

React에서 불필요한 웹소켓 재연결을 줄이는 방법

최종찬





WebSocket?

- “요청 ↔ 응답” 쌍으로 한차례 정보를 교환하는 HTTP API와는 다르게, 연결돼있는 동안 클라이언트와 서버간에 메시지를 자유롭게 주고받을 수 있음
- 하나의 웹소켓 연결 안에서 여러가지 주제를 구독하도록 설계할 수 있음
- 웹소켓 연결 및 주제 구독의 수명(시작과 끝)을 관리할 필요가 있음

WebSocket 에서 오가는 내용

클라이언트

→ 시세 정보를 구독할래

→ 잔고 정보를 구독할래

→ 시세 정보 구독을 해지할래

→ 잔고 정보 구독을 해지할래

서버

시세 정보 1 ←

시세 정보 2 ←

잔고 정보 1 ←

시세 정보 3 ←

잔고 정보 2 ←

잔고 정보 3 ←

잔고 정보 4 ←

...

WebSocket 에서 오가는 내용

클라이언트

→ 시세 정보를 구독할래

서버

시세 정보 1 ←

시세 정보 2 ←

→ 잔고 정보를 구독할래

잔고 정보 1 ←

시세 정보 3 ←

→ 시세 정보 구독을 해지할래

잔고 정보 2 ←

잔고 정보 3 ←

잔고 정보 4 ←

→ 잔고 정보 구독을 해지할래

...

WebSocket 에서 오가는 내용

클라이언트

→ 시세 정보를 구독할래

→ 잔고 정보를 구독할래

→ 시세 정보 구독을 해지할래

→ 잔고 정보 구독을 해지할래

서버

시세 정보 1 ←

시세 정보 2 ←

잔고 정보 1 ←

시세 정보 3 ←

잔고 정보 2 ←

잔고 정보 3 ←

잔고 정보 4 ←

...

WebSocket 에서 오가는 내용

클라이언트

→ 시세 정보를 구독할래

→ 잔고 정보를 구독할래

→ 시세 정보 구독을 해지할래

→ 잔고 정보 구독을 해지할래

서버

시세 정보 1 ←

시세 정보 2 ←

잔고 정보 1 ←

시세 정보 3 ←

잔고 정보 2 ←

잔고 정보 3 ←

잔고 정보 4 ←

...

WebSocket 에서 오가는 내용

클라이언트

→ 시세 정보를 구독할래

→ 잔고 정보를 구독할래

→ 시세 정보 구독을 해지할래

→ 잔고 정보 구독을 해지할래

서버

시세 정보 1 ←

시세 정보 2 ←

잔고 정보 1 ←

시세 정보 3 ←

잔고 정보 2 ←

잔고 정보 3 ←

잔고 정보 4 ←

...

Market

All selections are USDT Perpetual [i](#)

Market trend (1h)

↗ 135 Coins are up

Top categories for you

New listings

 CETUS
Cetus Protocol

0.26569
↓ 7.14% [>](#)

[See all](#)

 COW
CoW Protocol

0.4802
↗ 6.18% [>](#)

 PONKE
Ponke

0.51415
↗ 0.14% [>](#)



Portfolio balance [i](#)

10.75 USDT +1.96%
≈ \$10.76

- Unrealized P&L
- Net funding fee
- Funds invested
- Funds reserved
- Available funds

Your positions In USDT

Funds invested

Open

 BTC [>](#)

0.00035 BTC [Short ↘](#) [5x](#)

+0.20 (+3.97%) [<](#)

Entry price

74,569.9

Margin (invested funds)

5.2

Mark price

73,949.2

Take Profit

-

Market

All selections are USDT Perpetual

Market trend (1h)

↗ 135 Coins are up

Top categories for you

New listings

CETUS
Cetus Protocol

0.26569
↓ 7.14%

COW
CoW Protocol

0.4802
↑ 6.18%

PONKE
Ponke

0.51415
↑ 0.14%



Portfolio balance

10.75 USDT +1.96%

= \$10.76

거래 페이지

Unrealized P&L

• 차트 정보

Funds invested

Funds reserved

Available funds

Your positions

In USDT

Funds invested

5.21

Open

BTC >

0.00035 BTC Short 5x

+0.20 (+3.97%) <

Entry price

74,569.9

Mark price

73,949.2

Margin (invested funds)

5.2

Take Profit

-

Market

All selections are USDT Perpetual [i](#)

Market trend (1h)

↗ 135 Coins are up

Top categories for you

New listings

 CETUS
Cetus Protocol

0.26569
↓ 7.14%

 COW
CoW Protocol

0.4802
↑ 6.18%

 PONKE
Ponke

0.51415
↑ 0.14%

5m 15m 1h 4h D ⏪ ⏩ MA15 RSI History Mid

CUSDT.PERP · 1 · Flipster
'4377.4 H74390.2 L74377.4 C74390.2 +0.8 (+0.00%)

2



See all

17

22:30 22:45 23:00 23:15 23:30

23:45

Your positions Open 1 Pending 0

Symbol Side / Lev. Qty / Value (USDT) Unrealized P Margin

BTC Short 5x 0.00035 +0.06 5.2

Full close

지갑 페이지

• 잔고 정보

Portfolio balance [i](#)

10.75 USDT +1.96%

≈ \$10.76

Unrealized P&L

Net funding fee

Funds invested

Funds reserved

Available funds

Your positions In USDT

Funds invested

5.21

Open

 BTC >

0.00035 BTC Short 5x

+0.20 (+3.97%) <

Entry price

Mark price

74,569.9

73,949.2

Margin (Invested funds)

Take Profit

5.2

-

상태와 뷰의 의존 관계

-  : 상태
-  : 뷰

Market trend (1h)

Top categories for you

New listings

상점 페이지

시세 영역

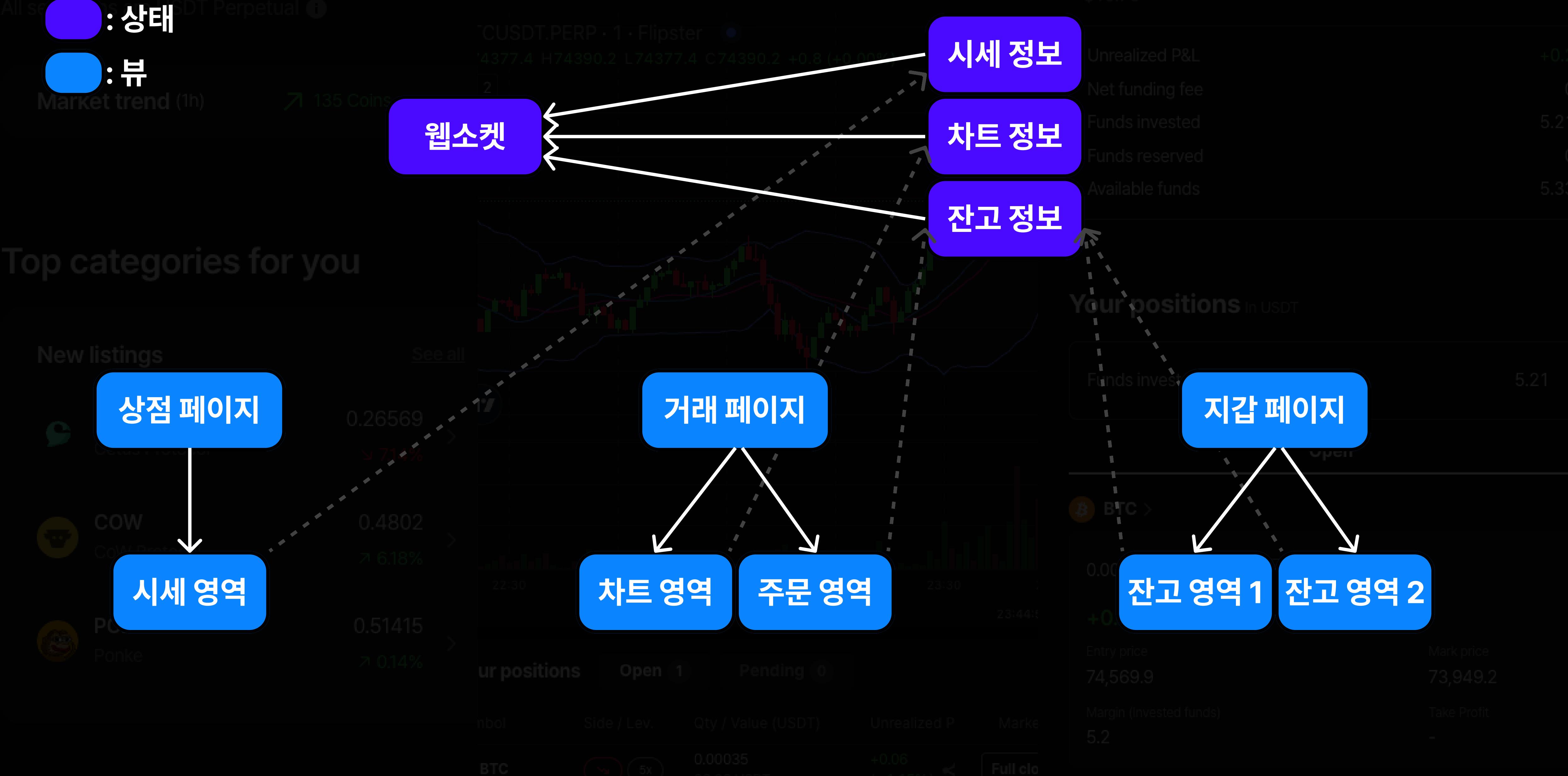
거래 페이지

차트 영역

주문 영역

지갑 페이지

잔고 영역 1 잔고 영역 2



기존 코드 (Naive한 구현)

: 상태

: 뷰

`function use웹소켓() { ... }`

웹소켓

시세 정보

`function use시세() { ... }`

차트 정보

`function use차트() { ... }`

잔고 정보

`function use잔고() { ... }`

`use웹소켓();`

상점 페이지

시세 영역

`use시세();`

`use웹소켓();`

거래 페이지

차트 영역

주문 영역

`use차트(); use잔고();`

`use웹소켓();`

지갑 페이지

잔고 영역 1 잔고 영역 2

`use잔고(); use잔고();`

기존 코드 (Naive한 구현)

상점 페이지

```
function 상점_페이지() {  
    use웹소켓();  
    return <시세_영역 />;  
}  
  
function 시세_영역() {  
    use시세();  
    return ...;  
}
```

거래 페이지

```
function 거래_페이지() {  
    use웹소켓();  
    return <>  
        <차트_영역 />  
        <주문_영역 />  
    </>;  
}  
  
function 차트_영역() {  
    use차트();  
    return ...;  
}  
function 주문_영역() {  
    use잔고();  
    return ...;  
}
```

지갑 페이지

```
function 지갑_페이지() {  
    use웹소켓();  
    return <>  
        <잔고_영역_1 />  
        <잔고_영역_2 />  
    </>;  
}  
  
function 잔고_영역_1() {  
    use잔고();  
    return ...;  
}  
function 잔고_영역_2() {  
    use잔고();  
    return ...;  
}
```

기존 코드 (Naive한 구현)

