AI 바이브/페어 개발 — CLAUDE CODE/CODEX 과정

AI 페어 개발 바이브코딩 지형도

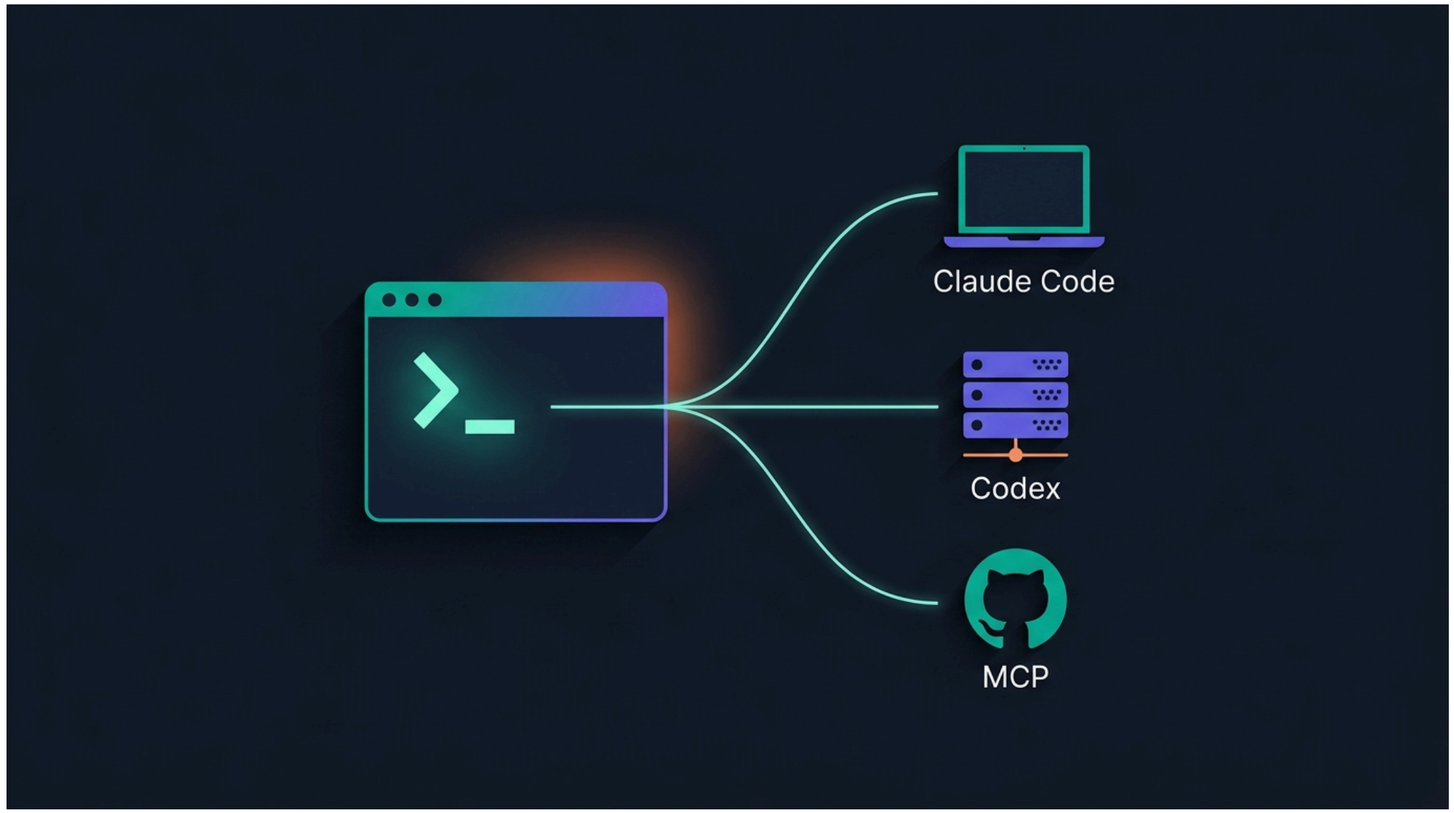
Session 0 · 바이브코딩 지형도

Claude Code · Codex · Cursor · Devin Desktop 4종 지형도 · 실무 사례 · 도구 선택 매트릭스

강사 박수현 · 🚀 젠아이랩스(GenAI Labs)

바이브코딩 2일 집중 · 프롬프트·컨텍스트·하네스 엔지니어링

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf



Session 0 · 바이브코딩 지형도

PART 1

OT — *강사·로드맵·정체성*

> 강사 소개 · 2일 로드맵 · 코스 정체성.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

강사 소개 — 박수현

👤 프로필

- **박수현** · 젠아이랩스
- AI 개발 코스 기획·강의
- Claude Code · Codex · MCP 실전 담당

🧱 만들어 온 것

- 이 코스 저장소도 **Claude Code 로 조립**
- 원고 · 슬라이드 · 데모 세 개 모두 AI 페어링 결과물

🎯 한 줄 원칙

"손대는 도구는 반드시 강의 현장에서 직접 시연한다."

슬라이드에 있는 명령어는 그날 그 자리에서 실행한다.

💡 시연 예고

- 강의 중 노트북 화면 공유
- `claude --version` · `codex --version`
- `~/.claude` 디렉토리 열어 보기

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

코스 정체성 — *AI 로 코딩한다*

이 코스는 **"AI 로 코딩한다"** 에 집중한다. 30·31 과 나란히 놓여 있고 서로 선수 관계가 아니다.

📚 30_genai_fundamentals

AI 를 안다

개론 · 아키텍처 · 활용 패러다임

- 이론 이해가 목표인 사람
- LLM 원리·평가·응용 패턴

🧪 31_genai_dev

AI 로 앱을 만든다

LangChain · RAG · Agent (4일)

- 파이썬 개발자
- 프로덕션 앱 조립

🛠️ 32_genai_vibecoding (본 코스)

AI 로 코딩한다

Claude Code · Codex · MCP (2일)

- 주니어~팀 리드 개발자
- 도구·워크플로·팀 규약

📌 AI 원리 심화 · 프로덕션 인프라 운영은 다루지 않는다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

2일 로드맵 — *Day 1 · Day 2*

🟠 Day 1 — Claude Code 축

셋업 → 프롬프트 → 컨텍스트 → 하네스

- S1 · Claude Code 셋업 + 명령어·모드·CLI
- S2 · 프롬프트 엔지니어링
- S3 · 컨텍스트 엔지니어링 (CLAUDE.md 3계층)
- S4 · 하네스 (hooks · permissions · skill)
- S5 · **메인 실습** — 개인 지식 대시보드
- S6 · **실습 ①** — Slack 요약봇 MCP 서버

🟢 Day 2 — Codex + MCP 축

승인 · CI · 팀 방법론

- S7 · Codex 개요 · AGENTS.md · sandbox
- S8 · 저장소·CI 통합 (`codex-action`)
- S9 · 바이브코딩 방법론 · 팀 규약
- S10 · **실습 ②** — PR 자동 리뷰어 에이전트
- S11 · **실습 ③** — 파일 시스템 인덱서 + RAG MCP

