useFormStatus

`useFormStatus` 는 마지막 폼 제출의 상태 정보를 제공하는 Hook입니다.

```
const { pending, data, method, action } = useFormStatus();
```


- 레퍼런스
  - `useFormStatus()`
- 사용법
  - 폼을 제출하는 동안 대기 중인 상태로 표시하기
  - 제출한 폼 데이터 읽기
- 문제 해결
  - `status.pending` 이 절대로 `true` 가 되지 않습니다

레퍼런스

useFormStatus()

`useFormStatus` Hook은 마지막 폼 제출의 상태 정보를 제공합니다.

```
import { useFormStatus } from "react-dom";
import action from './actions';

function Submit() {
  const status = useFormStatus();
  return <button disabled={status.pending}>Submit</button>
}
```

```
export default function App() {
  return (
    <form action={action}>
      <Submit />
    </form>
  );
}
```

상태 정보를 제공받기 위해 `Submit` 컴포넌트를 `<form>` 내부에 렌더링해야 합니다. 이 Hook은 폼이 현재 제출하고 있는 상태인지를 의미하는 `pending` 프로퍼티와 같은 상태 정보를 반환합니다.

위의 예시에서 `Submit` 컴포넌트는 폼이 제출 중일 때 `<button>` 을 누를 수 없도록 하기 위해 이 정보를 활용합니다.

아래 예시를 참고하세요.

매개변수

`useFormStatus` 은 어떤 매개변수도 받지 않습니다.

반환값

다음의 프로퍼티를 가지는 `status` 객체를 반환합니다.

- `pending`: 불리언 값입니다. `true` 라면 상위 `<form>` 이 아직 제출 중이라는 것을 의미합니다. 그렇지 않으면 `false` 입니다.
- `data`: `FormData` 인터페이스를 구현한 객체로, 상위 `<form>` 이 제출하는 데이터를 포함합니다. 활성화된 제출이 없거나 상위에 `<form>` 이 없는 경우에는 `null` 입니다.
- `method`: `'get'` 또는 `'post'` 중 하나의 문자열 값입니다. 이 프로퍼티는 상위 `<form>` 이 `GET` 또는 `POST` HTTP 메서드를 사용하여 제출되는지를 나타냅니다. 기본적으로 `<form>` 은 `GET` 메서드를 사용하며 `method` 프로퍼티를 통해 지정할 수 있습니다.
- `action`: 상위 `<form>` 의 `action` Prop에 전달한 함수의 레퍼런스입니다. 상위 `<form>` 이 없는 경우에는 이 프로퍼티는 `null` 입니다. `action` Prop에 URI 값이 제공되었거나 `action` prop를 지정하지 않았을 경우에는 `status.action` 은 `null` 입니다.

주의 사항

- `useFormStatus` Hook은 `<form>` 내부에 렌더링한 컴포넌트에서 호출해야 합니다.
- `useFormStatus` 는 오직 상위 `<form>` 에 대한 상태 정보만 반환합니다. 동일한 컴포넌트나 자식 컴포넌트에서 렌더링한 `<form>` 의 상태 정보는 반환하지 않습니다.

사용법

폼을 제출하는 동안 대기 중인 상태로 표시하기

폼을 제출하는 동안 대기Pending 상태를 표시하려면 `<form>` 내에서 렌더링한 컴포넌트에서 `useFormStatus` Hook을 호출하고 반환된 `pending` 프로퍼티를 확인하세요.

여기서는 `pending` 프로퍼티를 사용하여 폼이 제출 중인지를 나타냅니다.

App.js 새로고침 Clear 포크

```
import { useFormStatus } from "react-dom";
import { submitForm } from "./actions.js";

function Submit() {
  const { pending } = useFormStatus();
  return (
    <button type="submit" disabled={pending}>
      {pending ? "Submitting..." : "Submit"}
    </button>
  );
}
```

자세히 보기

주의하세요!

`useFormStatus` 는 동일한 컴포넌트에서 렌더링한 `<form>` 에 대한 상태 정보를 반환하지 않습니다.

`useFormStatus` Hook은 상위 `<form>` 에 대한 정보만 반환합니다. Hook을 호출하는 동일한 컴포넌트나 자식 컴포넌트에서 렌더링한 `<form>` 의 상태 정보는 반환하지 않습니다.

```
function Form() {
  // 🚩 `pending`은 절대 true가 되지 않습니다
  // useFormStatus는 현재 컴포넌트에서 렌더링한 폼을 추적하지 않습니다
  const { pending } = useFormStatus();
  return <form action={submit}></form>;
}
```

대신 `<form>` 내부에 위치한 컴포넌트에서 `useFormStatus` 를 호출합니다.

```
function Submit() {
  // ✅ `pending`은 Submit 컴포넌트를 감싸는 폼에서 파생됩니다
  const { pending } = useFormStatus();
  return <button disabled={pending}>...</button>;
}

function Form() {
  // `useFormStatus`가 추적하는 <form>입니다
  return (
    <form action={submit}>
      <Submit />
```

```
      </form>
    );
}
```

제출한 폼 데이터 읽기

`useFormStatus` 에서 반환된 상태 정보의 `data` 프로퍼티를 사용하여 사용자가 제출한 데이터를 표시할 수 있습니다.

여기에 사용자가 이름을 요청할 수 있는 폼이 있습니다. `useFormStatus` 를 사용하여 사용자가 요청한 사용자 이름을 확인하는 임시 상태 메시지를 표시할 수 있습니다.

UsernameForm.js App.js 새로고침 Clear 포크

```
import {useState, useMemo, useRef} from 'react';
import {useFormStatus} from 'react-dom';

export default function UsernameForm() {
  const {pending, data} = useFormStatus();

  return (
    <div>
      <h3>Request a Username: </h3>
      <input type="text" name="username" disabled={pending}/>
      <button type="submit" disabled={pending}>
        Submit
```

자세히 보기

문제 해결

`status.pending`이 절대로 `true`가 되지 않습니다

`useFormStatus`는 오직 상위 `<form>`에 대한 상태 정보만 반환합니다.

`useFormStatus`를 호출하는 컴포넌트가 `<form>`에 감싸져 있지 않다면, `status.pending`은 항상 `false`를 반환합니다. `useFormStatus`가 `<form>` 엘리먼트의 자식 컴포넌트에서 호출되는지 확인하세요.

`useFormStatus`는 동일한 컴포넌트에서 렌더링한 `<form>`의 상태를 추적하지 않습니다. 자세한 내용은 '주의하세요!'에서 확인할 수 있습니다.

이전
Hook

다음
컴포넌트

State 관리하기

탈출구

커뮤니티

행동 강령

팀 소개

문서 기여자

감사의 말

더 보기

블로그

React Native

개인 정보 보호

약관