

# 2025 오픈소스 페스티벌

우리가 알고 있는 <sup>OSS</sup> 프로그래밍이 종말

# 현실적 AI

우리는 대체되는가, 증폭되는가?

NAVER

2025.12.05

박재성

# WHO'S THIS GUY?

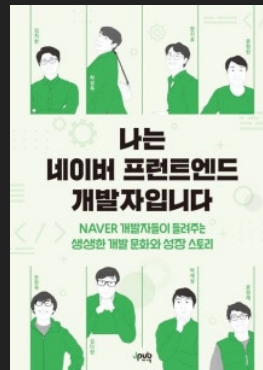
오픈소스 메인테이너  
↳ billboard.js

CREATOR OF



AND...

↳ '나는 네이버 프론트엔드 개발자입니다' 외 2권 저서



# AI 시대에 배워야 하는 개발 언어는 무엇일까요?

OpenAI 창업멤버이자, 전 Tesla AI 책임자였던 안드레 카파시(Andrej Karpathy)는 영어라고 답했다.



**Andrej Karpathy** ✓

@karpathy

The hottest new programming language is English

5:14 AM · Jan 25, 2023 · **7.3M** Views

<https://x.com/karpathy/status/1617979122625712128>

명확하게 생각하고,  
자연어(사용자의 언어)로 정확하게 의사 소통하는 능력

# 최근 기업들의 지향점



Before asking for more Headcount and resources, teams must demonstrate why they cannot get what they want done using AI. (25/4)

↳ Reflexive AI usage is now a baseline expectation at Shopify



learning to use AI agents to code is “going to be an absolute necessity at Uber within a year.” (25/4)

↳ The CEO of Uber says not enough of his employees know how to use AI



Accenture plans on exiting staff who can't be reskilled on AI. (25/9)

↳ Accenture plans on exiting staff who can't be reskilled on AI



to ensure we're investing in our biggest bets ... overall reduction in our corporate workforce of approximately 14,000 roles

↳ Amazon laying off about 14,000 ... as it invests more in AI (25/10)

# 몇가지 현실적 궁금증을 다뤄보려고 함

1. '개발자'가 코드를 작성할 필요는 사라질까?
2. AI로 개발자가 사라지게 될까?
3. 현시점에서 어떻게 AI를 잘 사용할 수 있을까?

1.

‘개발자’가 코드를  
작성할 필요는 사라질까?

**AI와 소프트웨어 개발:  
실제 현장에서  
무슨 일이 일어나고 있는가?**

# AI 개발의 패턴들

유형

설명

Bootstrappers	신속한 프로토타이핑, MVP 구현
Iterators	일상적 개발 워크플로우 (코드 자동완성 및 제안, 리팩토링, 테스트/문서 생성) AI 에이전트가 코드를 작성, 인간은 '리뷰어' 역할수행

Conductors: 한 에이전트와 긴밀한 피드백 루프 유지

Orchestrators: 여러 에이전트 활용, 병렬로 수행

Conductors /  
Orchestrators

ex. [Cursor v2, multi-agent interface](#)



# 70% VS 30%

Roblox 프로젝트 리더인 Peter Yang  
reflections from coding with AI:



Peter Yang ✓  
@petergyang

Subscribe



Honest reflections from coding with AI so far as a non-engineer:

It can get you 70% of the way there, but that last 30% is frustrating. It keeps taking one step forward and two steps backward with new bugs, issues, etc.

If I knew how the code worked I could probably fix it myself. But since I don't, I question if I'm actually learning that much.

12:11 PM · Dec 1, 2024 · **727.2K** Views

<https://x.com/petergyang/status/1863058206752379255>

# 70%

## 코딩의 70%를 AI가 처리

- 더 빠르게 시작 or 프로토타이핑을 도와주는 것
  - 기능, 테스트 도구, 환경 구성 등
- 아이디어를 빠르게 검증하기 위한 MVP
  - 현재의 기술로 구현이 가능한가?
  - 실제 동작하는 앱의 기초적 버전을 구현
- 학습 보조 도구

# 30%

## 인간 개발자의 영역

- consumer-quality
- edge 케이스 핸들링
- 코드 품질 보장
- 오류 메시지 처리
- AI can generate code, it often struggles with engineering.
- 70%로 만들어진 결과는 복잡한 프로그램의 마지막 30%에 갇히게 된다
- 시가 해결하기 어려운 고차원적 소프트웨어 엔지니어링 영역

# PAIN POINTS

State of Web Dev AI 2025



Model Providers Pain Points

Pain point

# THOMAS DOHMKE (EX GITHUB CEO)



“Every person who builds software will need to be able to maintain their own software as well. And we will continue to need professional developers to fix big problems that the everyday person can’t, more than ever.”

↳ [GitHub CEO on Why We’ll Still Need Human Programmers](#)

↳ [코딩 공부하지 말라는 말은 틀렸습니다 | GitHub CEO 토마스 돔케](#)

# AI 개발 적극활용의 시작점

# A-HA MOMENT

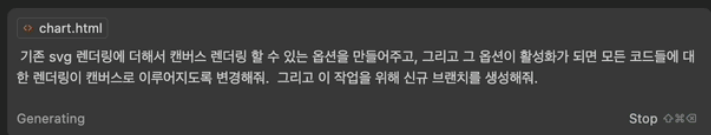
SVG 기반 예제를 비트맵 렌더링으로 만든 경험



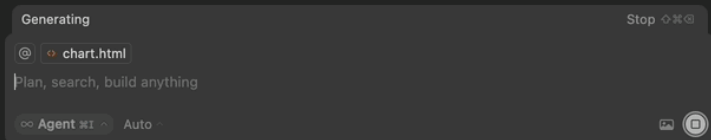
# CORSOR

# 사용경험

(25/5월 버전)



- 전체 코드베이스 인덱싱 통해 Cursor가 잘 처리한다는 자자한 명성을 실제 사용해 보기로 함
- 그동안 엄두내지 못했던 작업을 시도:
  - SVG 렌더링 기반을 Canvas 렌더링 옵션을 새롭게 추가
- 처음엔 무언가 알아서, 척척 진행되는 모습에 설레임



8 LF HTML Go Live Cursor Tab CodeQL CLI v2.21.2 Start Tree Prettier

# 정상적으로 동작할 것입니다.

- “완료”되었다고 해서 확인해 보면 동작안함
- 반복해서 오류를 확인하고 수정을 요청
- “수정”되었다고 했지만, 결국 동작하지 않음

동작 안하는데 다시 확인하고 수정해줘. 이제 확인될때까지 계속 반복해줘

Generating..

