act

`act` 는 테스트 헬퍼Helper로, 대기 중인 React 업데이트를 모두 적용한 뒤 단언Assert할 수 있게 도움을 줍니다.

```
await act(async actFn)
```

컴포넌트를 단언Assertion할 수 있도록 준비하려면 `await act()` 호출 안에 컴포넌트를 렌더링하고 업데이트하는 코드를 감싸세요. 이렇게 하면 테스트가 브라우저에서 작동하는 실제 React 방식과 더 유사하게 실행됩니다.

중요합니다!

`act()` 를 직접 사용하는 것이 다소 장황하다고 느껴질 수 있습니다. 반복되는 코드를 줄이고 싶다면 React Testing Library처럼 내부적으로 `act()` 로 감싼 헬퍼를 제공하는 라이브러리를 사용하는 것도 좋습니다.

레퍼런스

`await act(async actFn)`

UI 테스트를 작성할 때 렌더링, 사용자 이벤트, 데이터 가져오기 등은 사용자 인터페이스와의 상호작용 "단위"로 볼 수 있습니다. React는 `act()` 라는 헬퍼를 제공하는데 이는 이 "단위"와 관련된 모든 업데이트가 DOM에 적용되기 전까지 단언이 실행되지 않도록 보장해 줍니다.

`act` 라는 이름은 Arrange-Act-Assert 패턴에서 따온 것입니다.

```
it ('renders with button disabled', async () => {
  await act(async () => {
    root.render(<TestComponent />)
  });
  expect(container.querySelector('button')).toBeDisabled();
});
```

중요합니다!

`act` 는 `await` 와 `async` 함수와 함께 사용하는 것을 권장합니다. 동기 버전도 대부분의 경우 동작하지만 React가 내부적으로 업데이트를 예약하는 방식 때문에 언제 동기 버전을 써도 되는지 예측하기 어렵습니다.

앞으로 동기 버전은 더 이상 사용되지 않을 예정이며 제거될 예정입니다.

매개변수

- `async actFn` : 테스트할 컴포넌트를 렌더링하거나 상호작용을 수행하는 비동기 함수입니다. `actFn` 내부에서 발생하는 업데이트는 내부 act 큐에 추가되며 모두 모아서 DOM에 적용됩니다. 비동기 함수이기 때문에 React는 비동기 경계를 넘는 코드도 실행하고 예약된 업데이트도 함께 처리합니다.

반환값

`act` 는 아무 값도 반환하지 않습니다.

사용법

컴포넌트를 테스트할 때 `act` 를 사용하면 출력 결과에 대한 단언을 더 안전하게 할 수 있습니다.

예시로 `Counter` 라는 컴포넌트가 있다고 가정하고 아래 사용 예시는 이를 테스트하는 방법을 보여줍니다.

```
function Counter() {
  const [count, setCount] = useState(0);
  const handleClick = () => {
    setCount(prev => prev + 1);
  }

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  }, [count]);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={handleClick}>
        Click me
      </button>
    </div>
  )
}
```

테스트에서 컴포넌트를 렌더링하는 방법

컴포넌트의 렌더링 결과를 테스트하려면 렌더링 코드를 `act()` 로 감싸야 합니다.

```
import {act} from 'react';
import ReactDOMClient from 'react-dom/client';
import Counter from './Counter';

it('can render and update a counter', async () => {
```

```
  container = document.createElement('div');
  document.body.appendChild(container);

  // ✅ 컴포넌트를 act() 안에서 렌더링합니다.
  await act(() => {
    ReactDOMClient.createRoot(container).render(<Counter />);
  });

  const button = container.querySelector('button');
  const label = container.querySelector('p');
  expect(label.textContent).toBe('You clicked 0 times');
  expect(document.title).toBe('You clicked 0 times');
});
```

위 예시에서는 컨테이너를 만들고 문서에 추가한 뒤 `Counter` 컴포넌트를 `act()` 안에서 렌더링합니다. 이렇게 하면 컴포넌트가 렌더링되고 효과가 적용된 후에 단언을 수행할 수 있습니다.

`act` 를 사용하면 모든 업데이트가 적용된 뒤 단언을 실행할 수 있습니다.

테스트에서 이벤트 디스패칭하는 방법

이벤트를 테스트하려면 이벤트를 `act()` 로 감싸세요.

```
import {act} from 'react';
import ReactDOMClient from 'react-dom/client';
import Counter from './Counter';

it.only('can render and update a counter', async () => {
  const container = document.createElement('div');
  document.body.appendChild(container);

  await act( async () => {
    ReactDOMClient.createRoot(container).render(<Counter />);
  });

  // ✅ 이벤트 디스패치를 act() 안에서 실행합니다.
  await act(async () => {
    button.dispatchEvent(new MouseEvent('click', { bubbles: true }));
  });

  const button = container.querySelector('button');
```

```
  const label = container.querySelector('p');
  expect(label.textContent).toBe('You clicked 1 times');
  expect(document.title).toBe('You clicked 1 times');
});
```

위 예시에서는 컴포넌트를 먼저 `act` 로 감싸 렌더링하고, 이벤트 디스패치도 `act()` 로 감쌉니다. 이렇게 하면 해당 이벤트로 인한 모든 업데이트가 적용된 뒤 단언이 수행됩니다.

주의하세요!

DOM 이벤트를 디스패치할 때는 DOM 컨테이너가 문서에 추가되어 있어야 합니다. 반복되는 설정 코드를 줄이고 싶다면 React Testing Library를 사용하는 것도 고려해보세요.

문제 해결

“The current testing environment is not configured to support act(...)” 오류가 발생하는 경우

`act` 를 사용하려면 테스트 환경에서 `global.IS_REACT_ACT_ENVIRONMENT=true` 를 설정해야 합니다. 이 설정은 act가 올바른 환경에서만 사용되도록 보장합니다.

이 전역 설정이 없으면 다음과 같은 오류가 표시됩니다.

Console

```
Warning: The current testing environment is not configured to support act(…)
```

이 문제를 해결하려면 React 테스트를 위한 전역 설정 파일에 다음 코드를 추가하세요.

```
global.IS_REACT_ACT_ENVIRONMENT=true
```

중요합니다!

React Testing Library같은 테스트 프레임워크에서는 `IS_REACT_ACT_ENVIRONMENT` 가 이미 설정되어 있습니다.

이전
API

다음
addTransitionType

Meta Open Source

uwu?

React 학습하기

- 빠르게 시작하기
- 설치하기
- UI 표현하기
- 상호작용성 더하기
- State 관리하기
- 탈출구

API 참고서

- React APIs
- React DOM APIs

커뮤니티

- 행동 강령
- 팀 소개
- 문서 기여자
- 감사의 말

더 보기

- 블로그
- React Native
- 개인 정보 보호
- 약관