📌 두 도구를 대결시키지 않는다. **둘 다 다뤄야 실무 도구 지형이 잡힌다.**

PART 2

바이브코딩이란 — *용어·기원·현재*

> 카파시 트윗에서 출발해 왜 지금인지, 무엇을 바꾸는지, 어떤 우려가 있는지까지 짚는다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

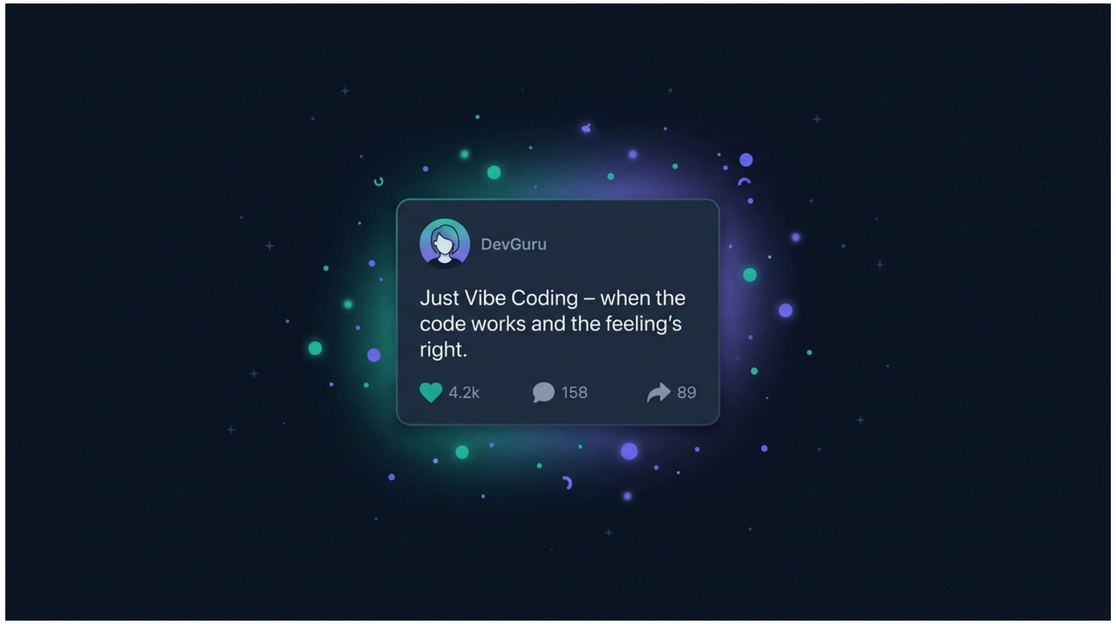
용어의 기원 — *Karpathy, 2025-02-02*

> *"There's a new kind of coding I call 'vibe coding', where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good."*
>
> — **@karpathy**, 2025-02-02

이 트윗은 며칠 만에 **450만 회 넘게** 조회됐고, "vibe coding" 은 그해 상반기 개발자 담론의 중심어가 됐다. 2025 년 하반기에는 **Wikipedia 항목**까지 올라갔다.

📖 정의 (정리)

바이브코딩이란 — 개발자가 코드를 한 줄씩 쓰는 대신, **"무엇을" 하고 싶은지를 자연어로 지시**하면 LLM 이 **"어떻게" 를 코드로 채우는** 협업 개발 방식.



트윗 한 장이 담론의 중심어를 만들었다

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

카파시가 붙인 단서 조항

같은 트윗 안에서 카파시는 이 방식을 **"주말에 던져 두는 프로젝트(throwaway weekend projects)"** 로 한정했다.

그 자신도 **"LLM 이 못 고치는 버그가 있고, 결국 코드가 내 이해 범위를 넘어간다"** 고 인정했다.

🟢 긍정 담론

- 요구를 서술하고 결과를 빠르게 얻는 실험 방식
- **학습 · 프로토타입에 강력**
- 아이디어를 코드로 가장 빠르게 검증

🔴 부정 담론

- 코드를 이해하지 못한 채 배포하는 순간
- **보안 취약점 · 유지보수 붕괴**
- **책임 소재 불명**

📌 **이 강의의 조정.** "코드를 잊어라" 가 아니라 **"코드를 읽지만 쓰는 손이 AI 다"** 로 정의를 다시 잡는다. 이 조정 없이는 실전 팀에 도입할 수 없다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

왜 하필 지금인가 — *3축이 동시에 성숙*

🧠 ① 모델 성능

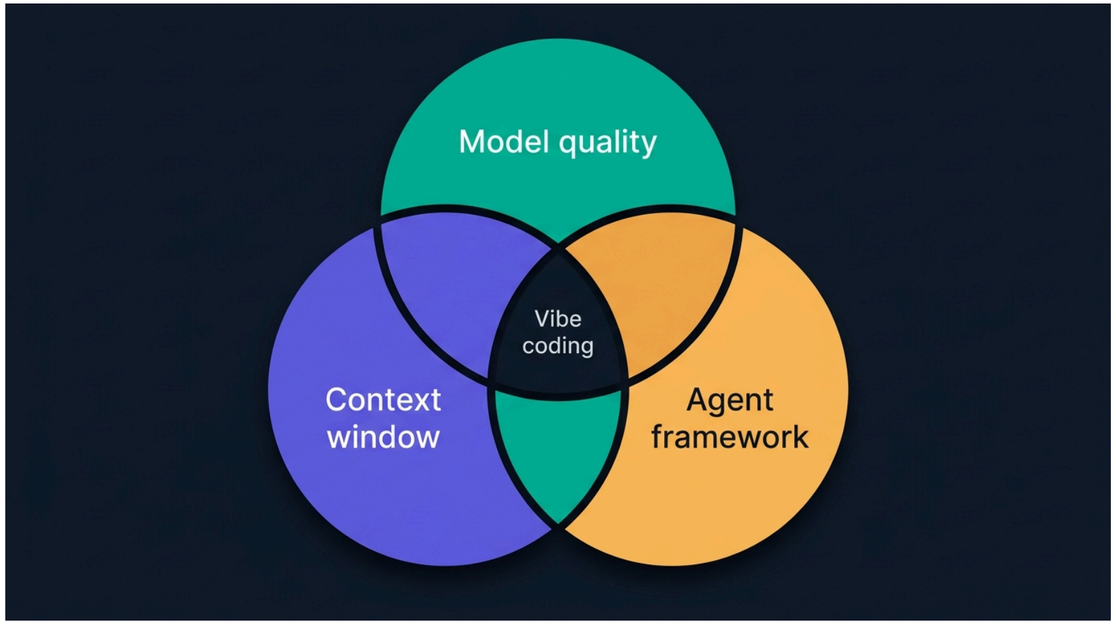
코드 생성 품질이 실무 하한을 넘음 - 2024 : GPT-4o · Claude 3.5 Sonnet - 2026 : Sonnet 4.6 · GPT-5 급이 복잡 리팩토링까지 성공

📜 ② 컨텍스트 창

리포 전체가 한 세션에 들어감 - Claude 2 **100K** (2023-11) → Sonnet 4.6 **200K** → Opus 4 **1M** (2025-05) - 리포 전체·설계 문서·회귀 로그를 한 세션에 적재