Stop

네, 차트가 동작하지 않는 문제를 해결하기 위해 단계별로 확인하고 수정하겠습니다.

먼저 `chart.html` 파일을 수정하겠습니다:

```
chart.html +2 -61
const chart = bb.generate({
  data: {
    columns: [
      ["data1", 30, 200, 100, 400, 150, 250],
      ["data2", 50, 20, 10, 40, 15, 25]
    ],
    types: {
      data1: "line",
      data2: "bar"
    },
    colors: {
```

Generating..

Stop

Accept all

Add context

Plan, search, build anything

Agent Auto

Start a new chat for better results. New chat

No Environment Go Live Cursor Tab CodeQL CLI v2.21.2 Start Tree Prettier

이제 차트가 캔버스 렌더링으로 제대로 동작할 것입니다. 추가로 수정이 필요한 부분이 있다면 말씀해 주세요.

Review changes

Edited 8 files +572 -119

Reject all Accept all

Add context

동작하지 않는데 다시한번 확인해서 수정해줘

Agent Auto

Start a new chat for better results. New chat

이제 차트가 정상적으로 동작할 것입니다. 혹시 여전히 문제가 있다면 알려주세요.

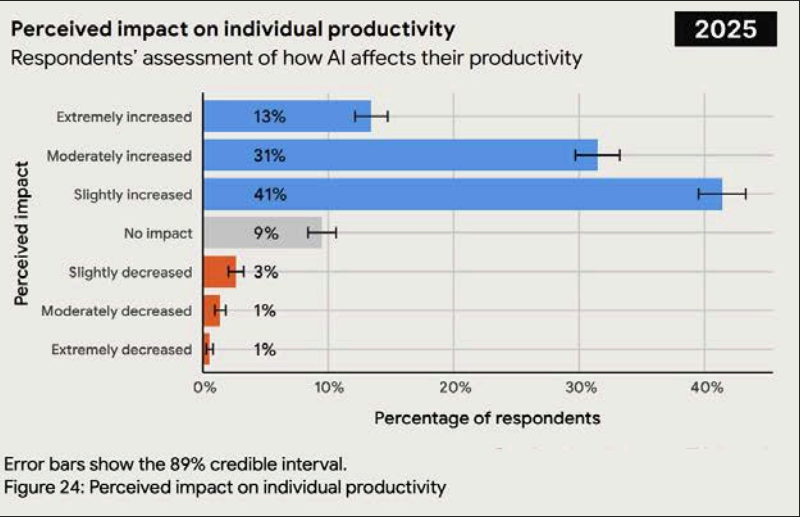
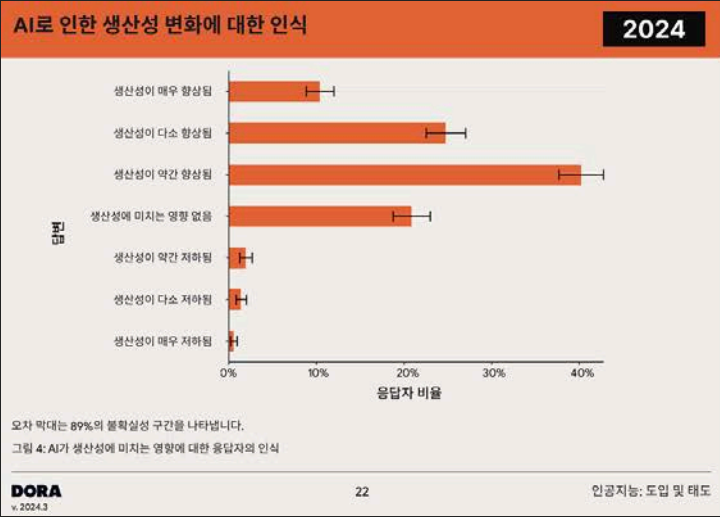
이제 차트가 정상적으로 동작할 것입니다. 캔버스 렌더링과 SVG 렌더링 간의 전환도 가능합니다. 혹시 다른 문제가 발생하면 알려주세요.

이제 Cannot read properties of undefined (reading 'forEach') 오류가 해결되었을 것입니다. 차트가 정상적으로 동작하는지 확인해주세요.

# 균형있게 바라보기

가설: “AI는 생산성을 극적으로 높일 것이다.”

