cacheSignal

React 서버 컴포넌트

`cacheSignal` 은 현재 React 서버 컴포넌트에서만 사용할 수 있습니다.

`cacheSignal` 을 사용하면 `cache()` 수명이 언제 끝나는지 알 수 있습니다.

```
const signal = cacheSignal();
```

레퍼런스

`cacheSignal`

`cacheSignal` 을 호출하면 `AbortSignal` 을 얻을 수 있습니다.

```
import {cacheSignal} from 'react';
async function Component() {
  await fetch(url, { signal: cacheSignal() });
```

```
}
```

React가 렌더링을 완료하면 `AbortSignal` 이 중단됩니다. 이를 통해 더 이상 필요하지 않은 진행 중인 작업을 취소할 수 있습니다.
렌더링이 완료된 것으로 간주하는 경우는 다음과 같습니다.

- React가 성공적으로 렌더링을 완료한 경우
- 렌더링이 중단된 경우
- 렌더링이 실패한 경우

매개변수

이 함수는 매개변수를 받지 않습니다.

반환값

`cacheSignal` 은 렌더링 중에 호출되면 `AbortSignal` 을 반환합니다. 그 외의 경우에 `cacheSignal()` 은 `null` 을 반환합니다.

주의 사항

- `cacheSignal` 은 현재 React 서버 컴포넌트에서만 사용할 수 있습니다. 클라이언트 컴포넌트에서는 항상 `null` 을 반환합니다. 향후 클라이언트 캐시가 갱신되거나 무효화될 때 클라이언트 컴포넌트에서도 사용될 예정입니다. 클라이언트에서 항상 `null` 을 반환한다고 가정하면 안 됩니다.
- 렌더링 외부에서 호출하면 `cacheSignal` 은 `null` 을 반환하여 현재 스코프가 영원히 캐시되지 않음을 명확히 합니다.

사용법

진행 중인 요청 취소하기

`cacheSignal` 을 호출하여 진행 중인 요청을 중단할 수 있습니다.

```
import {cache, cacheSignal} from 'react';
const dedupedFetch = cache(fetch);
```

```
async function Component() {
  await dedupedFetch(url, { signal: cacheSignal() });
}
```

주의하세요!

아래의 예시처럼 렌더링 외부에서 시작된 비동기 작업을 `cacheSignal`로 중단할 수 없습니다.

```
import {cacheSignal} from 'react';
// 🚩 Pitfall: The request will not actually be aborted if the rendering of `C
const response = fetch(url, { signal: cacheSignal() });
async function Component() {
  await response;
}
```

React가 렌더링을 완료한 후 오류 무시하기

함수가 오류를 던지는 경우 취소로 인한 것일 수 있습니다. (예를 들어, 데이터베이스 연결이 닫힌 경우) `aborted` 속성을 사용하여 오류가 취소로 인한 것인지 실제 오류인지 확인할 수 있습니다. 취소로 인한 오류는 무시할 수 있습니다.

```
import {cacheSignal} from "react";
import {queryDatabase, logError} from "./database";

async function getData(id) {
  try {
    return await queryDatabase(id);
  } catch (x) {
    if (!cacheSignal()?.aborted) {
      // only log if it's a real error and not due to cancellation
      logError(x);
    }
    return null;
```

```
  }
}

async function Component({id}) {
  const data = await getData(id);
  if (data === null) {
    return <div>No data available</div>;
  }
  return <div>{data.name}</div>;
}
```


이전

cache

다음

captureOwnerStack

uwu?