React_20220112

요약

1. **Context 와 Redux 는 같은 일을 하나**둘은 다른 도구이며, 다른 목적을 가지고 있다.
2. **Context 는 상태관리 도구인가 ?**아니다. Context API 는 단지 종속성 주입의 한 형태일뿐 아무것도 관리하지 않는다. 상태관리는 일반적으로 `useState` 와 `useReducer` 를 통해 일어난다.
3. **useReducer 는 Redux 의 대체품인가 ?**그렇지 않다. 유사한 부분들이 있지만 기능에는 큰 차이가 존재한다.
4. **언제 Context 를 사용해야하나**props drilling 을 피하고자 할 때
5. **언제 Context 와 useReducer 를 사용해야 하나**특정 컴포넌트에서 어느정도 복잡한 상태 관리가 필요한경우
6. **Redux 는 언제 사용해야하나**여러 위치에 많은 양의 상태 값이 존재 할 때업데이트 로직이 복잡 할 때거대한 코드 베이스를 여러 사람이 작업 할 때상태 변경 시각화가 필요 할 때사이드이펙트, 메모이제이션, 데이터 직렬화등 관리를 위해 더 강력한 기능이 필요 할 때

React Context 는 무엇인가 ?

일반적인 React 에서 Prop 전달의 흐름은 하향식이다. (부모에서 자식으로)전달되는 prop 이 많아지고 컴포넌트 깊이가 깊어지면 관리가 번거로워 질 수 있다. `Context Tree` 는 컨텍스트 안에 포함된 모든 레벨에서 명시적으로 prop 을 전달하지 않고, 어디서든 상태값에 접근 할 수 있는 방법을 제공한다.

목적 및 사례

Context 는 실제로 아무것도 관리하지 않는다. 단순 값을 전달하는 파이프와 같다. 사용하는 주요 목적은 `prop-drilling` 을 피하는 것이다.prop-passing 로직을 작성할 필요가 없기 때문에 코드가 단순해진다.

개념적으로는 종속성 주입의 한 형태이다. 자식 구성 요소에 특정한 상태값이 필요하다는 것은 알고 있지만 값 자체를 생성하거나 설정하려 하지 않는다. 대신 상위 요소가 런타임에 해당 값을 전달한다고 가정한다.

Redux 는 무엇인가 ?

애플리케이션 전체에 대한 상태 중앙 저장소 역할을 하며 액션 이라는 이벤트를 사용하여 애플리케이션의 상태를 예측 가능한 방식으로 업데이트 하기 위한 패턴 또는 라이브러리이다.

Redux 에서 제공하는 패턴과 도구들을 사용하면 상태가 언제, 어디서, 어떻게, 왜 업데이트 되었는지 쉽게 이해 할 수 있다

- 상태 관리
- Redux 의 목적은 시간이 지남에 따라 상태 값들이 어떻게 변하는지 쉽게 이해 할 수 있도록 돕는 것이다.

reducer 함수를 이용하여 상태 변경을 예측 가능하게 만들고, 사이드이펙트 및 저장소 확장 개념으로 미들웨어를 사용한다. 또한 Redux Devtools 를 이용하여 앱의 작업 기록 및 상태 확인 할 수 있다.

Redux 와 React

Redux 자체는 ui 에 구애받지 않는다. React, Vue, vanilla JS 등과 함께 사용 가능하다.

React 에서 Redux 를 사용할 때 사용되는 React-Redux 라이브러리는 Redux 에서 상태 값을 읽고 action 을 React 컴포넌트에게 전달하여 Redux 저장소와 상호 작용 할 수 있도록 도와주는 UI 바인딩 레이어이다.

React-Redux 를 사용하면 애플리케이션의 모든 React 요소들이 Redux 저장소에 접근이 가능한데 이는 React-Redux 내부에서 Context 를 사용하기 때문이다. 여기서 주의할 점은 React-Redux 는 현재의 상태값이 아닌 Context 를 통해 Redux 저장소 인스턴스만 전달한다는점이다.

Redux 저장소는 React-Redux 요소를 사용하여 런타임에 Context Tree 에 삽입된다.

(React-) Redux 의 목적 및 사례

- ui 레이어와 분리된 상태 관리 로직 작성이 필요할 때
- 서로 다른 ui 계층간에 상태 공유가 필요할 때
- Redux 미들웨어 기능을 빌려 액션이 전달 될 때 추가적인 로직이 필요할 때
- Redux 상태의 유지
- dev tool 을 이용한 버그 디버깅

Context가 '상태관리'가 아닌 이유

상태관리는 시간이 지남에 따라 상태가 변경되는 방식을 의미한다.

아래와 같은 경우를 상태관리라 한다.

- 초기 값을 저장한다.
- 현재 값을 읽을 수 있다.
- 값 업데이트가 가능하다.

