AI-Assisted Full Stack Development

React 18 + Spring Boot 3 + AI

강사 소개

강 연 경

- 데이터 분석 및 프로젝트 기획, PI /PM 다수 수행

주요 경력

- 공공기관·연구원 웹서비스 및 데이터 플랫폼 다수 구축
- 생명·의료 데이터 분석 및 AI 적용 프로젝트 수행

학력

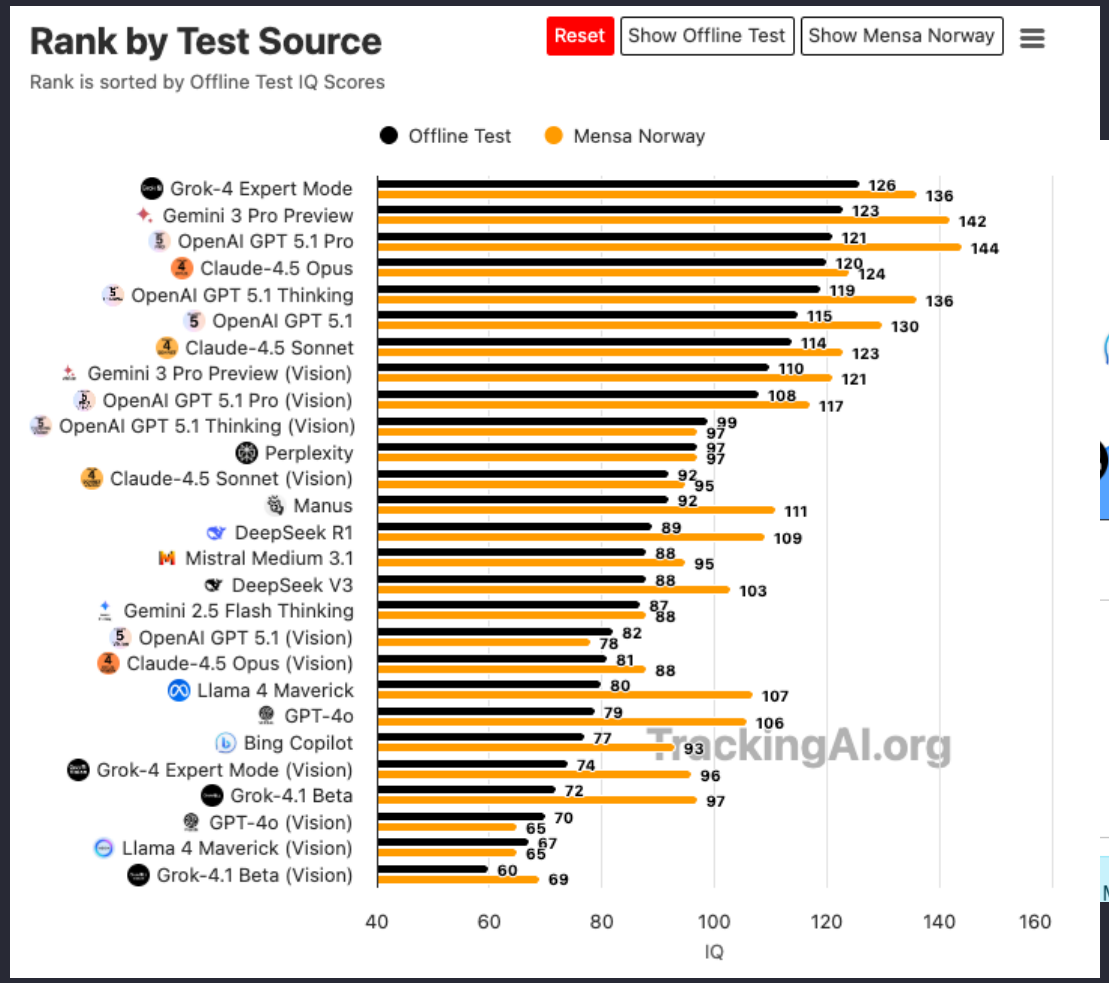
- 연세대학교 정보대학원 빅데이터 전공 박사과정
- 숭실대학교 소프트웨어공학 석사

