useInsertionEffect

주의하세요!

`useInsertionEffect` 는 CSS-in-JS 라이브러리 작성자를 위한 것입니다. CSS-in-JS 라이브러리 작업 중에 스타일을 주입할 위치가 필요한 것이 아니라면, `useEffect` 또는 `useLayoutEffect` 를 사용하세요.

`useInsertionEffect` 는 layout Effects 가 실행되기 전에 전체 요소를 DOM 에 주입 할 수 있습니다.

```
useInsertionEffect(setup, dependencies?)
```


레퍼런스

`useInsertionEffect(setup, dependencies?)`

`useInsertionEffect` 를 호출하여 layout 을 읽어야하는 Effects 가 호출 되기 전에 스타일을 주입할 수 있습니다.

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

// CSS-in-JS 라이브러리 안에서
function useCSS(rule) {
  useInsertionEffect(() => {
    // ... <style> 태그를 여기에서 주입하세요 ...
  });
  return rule;
}
```

더 많은 예시 보기

매개변수

- `setup` : Effects 의 로직이 포함된 함수입니다. setup 함수는 선택적으로 *cleanup* 함수를 반환할 수도 있습니다. 컴포넌트가 DOM에 추가되기 전에, layout Effects 가 실행되기 전에, React는 setup 함수를 실행합니다. `dependencies` 가 변경되어 다시 렌더링할 때마다, React는 먼저 이전 값으로 cleanup 함수(제공한 경우)를 실행한 다음 새 값으로 setup 함수를 실행합니다. 컴포넌트가 DOM에서 제거되기 전에 React는 cleanup 함수를 한 번 더 실행합니다.

- **선택사항** `dependencies` : `setup` 코드 내에서 참조된 모든 반응형 값의 목록입니다. 반응형 값에는 props, state, 그리고 컴포넌트 본문에 직접 선언된 모든 변수와 함수가 포함됩니다. linter가 React용으로 설정된 경우, 모든 반응형 값이 의존성으로 올바르게 지정되었는지 확인합니다. 의존성 목록에는 일정한 수의 항목이 있어야 하며 `[dep1, dep2, dep3]` 와 같이 작성해야 합니다. React는 `Object.is` 비교 알고리즘을 사용하여 각 의존성을 이전 값과 비교합니다. 의존성을 전혀 지정하지 않으면 컴포넌트를 다시 렌더링할 때마다 Effect가 다시 실행됩니다.

반환값

`useInsertionEffect` 는 `undefined` 를 반환합니다.

주의 사항

- 이펙트는 클라이언트에서만 실행됩니다. 서버 렌더링 중에는 실행되지 않습니다.
- `useInsertionEffect` 내부에서는 상태를 업데이트할 수 없습니다.
- `useInsertionEffect` 가 실행되는 시점에 ref는 아직 연결되지 않습니다.
- `useInsertionEffect` 는 DOM 의 업데이트 전 또는 후에 실행됩니다. DOM 이 업데이트 되는 특정시점에 의존해서는 안됩니다.

- 매번 모든 cleanup 을 실행하고 setup 하는 다른 Effects 와 달리, `useInsertionEffect` 는 하나의 컴포넌트에 대해 cleanup 과 setup 을 모두 실행합니다. 그 결과 cleanup 과 setup 이 'interleaving' 됩니다.

사용법

CSS-in-JS 라이브러리에서 동적 스타일 주입하기

전통적으로 plain CSS를 사용해 React 컴포넌트의 스타일을 지정했습니다.

```
// JS 파일 안에서
<button className="success" />

// CSS 파일 안에서
.success { color: green; }
```

일부 팀은 CSS 파일을 작성하는 대신 자바스크립트 코드에서 직접 스타일을 작성하는 것을 선호합니다. 이 경우 일반적으로 CSS-in-JS 라이브러리 또는 도구를 사용해야 합니다. CSS-in-JS에는 세 가지 일반적인 접근 방식이 있습니다:

1. 컴파일러를 사용하여 CSS 파일로 정적 추출
2. 인라인 스타일, 예: `<div style={{ opacity: 1 }}>`
3. 런타임에 `<style>` 태그 주입

CSS-in-JS를 사용하는 경우 처음 두 가지 접근 방식(정적 스타일의 경우 CSS 파일, 동적 스타일의 경우 인라인 스타일)을 조합하여 사용하는 것이 좋습니다. **런타임 `<style>` 태그 주입은 다음 두 가지 이유로 권장하지 않습니다.**

1. 런타임 주입은 브라우저에서 스타일을 훨씬 더 자주 다시 계산하도록 합니다.
2. 런타임 주입이 React 생명주기 중에 잘못된 시점에 발생하면 속도가 매우 느려질 수 있습니다.

첫 번째 문제는 해결할 수 없지만 `useInsertionEffect` 를 사용하면 두 번째 문제를 해결할 수 있습니다.

`useInsertionEffect` 를 호출하여 layout Effects 가 발생하기 전에 스타일을 주입합니다:

```
// CSS-in-JS 라이브러리 안에서
let isInserted = new Set();
function useCSS(rule) {
  useInsertionEffect(() => {
    // 앞서 설명했듯이 <style> 태그의 런타임 주입은 권장하지 않습니다.
    // 하지만 꼭 주입해야 한다면 useInsertionEffect에서 주입하는 것이 중요합니다.
    if (!isInserted.has(rule)) {
      isInserted.add(rule);
      document.head.appendChild(getStyleForRule(rule));
    }
  });
  return rule;
}

function Button() {
  const className = useCSS('...');
  return <div className={className} />;
}
```

`useEffect`와 마찬가지로 `useInsertionEffect`는 서버에서 실행되지 않습니다. 서버에서 어떤 CSS 규칙이 사용되었는지 수집해야 하는 경우 렌더링 중에 수집할 수 있습니다:

```
let collectedRulesSet = new Set();

function useCSS(rule) {
  if (typeof window === 'undefined') {
    collectedRulesSet.add(rule);
  }
  useInsertionEffect(() => {
    // ...
  });
  return rule;
}
```

런타임 인젝션이 있는 CSS-in-JS 라이브러리를 `useInsertionEffect`로 업그레이드하는 방법에 대해 자세히 알아보세요.

자세히 살펴보기

이것이 렌더링 중에 스타일을 주입하거나 useLayoutEffect를 사용하는 것보다 어떻게 더 나은가요?

자세히 보기

이전
useImperativeHandle

다음
useLayoutEffect

Meta Open Source

Copyright © Meta Platforms, Inc

uwu?