flushSync

> **주의하세요!**
>
> `flushSync`를 사용하는 것은 일반적이지 않으며 애플리케이션의 성능을 저하할 수 있습니다.

`flushSync`는 React에 제공된 콜백 내부의 모든 업데이트를 동기적으로 처리하도록 강제합니다. DOM이 즉시 업데이트되는 것을 보장합니다.

```
flushSync(callback)
```

레퍼런스

`flushSync(callback)`

`flushSync` 를 호출해서 React가 보류Pending 중인 모든 작업을 강제로 처리하고 DOM을 동기적으로 업데이트할 수 있습니다.

```
import { flushSync } from 'react-dom';

flushSync(() => {
  setSomething(123);
});
```

대부분의 경우 `flushSync` 의 사용을 권장하지 않습니다. `flushSync` 는 최후의 수단으로 사용하세요.

아래 예시를 참고하세요.

매개변수

- `callback` : 함수입니다. React는 즉시 콜백을 호출하고 콜백 내의 모든 업데이트를 동기적으로 처리합니다. 또한 보류 중인 업데이트나 Effect 또는 Effect 내부의 업데이트도 처리할 수 있습니다. `flushSync` 호출로 인해 업데이트가 중단되면 Fallback이 다시 표시될 수 있습니다.

반환값

`flushSync` 는 `undefined` 를 반환합니다.

주의 사항

- `flushSync` 를 사용하면 애플리케이션의 성능이 크게 저하될 수 있습니다. 가급적 사용하지 마세요.
- `flushSync` 는 보류 중인 Suspense 바운더리의 `fallback` State를 표시하도록 강제할 수 있습니다.
- `flushSync` 는 보류 중인 Effect를 실행하고 포함된 모든 업데이트를 반환하기 전에 동기적으로 적용할 수 있습니다.
- `flushSync` 는 콜백 내부의 업데이트를 처리할 때 필요한 경우 콜백 외부의 업데이트를 처리할 수 있습니다. 예를 들어 클릭으로 인한 보류 중인 업데이트가 있는 경우 React는 콜백 내부의 업데이트를 처리하기 전에 해당 업데이트를 처리할 수 있습니다.

사용법

서드 파티 통합을 위한 업데이트 Flushing

브라우저 API 또는 UI 라이브러리와 같은 서드 파티 코드를 통합할 때 React가 업데이트를 처리하도록 강제할 필요가 있을 수 있습니다. `flushSync` 를 사용해서 React가 콜백 내부의 모든 State 업데이트 를 동기적으로 처리하도록 할 수 있습니다.

```
flushSync(() => {
  setSomething(123);
});
// By this line, the DOM is updated.
```

이렇게 함으로써 React가 DOM을 이미 업데이트한 후에 다음 줄의 코드를 실행하는 것을 보장합니다.

`flushSync` 를 사용하는 것은 일반적이지 않고 자주 사용하면 애플리케이션의 성능이 크게 저하될 수 있습니다. 애플리케이션이 React API만 사용하고 서드 파티 라이브러리와 통합하지 않는다면 `flushSync` 는 필요하지 않습니다.

그러나 브라우저 API와 같은 서드 파티 코드와 통합할 때 유용할 수 있습니다.

일부 브라우저 API는 콜백 내부의 결과가 DOM에서 동기적으로 사용될 것으로 예상하므로 콜백이 완료될 때까지 렌더링된 DOM을 사용해서 브라우저가 작업할 수 있습니다. 대부분의 경우 React가 이를 자동으로 처리하지만, 때에 따라 강제로 동기적 업데이트를 해야 할 수 있습니다.

예를 들어 `onbeforeprint` 브라우저 API를 사용하면 프린트 대화 상자가 열리기 직전에 페이지를 변경할 수 있습니다. 문서를 더 잘 표시하기 위해 사용자가 정의한 프린트 스타일을 적용하는 데 유용합니다. 아래 예시에서는 `onbeforeprint` 콜백 내에서 `flushSync` 를 사용하여 React State를 DOM으로 즉시 “Flush”합니다. 그런 다음 프린트 다이얼로그가 열릴 때까지 `isPrinting` 이 “yes”로 표시됩니다.

App.js

다운로드 새로고침 Clear 포크

```
import { useState, useEffect } from 'react';
import { flushSync } from 'react-dom';
```

```
export default function PrintApp() {
  const [isPrinting, setIsPrinting] = useState(false);

  useEffect(() => {
    function handleBeforePrint() {
      flushSync(() => {
        setIsPrinting(true);
      })
```

자세히 보기

`flushSync`를 사용하지 않으면 프린트 대화 상자는 `isPrinting`을 "no"로 표시합니다. React가 업데이트를 비동기적으로 Batch하고 프린트 대화 상자를 State가 업데이트되기 전에 표시하기 때문입니다.

주의하세요!

`flushSync`를 사용하면 애플리케이션의 성능이 크게 저하될 수 있으며 보류 중인 Suspense 바운더리가 Fallback State를 표시하도록 강제할 수 있습니다.

대부분의 경우 `flushSync`를 사용하지 않을 수 있으므로 최후의 수단으로 사용하세요.

Troubleshooting

I'm getting an error: "flushSync was called from inside a lifecycle method"

React cannot `flushSync` in the middle of a render. If you do, it will noop and warn:

Console

- `Warning: flushSync was called from inside a lifecycle method. React cannot flush when React is already rendering. Consider moving this call to a scheduler task or micro task.`

This includes calling `flushSync` inside:

- rendering a component.
- `useLayoutEffect` or `useEffect` hooks.
- Class component lifecycle methods.

For example, calling `flushSync` in an Effect will noop and warn:

```
import { useEffect } from 'react';
import { flushSync } from 'react-dom';

function MyComponent() {
  useEffect(() => {
    // 🚩 Wrong: calling flushSync inside an effect
    flushSync(() => {
      setSomething(newValue);
    });
  }, []);

  return <div>{/* ... */}</div>;
```

```
}
```

To fix this, you usually want to move the `flushSync` call to an event:

```
function handleClick() {
  // ✅ Correct: flushSync in event handlers is safe
  flushSync(() => {
    setSomething(newValue);
  });
}
```

If it's difficult to move to an event, you can defer `flushSync` in a microtask:

```
useEffect(() => {
  // ✅ Correct: defer flushSync to a microtask
  queueMicrotask(() => {
    flushSync(() => {
      setSomething(newValue);
    });
  });
}, []);
```

This will allow the current render to finish and schedule another syncronous render to flush the updates.

주의하세요!

`flushSync` can significantly hurt performance, but this particular pattern is even worse for performance. Exhaust all other options before calling `flushSync` in a microtask as an escape hatch.

이전
createPortal

다음
preconnect