useOptimistic

`useOptimistic` 는 UI를 낙관적으로 업데이트할 수 있게 해주는 React Hook입니다.

```
const [optimisticState, addOptimistic] = useOptimistic(state, updateFn)
```

레퍼런스

`useOptimistic(state, updateFn)`

`useOptimistic` 은 React Hook으로, 비동기 작업이 진행 중일 때 다른 상태를 보여줄 수 있게 해줍니다. 인자로 주어진 일부 상태를 받아, 네트워크 요청과 같은 비동기 작업 기간 동안 달라질 수 있는 그 상태의 복사본을 반환합니다. 현재 상태와 작업의 입력을 취하는 함수를 제공하고, 작업이 대기 중일 때 사용할 낙관적인 상태를 반환합니다.

이 상태는 "낙관적" 상태라고 불리는데, 실제로 작업을 완료하는 데 시간이 걸리더라도 사용자에게 즉시 작업의 결과를 표시하기 위해 일반적으로 사용됩니다.

```
import { useOptimistic } from 'react';

function AppContainer() {
  const [optimisticState, addOptimistic] = useOptimistic(
```

```
    state,
    // updateFn
    (currentState, optimisticValue) => {
      // merge and return new state
      // with optimistic value
    }
  );
}
```

아래에 더 많은 예시를 참조하세요.

매개변수

- `state` : 작업이 대기 중이지 않을 때 초기에 반환될 값입니다.
- `updateFn(currentState, optimisticValue)` : 현재 상태와 addOptimistic에 전달된 낙관적인 값을 취하는 함수로, 결과적인 낙관적인 상태를 반환합니다. 순수 함수여야 합니다. `updateFn` 은 두 개의 매개변수를 취합니다. `currentState` 와 `optimisticValue` . 반환 값은 `currentState` 와 `optimisticValue` 의 병합된 값입니다.

반환값

- `optimisticState` : 결과적인 낙관적인 상태입니다. 작업이 대기 중이지 않을 때는 `state` 와 동일하며, 그렇지 않은 경우 `updateFn` 에서 반환된 값과 동일합니다.
- `addOptimistic` : `addOptimistic` 는 낙관적인 업데이트가 있을 때 호출하는 dispatch 함수입니다. 어떠한 타입의 `optimisticValue` 라는 하나의 인자를 취하며, `state` 와 `optimisticValue` 로 `updateFn` 을 호출합니다.

사용법

폼을 낙관적으로 업데이트하기

`useOptimistic` Hook은 네트워크 요청과 같은 백그라운드 작업이 완료되기 전에 사용자 인터페이스를 낙관적으로 업데이트하는 방법을 제공합니다. 폼의 맥락에서, 이 기술은 앱이 더 반응적으로 느껴지도록 도와줍니다. 사용자가 폼을 제출할 때, 서버의 응답을 기다리는 대신 인터페이스는 기대하는 결과로 즉시 업데이트됩니다.

예를 들어, 사용자가 폼에 메시지를 입력하고 “전송” 버튼을 누르면, `useOptimistic` Hook은 메시지가 실제로 서버로 전송되기 전에 “전송 중...” 라벨이 있는 목록에 메시지가 즉시 나타나도록

합니다. 이 “낙관적” 접근법은 속도와 반응성의 느낌을 줍니다. 그런 다음 폼은 백그라운드에서 메시지를 실제로 전송하려고 시도합니다. 서버가 메시지를 받았음을 확인하면, “전송 중...” 라벨이 제거됩니다.

App.js actions.js 새로고침 Clear 포크

```
import { useOptimistic, useState, useRef, startTransition } from "react";
import { deliverMessage } from "./actions.js";

function Thread({ messages, sendMessageAction }) {
  const formRef = useRef();
  function formAction(formData) {
    addOptimisticMessage(formData.get("message"));
    formRef.current.reset();
    startTransition(async () => {
      await sendMessageAction(formData);
    });
  }
```

자세히 보기

Hello! Send
Hello there!

useMemo

useReducer

React 학습하기

빠르게 시작하기

설치하기

UI 표현하기

상호작용성 더하기

State 관리하기

탈출구

API 참고서

React APIs

React DOM APIs

커뮤니티

행동 강령

팀 소개

문서 기여자

감사의 말

더 보기

블로그

React Native

개인 정보 보호

약관