App.jsx

내용 설명

## ✅ 1️⃣ 상태 선언

const [activeSection, setActiveSection] = useState('home')

* activeSection: 현재 화면에 표시 중인 섹션 이름을 저장하는 상태값 (기본값 'home')
* setActiveSection: 상태를 업데이트하는 함수

예를 들어,

* 사용자가 "About" 섹션을 보고 있으면 → activeSection 값이 "about"
* "Projects" 섹션을 보고 있으면 → "projects"

이 값은 Navbar 컴포넌트에 props로 전달되어  
현재 섹션에 해당하는 메뉴 버튼만 강조(active 클래스 추가)됩니다.

**✅ 2️⃣ useEffect — 스크롤 이벤트 등록**

useEffect(() => { const handleScroll = () => { *// ... 스크롤 시 실행될 함수* } window.addEventListener('scroll', handleScroll) return () => window.removeEventListener('scroll', handleScroll) }, [])

* useEffect는 컴포넌트가 **화면에 처음 렌더링될 때 한 번 실행**됩니다.
* 이 안에서 window.addEventListener('scroll', handleScroll)을 등록해서  
  사용자가 **스크롤할 때마다 handleScroll 함수**가 호출되도록 합니다.
* return 안의 코드는 **cleanup 함수**로,  
  컴포넌트가 사라질 때(unmount) 이벤트를 해제해 메모리 누수를 방지합니다.

**✅ 3️⃣ handleScroll 함수**

이 함수가 실제 “어떤 섹션이 보이는가”를 계산하는 부분이에요.

const handleScroll = () => { const sections = ['home', 'about', 'skills', 'projects', 'contact'] const scrollPosition = window.scrollY + 100

* sections: HTML 문서 내에 있는 섹션들의 ID 목록입니다.  
  (즉, <section id="home">, <section id="about">, 이런 식으로 되어 있어야 함)
* window.scrollY: 페이지의 현재 세로 스크롤 위치 (위에서부터 얼마나 내려왔는지)
* + 100: 헤더 높이 같은 상단 여백을 고려해 약간 보정해준 값이에요.  
  (메뉴바 높이가 있다면, 메뉴가 화면 위를 가릴 수 있으니까 살짝 아래로 계산)

**✅ 4️⃣ 각 섹션 위치 비교**

for (const section of sections) { const element = document.getElementById(section) if (element) { const { offsetTop, offsetHeight } = element if (scrollPosition >= offsetTop && scrollPosition < offsetTop + offsetHeight) { setActiveSection(section) break } } }

이 부분이 핵심입니다 🔍

* offsetTop: 페이지 맨 위에서부터 섹션이 시작되는 거리 (px 단위)
* offsetHeight: 섹션의 높이

즉,

사용자가 스크롤한 위치(scrollPosition)가  
특정 섹션의 **시작 위치(**offsetTop**)부터 끝 위치(**offsetTop + offsetHeight**) 사이**에 있다면  
지금 그 섹션이 화면에 보이고 있다는 뜻이에요.

그럼 setActiveSection(section)으로 해당 섹션 이름을 상태에 저장합니다.  
break를 사용하는 이유는 **가장 먼저 해당하는 섹션 하나만 찾으면 더 이상 검사할 필요가 없기 때문**이에요.

**✅ 5️⃣ 전체 동작 요약**

| **단계** | **동작** |
| --- | --- |
| 1 | 사용자가 스크롤을 내린다 |
| 2 | handleScroll() 실행됨 |
| 3 | window.scrollY로 현재 위치를 계산 |
| 4 | 각 섹션(home, about, skills...)의 위치와 높이를 비교 |
| 5 | 현재 위치가 해당 섹션 범위 안에 들어오면 |
| 6 | setActiveSection(section) 실행 → 상태 업데이트 |
| 7 | Navbar 컴포넌트로 전달되어, 해당 메뉴 버튼만 강조됨 |

**✅ 6️⃣ 예시 흐름**

예를 들어,

* 화면의 “Projects” 영역을 보고 있을 때:
  + scrollY + 100 = 2000
  + projects 섹션의 offsetTop = 1800, offsetHeight = 800  
    → 2000은 1800~2600 사이이므로  
    → setActiveSection('projects') 실행  
    → Navbar의 Projects 버튼이 하이라이트 표시 🎯

**✨ 한 줄 정리**

🔹 이 코드는 **스크롤 위치를 감시하여**,  
현재 보고 있는 섹션의 ID를 activeSection 상태로 저장하고,  
이 값을 Navbar가 받아 “현재 활성화된 메뉴”를 표시하도록 하는 로직입니다.