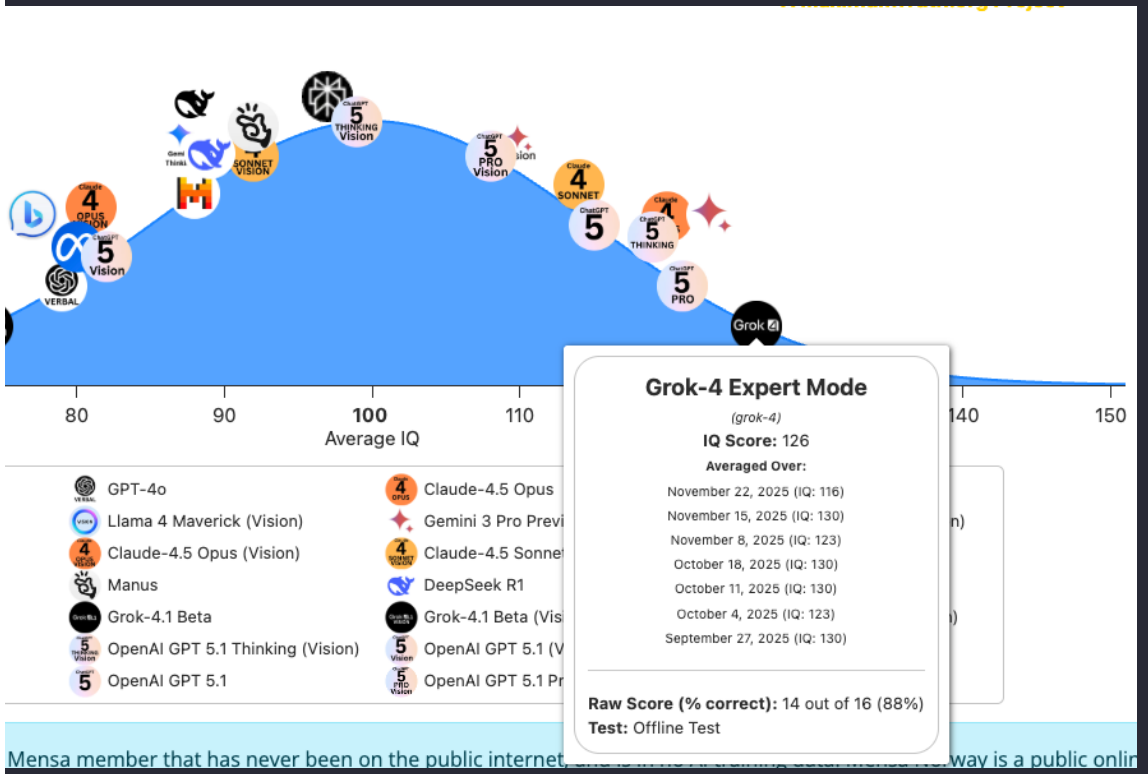
수업스케줄

Layer	Technology	Why?
Frontend	React 18, TS, Vite, Tailwind	빠른 렌더링, 타입 안정성
Backend	Spring Boot 3.2, Java 17, JPA	표준 아키텍처, 유지보수성
AI & Tool	Copilot, ChatGPT, Docker	생산성 극대화, 배포 용이성

Week 4 : Real-world Project

LLM은 이미 똑똑하다


Rank by Test Source
Reset
Show Offline Test
Show Mensa Norway
Rank is sorted by Offline Test IQ Scores
Offline Test
Mensa Norway
Grok-4 Expert Mode 126 136
Gemini 3 Pro Preview 123 142
OpenAI GPT 5.1 Pro 121 144
Claude-4.5 Opus 120 124
OpenAI GPT 5.1 Thinking 119 136
OpenAI GPT 5.1 115 130
Claude-4.5 Sonnet 114 123
Gemini 3 Pro Preview (Vision) 110 121
OpenAI GPT 5.1 Pro (Vision) 108 117
OpenAI GPT 5.1 Thinking (Vision) 99 97
Perplexity 97 97
Claude-4.5 Sonnet (Vision) 92 95
Manus 92 111
DeepSeek R1 89 109
Mistral Medium 3.1 88 95
DeepSeek V3 88 103
Gemini 2.5 Flash Thinking 87 88
OpenAI GPT 5.1 (Vision) 82 78
Claude-4.5 Opus (Vision) 81 88
Llama 4 Maverick 80 107
GPT-4o 79 106
Bing Copilot 77 93
Grok-4 Expert Mode (Vision) 74 96
Grok-4.1 Beta 72 97
GPT-4o (Vision) 70 65
Llama 4 Maverick (Vision) 67 65
Grok-4.1 Beta (Vision) 60 69
TrackingAI.org
40 60 80 100 120 140 160
IQ