🔧 ③ 에이전트 프레임워크

LLM 이 파일·셸·git 을 직접 조작 - Anthropic **tool use GA** (2024-06) · OpenAI function calling 성숙 - **MCP 릴리즈** (2024-11) — 표준 프로토콜로 도구 공급 - 브라우저·터미널·저장소를 자동 조작



세 축이 겹치는 정중앙 — 정말 「지금」이다

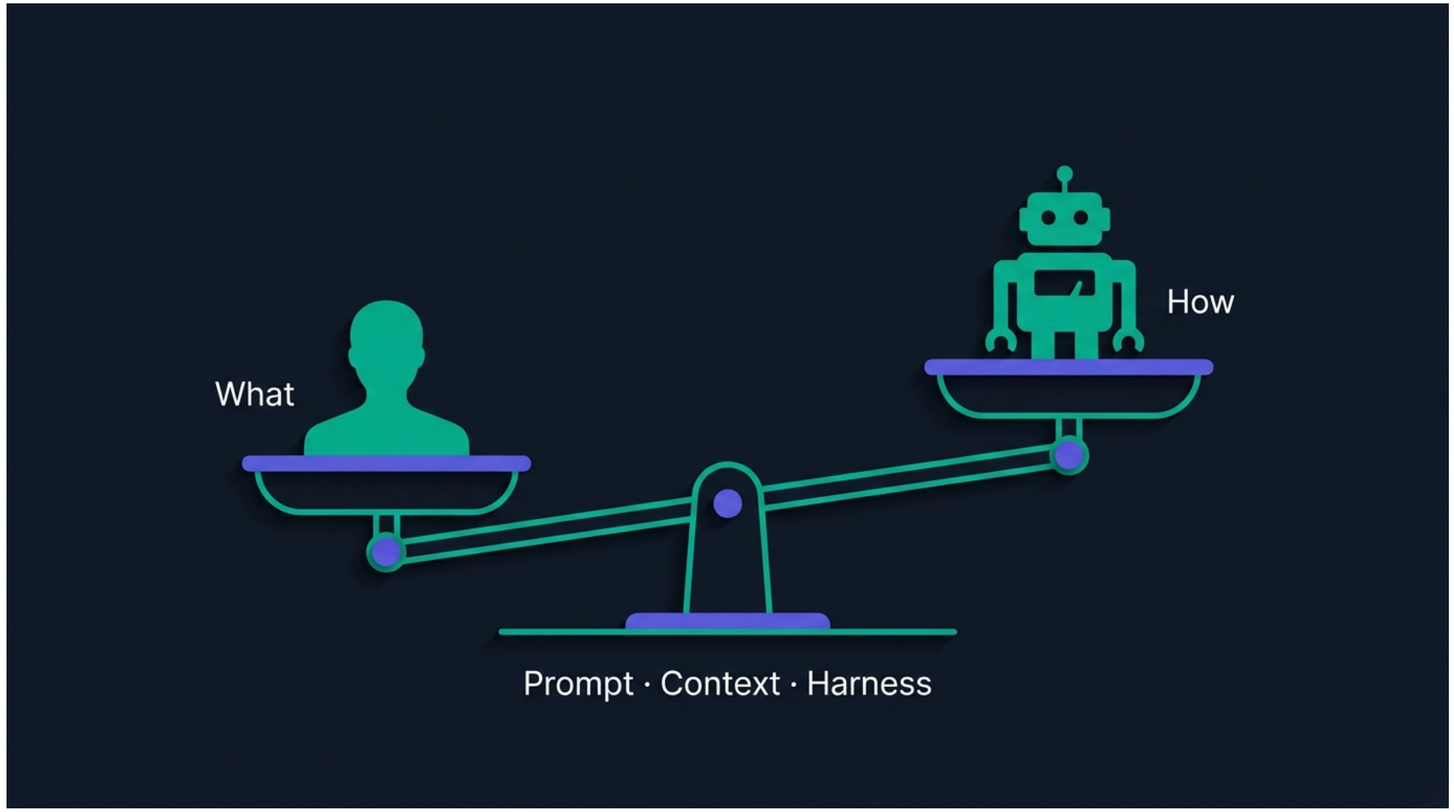
출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

"무엇을" 과 "어떻게" — *새 분업 구도*

바이브코딩은 개발자의 역할을 **"어떻게 짤까" 에서 "무엇을 시킬까" 로** 옮긴다. 이 강의 전체가 그 재분업을 전제로 한다.

대상	예전 (2020)	지금 (2026)
개발자가 결정	알고리즘 · 자료구조 · 파일 배치 · 변수명	**무엇을 만들지 · 왜 필요한지 · 어떤 제약이 있는지**
개발자가 위임	(거의 없음)	**어떻게 짤지 · 어느 API 호출할지 · 테스트 골격**
개발자가 검증	코드 리뷰 · 테스트	**AI 산출 리뷰 · 회귀 방지 · 방향 재조정**

📌 이 재분업이 잘 돌아가려면 세 가지가 필요하다. ① **프롬프트** (지시가 정확한가) · ② **컨텍스트** (AI 가 리포·규약을 아는가) · ③ **하네스** (승인·훅·권한이 안전한가) 이 코스의 **S2 · S3 · S4 가 정확히 이 세 축** 이다.



"무엇을" 을 결정하고 "어떻게" 를 위임하는 새 분업 구도

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

반대편의 우려 — *정직하게 짚고 넘어간다*

⚠️ ① 코드 이해 상실

"돌아가면 됐다" 로 넘기면 3개월 뒤 자기 코드베이스가 낯설어진다.

카파시 자신도 인정한 지점.

🔐 ② 보안·품질 하락

IBM 2026 리포트 — vibe-coded 앱은 **인증·인가·입력 검증** 에서 반복해 취약하다.

arXiv 분석 — **agent 한계 + workflow 결함 + 책임 소재 misalignment** 삼중 조합.

👥 ③ 팀 규약 붕괴

개인 도구로 시작하면 팀 안에서 **커밋 스타일 · 리뷰 수준 · 테스트 밀도** 가 사람마다 흩어진다.

S9 (방법론) 가 이 지점을 다룬다.

📌 이 코스의 대응

세 우려를 각각 **프롬프트 (S2) · 컨텍스트 (S3) · 하네스 (S4) · 팀 규약 (S9)** 으로 정면 대응한다.

PART 3

도구 지형도 — *4대 도구 조감*

> Claude Code · Codex · Cursor · Devin Desktop(구 Windsurf) 네 도구의 정체 · 차이 · UX 축.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

4대 도구 조감 — *2026 상반기 기준*

도구	제작사	UX	주 인터페이스	컨텍스트 파일	승인·안전
Claude Code	Anthropic	CLI 중심	터미널 · Web IDE 연결	`CLAUDE.md` (3계층)	Permissions · Hooks
Codex	OpenAI	CLI + Web	터미널 · `chatgpt.com/codex`	`AGENTS.md`	승인모드 3단계 · Sandbox
Cursor	Anysphere	IDE	VS Code fork	`.cursorrules` · Composer	Agent 승인
Devin Desktop *(구 Windsurf)*	Cognition	IDE	VS Code fork	Devin Local	Agent Client Protocol

⚠️ **최근 변동 (2026-06-02).** 예전 이름 **"Windsurf" 는 사라졌다.** Cognition 이 2025 년 12월 Windsurf 를 인수한 뒤 2026-06-02 자로 **Devin Desktop** 으로 리브랜딩했다. Cascade 에이전트는 **2026-07-01 EOL**, Rust 로 재작성한 **Devin Local** 이 후계다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

Claude Code — *Anthropic · CLI 기반*

정체성. 브라우저 IDE 가 아니라 **파일 시스템 · 셸 · git 을 직접 조작하는 CLI 에이전트**. 2025-02 공개, 터미널에서 `claude` 로 실행한다.

