[사용자] ({ coinSymbol }: LikeButtonProps)와 React.FC<LikeButtonProps> = ({ coinSymbol })의 비교

구현 어려움 정도	개념
link	[사용자] ({ coinSymbol }: LikeButtonProps)와 React.FC<LikeButtonProps> = ({ coinSymbol })의 비교
구현 완료	☑
기간	@December 14, 2024
정리 완료	☑

TypeScript에서 React 컴포넌트를 작성할 때 사용되는 방식으로 두가지에 대해 차이점과 장단점을 정리해보려 합니다.

1. ({ coinSymbol }: LikeButtonProps)

```
interface LikeButtonProps {
  coinSymbol: string;
}

const LikeButton = ({ coinSymbol }: LikeButtonProps) => {
```

특징

- Typescript 함수 컴포넌트로 작성된 일반 함수 방식입니다.
- LikeButtonProps는 컴포넌트의 props 타입을 지정합니다.
- coinSymbol은 props에서 구조분해할당을 통해 추출됩니다.

장점

- 일반 함수 선언 스타일로 문법이 간단합니다.
- 함수 선언 방식이라 디버깅 시, 스택 트레이스에서 함수 이름이 명확하게 출력됩니다.
- 타입을 유연하게 지정할 수 있습니다.

단점

- React에서 제공하는 React.FC(Functional Component)의 몇 가지 타입 관련 기능을 직접 제공하지 않습니다.
- 예) React.FC는 children 자동 포함

2. React.FC<LikeButtonProps> = ({ coinSymbol })

```
interface LikeButtonProps {
  coinSymbol: string;
}

const LikeButton: React.FC<LikeButtonProps> = ({ coinSymbol }) => {
```

특징

- React의 `React.FC` (Functional Component)를 활용한 컴포넌트 선언 방식입니다.
- React.FC<LikeButtonProps>를 사용해 컴포넌트의 props를 명시적으로 지정합니다.

장점

- `React.FC`는 children을 기본적으로 타입에 포함시켜줍니다. (이 경우, `LikeButton` 컴포넌트가 `children`을 지원해야 한다면 별도로 추가 타입 정의를 하지 않아도 됩니다.)
- 컴포넌트가 더 명확히 React 함수형 컴포넌트임을 나타냅니다.

단점

- `React.FC`는 불필요하게 children 타입이 포함되어 기본적으로 `props.children`을 사용하지 않는 경우에도 이를 허용하게 됩니다. 이는 의도치 않은 오류를 불러올 수 있다고 합니다.
- 타입스크립트에서는 최근 React 팀이 `React.FC`의 사용을 권장하지 않는 추세입니다.
  - **이유**
    1. **불필요한** children **타입 포함**: 명시적으로 포함할 의도가 없는 **children**을 기본으로 허용
    2. `React.FC`를 사용하지 않을 때 더 유연한 타입 추론 가능
    3. 코드가 더 복잡함

결론적으로 1번 방식을 사용하는 것이 좋다고 합니다. 👏