React `useState` 와 `useReducer` 가 상태 관리의 좋은 예이다.

- hook 을 호출하여 초기 값 저장
- hook 을 호출하여 현재 값을 읽는다
- 제공된 setState, dispatch 함수를 호출하여 값을 업데이트한다.
- 구성 요소가 re-render 되었기 때문에 값이 업데이트 됐음을 확인 할 수 있다.
- 마찬가지로 Redux, Mobx 도 위의 조건들을 충족하기 때문에 상태관리라 할 수 있다.
- React-Query, SWR, Apollo 및 Urql 과 같은 서버 캐싱 도구들은 가져온 데이터를 기반으로 초기 값을 설정하고 hook 들을 통해 현재 값을 반환하며 '서버 변화' 를 확인하여 구성요소를 다시 렌더링하기 때문에 '상태관리' 라고 정의 할 수 있다.

React Context 의 경우 위의 조건을 충족하지 않기 때문에 상태관리 도구라고 할 수 없다. Context 는 전달되는 값을 결정하는 역할을 하고 일반적으로 실제 상태관리는 useState/useReducer hook 과 함께 발생한다.

> 정리하면 다음과 같다. Context 는 상태(이미 존재하는, 어딘가에 있는)가 Context Tree 내부에 포함된 다른 컴포넌트들과 공유되는 방식이다.

Context 와 useReducer

useReducer 는 Redux + React-Redux 와 비슷한 구조를 가지고 있다.

- 값의 저장
- reducer 함수
- action 전달
- 값을 전달하고 컴포넌트에서 읽는 방법

그러나 기능과 동작에는 많은 차이점이 존재한다.

- Context + useReder 는 Context 를 통해 현재 상태 값을 전달하는데 의존한다. React-Redux 는 Context 를 통해 현재 Redux 스토어 인스턴스를 전달한다.
- useReducer 의 경우 새로운 상태 값을 생성 할 때 해당 Context 내부에 포함된 컴포넌트들이 상태값의 일부에만 관심이 있더라도 강제로 re-render 되기 때문에 성능 문제가 발생 할 수 있다. React-Redux 를 사용하면 저장소 상태의 특정 부분만 사용하고 해당 값이 변경 될 때만 re-render 할 수 있다.
- Context + useReducer 는 React 의 기능이기 때문에 React 외부에서는 사용이 불가하다. Redux 는 UI 독립적이기 때문에 React 와 별도로 사용이 가능하다.
- React DevTools 를 사용하면 현재의 상태 값은 볼 수 있지만 전달된 action, 과 payload, 처리 된 후의 상태등 시간에 따른 변화를 볼 수 없다. Redux Devtools 을 이용하면 시간에 따른 상태 차이를 볼 수 있다.
- useReducer 는 미들웨어가 없다.

> Context API + useReducer 는 낮은 규모와 빈도의 업데이트와 같은 정적인 상태의 전달에는 괜찮지만, Flux 와 유사한 상태 전파의 대체물로는 부족하다.

불필요한 re-render 를 줄이고 문제의 범위를 지정하기 위해 개별 컨텍스트를 설정하고 React.memo, useMemo 등을 이용하여 코드를 작성하길 권장하지만, React-Redux 를 재창조하는 행위와 같다.

권장사항

해결하려는 문제에 가장 적합한 도구를 선택해라

- 단순 prop-drilling 을 피하는 것이 목적이라면 Context 를 사용해라
- 적당히 복잡한 컴포넌트가 있거나 외부 라이브러리를 사용하고 싶지 않다면 Context + useReducer 를 사용해라

- 특정 구성 요소만 re-render 시키거나, 사이드이펙트를 줄이기 위해 더 강력한 기능이 필요하다면 Redux + React-Redux 를 사용해라

Ref.

Context API vs Redux

일반적인 React 에서 Prop 전달의 흐름은 하향식이다. (부모에서 자식으로) 전달되는 prop 이 많아지고 컴포넌트 깊이가 깊어지면 관리가 번거로워 질 수 있다. Context Tree 는 컨텍스트 안에 포함된 모든 레벨에서 명시적으로 prop 을 전달하지 않고, 어디서든 상태값에 접근 할 수 있는 방법을 제공한다. Context 는 실제로 아무것도...

https://olaf-go.medium.com/context-api-vs-redux-e8a53df99b8

Blogged Answers: Why React Context is Not a "State Management" Tool (and Why It Doesn't Replace Redux)

Definitive answers and clarification on the purpose and use cases for Context and Redux "Context vs Redux" has been one of the most widely debated topics within the React community ever since the current React Context API was released. Sadly, most of this "debate" stems from confusion over the purpose and use cases for these two tools.

https://blog.isquaredsoftware.com/2021/01/context-redux-differences/