웹소켓

```
function use웹소켓() {  
  useEffect(() => {  
    웹소켓_연결();  
    return () => 웹소켓_정리();  
  });  
}
```

시세 정보

```
function use시세() {  
  useEffect(() => {  
    시세_구독시작();  
    return () => 시세_구독해지();  
  });  
}
```

차트 정보

```
function use차트() {  
  useEffect(() => {  
    차트_구독시작();  
    return () => 차트_구독해지();  
  });  
}
```

잔고 정보

```
function use잔고() {  
  useEffect(() => {  
    잔고_구독시작();  
    return () => 잔고_구독해지();  
  });  
}
```

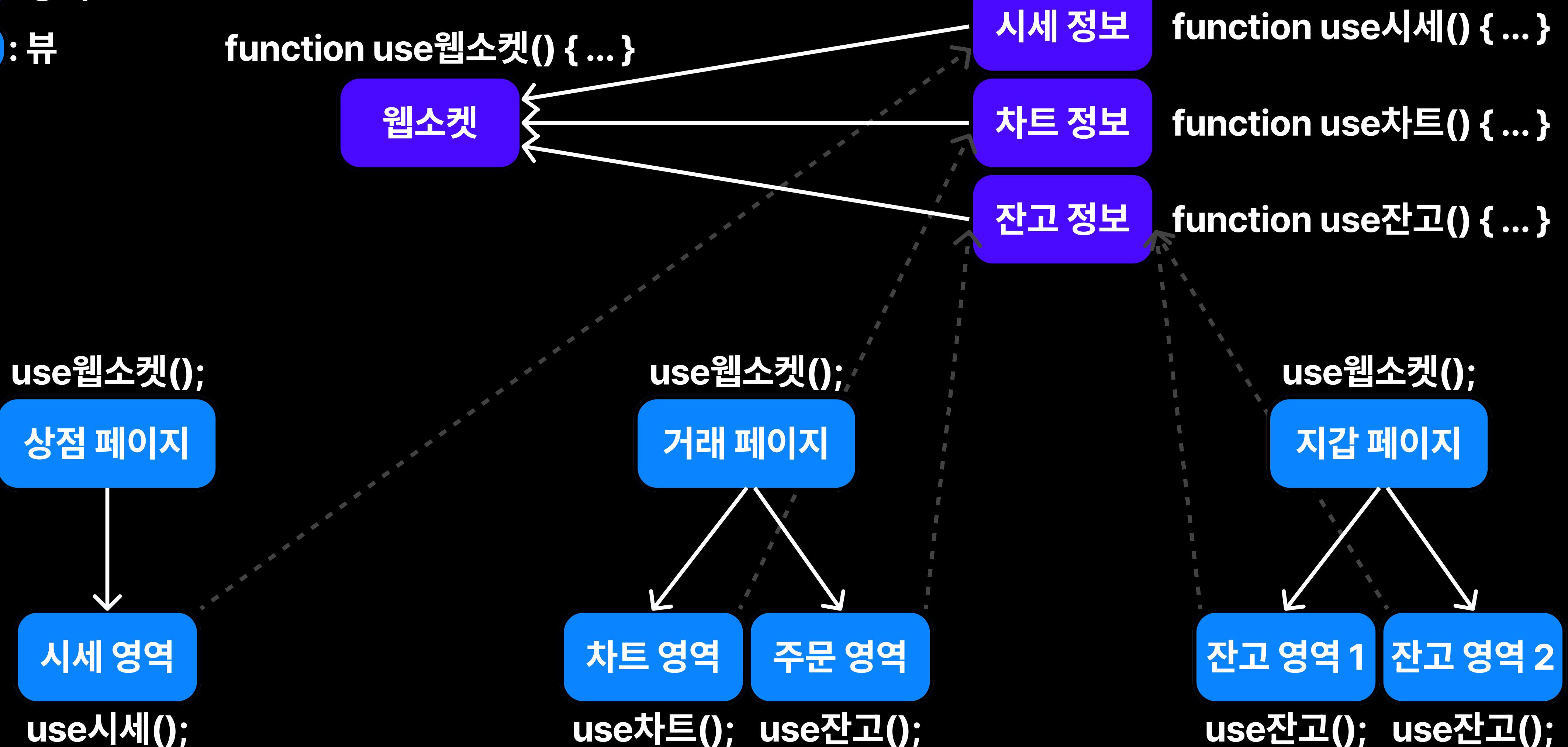
기존 코드의 문제

1. 웹소켓의 연결과 정리를 담당하는 곳과 웹소켓의 상태를 사용하는 곳이 달라서 같은 관심사 간에 Colocation되지 않음
2. 같은 리소스를 바라보는 곳이 여러군데가 되면 땅글링 구독 문제가 생김
3. SPA임에도 불구하고 매 페이지 이동시 웹소켓이 재연결됨