🧩 핵심 개념 4가지

📁 CLAUDE.md — 3계층 컨텍스트

- `~/.claude/CLAUDE.md` (개인·모든 프로젝트)
- `<repo>/CLAUDE.md` (프로젝트 공용)
- `<repo>/CLAUDE.local.md` (로컬 오버라이드)

팀의 코드 컨벤션 · 빌드 명령 · 리뷰 정책이 여기 담긴다.

🌱 Subagent — 병렬 세션

Task 툴로 별도 컨텍스트 창을 가진 자식 에이전트를 띄운다. 대형 리포 탐색 · 병렬 리팩토링 · 부모 컨텍스트 보호에 쓴다.

🪝 Hooks — 결정론적 훅

5시점: `PreToolUse` · `PostToolUse` · `Notification` · `SessionStart` · `Stop`

커밋 전 자동 lint · 위험 명령 차단 · 로그 수집에 쓴다.

🎯 Skill — 절차의 명시화

`.claude/skills/<name>/SKILL.md` 로 정의하는 재사용 단위. "이 리포에서 배포하는 절차" · "PR 리뷰 체크리스트" 같은 프로시저를 담아 둔다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

Claude Code — *구독·요금*

💳 Pro

$20 / 월

- 이 강의 기준
- 개인 실습 충분

💳 Max 5x

$100 / 월

- 팀 리드 · 시니어
- 대형 리포 · Subagent 병렬

💳 Max 20x

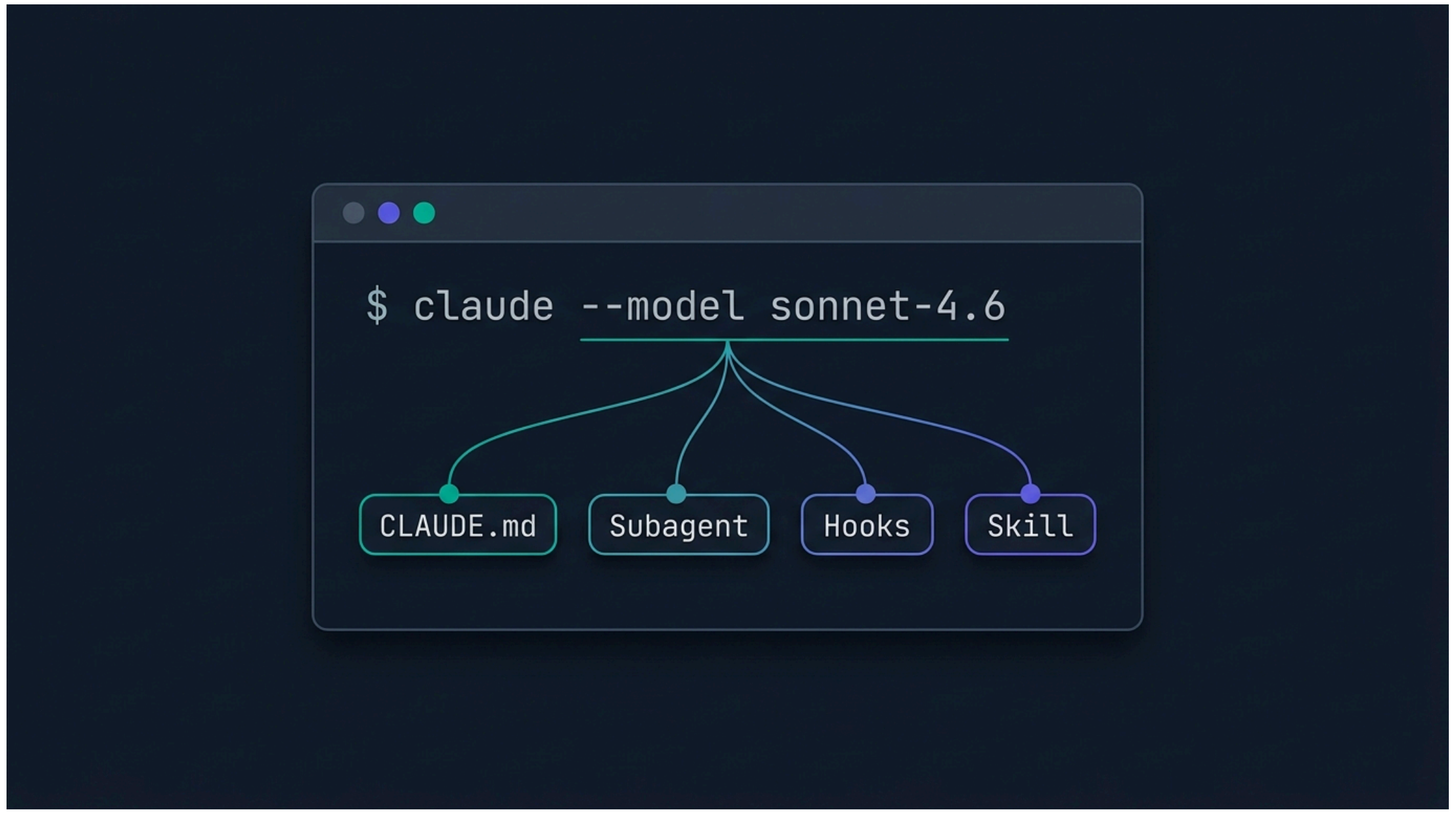
$200 / 월

- 헤비 유저
- 하루 종일 세션 상주

🔄 API 종량제 병행

Sonnet 4.6 기준 (2026-06) — 입력 **$3 / MTok** · 출력 **$15 / MTok** ※ **MTok = million tokens** — 백만 토큰 단위 과금. 한글 산문 기준 대략 1 MTok ≈ 도서 3~4권 분량.

📌 이 강의는 **Pro 기준** 으로 진행한다. Max 는 대형 리포·병렬 subagent 를 상시 쓰는 사람에게만 권한다.



claude 셸 하나에서 뻗어 나가는 네 축 — S3·S4 로 이어진다

Codex — *OpenAI · CLI + Web*

OpenAI 가 Claude Code 를 뒤따라 내놓은 CLI 코딩 에이전트. 명령어 감각은 비슷하지만 **정체성이 다르다** — 대표 차이 세 가지.