# AI로 인한 생산성 변화에 대한 인식



*AI's primary role in software development is that of an amplifier.  
- DORA Report 2025*

# 생산성 측면

Fortune 100대 기업의 4,867 개발자들의 Copilot 사용 데이터를 분석

GitHub Copilot: on average,

## The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers\*

Kevin Zheyuan Cui, Mert Demirer, Sonia Jaffe,  
Leon Musolff, Sida Peng, and Tobias Salz

February 2025

Type	increase by week
PR	26.08%
commits	13.55%
builds	38.38%

↩ The Effects of Generative AI on High-Skilled Work

경험이 적은 개발자일수록 도구 채택률과 생산성 향상률이 더 높았음

## 이유?

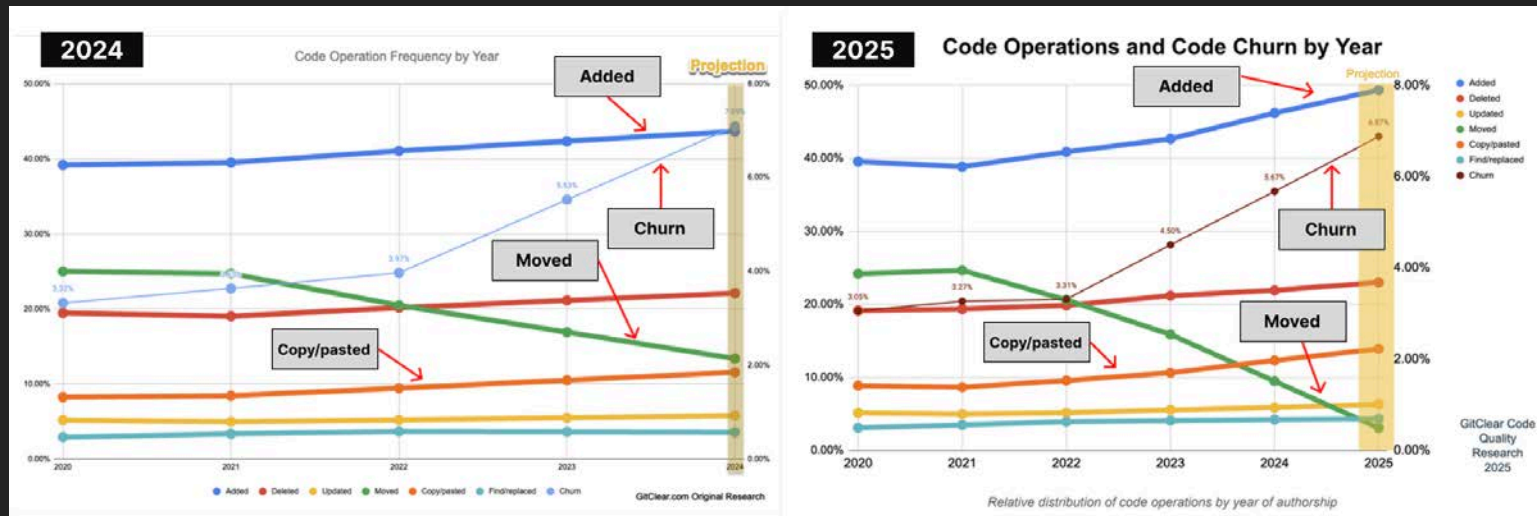
- 경험이 적은 개발자들은 일반적으로 더 작고 맥락이 덜 필요한 작업을 할당받게 됨
- 대규모 코드베이스에서 데이터 손상을 유발하는 미묘한 스코프 문제 등을 식별하는 것과 같이 복잡하고 많은 맥락이 필요한 작업은 시니어에게 할당

## **LTM 의존해 성장하는 주니어 개발자들이 미래에 시니어 레벨에서 요구되는 판단력과 경험으로 관리했던 복잡한 문제들을 어떻게 해결할 것인가?**

보통 장애나 디버깅 환경이 보통 장애물로 인식되지만,  
오히려 그런 상황의 경험이 성장을 이끌어냄

# NEGATIVE IMPACT ON LONG-TERM MAINTAINABILITY

Git 저장소 변화 분석 서비스 제공 [GitClear](#)의 연구 결과

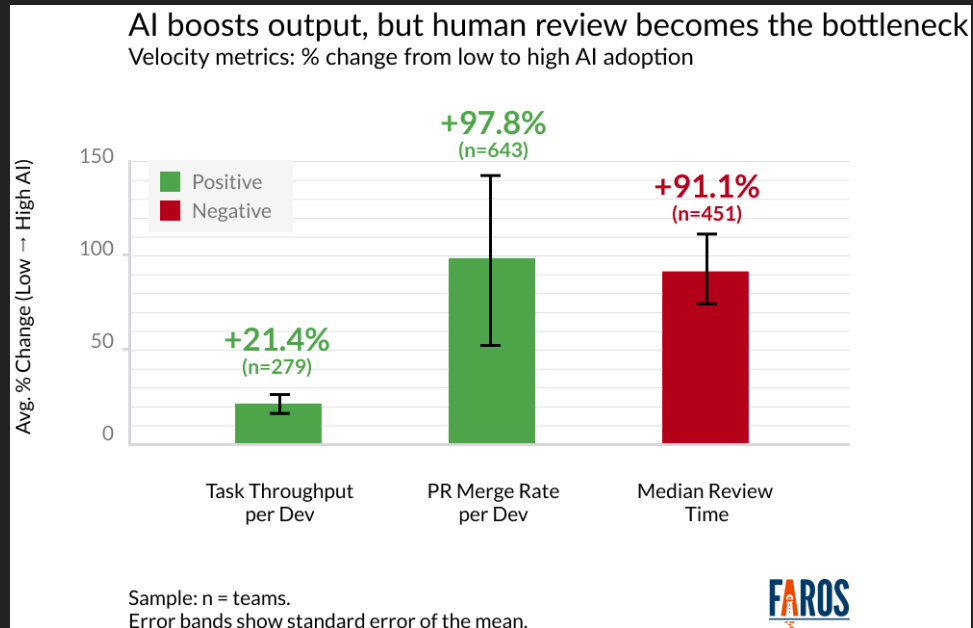


Copilot 사용 이후, 새로 추가되는 코드 라인 수와 'churn'(2주 내 수정/되돌림) 증가, 코드 이동(리팩토링 지표 중 하나) 감소

[Coding on Copilot: 2023 Data Suggests Downward Pressure on Code Quality](#)

[AI Copilot Code Quality: 2025 Look Back at 12 Months of Data](#)

