

스냅샷 테스트의 한계

실무에 바로 적용하는 프론트엔드 테스트

스냅샷 테스트의 가독성

```
import React from 'react';

import PageTitle from '@pages/cart/components/PageTitle';
import render from '@utils/test/render';

it('pageTitle 스냅샷 테스트', async () => {
  const { container } = await render(<PageTitle />);

  expect(container).toMatchInlineSnapshot(`
    <div>
      <h1
        class="MuiTypography-root MuiTypography-h4 css-1lnl64-MuiTypography-root"
      >
        상품 리스트
      </h1>
      <div
        style="position: fixed; z-index: 9999; top: 16px; left: 16px; right:
16px; bottom: 16px; pointer-events: none;"
      />
    </div>
  `);
});
```

일관되지 않은 스냅샷 업데이트

```
FAIL Tests failed. Watching for file changes...  
press u to update snapshot, press h to show help
```

u 입력 한 번이면 통과할 수 있는 테스트

스냅샷을 잘 관리하기 위해서는

1. 일반 코드처럼 관리해야 한다.
2. 스냅샷 단위를 작게 잡아야 한다. 가능하다면 no-large-snapshots eslint 규칙을 사용해 길이를 제한하자

Disallow large snapshots (`vitest/no-large-snapshots`) [↗](#)

⚠ This rule *warns* in the [all](#) config.

Rule Details [↗](#)


This rule aims to prevent large snapshots.

Options [↗](#)

This rule accepts an object with the following properties:

- `maxSize` (default: `50`): The maximum size of a snapshot.
- `inlineMaxSize` (default: `0`): The maximum size of a snapshot when it is inline.
- `allowedSnapshots` (default: `[]`): The list of allowed snapshots.

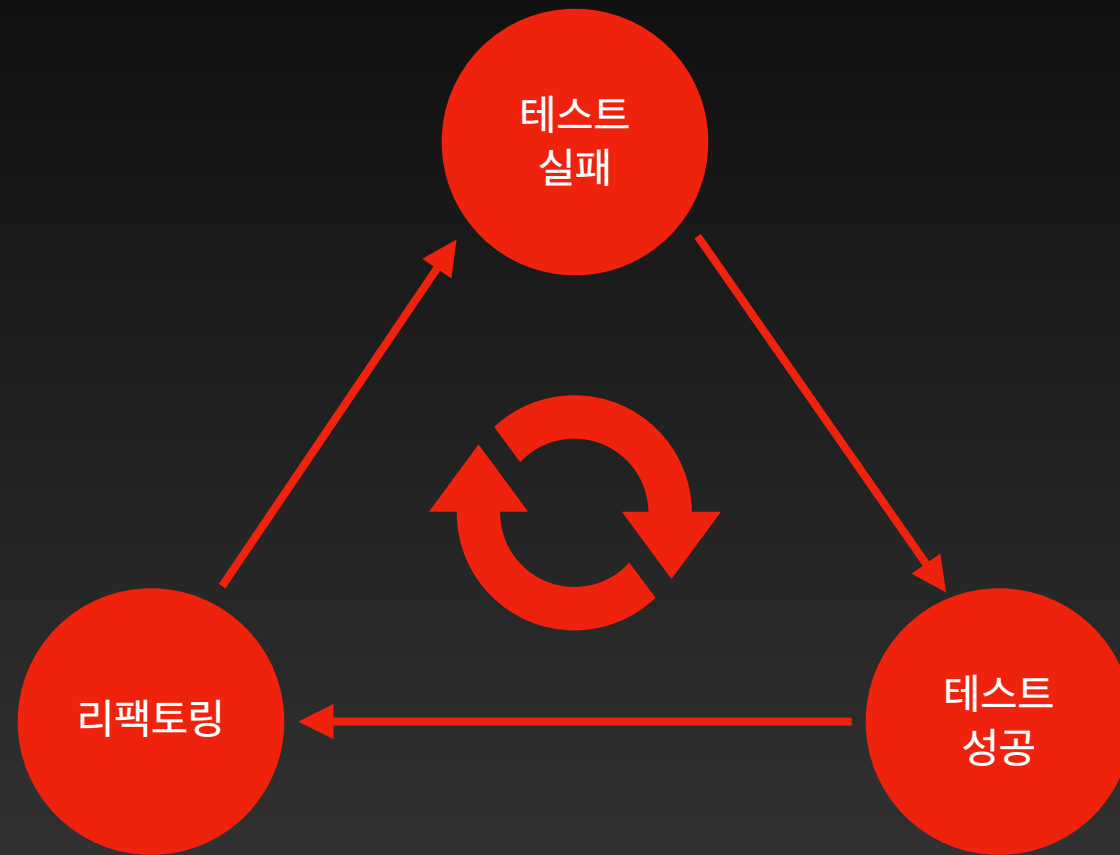
실제 UI 렌더링은 확인할 수 없다



```
/* 이 코드가 */  
.my-class1 {  
  width: 25px;  
  height: 25px;  
  border: 1px solid #ccc;  
}  
  
/* 이렇게 변경된다면? */  
.my-class1 {  
  width: 50px;  
  height: 50px;  
  border: 4px solid #ccc;  
}
```

UI상으로 변화가 있지만,
스냅샷 테스트는 변화를 알아차리지 못할 것

TDD를 적용할 수 없다



계속해서 스냅샷 업데이트가 필요하다

정리

스냅샷 테스트 관리는 어렵다

- 컴포넌트의 크기가 커지고 복잡할수록 스냅샷 결과는 가독성이 떨어진다.
- 개개인에 따라 무분별한 스냅샷 업데이트가 발생할 수 있다.

이런 문제를 해결하기 위해

- eslint의 no-large-snapshots와 같은 규칙을 사용해 스냅샷을 간결하게 유지하자

그럼에도..

- 여전히 스냅샷 업데이트는 쉽고, 잘못 업데이트될 가능성은 크다.
- 실제로 렌더링을 하는 것이 아니기 때문에 CSS의 변경 또는 UI상에서 어떤 변화가 있는지 정확하게 감지할 수 없다.
- TDD 사이클과는 맞지 않는다.