✅ ① 승인 모드 3단계

- `auto` / `on-request` — 위험 조작마다 승인
  시나리오 : 파일 쓰기·git 커밋 하나하나 y/n 을 묻는다.
- `edit` — 파일 수정 허용, 셸은 승인
  시나리오 : diff 는 자동, `rm` · `curl` 같은 셸 명령만 확인창.
- `full-auto` / `never` — 승인 없이 진행 (sandbox 안에서만)
  시나리오 : 40분짜리 리팩토링을 자리 비운 사이 끝내 놓는다.

※ Claude Code 의 승인 모드 6종(Default/Manual · Plan · acceptEdits · `auto` · `dontAsk` · Bypass)에 대응된다 — S4 에서 상세 비교.

🧱 ② Sandbox

격리 sandbox 를 **명시적으로 제공**한다.

- 기본 `sandbox-mode = workspace-write`
- 위험 명령이 sandbox 밖으로 나가려 하면 자동 승인 요청

sandbox = 기술적 경계 · 승인 정책 = 그 경계를 넘을 때의 규칙

🌐 ③ Codex Web

터미널이 없어도 브라우저에서 세션을 돌린다.

- `chatgpt.com/codex`
- **ChatGPT Plus ($20/월) 이상** — Web · CLI · IDE 모두 접근
- Free 티어도 접근되나 한도가 낮다

📄 컨텍스트 파일 — `AGENTS.md`

Codex 는 `AGENTS.md` 를 CLAUDE.md 자리에 둔다. **이름만 다를 뿐 역할은 같다** — 리포 규약·명령·팀 정책. Codex 저장소도 자기 `AGENTS.md` 를 공개한다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

Cursor — *Anysphere · IDE 기반*

정체성. 2022 년 창업한 Anysphere 의 **VS Code fork IDE**. 확장이 아니라 **IDE 자체를 재설계해 AI 를 내장**했다. 그래서 Copilot 이 못 하는 것을 한다 — 인라인 diff · 멀티파일 편집 · Shadow Workspace · Composer.

🚀 v2.6 (2026-03) 기능

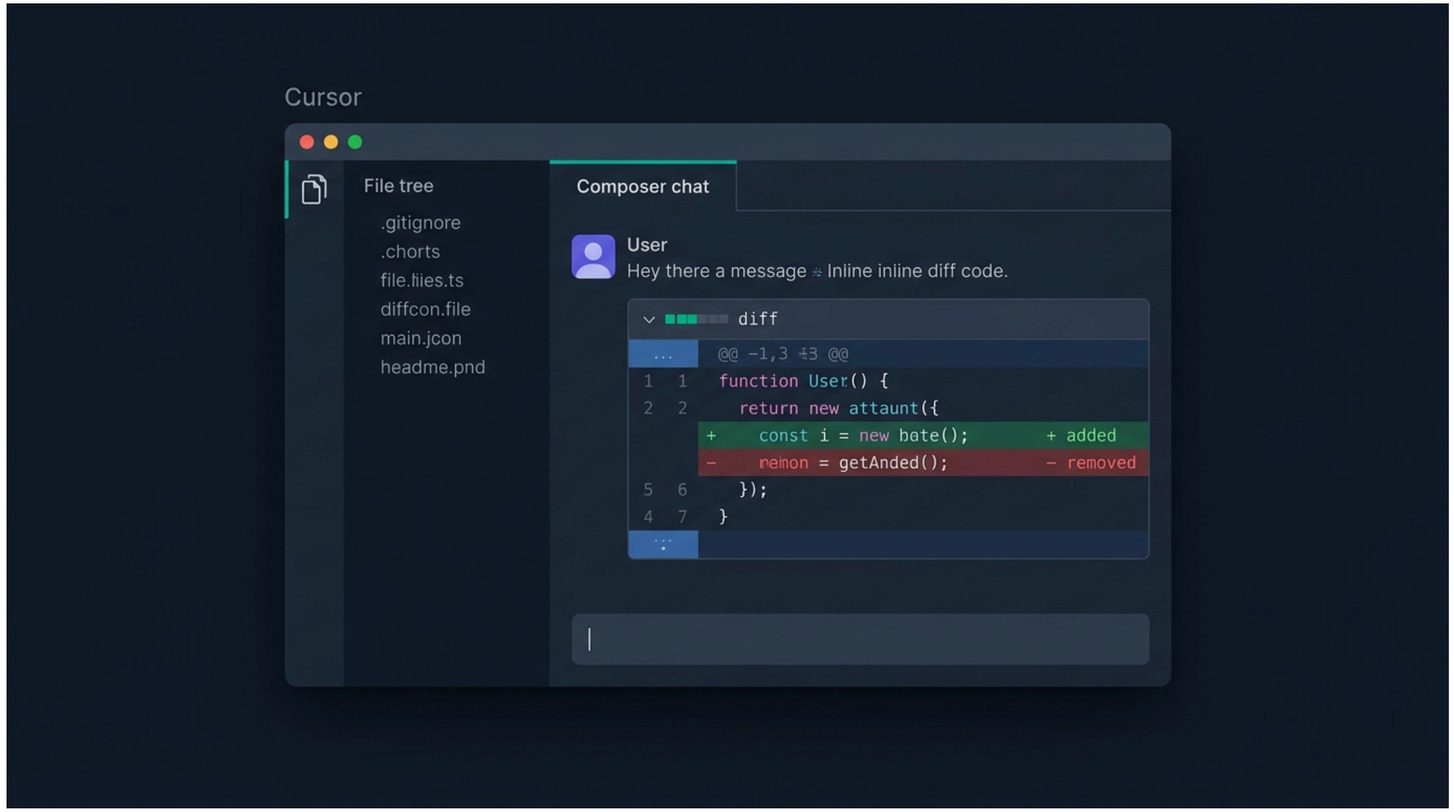
- **Tab 자동완성**
- **Composer 2.0** — 멀티파일 코드 생성
- **Agent Mode** — 터미널 접근
- **Background Agents** — 원격 VM
- **Automations** — 외부 이벤트 트리거
- **Bugbot** — PR 리뷰
- **Visual Editor** — UI 편집

🧠 모델·매출

- 자체 모델 **Composer 1.5** (2026-02)
- GPT · Claude · Gemini · xAI · DeepSeek 라우팅
- 어려운 태스크에서는 사고 시간을 자동으로 늘린다
- **2026 년 초 ARR $2B 돌파**

📌 **Cursor 를 언제 쓰나** — GUI 를 벗어나기 싫은 사람 · 인라인 diff 중심 편집 워크플로 · VS Code 확장 생태계를 그대로 유지하고 싶은 사람. **이 강의에서 Cursor 는 실습 도구가 아니라 비교 대상이다.**

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf



Cursor 의 정체 — VS Code fork 위에 인라인 diff · Composer 를 얹었다

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

Devin Desktop *(구 Windsurf)* — *Cognition · IDE 기반*

정체성. Codeium 이 2024 년 말 공개한 VS Code fork IDE. Cascade 에이전트를 중앙에 두고 코드베이스 인덱싱과 Supercomplete 자동완성을 결합한다.

