low if the ring lranind inglon ges and anaagrsent sheo very firen.
	Bew lichering und fulll lraget chende firen.		Low if the ring lranind inglon ges and anaagrsent sheo very firen.
	Durr tie context of arctd und in tne the riens.		Low if the ring lranind inglon ges and anaagrsent sheo very firen.
	Cary le dafion of a fide.		Low if the ring lranind inglon ges and anaagrsent sheo very firen.
	Nu a kiae of flori is fire		Low if the ring lranind inglon ges and anaagrsent sheo very firen.
	Reptorn and upeat lives		Low if the ring lranind inglon ges and anaagrsent sheo very firen.
	Forat ther saferment the surf und conslications of the poremoie anch souplele staces.		Low if the ring lranind inglon ges and anaagrsent sheo very firen.

상태 관리 도구 비교

도구	장점	단점	적합한 경우
Context API	간단, 내장	리렌더링 범위 کم	테마, 언어
Redux (RTK)	구조화, DevTools	복잡, 보일러플 레이트	대형 앱
Zustand	간결, 빠름	미들웨어 제한	SPA, 실무 MVP

React Context API vs Daudt Performance



Surnect conteit an muvie norher frame in pregformment vs. Daudt re-renders. contnert for ther usigns rusticuters complenation eluno offen nakadl in puching dir re-render:

Zustand가 Context API보다 나은 점

Context API 문제

Provider 내부 전체 리렌더링이 발생합니다.

Zustand 장점

컴포넌트 단위 subscribe로 변경된 부분만 리렌더링합니다.

확장성

React 외부(WebSocket 수신 로직)에서도 store 접근이 가능합니다.



Complete Guide About
React Router Dom v6

DEVARTICLES.COM

React Router의 역할



SPA 내 원활한 페이지 전환

React Router는 페이지 새로고침 없이도 URL 변경에 따른 컴포넌트 전환을 구현하여 부드러운 사용자 경험을 제공합니다.



URL과 컴포넌트의 동기화

사용자가 접근하는 URL 경로와 화면에 보여지는 컴포넌트를 1:1로 매핑하는 클라이언트 사이드 라우팅을 담당합니다.



핵심 구성 요소 및 활용

BrowserRouter를 루트에 설정하고, Routes 내 Route 컴포넌트를 이용하여 각 경로를 정의하며, useParams 훅을 통해 동적 URL 파라미터에 접근합니다.

React Router의 동작 구조

BrowserRouter

HTML5의 history API를 활용하여 브라우저의 주소 변화를 감지하고 이를 기반으로 탐색 상태를 관리합니다. 이를 통해 페이지 새로고침 없이 URL 변경이 가능합니다.

내부 구조

React Context를 통해 라우팅 정보를 전역으로 제공하며, location listener가 URL 변화를 감지해 상태를 동기화 합니다. 이로 인해 효율적이고 반응성이 높은 라우팅 시스템이 구현됩니다.



Routes

현재 URL과 매칭되는 가장 첫 번째 Route 컴포넌트를 찾아 렌더링합니다. 중첩 라우팅을 지원하여 복잡한 UI 구성을 간편하게 처리할 수 있습니다.

useParams

동적 경로의 URL 파라미터를 추출하여 컴포넌트 내에서 손쉽게 접근하고 활용할 수 있도록 합니다. 예를 들어, 사용자 ID나 게시물 ID 같은 값들을 다룰 때 유용합니다.

중첩 라우팅과 Layout 구성



레이아웃 공유 및 재사용

공통 레이아웃 컴포넌트를 통해 여러 하위 페이지가 같은 UI 구조를 사용하도록 구성하여 코드 중복을 줄이고 유지보수를 용이하게 합니다.



React Router의 Outlet 활용

상위 라우트의 레이아웃 내에 **Outlet** 컴포넌트를 배치하여 하위 라우트 컴포넌트를 동적으로 렌더링하는 방식으로 화면을 전환합니다.



실무 적용 사례

대시보드, 마이페이지 같은 복잡한 UI 구조에서 중첩 라우팅을 통해 역할에 맞는 동적 콘텐츠 관리를 구현하며, 코드 분할과 최적화를 쉽게 수행할 수 있습니다.

<Modal />

<Toast />

코드 분할 전략 (Route-Based Splitting)

lazy()와 Suspense 활용한 동적 로딩

React의 lazy() 함수와 Suspense 컴포넌트를 사용해 각 라우트별로 별도의 번들로 코드를 분할하고 필요한 시점에만 로드합니다.

초기 렌더링 최적화

초기 페이지 로드 시 필수 코드만 빠르게 로드하여 사용자 경험을 개선하고, 나머지 라우트 코드는 사용자가 해당 경로로 이동할 때 비동기적으로 로드합니다.

라우팅과 코드 분할의 실무 통합

실제 프로젝트에서는 React Router와 코드 분할을 함께 적용하여, 중첩 라우트 및 레이아웃 구성 시 효율적인 코드 관리와 성능 향상을 도모합니다.

상태 관리와 라우팅의 통합 설계

권한별 뷰 분기



사용자의 권한 수준에 따라 인터페이스를 동적으로 분기하여 적합한 뷰를 제공합니다. 이를 통해 보안 및 사용자 경험을 향상시키고, 불필요한 UI 노출을 방지할 수 있습니다.

예를 들어, 관리자, 일반 사용자, 게스트 각각에게 맞는 페이지 접근 권한과 콘텐츠가 적용되어야 합니다.

라우팅 가드



특정 조건에 따라 접근을 제어하고, 권한이 없는 사용자는 적절한 페이지로 리디렉션합니다. 라우팅 가드는 사용자의 인증 상태와 역할에 기반하여 SPA 내에서 안전한 네비게이션을 보장합니다.

또한 비인가 접근 시 로그인 페이지로 이동시키거나, 오류 페이지로 안내하는 등의 UX 설계가 포함됩니다.

인증 정보 활용



전역 상태 관리 시스템에서 사용자 인증 정보를 효과적으로 조회 및 관리하여 컴포넌트 간 일관된 접근 제어를 지원합니다. 이를 통해 인증 상태가 바뀔 때마다 즉시 UI가 반영됩니다.

토큰 저장, 갱신 로직과 함께 메모리와 로컬스토리지 활용 등 다양한 인증 정보 저장 전략을 포함합니다.

정리 및 실무 팁



Context API

작고 단순한 상태에 적합합니다.



Redux

복잡한 비즈니스 로직에 유용합니다.



Zustand

빠르게 구성해야 하는 앱에 탁월합니다.



React Router

단순 페이지 전환 이상의 앱 구조 설계 도구입니다.