문제 1 - Colocation 안 됨

: 상태

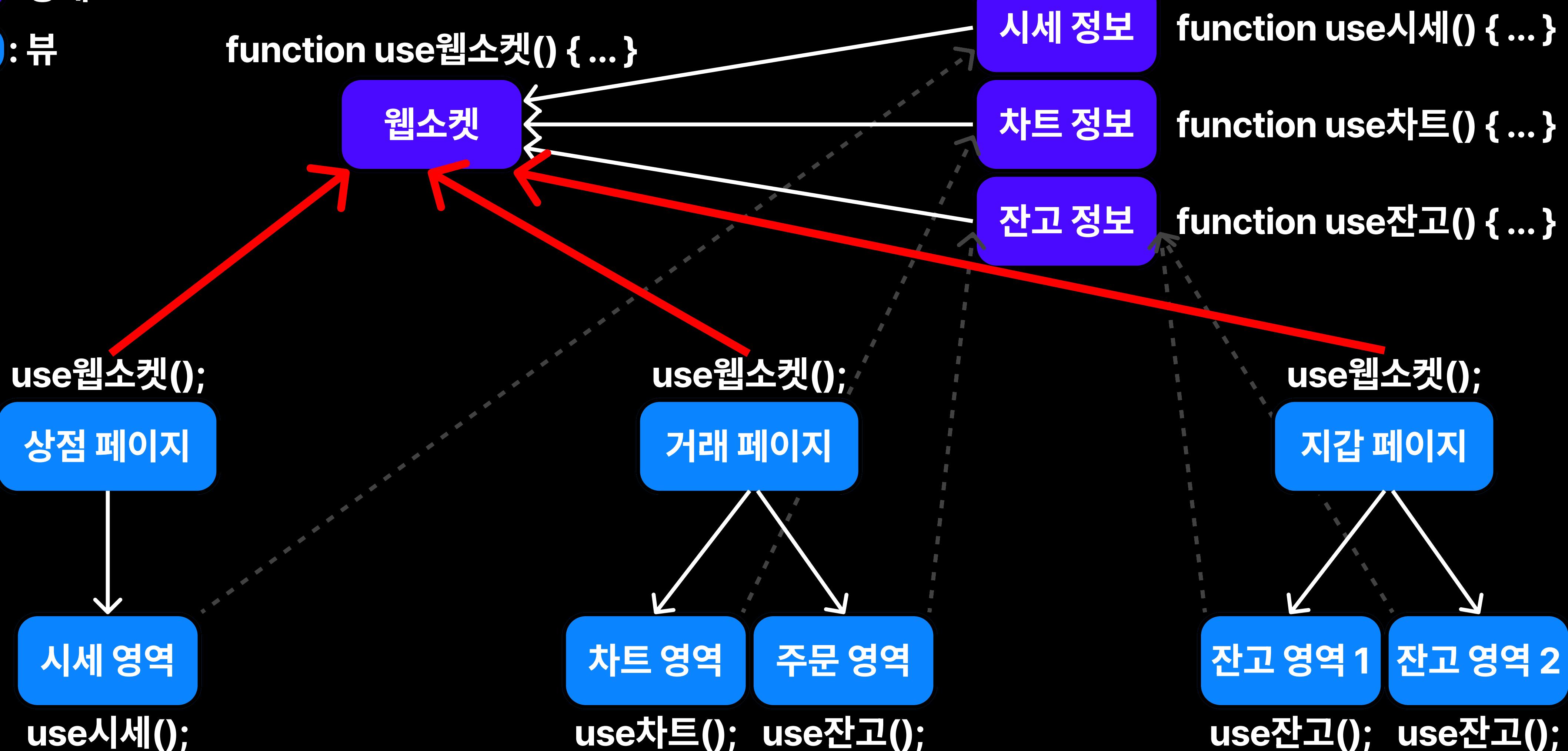
: 뷰



문제 1 - Colocation 안 됨

: 상태

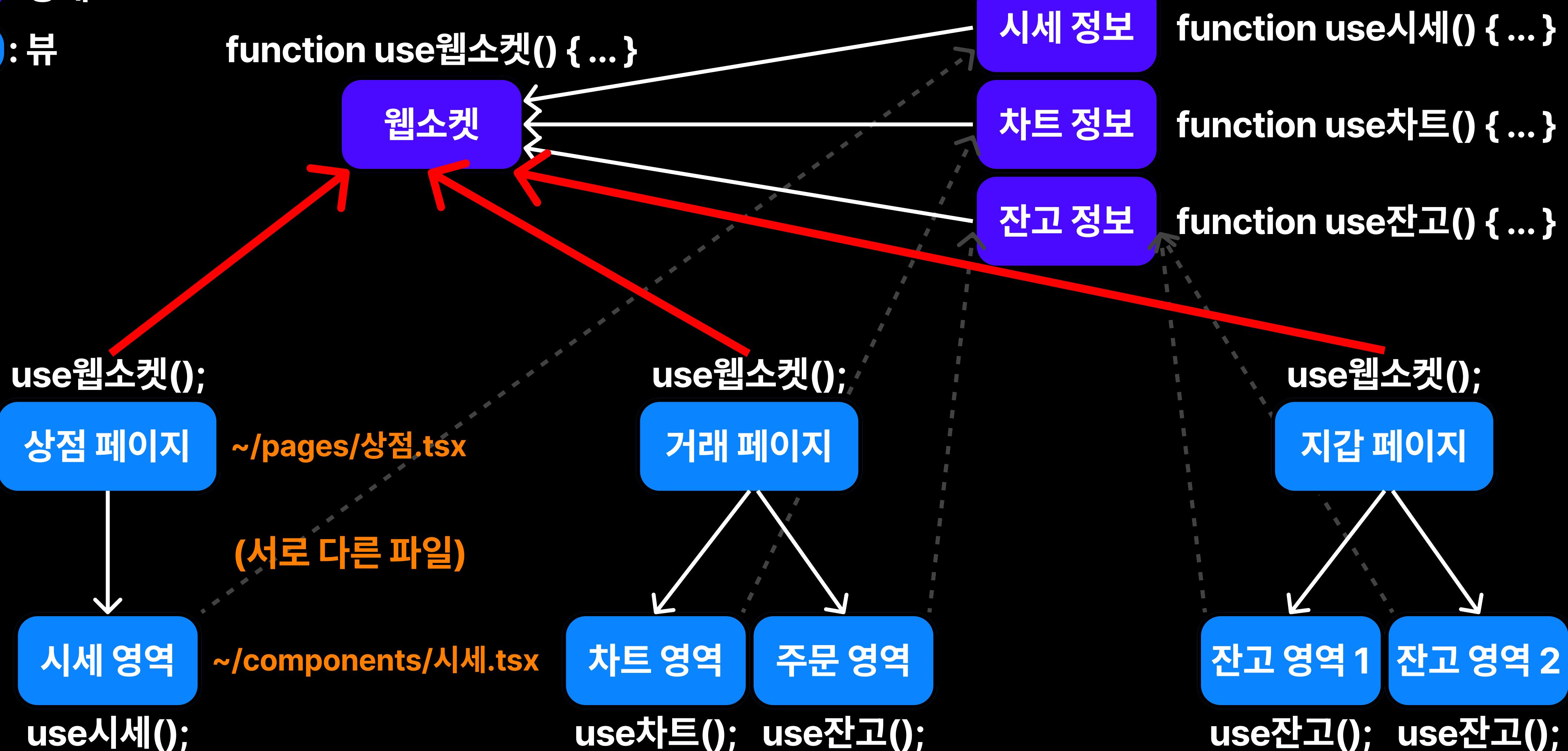
: 뷰



문제 1 - Colocation 안 됨

: 상태

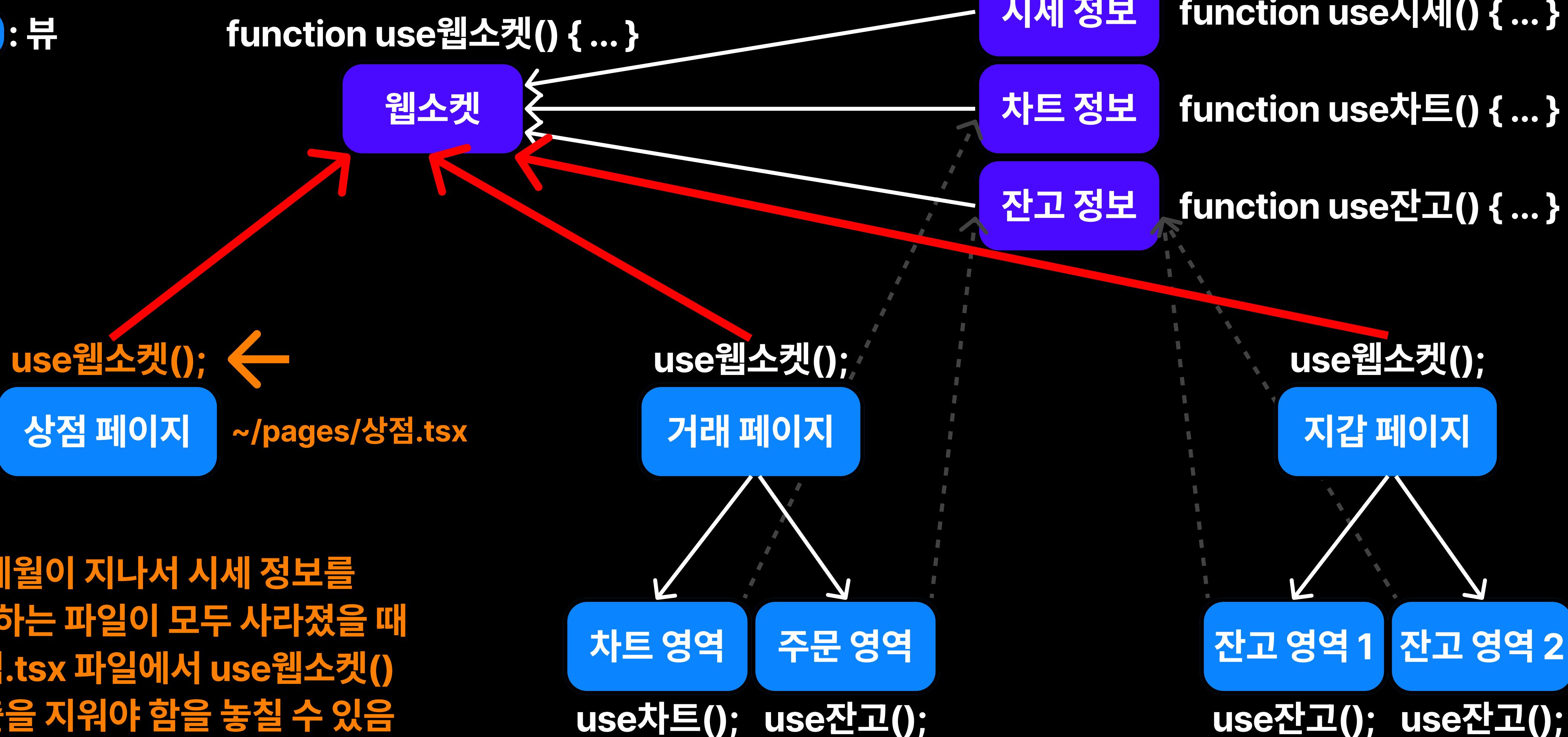
: 뷰



문제 1 - Colocation 안 됨

: 상태

: 뷰



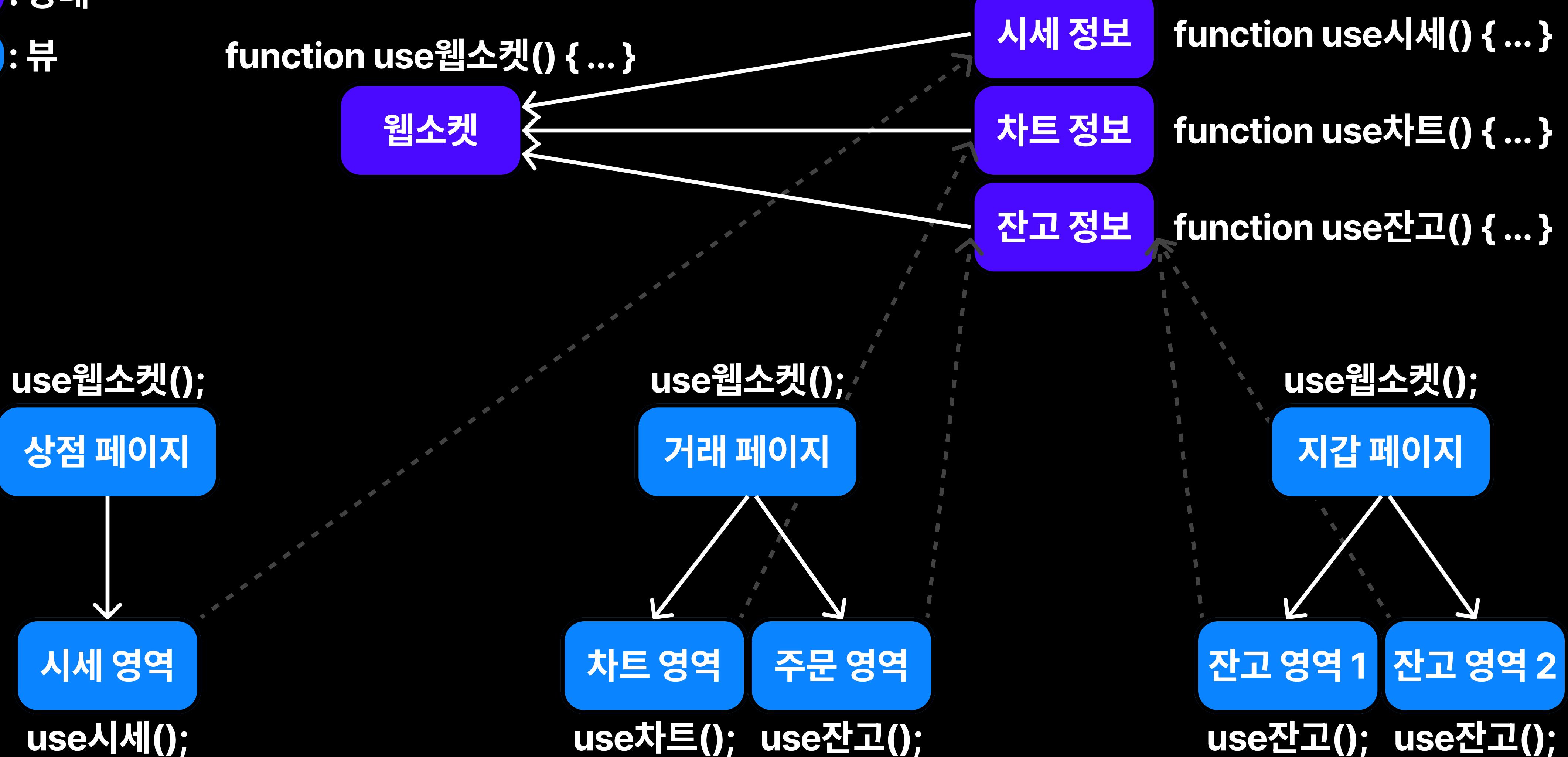
기존 코드의 문제

1. 웹소켓의 연결과 정리를 담당하는 곳과 웹소켓의 상태를 사용하는 곳이 달라서 같은 관심사 간에 Colocation되지 않음
2. 같은 리소스를 바라보는 곳이 여러군데가 되면 땅글링 구독 문제가 생김
3. SPA임에도 불구하고 매 페이지 이동시 웹소켓이 재연결됨