80 90 100 110 140 150
Average IQ
GPT-4o
Llama 4 Maverick (Vision)
Claude-4.5 Opus (Vision)
Manus
Grok-4.1 Beta
OpenAI GPT 5.1 Thinking (Vision)
OpenAI GPT 5.1
Claude-4.5 Opus
Gemini 3 Pro Previ
Claude-4.5 Sonne
DeepSeek R1
Grok-4.1 Beta (Vis
OpenAI GPT 5.1 (V
OpenAI GPT 5.1 Pr
Grok-4 Expert Mode
(grok-4)
IQ Score: 126
Averaged Over:
November 22, 2025 (IQ: 116)
November 15, 2025 (IQ: 130)
November 8, 2025 (IQ: 123)
October 18, 2025 (IQ: 130)
October 11, 2025 (IQ: 130)
October 4, 2025 (IQ: 123)
September 27, 2025 (IQ: 130)
Raw Score (% correct): 14 out of 16 (88%)
Test: Offline Test
Mensa member that has never been on the public internet

Vibe Coding

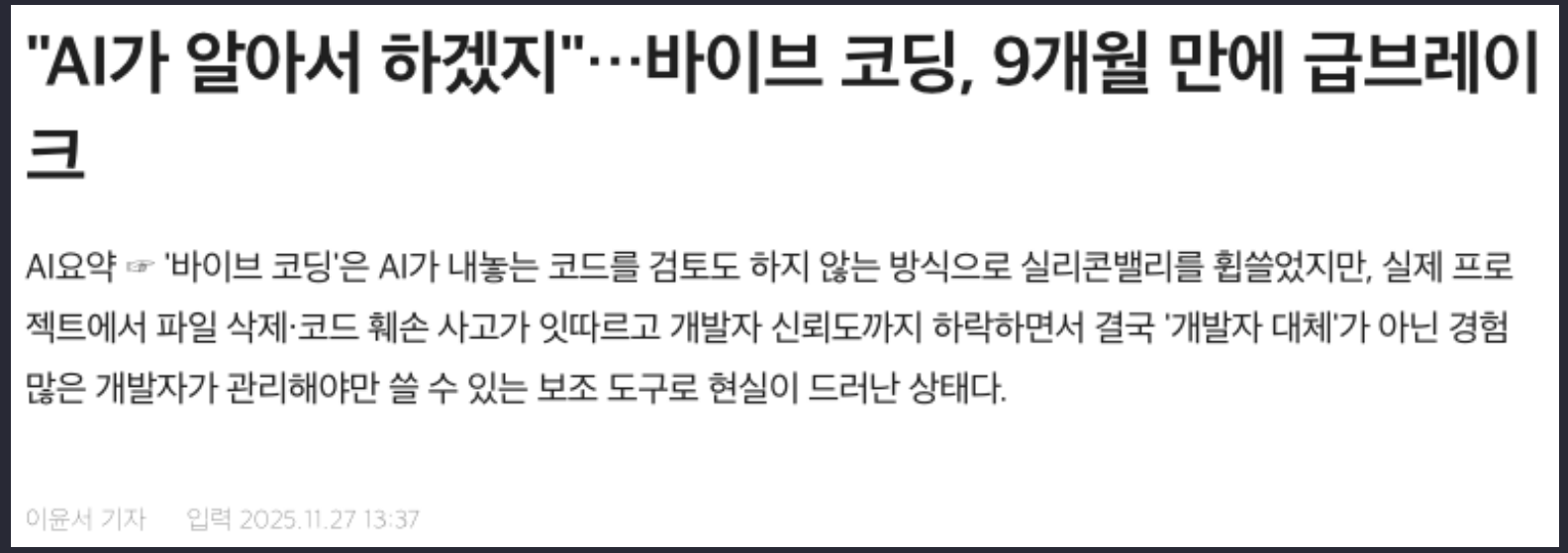
"코딩은 AI가 하고, 인간은 기분(Vibe)만 맞춘다?"