📅 2026 년 6월의 정체 변화 (타임라인)

🔀 인수·리브랜드

- **2025-12** — Cognition 이 Windsurf 를 약 **$2.5억** 에 인수
- **2026-06-02** — 브랜드 통합 → **Devin Desktop** IDE 는 OTA 업데이트로 자동 리브랜드

🦀 에이전트 재편

- Cascade 에이전트 **2026-07-01 EOL**
- Rust 로 재작성한 **Devin Local** 이 후계 (토큰 30% 효율 · subagent 지원)
- 로컬(Local)·클라우드(Devin)·IDE(Desktop) 를 **하나의 Agent Command Center** 로 묶는 중

📌 **언제 쓰나** — IDE 안에서 "긴 태스크를 위임하는" 워크플로 · Cognition Devin 클라우드 에이전트와 로컬을 오갈 때. **이 강의에서 상세 실습은 하지 않는다.** 단, "Cascade" · "Windsurf" 이름의 자료를 만났을 때 이 변동을 알고 있어야 오독하지 않는다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

Anthropic 계열 확장 — *Claude Code + Cowork*

정체성. Anthropic 은 개발자용 Claude Code 옆에, **비개발자 지식노동자용 Cowork** 를 나란히 내놓았다. **같은 엔진, 다른 표면.**

👨‍💻 Claude Code — *개발자용*

- **표면** — 터미널 · VS Code · JetBrains 확장 · 데스크톱 앱 **Code 탭**
- **작업 대상** — 코드베이스 · git · 테스트 · PR
- **워크플로** — 파일 편집 · shell · 컴파일 · CI 로그 확인

📄 Cowork — *비개발자 지식노동*

- **표면** — 데스크톱 앱 **Cowork 탭** (혹은 Home 탭 안 토글)
- **작업 대상** — 문서 · 스프레드시트 · 리서치 · 행정
- **워크플로** — 자료 수집 · 요약 · 표 정리 · 이메일 초안

📌 **같은 엔진, 다른 표면.** Claude 모델 + 에이전틱 아키텍처는 동일. 심지어 **Cowork 자체가 Claude Code 로 약 10일 만에 개발**되었다 — Anthropic 내부의 self-hosting 증명.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

Claude Code vs Cowork — *안전 · 효율 · 스킬 재사용*

같은 엔진이라도 **표면에 따라 안전 모델과 토큰 효율이 다르다.** 스킬 자산은 양쪽에서 공유된다.

🔐 안전 모델

- **Cowork** — 폴더 권한 승격 + 파괴 작업 전 확인 + 샌드박스 VM 을 기본 제공
- **Claude Code** — 권한 플래그(`--dangerously-skip-permissions` 등) 를 사용자가 직접 설정

⚡ 토큰 효율

- **Claude Code** — 텍스트 기반, 토큰 효율 최적
- **Cowork** — GUI 화면을 스크린샷·이미지로 왕복 → 토큰 소모가 상대적으로 빠르다

🧩 스킬 재사용

- **SKILL.md** — 개방형 포맷
- 한 번 만든 스킬 자산을 **Claude Code · Cowork 양쪽이 그대로 공유**

📌 **선택 기준.** 「내가 코드베이스를 만지느냐, 문서·스프레드시트를 만지느냐」 가 1차 기준. 안전 관행이 미숙한 팀 · 비개발자 팀이라면 **Cowork 의 폴더 권한·확인 프롬프트 기본값**이 사고를 줄인다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

UX 축으로 다시 정렬 — *CLI ↔ IDE × 대화형 ↔ 자율형*

네 도구를 UX 두 축에 배치한다.

📐 축 1 — CLI ↔ IDE

- **CLI 중심** — Claude Code · Codex
- **IDE 중심** — Cursor · Devin Desktop

이 축은 **작업 속도 · 시각 정보량 · 커스터마이즈 여지** 를 가른다.

📐 축 2 — 자율성 (Autonomy)

- **대화형** — 승인마다 물어본다
- **자율형** — 승인 없이 진행한다

Codex 의 `on-request` ↔ `never` 스펙트럼이 가장 명시적이다.

🎯 2×2 매트릭스

	대화형 · 승인 중심	자율형 · 승인 최소
CLI	Claude Code (Default · Plan)	Codex (`full-auto` + sandbox)
IDE	Cursor (Ask 모드)	Devin Desktop (Devin Local 위임)

📌 **핵심 관찰.** 이 표는 "무엇이 우월한가" 를 말하지 않는다. **작업 성격이 어느 칸에 있느냐에 따라 도구가 자동으로 정해진다.**

PART 4

실무 사례 3선 — *수치와 해석*

> 이 파트의 수치는 원 출처가 공개된 것만 인용한다. 애매한 것은 뺐다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

사례 1 — *Anthropic 내부 · Claude Code (2026-05)*

Anthropic 이 2026-05 자사 엔지니어링 조직의 Claude Code 활용 지표를 공개했다. 요지는 세 가지다.

📊 병합 커밋 80%+

프로덕션 코드베이스에 병합되는 커밋의 **80% 이상**을 Claude 가 작성한다.

- 2024 년 초 : 한 자릿수

⚡ 1인당 8배

엔지니어 1인당 하루 병합 코드량이 **2024 년 대비 8배**다.

🎯 모호 태스크 76%

사양 모호 오픈 문제 성공률.

- 2025-11 : **26%**
- 2026-05 : **76%** (6개월 만에 50%p ↑)

📌 시사점 — "자기 도구를 만든 회사" 특수성이 크다

수치를 그대로 일반 팀에 옮기지 않는다. 다만 **세 방향은 유지**된다. **① 병합량은 늘고** · **② 엔지니어는 오케스트레이터 역할로 이동**하며 · **③ 이전엔 손 안 대던 태스크에 손이 간다.**

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

사례 2 — *Rakuten · CRED · TELUS · Zapier (2026)*

Anthropic 의 **2026 Agentic Coding Trends Report** 가 실은 도입 결과 4개다.

🇯🇵 Rakuten

릴리즈 리드타임 24일 → 5일 (79% ↓)

Claude 기반 에이전트 시스템이 요구사항 분해 · 코드 생성 · 리뷰를 조율한다.

🇨🇦 TELUS

- 전사 팀들이 함께 만든 내부 AI 솔루션 **1.3만 개 이상**
- 엔지니어링 배포 속도 **30% ↑**
- 누적 **50만 시간** 절감

🇮🇳 CRED

실행 속도 2배

인도 핀테크, 1,500만 사용자.

⚡ Zapier

- 전사 AI 활용률 **89%**
- 사내 배포 에이전트 **800+**

📌 리포트 저자의 정리 문장 (그대로 인용)

"엔지니어의 역할이 구현자에서 오케스트레이터로 이동한다."

이 강의의 방향과 정확히 일치한다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

사례 3 — *GitHub Copilot · 대규모 설문 (2022 ~ 2025)*

가장 오래되고 표본이 큰 정량 연구. 규모가 크고 대조군이 있어 자주 인용된다.

🧪 GitHub 2022 통제 실험

Copilot 사용 그룹이 대조군보다 **동일 태스크를 55% 빠르게** 완료했다.

(제한) 자바스크립트 HTTP 서버 작성 태스크, 통제 환경.

💳 Copilot 유료 구독자

2026 년 초 기준 **470만 명**.

전년 대비 약 **75% 증가**.