문제 2 - 땅글링 구독 발생

: 상태

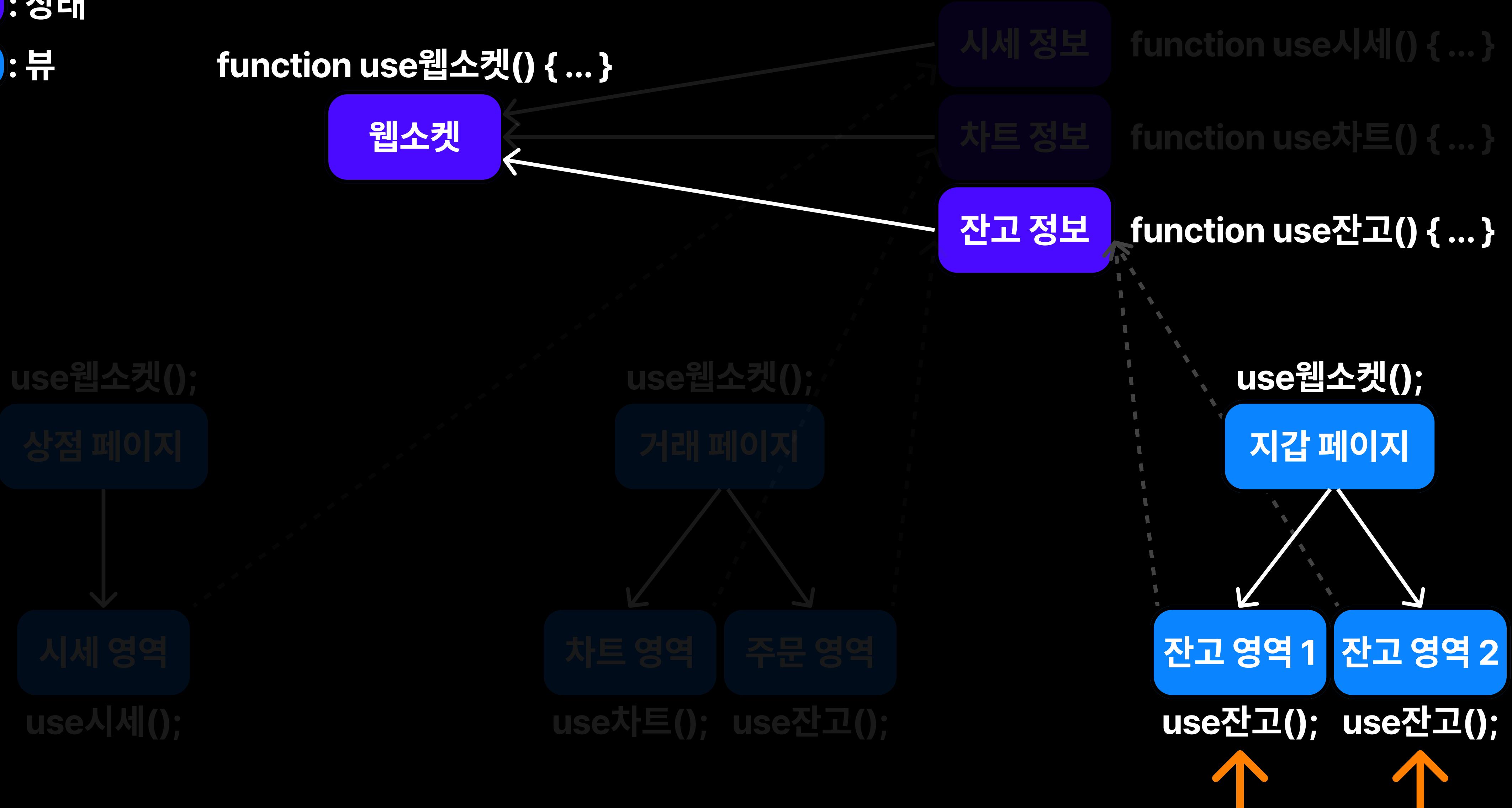
: 뷰



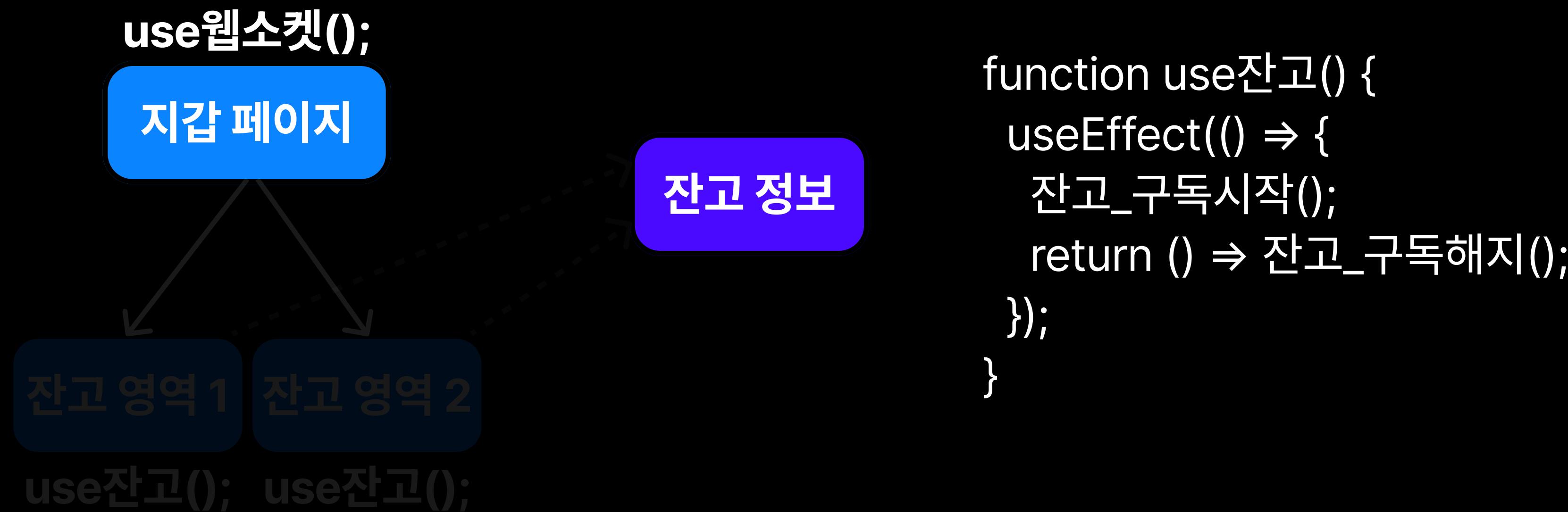
문제 2 - 땅글링 구독 발생

: 상태

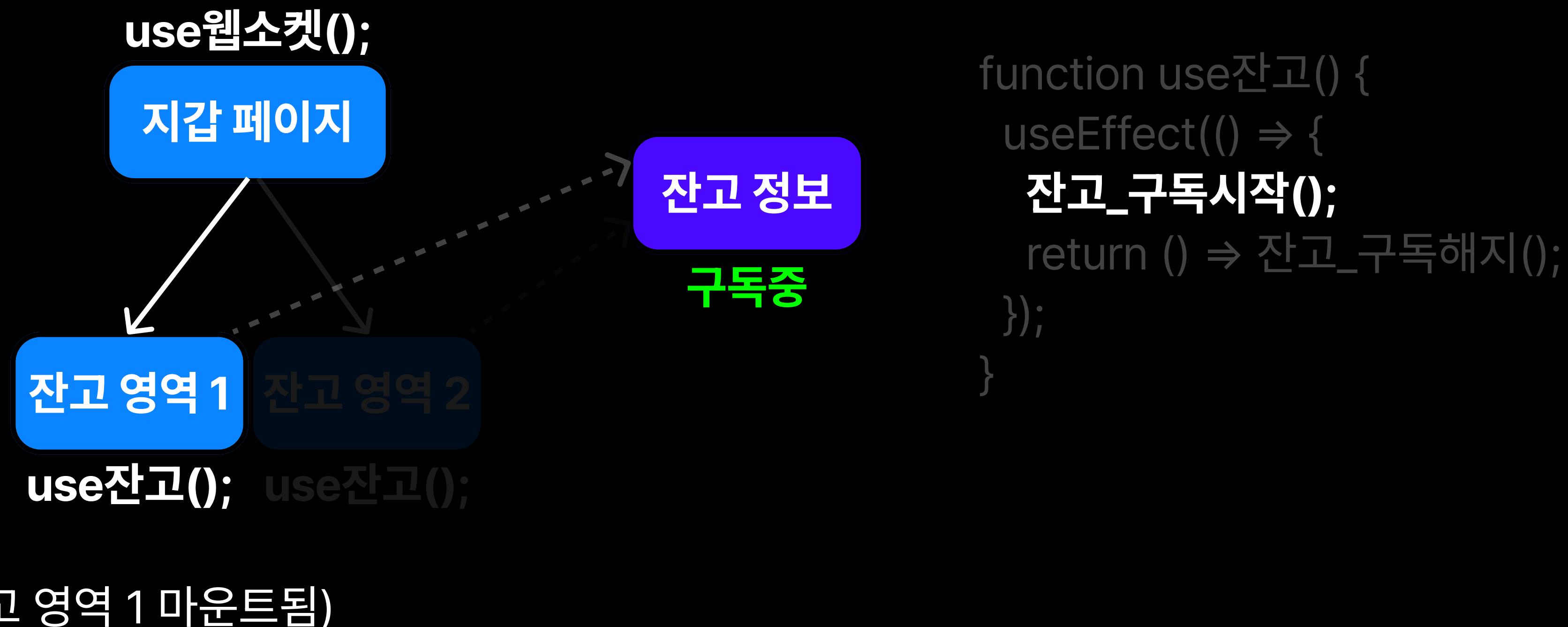
: 뷰



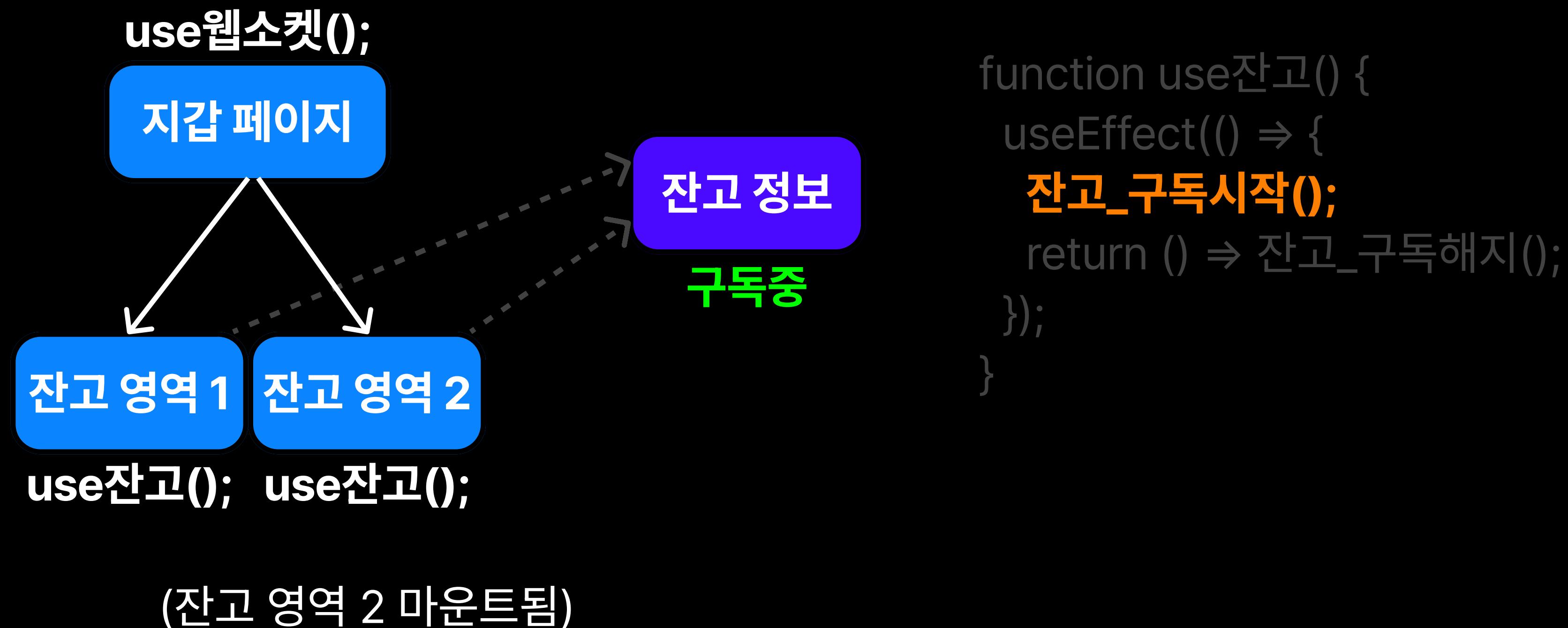
문제 2 - 땅글링 구독 발생



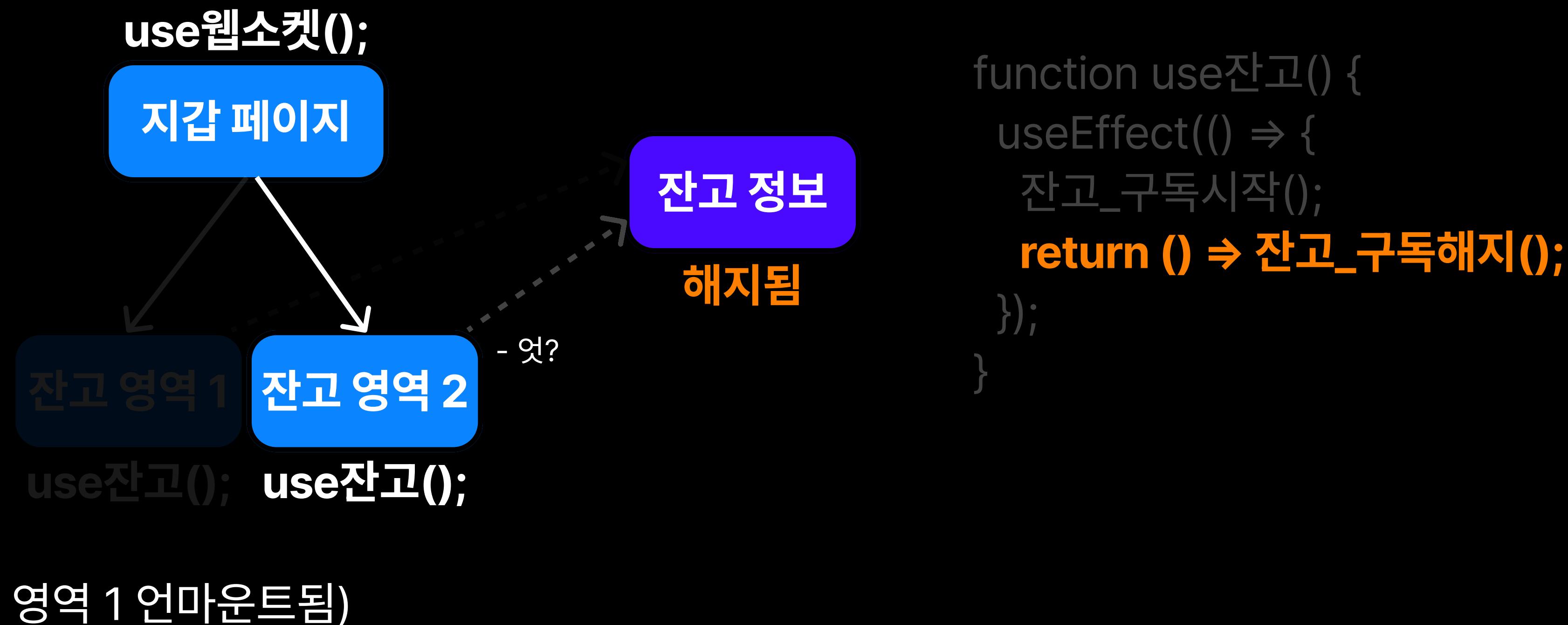
문제 2 - 땅글링 구독 발생



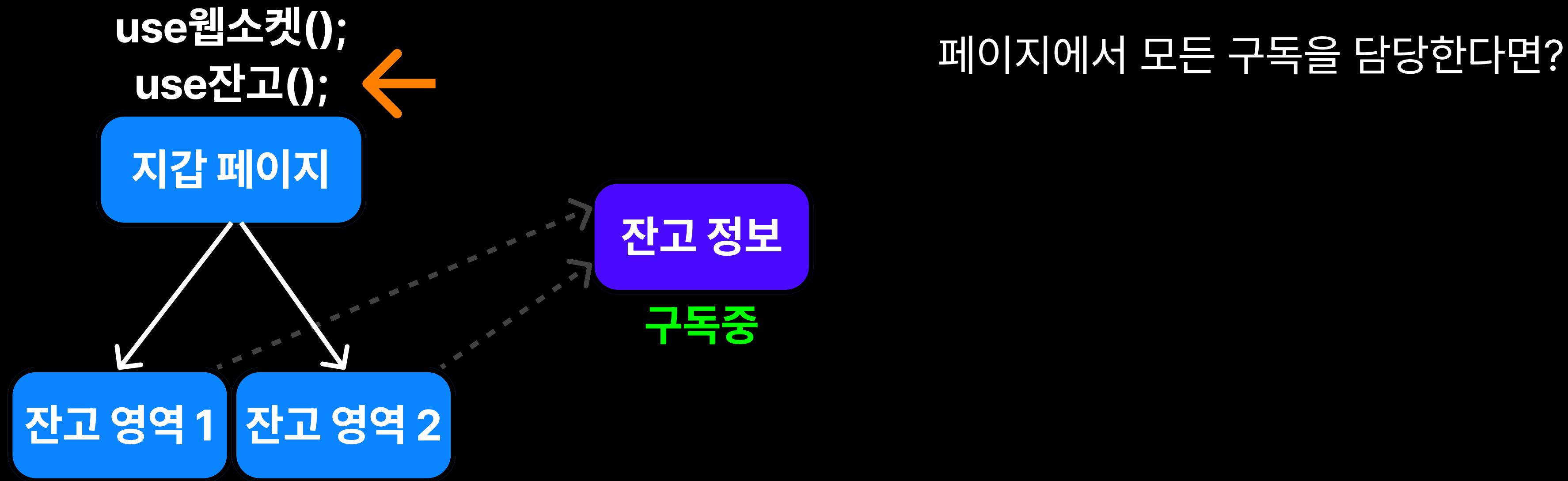
문제 2 - 땅글링 구독 발생



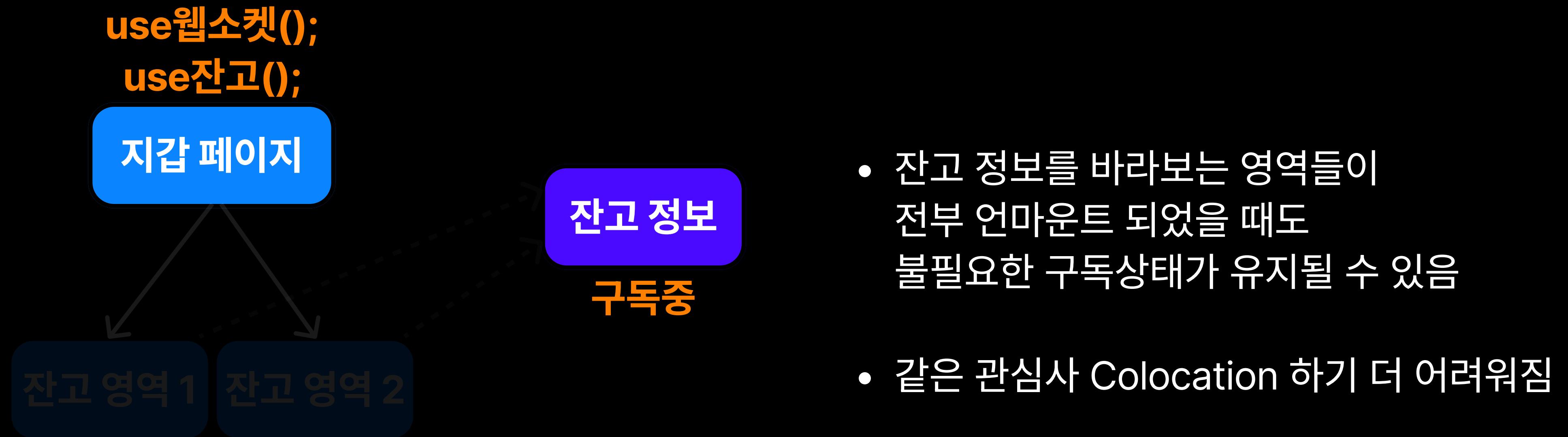
문제 2 - 땅글링 구독 발생



문제 2 - 땅글링 구독 발생



문제 2 - 냉글링 구독 발생



기존 코드의 문제

1. 웹소켓의 연결과 정리를 담당하는 곳과 웹소켓의 상태를 사용하는 곳이 달라서 같은 관심사 간에 Colocation되지 않음
2. 같은 리소스를 바라보는 곳이 여러군데가 되면 땅글링 구독 문제가 생김
3. SPA임에도 불구하고 매 페이지 이동시 웹소켓이 재연결됨

문제 3 - 페이지 이동시 웹소켓 재연결

: 상태

: 뷰

`function use웹소켓() { ... }`

웹소켓

연결됨

시세 정보

차트 정보

잔고 정보

`function use시세() { ... }`

`function use차트() { ... }`

`function use잔고() { ... }`

`use웹소켓();`

상점 페이지

```
function use웹소켓() {
  useEffect(() => {
    웹소켓_연결();
    return () => 웹소켓_정리();
  });
  use시세();
}
```

`use웹소켓();`

거래 페이지

차트 영역 주문 영역
`use차트(); use잔고();`

`use웹소켓();`

지갑 페이지

잔고 영역 1 잔고 영역 2
`use잔고(); use잔고();`

문제 3 - 페이지 이동시 웹소켓 재연결

: 상태

