# [개발환경] Error: useSyncExternalStore only works in Client Components. Add the "use client" directive at the top of the file to use it.

| | |
|---|---|
| 구현 어려움 정도 | ⭐ |
| 구현 완료 | ☑ |
| 기간 | @November 29, 2024 |
| 정리 완료 | ☑ |

“use client”를 사용하지 않아 발생한 문제였고, 오류가 났던 이유에 대해 찾아봤습니다.

서버 컴포넌트와 클라이언트 컴포넌트에 대해 먼저 알아보겠습니다.

## Client Component

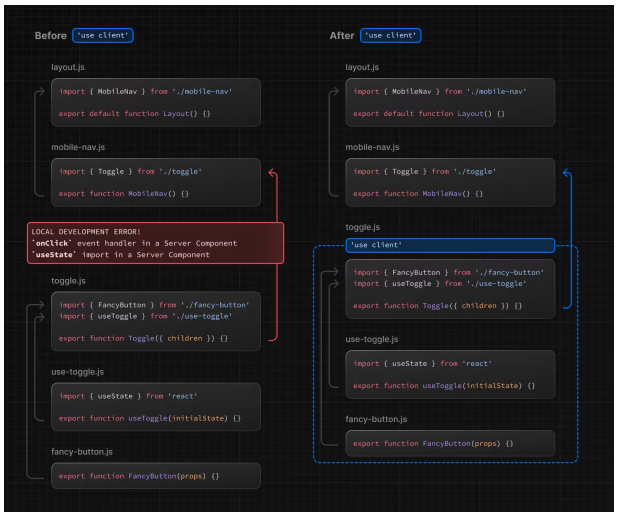
클라이언트 컴포넌트는 버튼 또는 검색창과 같이 사용자와 상호작용하는 작은 UI들을 클라이언트 단에서 랜더링되는 컴포넌트를 뜻합니다.

```
'use client';

import { useState } from 'react'

export default function Counter() {
  const [count, setCount] = useState(0)

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  )
}
```




‘use client’를 파일의 최상단에 입력하면 하위 구성 요소를 포함하여 해당 파일로 가져온 다른 모듈이 클라이언트 번들의 일부로 간주됩니다.

즉, 모든 파일에서 ‘use client’를 선언하지 않고, 진입전에서 한 번 정의해주면 됩니다.

## Server Component

서버 컴포넌트란 말 그대로 서버 부분에서 랜더링되는 컴포넌트입니다.

사용자와 상호작용하는 버튼 또는 검색 UI 등을 제외한 나머지가 서버에서 랜더링되는 방식입니다.

### 서버컴포넌트의 특징

- 모든 컴포넌트는 `서버 컴포넌트`가 기본입니다.
- 따라서 클라이언트 컴포넌트를 사용하고 싶다면, `'use client'` 를 최상단에 입력해줘야 합니다.
- 이로 인해 자바스크립트 번들이 감소합니다.
- Hydration 과정이 없게 됩니다.
  - Hydration: 정적인 HTML 요소들에 JS 코드들이 결합되면 버튼 클릭에 따른 이벤트 처리 등이 가능한 것
- Event Listener(onClick 등) 사용이 불가합니다.
- React Hooks(useEffect 등) 사용이 불가합니다.
- Client Component 사용이 불가합니다.
- async 함수로 정의 가능합니다.

### 클라이언트 컴포넌트를 서버 컴포넌트로 바꾸려면?

변경 전

```
// 클라이언트 컴포넌트

"use client";

import { useState, useEffect } from "react";

export default async function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
   const [ topics, setTopics ] = useState();

   useEffect(() => {
      const setPage = async () => {
      const { data } = fetch(`http://localhost:9999/topics`);
      setTopics(data}
    }

    setPage();
  },[])

  return (
    <html>
      <body>
        <ol>
          {topics.map((topic: ITopics) => {
            return (
              <li key={topic.id}>
                <Link href={`/read/${topic.id}`}>{topic.title}</Link>
              </li>
              ...
```

변경 후

```
// 서버 컴포넌트 코드

export default async function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  const resp = await fetch(`http://localhost:9999/topics`);
  const topics = await resp.json();

  return (
    <html>
      <body>
        <ol>
          {topics.map((topic: ITopics) => {
            return (
              <li key={topic.id}>
                <Link href={`/read/${topic.id}`}>{topic.title}</Link>
              </li>
              ...
```