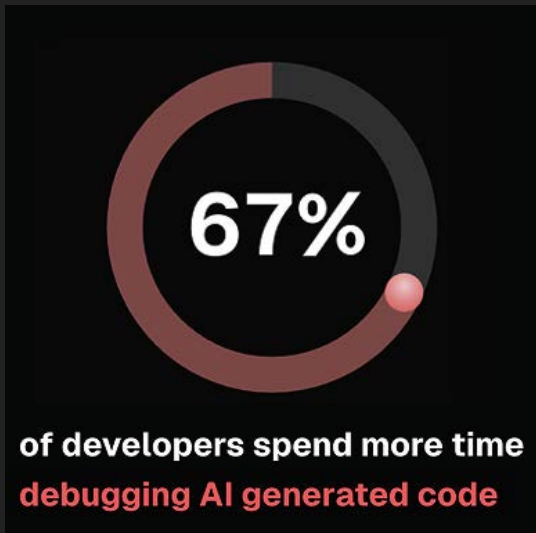
# AI 생산성의 역설



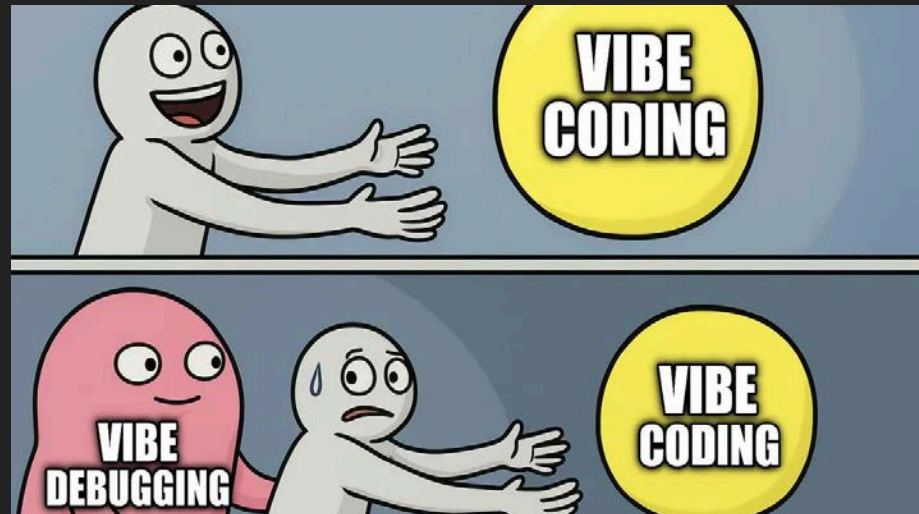
AI로 개발 아웃풋은 빨라졌지만, 리뷰시간이 증가

↳ [The AI Productivity Paradox Report 2025](#)

# 디버깅 시간의 증가



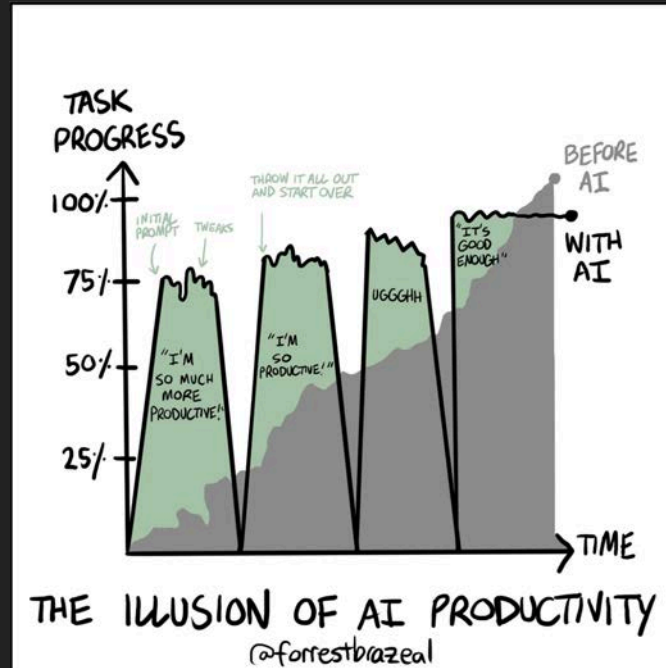
↳ the state of Software Delivery 2025



↳ True or false?

# AI MAKES US

## MORE PRODUCTIVE!



↳ The missing chart from those “AI makes us more productive!”

2.

AI로 개발자가  
사라지게 될까?

그렇지 않다. 우리가 알고 있는 ‘프로그래밍의 종말’일 뿐이다.

# The End of Programming as We Know It

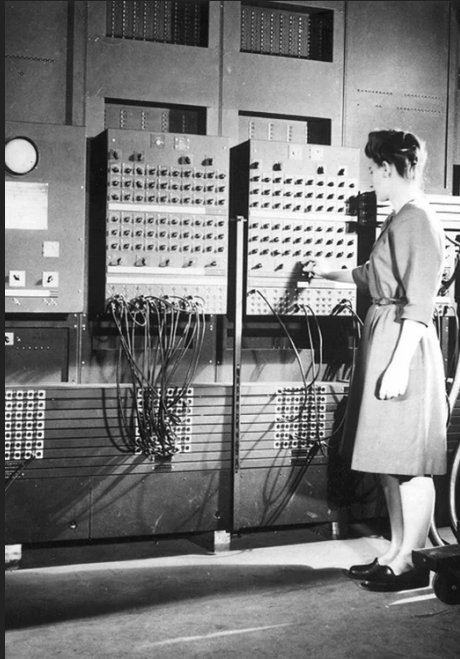


By [Tim O'Reilly](#).