: 뷰

```
function use웹소켓() { ... }
```

웹소켓

끊어졌다가
재연결됨

시세 정보

차트 정보

잔고 정보

```
function use시세() { ... }
```

```
function use차트() { ... }
```

```
function use잔고() { ... }
```

use웹소켓();
상점 페이지 언마운트 되면서
useEffect cleanup 호출

```
function use웹소켓() {  
  useEffect(() => {  
    웹소켓_연결();  
    return () => 웹소켓_정리();  
  });  
  use시세();  
}
```

use웹소켓();

거래 페이지

```
function use웹소켓() {  
  useEffect(() => {  
    웹소켓_연결(); <- 다시 연결  
    return () => 웹소켓_정리();  
  });  
  use차트(); use잔고();  
}
```

use웹소켓();

지갑 페이지

```
잔고 영역 1 잔고 영역 2  
use잔고(); use잔고();
```

문제 3 - 페이지 이동시 웹소켓 재연결

: 상태

: 뷰

`function use웹소켓() { ... }`

웹소켓

또 끊어졌다가
재연결됨

시세 정보

차트 정보

잔고 정보

`function use시세() { ... }`

`function use차트() { ... }`

`function use잔고() { ... }`

`use웹소켓();`

상점 페이지

`use웹소켓();`
거래 페이지 언마운트 되면서
`useEffect cleanup` 호출

`use웹소켓();`

지갑 페이지

`function use웹소켓() {
 useEffect(() => {
 웹소켓_연결();
 return () => 웹소켓_정리();
 });
}`

시세 영역

`use시세();`

차트 영역

`use차트();`

잔고 영역

`function use웹소켓() {
 useEffect(() => {
 웹소켓_연결(); ← 또 다시 연결
 return () => 웹소켓_정리();
 });
}`