"자연어로 명령하고, AI가 짠 코드는 읽지도 않고 수락한다.
에러가 나면 복사해서 고쳐질 때까지 돌린다."

- Andrej Karpathy (Former Tesla AI Chief)

“AI 가 알아서 하겠지”의 결말

- **신뢰도 하락:** 40% ➔ 29%
- **환각 연쇄(Confabulation Cascade):** 없는 파일을 참조하거나 코드를 삭제하는 사고 발생
- **Back to Basics:** Karpathy 조차 최신 프로젝트(Nanochat)는 직접(Hand-coding) 작성함



"AI가 알아서 하겠지"…바이브 코딩, 9개월 만에 급브레이크

AI요약 ☞ '바이브 코딩'은 AI가 내놓는 코드를 검토도 하지 않는 방식으로 실리콘밸리를 휩쓸었지만, 실제 프로젝트에서 파일 삭제·코드 훼손 사고가 잇따르고 개발자 신뢰도까지 하락하면서 결국 '개발자 대체'가 아닌 경험 많은 개발자가 관리해야만 쓸 수 있는 보조 도구로 현실이 드러난 상태다.

이윤서 기자 입력 2025.11.27 13:37

우리는 "Vibe Coder"가 아닌 **"Engineering Pilot "** 로

Vibe Coder (Amateur)	Engineering Pilot (Pro)
코드를 읽지 않고 수락함	AI 코드를 **Review & Refactor**함
에러 나면 무한 프롬프팅	원인을 파악하고 **Architecture**를 수정함
간단한 기능만 구현 가능	**Spring Boot + React** 통합 시스템 구축

React ?

- UI(User Interface) 구축을 위한 JavaScript 라이브러리
- Meta(Facebook)에서 개발
- 복잡한 웹 UI를 효율적이고 예측 가능하게 만들기 위해 등장

React ?

기존 방식(=Vanilla JS, jQuery 시대)에서의 문제

DOM 직접 조작이 너무 많음	UI가 복잡해 질수록 코드도 폭발적으로 복잡해짐 **Virtual DOM으로 효율적 업데이트**
상태 관리 혼란	데이터가 바뀔 때 어떤 부분을 어떻게 업데이트해야 하는지 점점 난해 **단방향 데이터 흐름으로 예측 가능성 ↑** **State 기반 자동 렌더링**
UI 재사용 어려움	동일 컴포넌트를 여러 곳에서 쓰기 어려움 **Component 기반 구조**
성능 저하	DOM 변경이 비효율적 → 렌더링 느려짐

Vanilla JS vs React

📂 01-vanilla/counter-vanilla.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Vanilla JS Counter</title>
</head>
<body>
  <h2>Vanilla JS Counter</h2>
  <button id="btn">Click</button>
  <p id="count">0</p>

  <script>
    let count = 0;
    const btn = document.getElementById("btn");
    const countDisplay = document.getElementById("count");

    btn.addEventListener("click", () => {
      count++;
      countDisplay.innerText = count;
    });
  </script>
</body>
</html>
```