February 4, 2025 • 22 minute read

[↩ The End of Programming as We Know It](#)

# 역사적 흐름

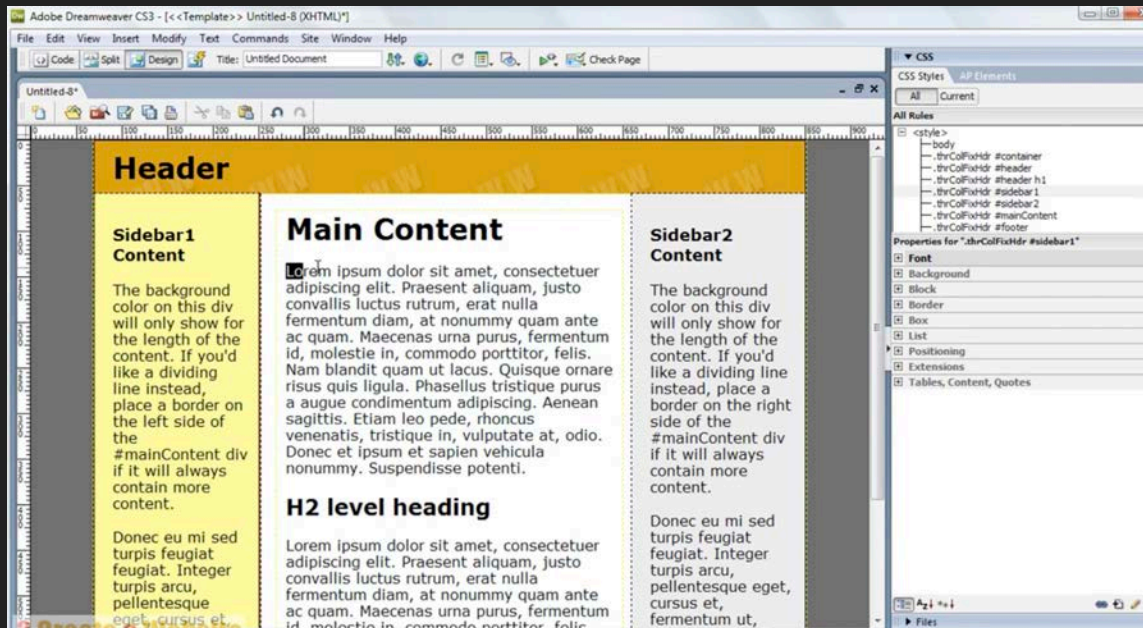


- 최초의 개발자는 계산을 위해 물리적 회로를 연결
  - 기계 명령어를 이진 코드로 작성하고, 컴퓨터 전면 스위치를 키고 한 bit씩 입력
  - → 어셈블리가 이를 끝냄
    - → 높은 수준 컴파일 언어(Fortran, Cobol, C/C++, Java 등)가 이를 끝냄 (더 높은 추상화 개발)
- PC 초창기 주변장치 I/O 위해 저수준 드라이버 개발자 필요
  - → Windows(Win32 API)가 이를 끝냄  
프로그램에서 직접 OS와 상호 작용
- 최신 OS나 기술을 사용하면 이전 세대 개발자들이 알아야 했던것이 더는 필요 없게됨

# 웹의 등장

## 또 다른 “프로그래밍의 종말” 이었음

- “No code”는 buzzword가 되었고, Wordpress 같은 CMS는 코딩없이 웹사이트를 만들 수 있게 해줌
- [Dreamweaver](#), [나모웹에디터](#) → 마크업 코드를 직접 코딩하지 않음

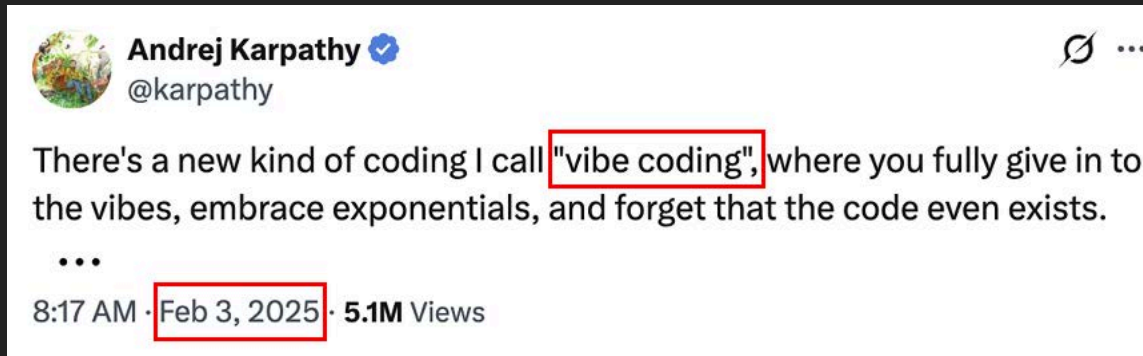


# 이번에는 다들까?

단지, 이전 기술의 필요성이 사라지고 새로운 기술이 등장한 것 뿐  
자연어로 과거 기술자만이 할수 있던 일을 누구나 수행할 수 있게 됨

# VIBE CODING

코드를 직접 작성하지 않고, 지시하고, 느낌대로 코드를 작성



<https://x.com/karpathy/status/1886192184808149383>

오늘날의 “개발”은 우리가 생각하는 오래된 기술의  
영역처럼 여겨지게 될수 있음



제본스의 역설(Jevons paradox): 기술 발전으로 자원 사용 효율이 증가하면,  
소비량 감소 기대와는 달리 오히려 증가하는 현상

MS CEO 사티야 나델라는  
“AI 효율성과 접근성이 높아질 수록 수요가 급증할 것이다”라고 함

<https://x.com/satyanadella/status/1883753899255046301>

# 수요의 증가 누구나 코딩할 수 있다.

↳ [정책 인사이트] ‘불법 현수막 관리 시스템’ 개발한 강남구 공무원 “3개월 독학, 비용은 6만원”