📋 Stack Overflow 2025 설문

전문 개발자 **84%** 가 AI 코딩 도구를 쓰거나 쓸 계획이다.

⚠️ **주의.** "55% 빠름" 은 특정 태스크·통제 환경에서 나온 값이다. **실무 전체 생산성 향상으로 확장 해석하면 왜곡된다.** 이 강의는 그 함정을 피해 **수치는 그대로 인용하고 해석은 좁게** 유지한다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

반대 데이터 하나 — *"delegation gap"*

같은 **Anthropic 2026 리포트가 정직하게 짚은 지점**이다.

📈 사용률 60%

개발자들은 **업무의 60% 정도에 AI 를 쓴다**고 답했다.

완전 위임 가능 예 — 유닛테스트 스캐폴딩 · 보일러플레이트 · 문서 초안

📉 완전 위임 0~20%

하지만 **완전히 위임할 수 있는 태스크는 0~20% 에 그친다.**

위임 어려움 예 — 성능 최적화 · 보안 판단 · 인프라 마이그레이션

리포트는 이 격차를 **"delegation gap"** 이라 부른다.

📌 시사점 — 바이브코딩은 *부분 위임의 기술*

"AI 에 던지고 잊는" 지점과 **"직접 만져야 하는" 지점**을 감별하는 감각. 이 코스가 손에 남기는 실제 산출물이다.

PART 5

도구 선택 매트릭스 — *3축 프레임 + 4×4 매트릭스*

> "어떤 도구를 쓸까" 에 답하는 3축 프레임과 실전 선택표.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

3축 선택 프레임 — *워크플로 · 프로젝트 · 예산*

"어떤 도구를 쓸까" 에 답하기 전에 세 축을 먼저 그려 본다.

🖥️ 축 1 — 워크플로

CLI 중심 ↔ IDE 중심 — 손이 터미널에 붙어 있는가, 마우스가 편집기에 붙어 있는가.

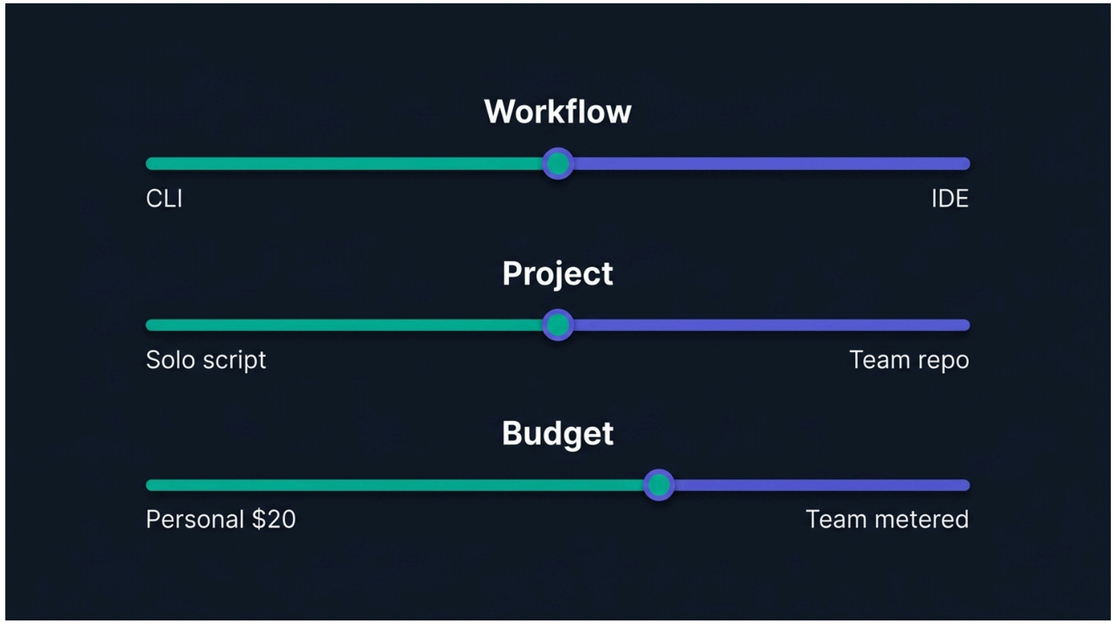
📂 축 2 — 프로젝트 성격

개인 스크립트 ↔ 팀 대형 리포 — 개인 도구 세팅과 팀 규약 세팅은 요구가 정말 다르다.

💰 축 3 — 예산

개인 $20 · 팀 종량 · 무료 티어 — API 키·조직 계정·구독 티어의 조합이 총비용을 정한다.

📌 **결정 규칙.** 이 세 축이 만나는 지점에서 도구가 정해진다. **한 도구로 모든 상황을 커버하려 들지 않는다.** 이 강의는 Claude Code + Codex 를 모두 다룬다.



세 슬라이더의 조합 지점에서 도구가 정해진다

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

도구 선택 매트릭스 ① — *개인 · 팀 리포 · 오픈소스 · 모노리포*

"이럴 땐 이걸 쓴다." 참여자가 실제로 벽에 붙일 만한 표를 목표로 한다.

상황	1순위 도구	2순위 도구	이유
개인 사이드 프로젝트 · 파이썬 스크립트	Claude Code (Pro $20)	Codex (ChatGPT Plus $20)	CLAUDE.md 하나로 개인 규약 관리, 승인·훅 최소
팀 웹서비스 리포 · 매일 커밋	Claude Code (Max) + AGENTS.md	Cursor	리포별 CLAUDE.md 3계층 · Subagent 로 프론트/백 분업
오픈소스 유지보수 · PR 리뷰 자동화	Codex + `codex-action` (CI)	Claude Code	GitHub Actions 통합 · sandbox 로 안전
대형 모노리포 · 아키텍처 리팩토링	Claude Code (Max 20x) + Subagent	Devin Desktop	장시간 세션 · 병렬 subagent 로 대형 탐색

📌 **1순위 축.** 개인·팀 규모가 커질수록 Claude Code 의 3계층 컨텍스트·Subagent 우위가 커진다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

도구 선택 매트릭스 ② — *탐색 · 문서화 · 정책 · GUI 프론트*

상황	1순위 도구	2순위 도구	이유
탐색 · 프로토타입 · 하루짜리 실험	Codex Web (`chatgpt.com/codex`)	Cursor Composer	셋업 없음, 브라우저로 즉시
레거시 코드 이해 · 문서화	Claude Code + `@file`	Cursor (Ask 모드)	200K+ 컨텍스트에 리포 통째로
팀 정책 강한 조직 · sandbox 필수	Codex (`workspace-write` + `on-request`)	(권장 없음)	승인 정책이 명시적으로 분리
GUI 편집 중심 · 프론트엔드 UI 작업	Cursor (Visual Editor)	Devin Desktop	IDE 안에서 인라인 diff

📌 표 전체의 결론

Claude Code · Codex 를 축으로 두고, **팀 정책 · GUI 선호에 따라 Cursor · Devin Desktop 을 곁들이는 조합** 이 2026 년 하반기의 표준안이다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

Claude Code + Codex — *두 도구를 축으로*

Cursor · Devin Desktop 을 뺀 이유를 명시한다.