```
02-react/
├── index.html        → 브라우저가 처음 읽는 HTML (진입점)
├── package.json      → 프로젝트 정보 + 의존성 목록
├── vite.config.js    → Vite 설정 파일
└── src/              → React 실제 소스코드
    ├── main.jsx      → React 앱을 DOM에 렌더링하는 진입 스크립트
    ├── App.jsx       → 최상위 컴포넌트(UI 뼈대)
    └── Counter.jsx   → 컴포넌트(재사용 가능한 UI 단위)
```

index.html → React가 붙을 자리 제공
main.jsx → 최상위 컴포넌트(App) 렌더링
App.jsx → 화면 전체 레이아웃 담당
Counter.jsx → 상태(state) 사용 UI 컴포넌트

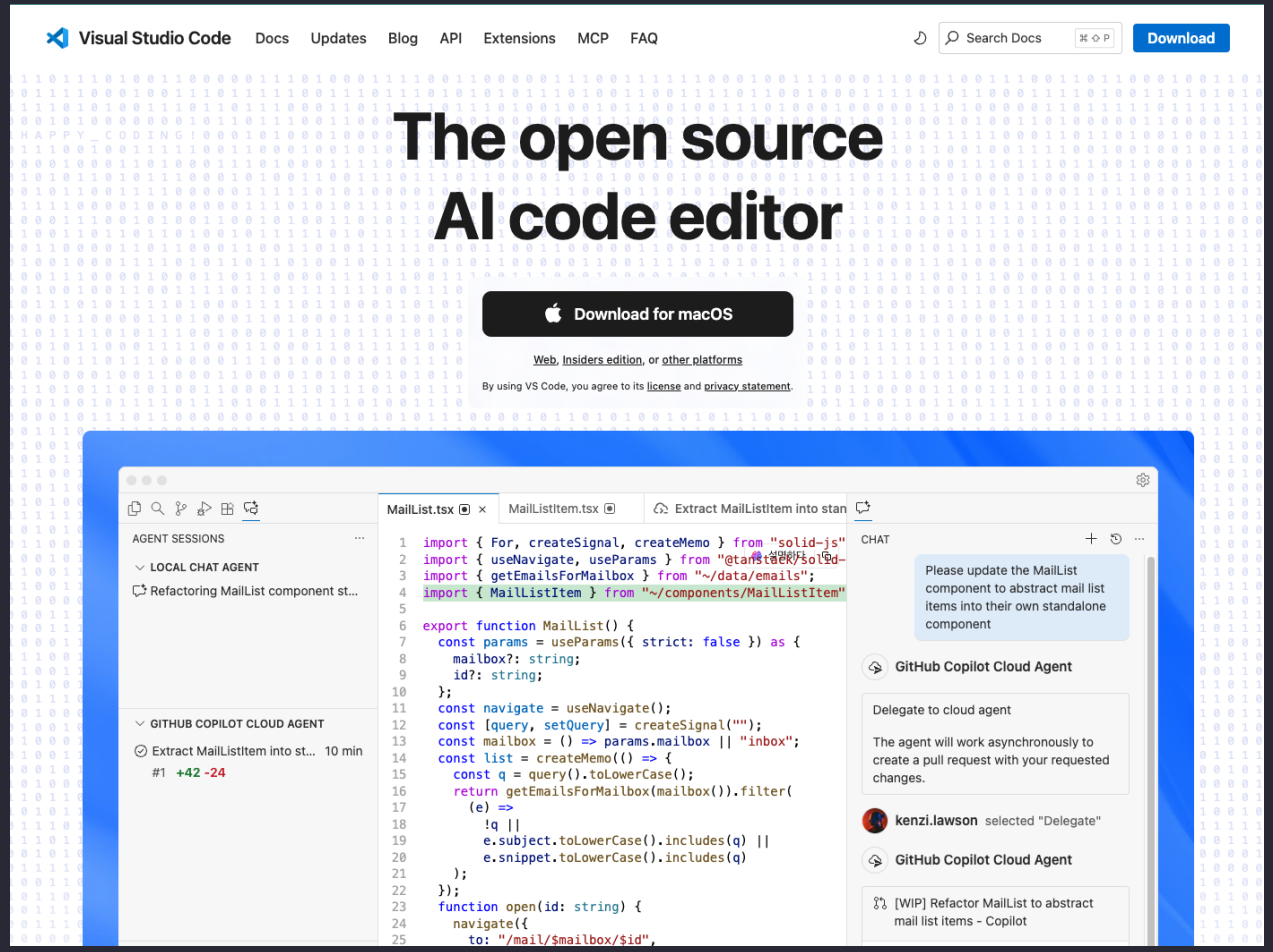
실습 환경 셋팅

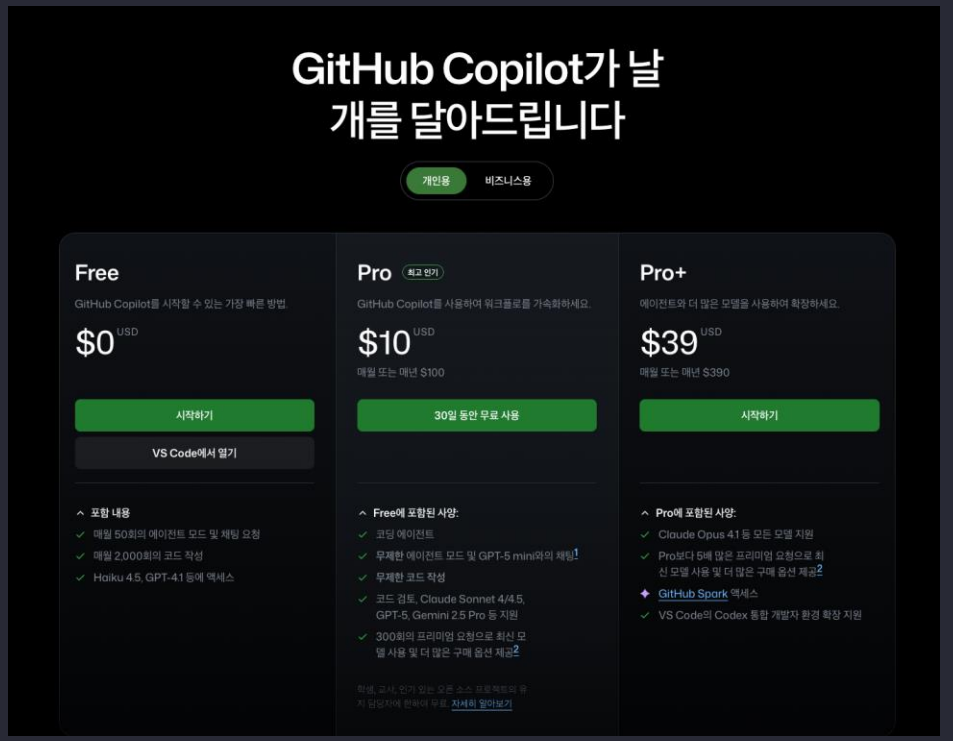
1. **Node.js (LTS):** 자바스크립트 실행 엔진
2. **VS Code Extensions:** 생산성 도구 모음
3. **pnpm:** 고속 패키지 매니저
4. **Vite Project:** 리액트 앱 생성