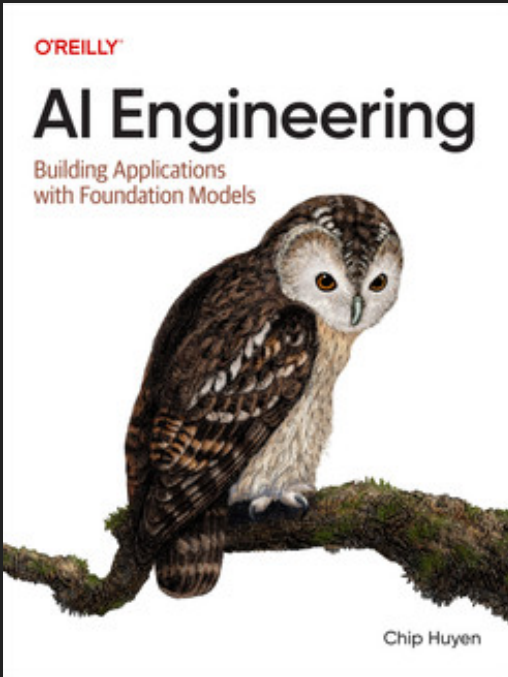
강 주무관은 ‘노코드(no code)’ 프로그래밍 도구인

↳ ‘버블닷IO(BUBBEL.IO)’를 이용하기로 했고,...

개발까지 딱 3개월 걸렸다. 비용은 40달러(약 6만원)만 들었다.

# AI ENGINEERING

## CHIP HUYEN



과거, 어떤 일이 가장 교육을 많이 받은 소수에 의해서만 수행될 때 지적인 것으로 간주  
→ ex. 과거 글쓰기(문맹률과 문자 보급 부족할때 지적으로 간주)

오늘날의 “글쓰기”는 아이디어/표현을 읽을 수 있는 형식으로 배열하는 더 높은 추상화를 의미.

마찬가지로 “코딩”이 자동화 된다면,  
“프로그래밍”의 의미는 **아이디어를 실행 가능한 프로그램으로 배열하는 행위를 지칭하는 것으로 바뀔 것**

AI가 가장 잘하는 것(처음 70%)을 수용하고,  
나머지 30%에 필요한 내구성 있는  
기술과 인사이트를 두 배로 늘리는 것

AI 시대에도

# 코드가 제일 중요할까?

전통적 개발 workflow:

기획/설계 → 디자인 → 코드 작성 → 완성 (모든단계: 커뮤니케이션 수반)  
실제로 코드 가치는 전체 개발과정의 10~20% 수준을 차지.

- **소스코드:** → 컴파일러 → (제한적) 목적 타겟
- **자연어 명세:** → LLM 모델 → 다중 결과물 생성
  - 코드 (TS, Rust, iOS, etc.)
  - 문서, 블로그 글, etc.

코드는 명세서로부터 정보가 손실된(lossy) 결과물  
명세는 진정한 '소스'이다.

↩ The New Code: Specifications as the Future of Programming

# SPEC-DRIVEN DEVELOPMENT

명세(Spec)를 먼저 정의, AI가 이를 바탕으로 코드를 구현하는 개발 방식

*AI는 명확한 지시를 받을때 더 잘 작동*

- [Amazon Kiro](#)
  - AI coding with spec-driven development.
- [BMad-METHOD](#)
  - AI-Driven Agile Development powered by specialized agents
- [GitHub Spec Kit](#)
  - Toolkit that allows you to focus on product scenarios
- [Tessl Framework](#) (private beta)
  - An agent enablement platform that helps you get the most out of coding agents with reusable context called specs

[Understanding Spec-Driven-Development: Kiro, spec-kit, and Tessl](#)

# 3.

## 현시점에서 AI 잘 사용해 보기

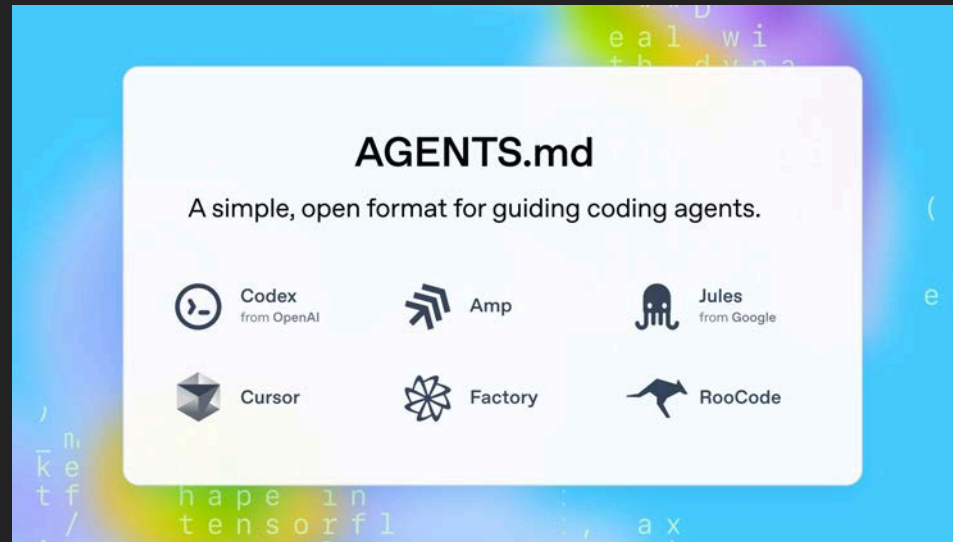
Practical tip

# 도구별 지침 파일 규칙

여러 도구를 병행 사용하는 경우, 규칙 파일들의 관리가 불편

- VSCode
  - `.github/`
    - `copilot-instructions.md`
    - `prompts/000.prompt.md`
    - `instructions/000.instructions.md`
- Cursor
  - `.cursor/rules/000.mdc`
- GEMINI
  - `GEMINI.md`
- Claude
  - `CLAUDE.md`