시세 영역

`use시세();`

차트 영역

`use차트();`

잔고 영역

`use잔고();`

`use잔고();`

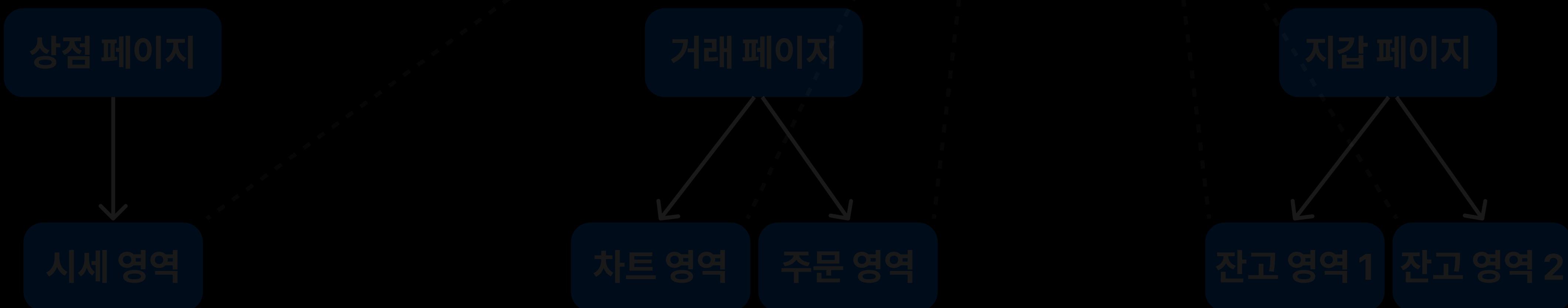
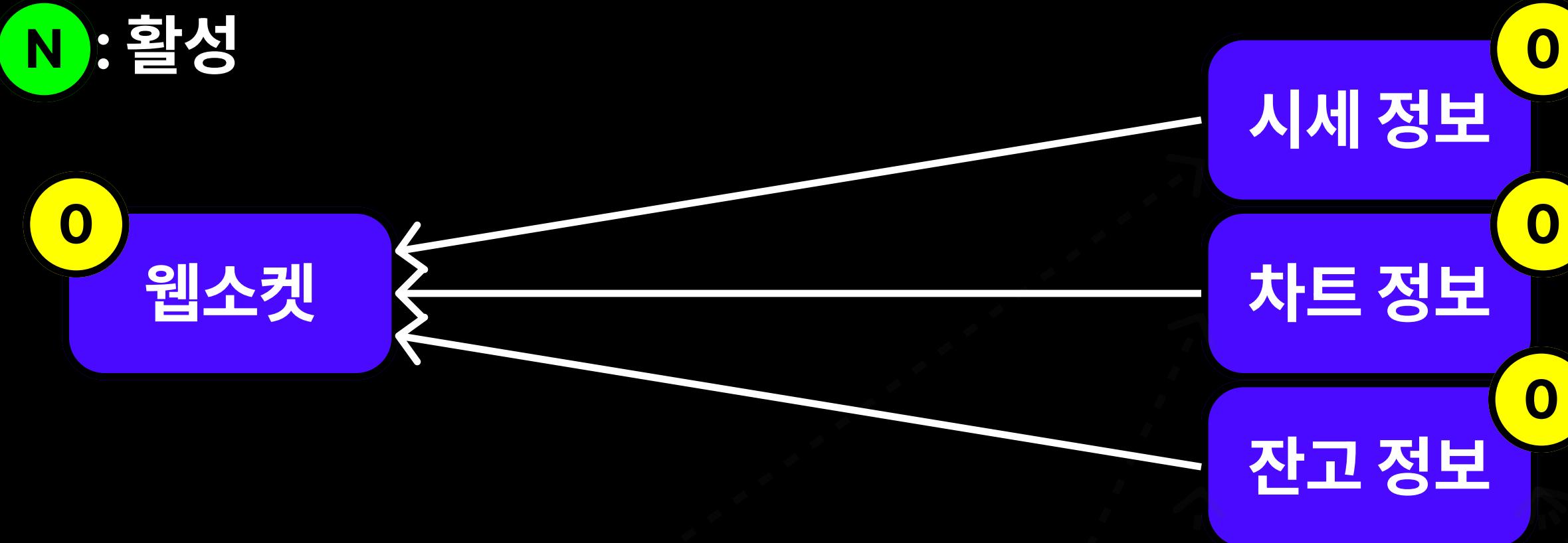
Solution

참조 카운트

- 웹소켓, 시세 정보 등 시작과 끝이 존재하는 개념에 참조 카운트 적용
- 어디선가 사용되기 시작하면 +1, 사용이 종료되면 -1
- 카운트가 0보다 커졌다면 초기화
- 카운트가 다시 0이 됐다면 정리

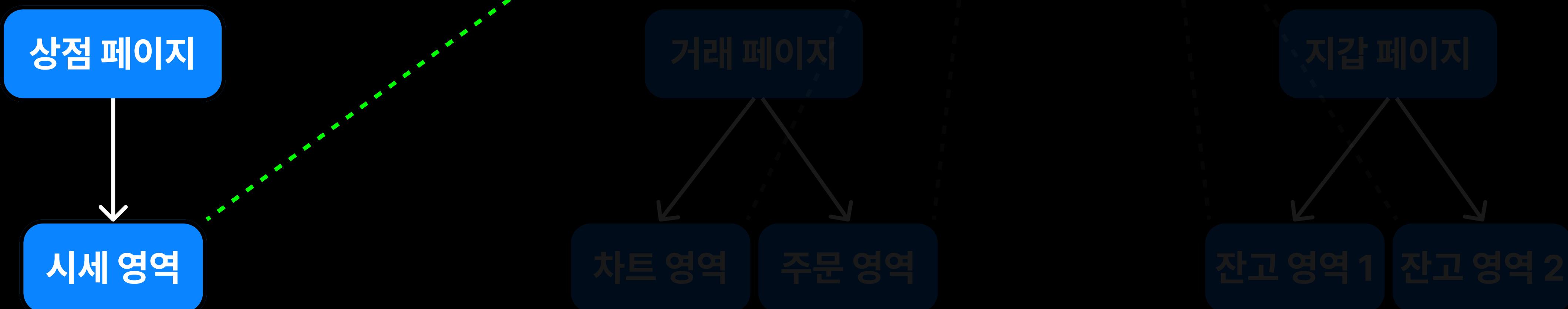
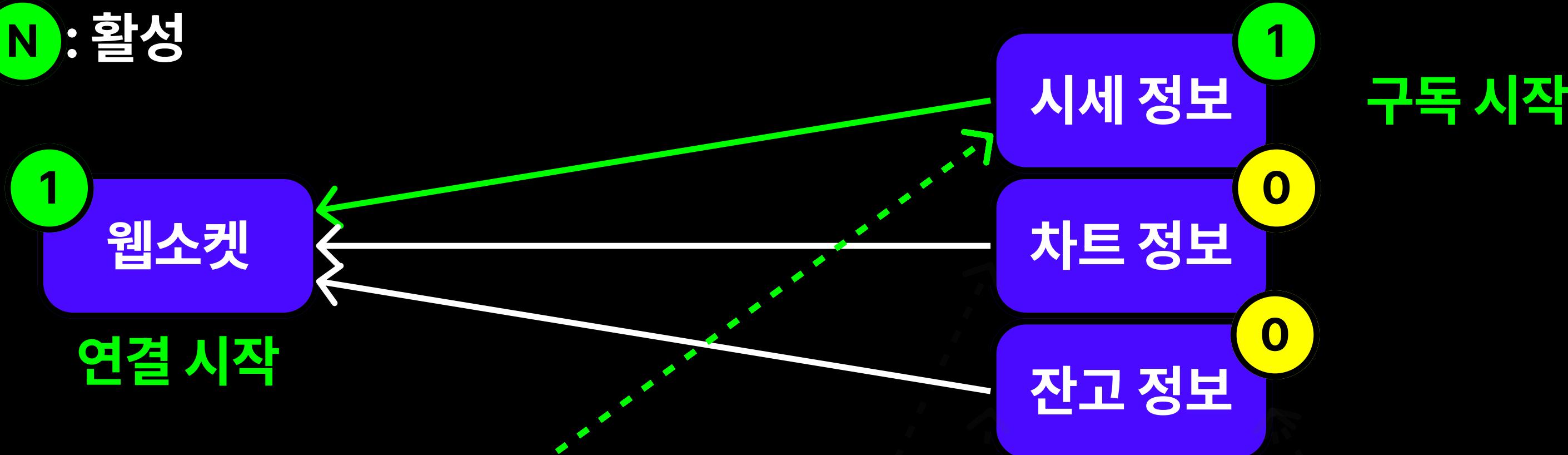
해결책 - 참조 카운트