React 개발 환경 설정 (VS Code 기반) : https://buly.kr/Chpz17G

STEP 1 VS Code 설치

다운로드 : https://code.visualstudio.com/





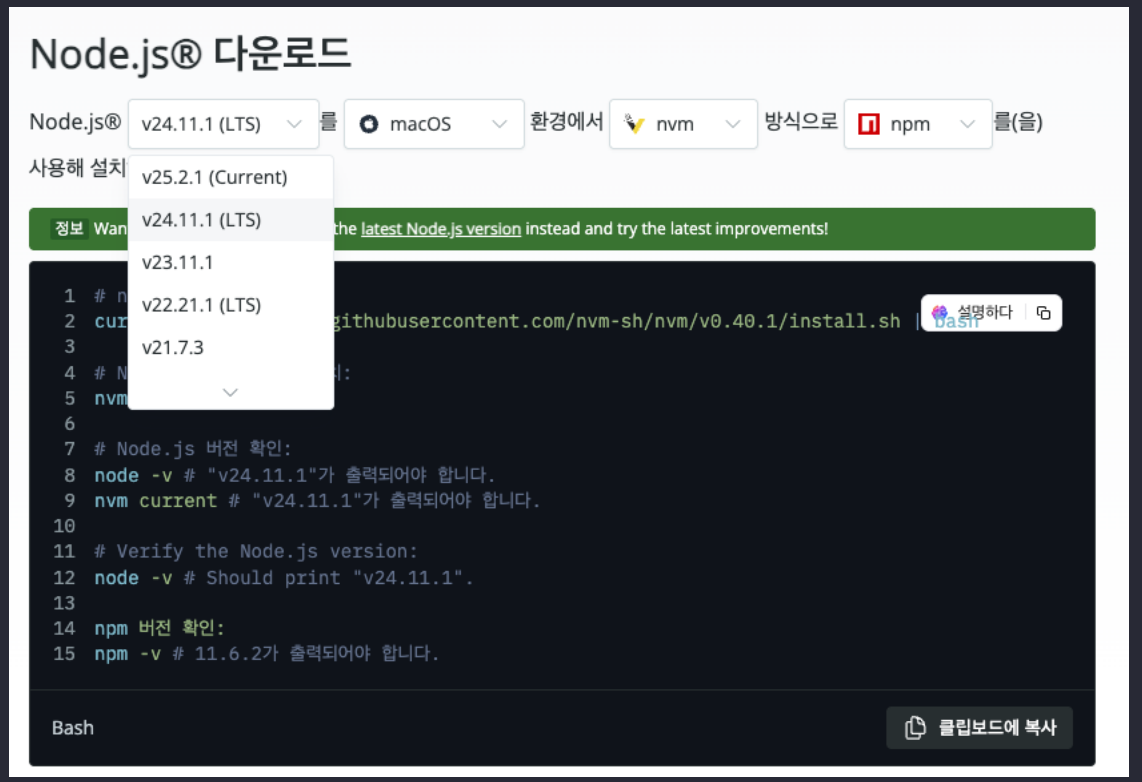
STEP 2 Node.js 설치

React를 실행하기 위한 필수 런타임입니다.

URL: nodejs.org

Version: LTS (Long Term Support) 선택

⚠ "Current" 버전은 실험적 기능이 포함될 수 있어 비추천



STEP 3 VS Code Extensions 설치

확장 프로그램명 (검색어)	용도 및 설명
Reactjs code snippets	React 컴포넌트 코드 자동 생성
Tailwind CSS IntelliSense	Tailwind CSS 클래스 자동 완성, 색상을 미리 보기
Prettier - Code formatter	저장 시 코드 자동 정렬 (팀 협업 필수)
ESLint	코드의 잠재적인 오류나 안티 패턴을 미리 잡아줌
GitHub Copilot	AI 페어 프로그래밍

STEP 4 Package Manager (pnpm)

npm보다 빠르고 디스크를 적게 쓰는 **pnpm**을 사용합니다.

