experimental_taintUniqueValue

실험적 기능

이 API는 실험적이며 React 안정 버전에서는 아직 사용할 수 없습니다.

이 API를 사용하려면 React 패키지를 가장 최신의 실험적인 버전으로 업그레이드해야 합니다.

- `react@experimental`
- `react-dom@experimental`
- `eslint-plugin-react-hooks@experimental`

실험적인 버전의 React에는 버그가 있을 수 있습니다. 프로덕션에서는 사용하지 마세요.

이 API는 React 서버 컴포넌트에서만 사용할 수 있습니다.

`taintUniqueValue`를 사용하면 패스워드, 키 또는 토큰과 같은 고유 값을 클라이언트 컴포넌트로 전송하는 것을 방지할 수 있습니다.

```
taintUniqueValue(errMessage, lifetime, value)
```

민감한 데이터가 포함된 객체가 전달되는 것을 방지하는 방법은 `taintObjectReference`를 참고하세요.

- 토큰이 클라이언트 구성 요소로 전달되지 않도록 방지하기

레퍼런스

`taintUniqueValue(message, lifetime, value)`

클라이언트에 전달되지 않아야 할 패스워드, 토큰, 키, 해시를 `taintUniqueValue`와 함께 호출하여 React에 등록합니다.

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

experimental_taintUniqueValue(
  '시크릿 키를 클라이언트로 전달하지 마세요.',
  process,
  process.env.SECRET_KEY
);
```

아래 예시를 참고하세요.

매개변수

- `message`: 클라이언트 컴포넌트에 `value`가 전달될 경우 표시하고자 하는 메시지입니다. 이 메시지는 `value`가 클라이언트 컴포넌트에 전달될 경우 발생하는 오류의 일부로 표시됩니다.
- `lifetime`: `value`가 얼마나 오랫동안 오염Taint 상태를 유지해야 하는지를 나타내는 객체입니다. `value`는 이 객체가 존재하는 동안 클라이언트 컴포넌트로 전달되지 않도록 차단됩니다. 예를 들어 `globalThis`를 전달하면 앱이 종료될 때까지 값이 차단됩니다. `lifetime`은 일반적으로 `value`를 프로퍼티로 가지는 객체입니다.
- `value`: 문자열, bigint 또는 TypedArray입니다. `value`는 암호화 토큰, 개인 키, 해시, 긴 비밀번호와 같이 높은 엔트로피를 가진 고유한 문자 또는 바이트 시퀀스여야 합니다. `value`는 클라이언트 컴포넌트로 전송되지 않도록 차단됩니다.

반환값

`experimental_taintUniqueValue`는 `undefined`를 반환합니다.

주의 사항

- 오염된 값을 이용해서 새로운 값을 만들어 내면 오염 보호가 손상될 수 있습니다. 오염된 값을 대문자로 변경하거나, 다른 문자열과 연결하거나, Base64로 변환하거나, 잘라내는 등 기타 유사한 변환을 통해서 새롭게 생성된 값은 `taintUniqueValue` 을 명시적으로 호출하지 않으면 오염되지 않습니다.
- PIN 코드나 전화번호와 같이 복잡도가 낮은 값을 보호하기 위해 `tainUniqueValue` 를 사용하지 마세요. 공격자가 요청의 값을 이용하여 암호의 가능한 모든 값을 열거하여 어떤 값이 오염되었는지 추론할 수 있습니다.

사용법

토큰이 클라이언트 구성 요소로 전달되지 않도록 방지하기

패스워드, 세션 토큰 또는 기타 고유 값과 같은 민감한 정보가 실수로 클라이언트 컴포넌트로 전달되지 않도록 `taintUniqueValue` 함수는 보호 레이어을 제공합니다. 값이 오염되면 클라이언트 컴포넌트로 전달하려는 시도는 오류를 발생시킵니다.

`lifetime` 인자는 값이 오염된 상태로 남아 있는 기간을 정의합니다. 오염된 상태로 무기한 유지되어야 하는 값의 경우 `globalThis` 또는 `process` 와 같은 객체가 `lifetime` 인자로 사용될 수 있습니다. 이 객체들은 앱이 실행되는 전체 기간을 수명으로 가집니다.

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

experimental_taintUniqueValue(
  '패스워드를 클라이언트로 전달하지 마세요.',
  globalThis,
  process.env.SECRET_KEY
);
```

만약 오염된 값의 수명이 객체에 묶여 있다면, `lifetime` 은 그 값을 캡슐화하는 객체이어야 합니다. 이렇게 하면 오염된 값이 캡슐화 객체의 수명 동안 보호될 수 있습니다.

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

```
export async function getUser(id) {
  const user = await db`SELECT * FROM users WHERE id = ${id}`;
  experimental_taintUniqueValue(
    '세션 토큰을 클라이언트로 전달하지 마세요.',
    user,
    user.session.token
  );
  return user;
}
```

이 예시에서 `user` 객체는 `lifetime` 인수 역할을 합니다. 이 객체가 전역 캐시에 저장되거나 다른 요청에 의해 접근할 수 있다면 세션 토큰은 오염된 상태로 유지됩니다.

주의하세요!

보안을 오염에만 의존하지 마세요. 값을 오염시킨다고 해서 모든 파생 값의 누출이 방지되는 것은 아닙니다. 예를 들어 오염된 문자열을 대문자로 바꾸어 새로운 값을 만들면 오염되지 않은 새로운 값이 만들어집니다.

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

const password = 'correct horse battery staple';

experimental_taintUniqueValue(
  '패스워드를 클라이언트로 전달하지 마세요.',
  globalThis,
  password
);

const uppercasePassword = password.toUpperCase() // `uppercasePassword`는 오염되
```

이 예시에서는 상수 `password`가 오염되어 있습니다. 이러한 `password`에 `toUpperCase` 메서드를 사용하여 `uppercasePassword`라는 새로운 값을 만들었습니다. 이렇게 새로 생성된 `uppercasePassword`는 오염되지 않았습니다.

오염되지 않은 새로운 값이 만들어지는 다른 유사한 방법에는 오염된 값을 다른 문자열과 연결하거나, Base64로 변환하거나, 잘라내는 것이 있습니다.

오염은 비밀 값을 클라이언트에 전달하는 것과 같이 단순한 실수만 방지합니다. `lifetime` 객체 없이 React 외부의 전역 스토어를 사용하는 것과 같이 `taintUniqueValue`를 호출하는 실수는 오염된 값을 오염되지 않은 값으로 만들 수 있습니다. 오염은 보호 레이어이며 안전한 앱에는 여러 개의 보호 레이어와 잘 설계된 API, 격리 패턴이 있습니다.

자세히 살펴보기

비밀 누출 방지를 위해서 `server-only`와 `taintUniqueValue` 사용하기

자세히 보기

이전
experimental_taintObjectReference

커뮤니티

더 보기