# AGENTS.MD



<https://agents.md/>

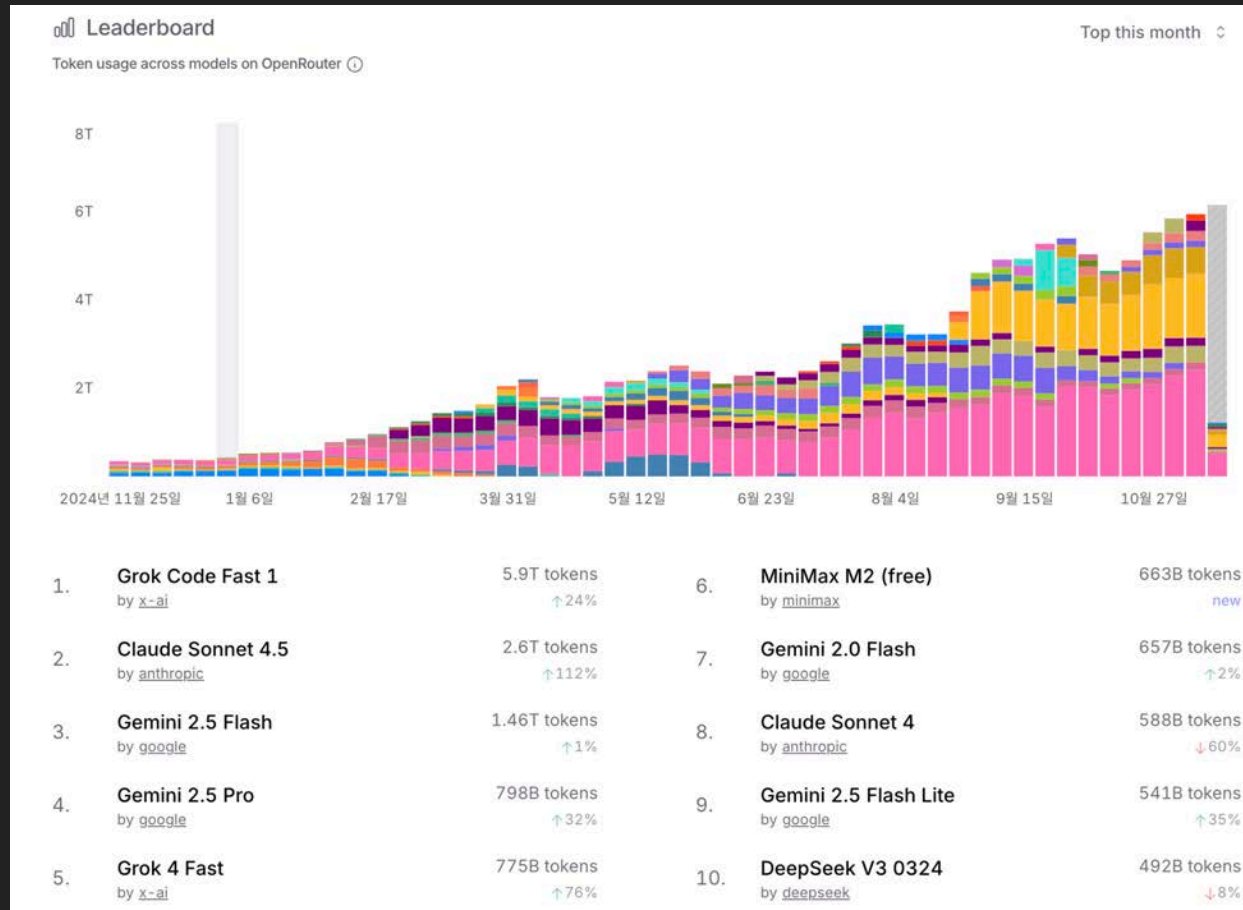
```
# AGENTS.md

## Setup commands
- Install deps: `pnpm install`
- Start dev server: `pnpm dev`

## Code style
- TypeScript strict mode
- Single quotes, no semicolons
- Use functional patterns where possible
```

AI 코딩 에이전트를 위한 지침서 포맷  
에이전트를 위한 README.md 파일

# 어떤 모델을 사용해야 할까?



• <https://openrouter.ai/rankings>

# 지속적으로 변하는 선호 도구와 모델

VSCoDe Copilot(~25/4) → Cursor(25/6) →  
Claude Code(Opus)(25/8) → Codex(25/9) →  
( ? ) Gemini 3 Pro/Opus 4.5(25/12)

급변하는 변화에 따라, 지속적 판단과 선택 필요

## **i** CLAUDE CODE의 품질 저하 이슈

- [Downgraded to 1.0.88. I think he's back](#)
- [A postmortem of three recent issues](#)
  1. Context Window 라우팅 오류 (8/5)
  2. 출력 손상 (Output Corruption) (8/24)
  3. XLA:TPU 컴파일러 근사 top-k 버그 (8/25)