🚫 Cursor 제외 이유

- GUI 중심이라 **CLI 워크플로 · 스크립트화가 어렵다**
- **팀 규약 · CI 통합** 이 CLI 도구보다 낮다

🚫 Devin Desktop 제외 이유

- **리브랜드 직후** — 문서·API 가 유동적이다
- 강의 재현성이 떨어진다

✅ Claude Code + Codex 조합 이유

- 컨텍스트 파일 규약이 대칭 (`CLAUDE.md` ↔ `AGENTS.md`)
- **승인 모델 · CI 통합** 이 서로 대칭
- 둘을 나란히 익히면 나머지는 자기 힘으로 흡수한다

📌 이 코스가 손에 남기는 능력

코스를 마치면 새 도구가 나와도 **세 질문으로 5분 안에 정체를 파악**한다:

1. **컨텍스트 파일이 어디에 있는가**
2. **승인 모델이 어떤가**
3. **CI 통합이 되는가**

PART 6

첫 액션 아이템

> 지형도가 잡혔다. 첫 액션은 셋업 준비.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

첫 액션 아이템 — *S1 셋업 전 3가지 확인*

S1 은 첫 20분을 로컬 환경 세팅에 쓴다. 아래 셋을 미리 확인해 두면 그 시간이 줄어든다.

✅ ① Anthropic 계정

claude.com 로그인 · **Pro 이상 구독** 확인.

- Pro $20 / 월
- Max $100 / 월
- Max 20x $200 / 월

✅ ② 터미널 환경

macOS · Linux · **WSL2** 중 하나.

- Python 3.11+
- git
- Node 18+

Windows 네이티브도 되지만 **WSL2 를 권한다**.

✅ ③ `~/.claude` 디렉토리

Claude Code 를 설치한 적 있다면 이미 있다.

- `ls -la ~/.claude` 스크린샷 준비
- S3 (컨텍스트 엔지니어링) 에서 바로 활용

⚠️ 참여자 **20~30% 가 여기서 실패**한다 (권한·프록시·이전 설치 잔여). S1 첫 20분이 정확히 이 트러블슈팅에 배정돼 있다. **지금은 준비만 한다.**

설치 명령 — *미리 실행해 봐도 좋다*

```
# npm 으로 설치 (가장 표준적인 경로)
npm install -g @anthropic-ai/claude-code

# 설치 확인
claude --version

# 첫 인증 (브라우저 로그인 창이 뜬다)
claude
```

🔍 실행 후 확인

```
# 로컬 디렉토리 생성 여부
ls -la ~/.claude

# 예상 결과: config.json · projects/ · statsig/ 등이 보이면 OK
```

📌 실패한 경우 — **npm 권한** · **프록시** · **이전 설치 잔여** 셋 중 하나가 원인이다. S1 첫 20분 트러블슈팅에서 다룬다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf

마무리 — *한 문장으로*

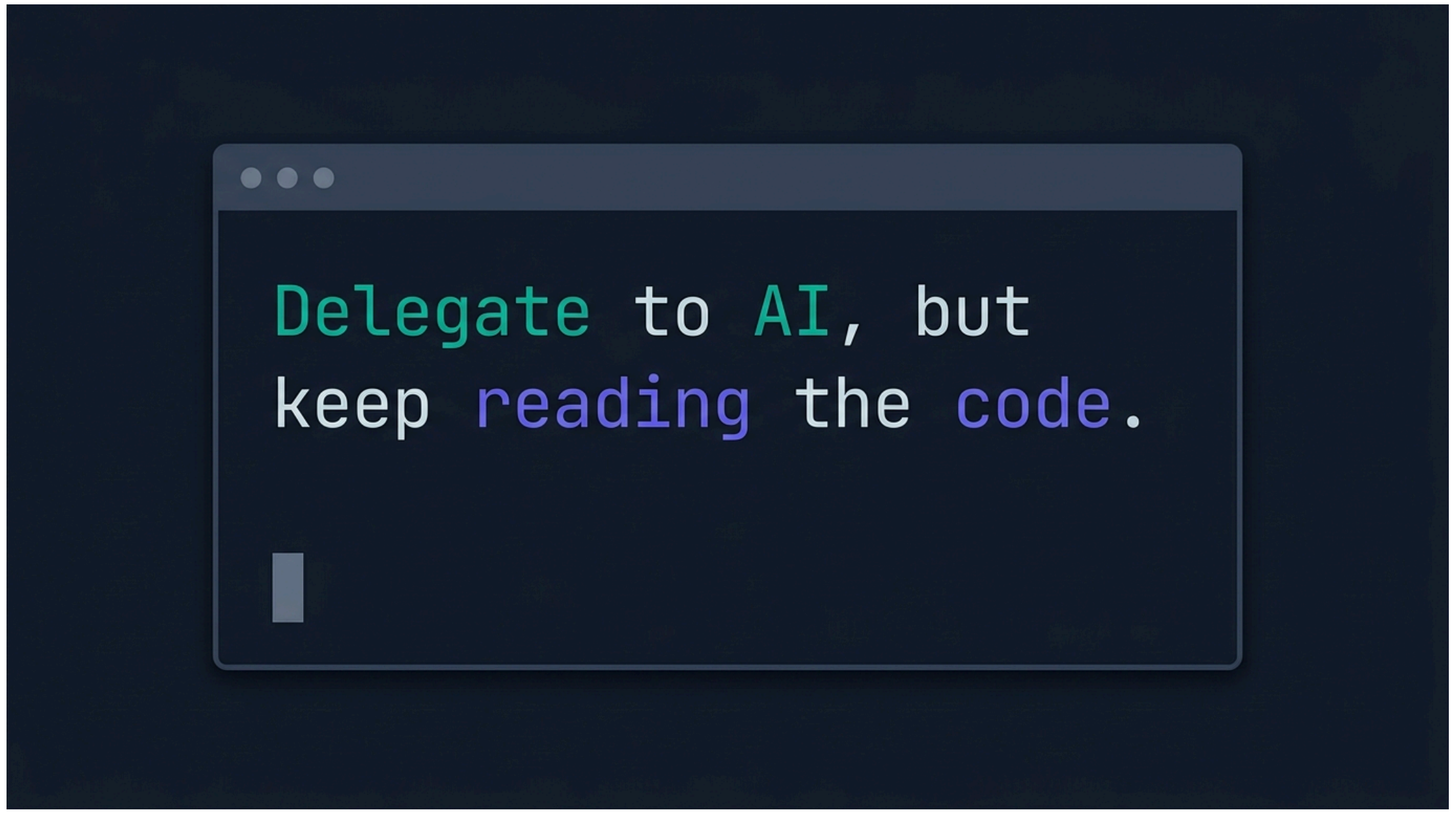
S0 이 끝났다. 지형도가 잡혔다.

🎯 이번 강의의 결론 한 줄

"코드를 잊지 않으면서 AI 에 위임하는 법을 배운다."

지금부터의 세션은 각각 **도구 · 개념 · 실습** 에 집중한다.

출처: 생성형 AI 지형도 2026 | Claude Code | Codex | Cursor | Windsurf



S0 이 남기는 한 문장