useId

`useId`는 접근성 어트리뷰트에 전달할 수 있는 고유 ID를 생성하기 위한 React Hook입니다.

```
const id = useId()
```

레퍼런스

`useId()`

`useId`를 컴포넌트의 최상위에서 호출하여 고유 ID를 생성합니다.

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

function PasswordField() {
  const passwordHintId = useId();
  // ...
```

아래에서 더 많은 예시를 확인하세요.

매개변수

`useId` 는 어떤 매개변수도 받지 않습니다.

반환값

`useId` 를 호출한 특정 컴포넌트와 특정 `useId` 에 관련된 고유 ID 문자열을 반환합니다.

주의 사항

- `useId` 는 Hook이므로 **컴포넌트의 최상위** 또는 커스텀 Hook에서만 호출할 수 있습니다. 반복문이나 조건문에서는 사용할 수 없습니다. 필요한 경우 새로운 컴포넌트를 추출하고 해당 컴포넌트로 state를 이동해서 사용할 수 있습니다.
- `useId` 를 리스트의 **key를 생성하기 위해 사용하면 안 됩니다.** Key는 데이터로부터 생성해야 합니다.
- `useId` 는 현재 비동기 서버 컴포넌트에서 사용할 수 없습니다.

사용법

> **주의하세요!**
>
> **`useId` 를 리스트의 key를 생성하기 위해 사용하면 안 됩니다.** Key는 데이터로부터 생성해야 합니다.

접근성 어트리뷰트를 위한 고유 ID 생성하기

고유 ID를 생성하기 위해 `useId` 를 컴포넌트의 최상단에서 호출합니다.

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

function PasswordField() {
```

```
  const passwordHintId = useId();
  // ...
```

생성된 ID 를 다른 어트리뷰트로 전달할 수 있습니다.

```
<>
  <input type="password" aria-describedby={passwordHintId} />
  <p id={passwordHintId}>
</>
```

예시를 통해 유용한 상황에 대해 알아보겠습니다.

aria-describedby 와 같은 HTML 접근성 어트리뷰트를 사용하면 두 개의 태그가 서로 연관되어 있다는 것을 명시할 수 있습니다. 예를 들어 엘리먼트(input)를 다른 엘리먼트(paragraph)에서 설명하도록 명시할 수 있습니다.

HTML에서는 일반적으로 다음과 같이 작성합니다.

```
<label>
  Password:
  <input
    type="password"
    aria-describedby="password-hint"
  />
</label>
<p id="password-hint">
  The password should contain at least 18 characters
</p>
```

React에서 ID를 직접 코드에 입력하는 것은 좋은 사례가 아닙니다. 페이지에서 컴포넌트는 몇 번이고 렌더링 될 수 있지만 ID는 고유해야 합니다. ID를 직접 입력하는 대신 useId 를 활용해서 고유한 ID를 생성할 수 있습니다.

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

```
function PasswordField() {
  const passwordHintId = useId();
  return (
    <>
      <label>
        Password:
        <input
          type="password"
          aria-describedby={passwordHintId}
        />
      </label>
      <p id={passwordHintId}>
        The password should contain at least 18 characters
      </p>
    </>
  );
}
```

이제 `PasswordField` 가 화면에 여러 번 나타나도 생성된 ID는 충돌하지 않습니다.

App.js 다운로드 새로고침 Clear 포크

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

function PasswordField() {
  const passwordHintId = useId();
  return (
    <>
      <label>
        Password:
        <input
          type="password"
          aria-describedby={passwordHintId}
        />
```

자세히 보기

영상을 통해 보조 기술을 활용했을 때 사용자 경험의 차이점을 확인할 수 있습니다.

주의하세요!

서버 렌더링에서 **`useId` 는 서버와 클라이언트에서 동일한 컴포넌트 트리가 필요합니다.** 서버와 클라이언트에서 렌더링하는 트리가 정확히 일치하지 않으면 생성된 ID는 일치하지 않습니다.

자세히 살펴보기

useId를 사용하는 것이 카운터를 증가하는 것보다 나은 이유는 무엇일까요?

자세히 보기

여러 개의 연관된 엘리먼트의 ID 생성하기

여러 개의 연관된 엘리먼트에 ID를 전달하는 과정이 필요할 때 `useId` 를 사용해서 공유 접두사를 생성할 수 있습니다.

App.js 다운로드 새로고침 Clear 포크

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

export default function Form() {
  const id = useId();
  return (
    <form>
      <label htmlFor={id + '-firstName'}>First Name:</label>
      <input id={id + '-firstName'} type="text" />
      <hr />
      <label htmlFor={id + '-lastName'}>Last Name:</label>
      <input id={id + '-lastName'} type="text" />
    </form>
```

`useId` 를 고유한 ID가 필요한 모든 엘리먼트에서 실행하는 것을 방지할 수 있습니다.

생성된 모든 ID에 대해 공유 접두사 지정하기

여러 개의 독립된 React 애플리케이션을 하나의 페이지에서 렌더링한다면 `identifierPrefix` 를 `createRoot` 또는 `hydrateRoot` 호출에 대한 옵션으로 전달합니다. `useId` 로 생성된 모든

식별자가 별개의 접두사로 시작하므로 서로 다른 두 개의 앱에서 생성된 ID가 충돌하지 않는 것을 보장합니다.

index.js index.html App.js 새로고침 Clear 포크

```
import { createRoot } from 'react-dom/client';
import App from './App.js';
import './styles.css';

const root1 = createRoot(document.getElementById('root1'), {
  identifierPrefix: 'my-first-app-'
});
root1.render(<App />);

const root2 = createRoot(document.getElementById('root2'), {
  identifierPrefix: 'my-second-app-'
});
```

클라이언트와 서버에서 동일한 ID 접두사 사용하기

동일한 페이지에서 여러 독립적인 React 앱을 렌더링하는 경우, 이러한 앱 중 일부가 서버에서 렌더링되는 경우, 클라이언트 측에서 `hydrateRoot` 호출에 전달하는 `identifierPrefix`가

`renderToPipeableStream` 와 같은 서버 API에 전달하는 `identifierPrefix` 와 동일한지 확인해야 합니다.

```
// Server
import { renderToPipeableStream } from 'react-dom/server';

const { pipe } = renderToPipeableStream(
  <App />,
  { identifierPrefix: 'react-app1' }
);
```

```
// Client
import { hydrateRoot } from 'react-dom/client';

const domNode = document.getElementById('root');
const root = hydrateRoot(
  domNode,
  reactNode,
  { identifierPrefix: 'react-app1' }
);
```

페이지에 React 앱이 하나만 있는 경우에는 `identifierPrefix` 를 전달할 필요가 없습니다.

이전
useEffectEvent

다음
useImperativeHandle

커뮤니티

더 보기