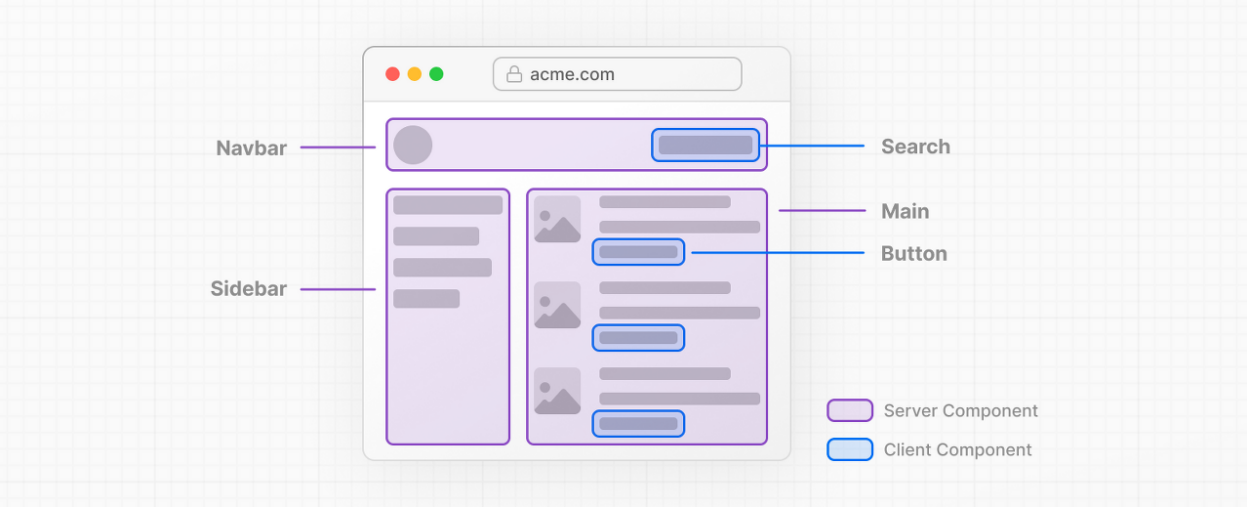
### 각각 언제 사용하는 것이 좋을까?

## 서버 및 클라이언트 구성 요소를 언제 사용합니까?

다음은 서버 및 클라이언트 구성 요소의 다양한 사용 사례에 대한 간략한 요약입니다.

| 뭘하길 원해? | 서버 구성요소 | 클라이언트 구성요소 |
|---|---|---|
| 데이터 가져오기 | ✓ | ✗ |
| 백엔드 리소스에 (직접) 액세스 | ✓ | ✗ |
| 민감한 정보를 서버에 보관하세요(액세스 토큰, API 키 등) | ✓ | ✗ |
| 서버에 대한 큰 의존성 유지 / 클라이언트 측 JavaScript 감소 | ✓ | ✗ |
| 상호작용 및 이벤트 리스너( `onClick()` , `onChange()` 등) 추가 | ✗ | ✓ |
| 상태 및 수명주기 효과 사용( `useState()` , `useReducer()` , `useEffect()` 등) | ✗ | ✓ |
| 브라우저 전용 API 사용 | ✗ | ✓ |
| 상태, 효과 또는 브라우저 전용 API에 따라 달라지는 사용자 정의 후크를 사용하세요. | ✗ | ✓ |
| React 클래스 구성요소 사용↗ | ✗ | ✓ |

`onClick` , `useState` 등 과 같은 `클라이언트 컴포넌트` 사용이 불가피한 경우를 모든 제외한 경우라고 할 수 있을 것 같다.
즉시 데이터를 `fetch` 하여 화면에 보여주거나, `env` 에서 가져오는 `API_KEY` 등의 보안이 필요한 곳에서 사용하면 된다.




## SSR과 Server Component

서버 컴포넌트는 서버에서 랜더링된다는 점이 SSR과 유사하다는 것을 알 수 있다.

둘의 차이점은

- **SSR은 서버에서 랜더링된 HTML**을 가져오지만 **서버 컴포넌트는 HTML이 아닌 랜더링할 트리 객체**를 가져온다.
- 트리 객체는 서버 컴포넌트, 클라이언트 컴포넌트로 구성되어 있으며(서버 컴포넌트 안에 클라이언트 컴포넌트가 포함돼 있음)
- 트리를 보고 DOM을 업데이트한다고 합니다.

---

### 참고

- https://curryyou.tistory.com/539
- https://html-jc.tistory.com/657
- https://velog.io/@2ast/React-%EC%84%9C%EB%B2%84-%EC%BB%B4%ED%8F%AC%EB%84%8C%ED%8A%B8React-Server-Component%EC%97%90-%EB%8C%80%ED%95%9C-%EA%B3%A0%EC%B0%B0