 : 상태  : 비활성  : 활성
 : 뷰



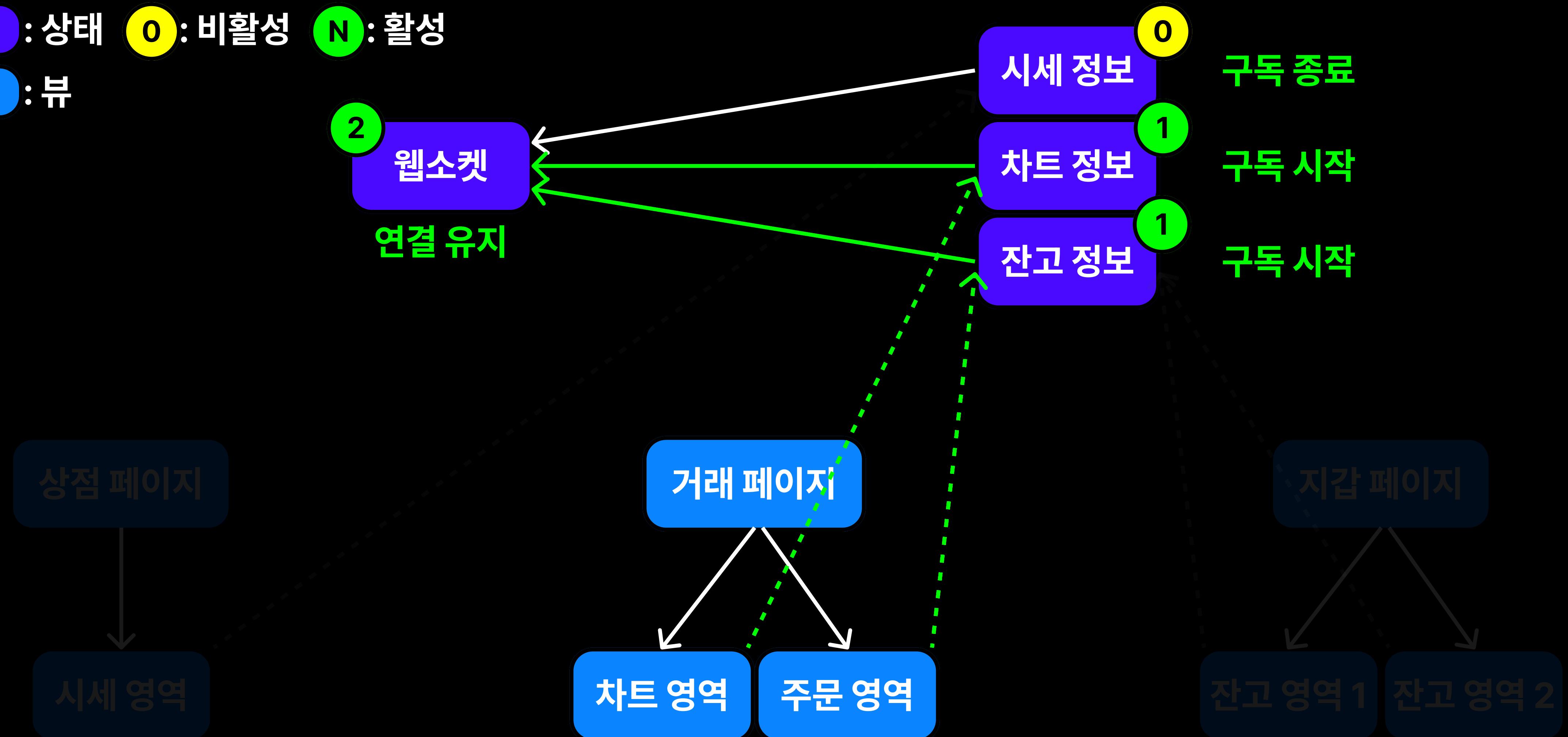
해결책 - 참조 카운트

 : 상태  : 비활성  : 활성
 : 뷰



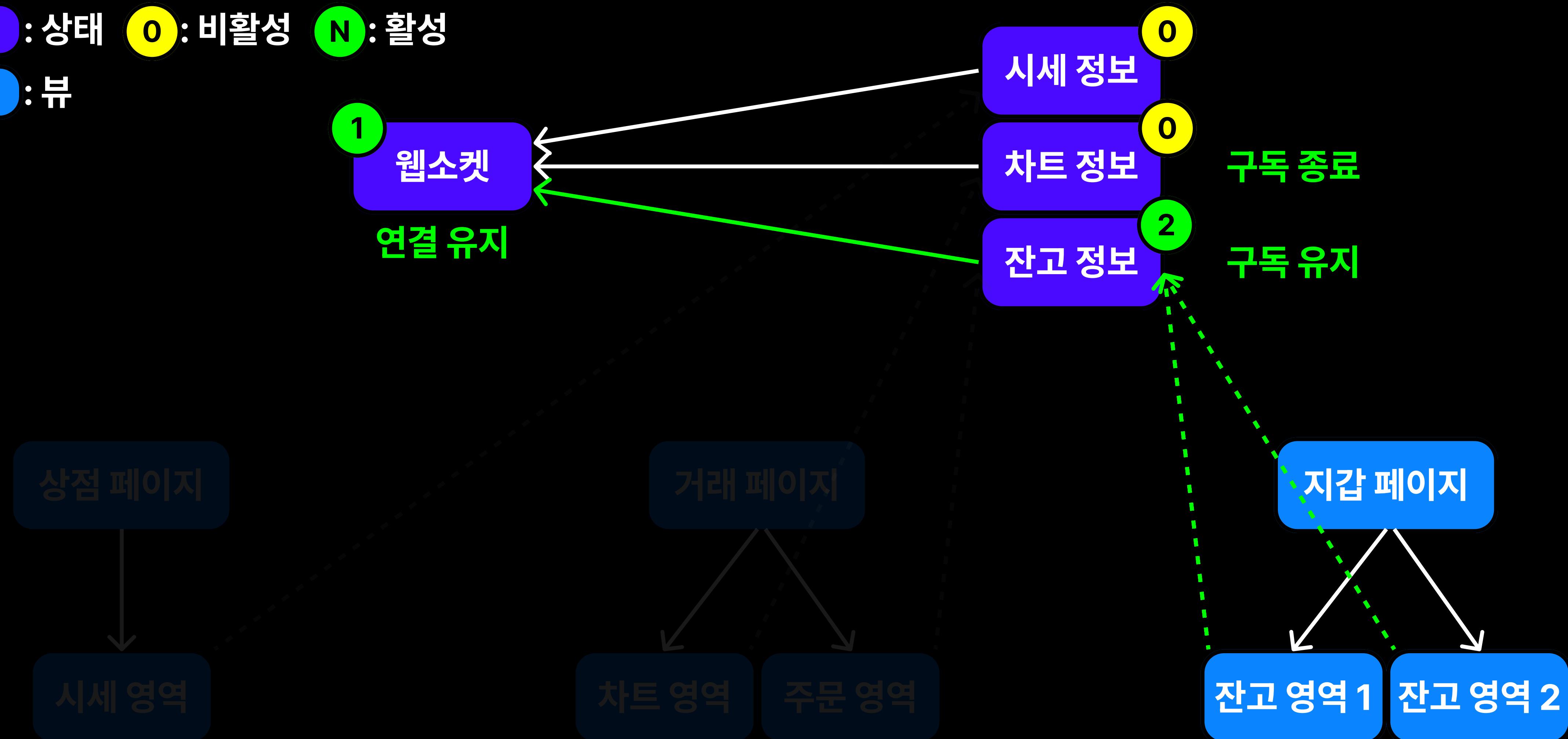
해결책 - 참조 카운트

: 상태 0: 비활성 N: 활성
: 뷰

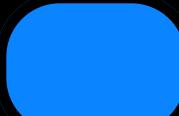


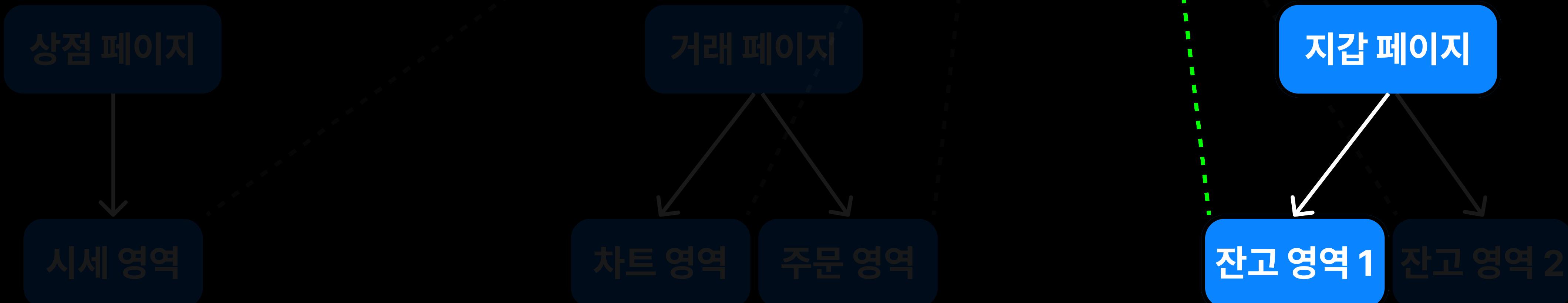
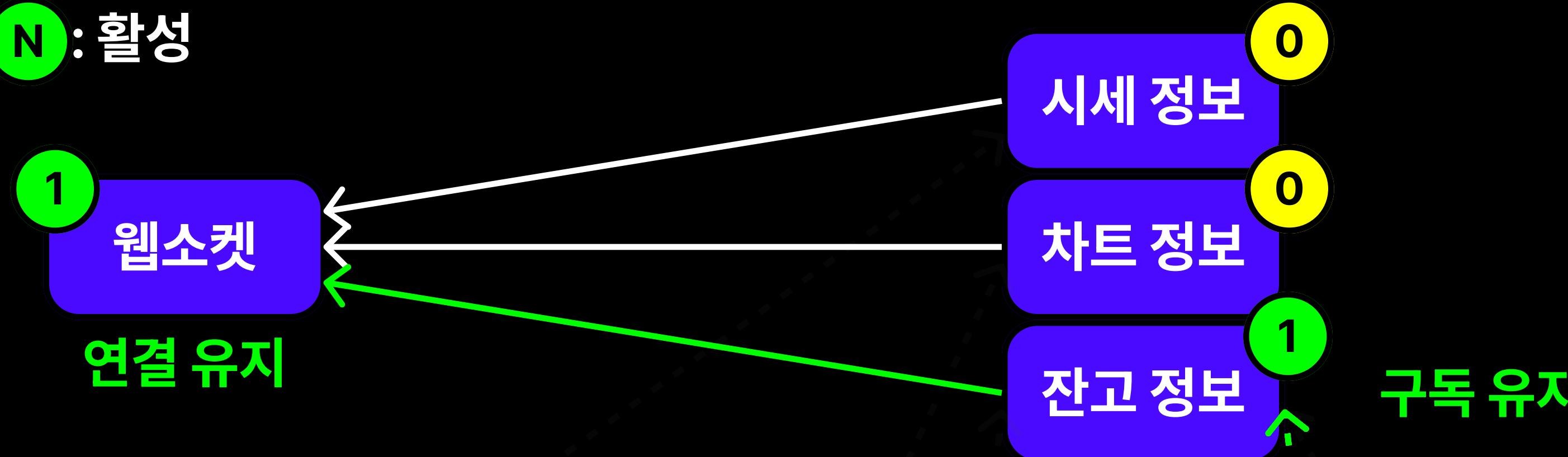
해결책 - 참조 카운트

: 상태 0: 비활성 N: 활성
: 뷰



해결책 - 참조 카운트

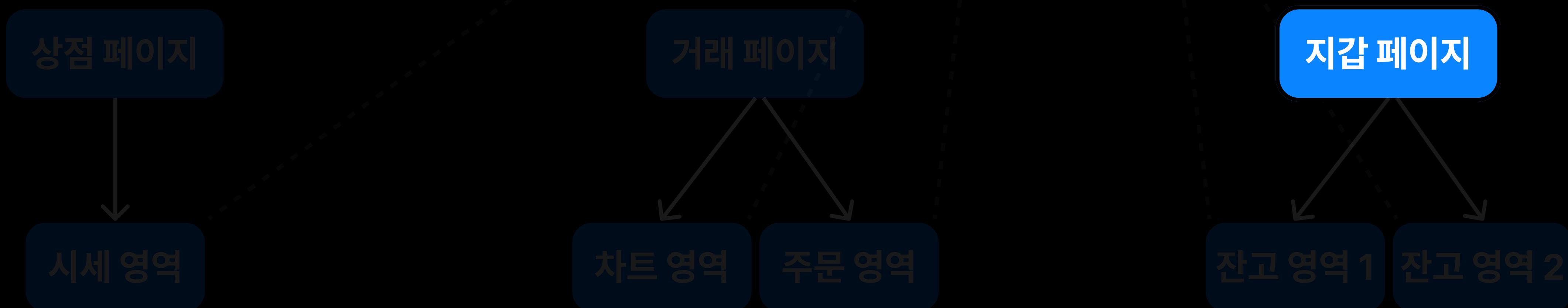
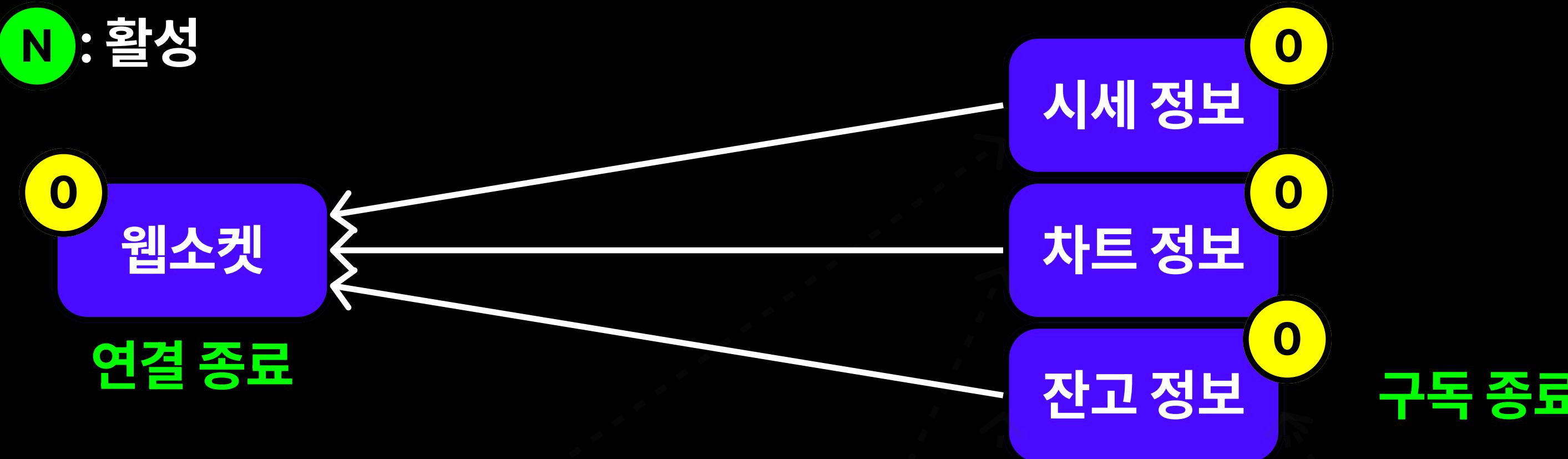
 : 상태  : 비활성  : 활성
 : 뷰