```
# 터미널에 입력하여 전역 설치

npm install -g pnpm

# 설치 확인

pnpm -v
```

! 보안 정책으로 설치가 안 될 경우, 기본 npm을 사용해도 무방합니다.

STEP 5 Project Creation (Vite)

CRA(Create-React-App) 대신 100배 빠른 Vite를 씁니다.

```
# 1. 프로젝트 생성 (React 템플릿)
pnpm create vite my-first-app --template react

◇ Install with pnpm and start now?
│  Yes  <- 선택 시 아래 명령어 실행 X

# 2. 폴더 이동 및 패키지 설치
cd my-first-app
pnpm install

# 3. 개발 서버 실행
pnpm run dev
```

🚀 실행 결과 확인

http://localhost:5173

my-first-app > index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<link rel="icon" type="image/svg+xml" href="/vite.svg" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>my-first-app</title>
</head>
<body>
<div id="root"></div>
<script type="module" src="/src/main.jsx"></script>
</body>
</html>
```

my-first-app > src > main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'
```

컴포넌트와 정적 리소스 로드

```
createRoot(document.getElementById('root')).render(
<StrictMode>
<App />
</StrictMode>,
)
```

my-first-app > src > App.jsx

```jsx
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

function App() {
  const [count, setCount] = useState(0)

  return (
    <>
      <div>
        <a href="https://vite.dev" target="_blank">
          <img src={viteLogo} className="logo" alt="Vite logo" />
        </a>
        <a href="https://react.dev" target="_blank">
          <img src={reactLogo} className="logo react" alt="React logo" />
        </a>
      </div>
      <h1>Vite + React</h1>
      <div className="card">
        <button onClick={() => setCount((count) => count + 1)}>
          count is {count}
        </button>
        <p>
          Edit <code>src/App.jsx</code> and save to test HMR
        </p>
      </div>
      <p className="read-the-docs">
        Click on the Vite and React logos to learn more
      </p>
    </>
  )
}

export default App
```

여러 요소를 반환한다면 하나의 상위 요소 안에 넣거나 <React.Fragment></React.Fragment> 이용할 수 있다. 빈 JSX 태그와 비슷한 더 간단한 프래그먼트 구문을 사용하기도 함

소스 코드 끝에는 컴포넌트를 내보내는 export default 문이 있으며 import 를 이용하여 다른 컴포넌트에서 이용할 수 있다