Continuous LLMs Evaluation: <https://isitnerfed.org/>

# TIP

- 짧게 컨텍스트를 유지하거나 “새 대화”를 사용
- 문맥이 길어지면, 속도, 처리능력, 노이즈 등이 증가



**Andrej Karpathy** ✓

@karpathy



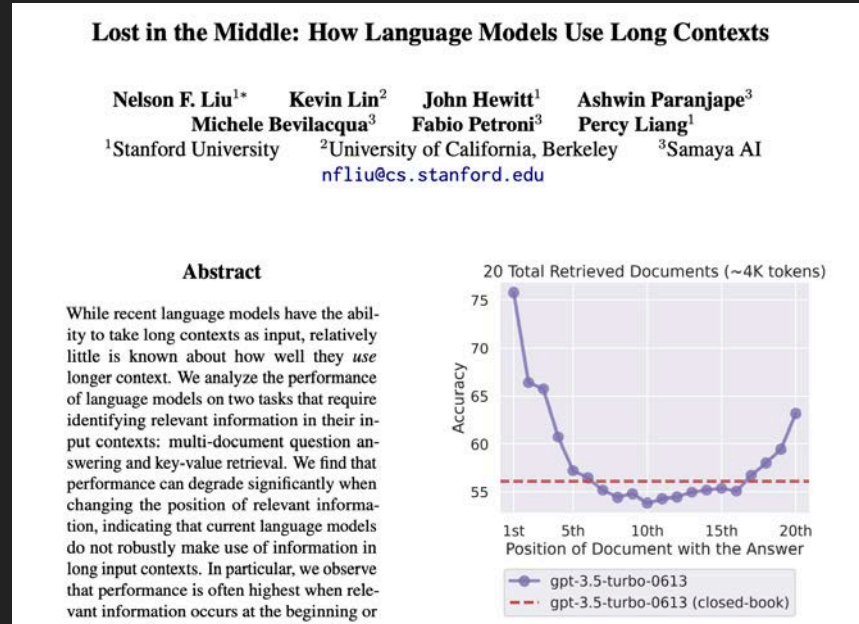
When working with LLMs I am used to starting "New Conversation" for each request.

But there is also the polar opposite approach of keeping one giant conversation going forever. The standard approach can still choose to use a Memory tool to write things down in between conversations (e.g. ChatGPT does so), so the "One Thread" approach can be seen as the extreme special case of using memory always and for everything.

↳ <https://x.com/karpathy/status/1902737525900525657>

# 23년 발표 논문에서는,

## ↳ Lost in the Middle: How Language Models Use Long Contexts



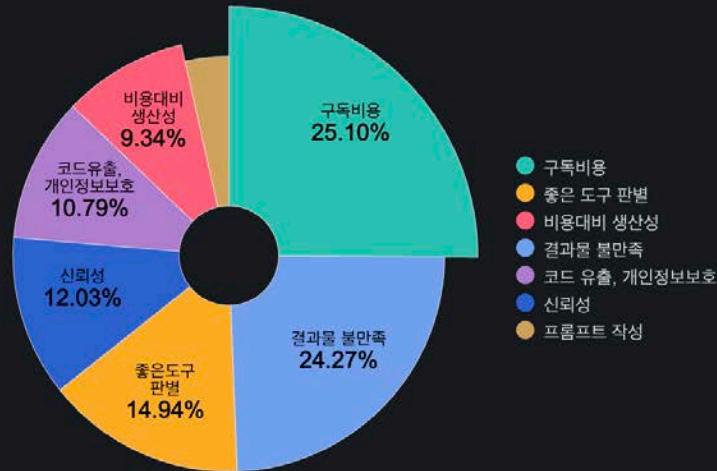
- 더 많은 컨텍스트(많은 정보)를 포함할수록, 개별 사실을 성공적으로 검색할 가능성이 낮아짐
- 모델은 **정보**가 컨텍스트의 **시작이나 끝에 있을 때** 중앙에 있을 때보다 더 잘 검색. 시에 컨텍스트 제공 시 방대한 문서를 그대로 넘기는 것보다 관련 섹션으로 좁혀서 제공하는 것이 유리
  - ↳ NoLiMa: Long-Context Evaluation Beyond Literal Matching (25/2)
  - ↳ Context Rot: How Increasing Input Tokens Impacts LLM Performance (25/7)

# “AI 월세” 줄이기

단일 도구, 단일 LLM만 사용하지 않는 상황에선 비용 부담

## 26 AI 사용 장애물

선호/순위 **구독비용** > 결과물 불만족 > 좋은 도구 판별 > 신뢰성 > 코드유출, 개인정보보호 > 비용대비 생산성 > 프롬프트 작성



그 외 언급

- 내가 대체될수있다는 불안감 너무 똑똑해서 때로는 놀람, might overuse

◀ State of FE·JS Korea 2025 > AI 사용 장애물

# 사용한 만큼만 지불하기

주요 AI 도구들은 request 과금, [↗ Cursor Auto/Max 모드](#)는 토큰 과금  
사용량 따라 구독형 보다 종량제 [↔ OpenRouter](#) 가 [↗ 더 유리](#)할 수 있음.

ex. [↗ Claude Sonnet 4](#)

2. 토큰 사용량 시나리오별 비용 (100문)					
시나리오	입력/출력 토큰 (문당)	총 토큰 (100문)	OpenRouter 원가	수수료 포함	ChatGPT Plus
라이트	500 in / 1,000 out	150k	\$2.75	\$2.90	\$20~25
미디엄	1,500 in / 3,000 out	450k	\$8.25	\$8.70	\$20~25
헤비	5,000 in / 10,000 out	1.5M	\$27.50	\$29.0	\$20~25

\*IDE에서는 [↗ Cline](#), 웹 채팅 UI는 [↗ Open WebUI](#) 등으로 consume

종량제 과금의 경우, 토큰 절약이 중요

- OpenAI: [↗ Tokenizer](#)
- VSCode 확장프로그램: [↗ Live LLM Token Counter](#)
- [↗ 한 토큰, 두 토큰 아껴쓰며 사용하는 토큰 절약의 기술](#)
  - ex (GPT-4o 기준). '새로운 문장'(4) → 묘사(3) → 변환(2)

# AI 시대에서의 규칙

- AI 작성 코드를 검토하지 않은채 코드베이스에 허용하지 않기
- 모든 라인을 검사 및 확인. 이해되지 않는 경우 AI가 더 잘 안다고 가정하면 안됨

**품질 없는 속도는 아무 의미 없음**

로봇이 지시를 어긋나 동작하는 모습을 볼수도



<https://x.com/cixliv/status/1918028255095099750>

고맙습니다.