해결책 - 참조 카운트

: 상태 0: 비활성 N: 활성

: 뷰



Library

Bunja

- bunja, useBunja 함수 제공
 - bunja – 의존관계를 기술할 수 있는 상태 단위
 - **useBunja** – 참조 카운트가 자동 관리되는 흑
- 의존성 주입 라이브러리인 bunshi에 영감을 받음
- 원자(atom)를 조합해서 문자가 된다는 은유
 - jotai 등과 같이 사용하기에 좋음

Bunja 적용 코드

웹소켓

```
const 웹소켓 = bunja([], () => ({
  [bunja.effect]() {
    웹소켓_연결();
    return () => 웹소켓_정리();
  }
}));
```

시세 정보

```
const 시세 = bunja([웹소켓], () => ({
  [bunja.effect]() {
    시세_구독시작();
    return () => 시세_구독해지();
  }
}));
```

차트 정보

```
const 차트 = bunja([웹소켓], () => ({
  [bunja.effect]() {
    차트_구독시작();
    return () => 차트_구독해지();
  }
}));
```

잔고 정보

```
const 잔고 = bunja([웹소켓], () => ({
  [bunja.effect]() {
    잔고_구독시작();
    return () => 잔고_구독해지();
  }
}));
```

Bunja 적용 코드

상점 페이지

```
function 상점_페이지() {  
    return <시세_영역 />;  
}  
  
function 시세_영역() {  
    useBunja(시세);  
    return ...;  
}
```

거래 페이지

```
function 거래_페이지() {  
    return <>  
        <차트_영역 />  
        <주문_영역 />  
    </>;  
}  
  
function 차트_영역() {  
    useBunja(차트);  
    return ...;  
}  
function 주문_영역() {  
    useBunja(잔고);  
    return ...;  
}
```

지갑 페이지

```
function 지갑_페이지() {  
    return <>  
        <잔고_영역_1 />  
        <잔고_영역_2 />  
    </>;  
}  
  
function 잔고_영역_1() {  
    useBunja(잔고);  
    return ...;  
}  
function 잔고_영역_2() {  
    useBunja(잔고);  
    return ...;  
}
```

Bunja 적용 코드

: 상태

: 뷰

const 웹소켓 = bunja(...);

웹소켓

시세 정보

const 시세 = bunja(...);

차트 정보

const 차트 = bunja(...);

잔고 정보

const 잔고 = bunja(...);

상점 페이지

시세 영역

useBunja(시세);

거래 페이지

차트 영역

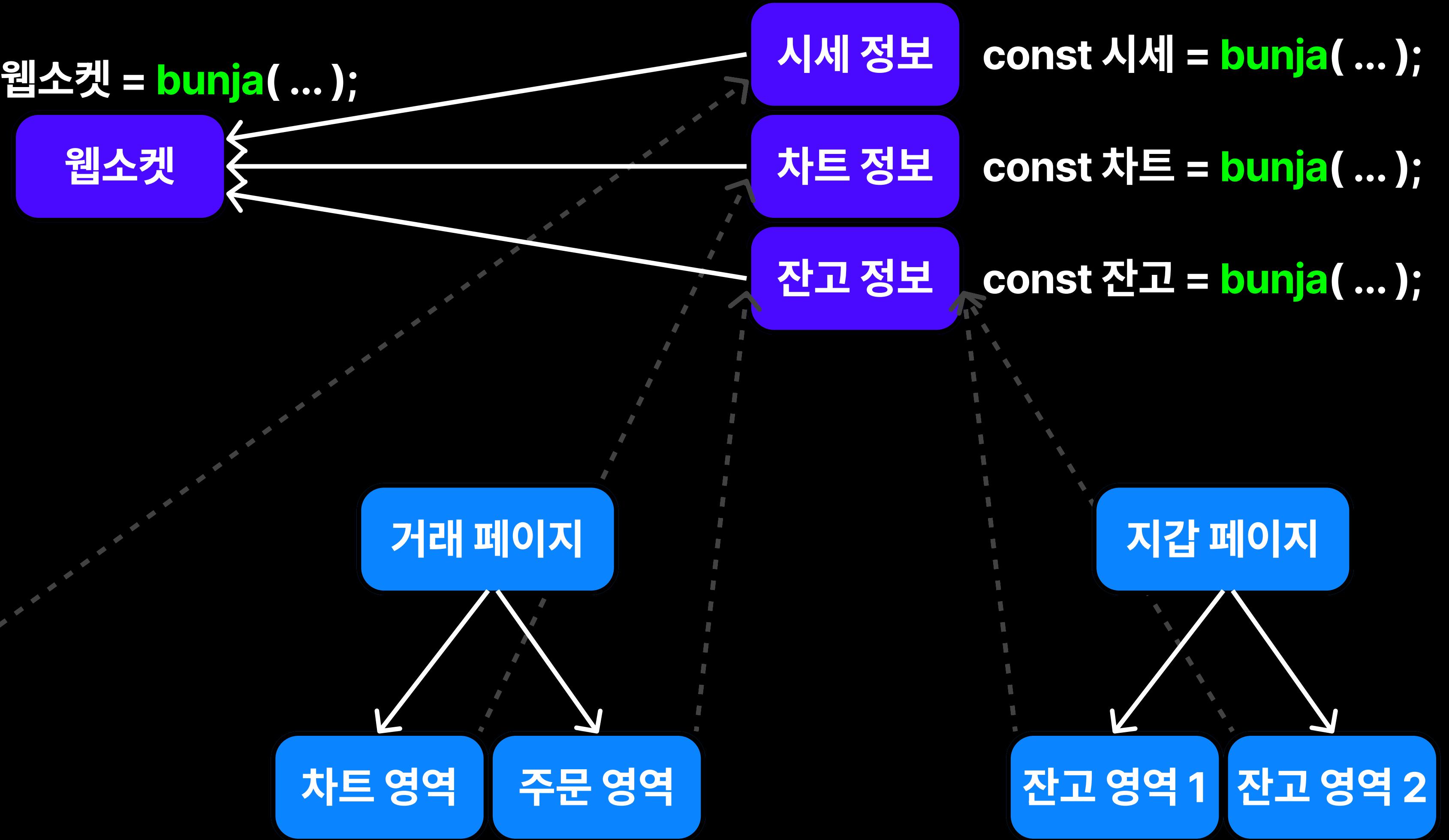
주문 영역

useBunja(차트); useBunja(잔고);

지갑 페이지

잔고 영역 1 잔고 영역 2

useBunja(잔고); useBunja(잔고);



Bunja가 해결한 문제

1. 웹소켓의 연결과 정리를 담당하는 곳과 웹소켓의 상태를 사용하는 곳이 달라서 같은 관심사 간에 Colocation되지 않던 문제
2. 같은 리소스를 바라보는 곳이 여러군데가 되면 땅글링 구독이 생기던 문제
3. SPA임에도 불구하고 매 페이지 이동시 웹소켓이 재연결되던 문제

Bunja가 해결한 문제

1. 웹소켓의 연결과 정리를 담당하는 곳과 웹소켓의 상태를 사용하는 곳이 달라서 같은 관심사 간에 Colocation되지 않던 문제
 - 시작과 끝이 존재하는 개념간의 의존관계를 기술해서 리액트 컴포넌트에서는 상태를 가져다 쓰는 것만 신경쓰도록 해서 해결
 - useBunja(분자) 호출에서 의존하는 분자로 쉽게 go to definition 가능
 - 분자를 이용하는 코드라면 어디서든 초기화/정리 코드를 쉽게 찾아갈 수 있음

Bunja가 해결한 문제

1. 웹소켓의 연결과 정리를 담당하는 곳과 웹소켓의 상태를 사용하는 곳이 달라서 같은 관심사 간에 Colocation되지 않던 문제
2. 같은 리소스를 바라보는 곳이 여러군데가 되면 땅글링 구독이 생기던 문제
3. SPA임에도 불구하고 매 페이지 이동시 웹소켓이 재연결되던 문제

Bunja가 해결한 문제

2. 같은 리소스를 바라보는 곳이 여러군데가 되면 땡글링 구독이 생기던 문제

- `useBunja(분자)`가 렌더트리 어딘가에 존재하는 동안은 리소스 구독이 해지되지 않음
- `useBunja(분자)`를 호출하는 컴포넌트들이 렌더트리에서 전부 사라지면 자동으로 리소스 구독이 해지됨

Bunja가 해결한 문제

1. 웹소켓의 연결과 정리를 담당하는 곳과 웹소켓의 상태를 사용하는 곳이 달라서 같은 관심사 간에 Colocation되지 않던 문제
2. 같은 리소스를 바라보는 곳이 여러군데가 되면 땅글링 구독이 생기던 문제
3. SPA임에도 불구하고 매 페이지 이동시 웹소켓이 재연결되던 문제

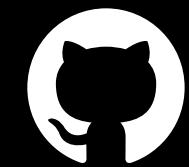
Bunja가 해결한 문제

3. SPA임에도 불구하고 매 페이지 이동시 웹소켓이 재연결되던 문제

- 땅글링 구독 문제를 해결한 것과 마찬가지로 `useBunja(분자)`가 렌더트리 어딘가에서 사용되는 동안은 웹소켓 연결도 계속 유지됨
- 사실 참조카운트 개념만 존재하면 카운트가 0이 된 즉시 리소스 정리를 하게 되므로 페이지 이동간에 여전히 재연결 문제가 발생할 수 있음
- `bunja`에서는 카운트가 깎였을 때 `setTimeout`으로 한 틱을 기다린 뒤, 그 때에도 카운트가 0이면 리소스를 정리하도록 함

끝

Questions?



<https://github.com/disjukr/bunja>



<https://www.npmjs.com/package/bunja>



<https://jsr.io/@disjukr/bunja>