읽을거리 · 칼럼 정리

개발자는 코딩을 해야 하나?

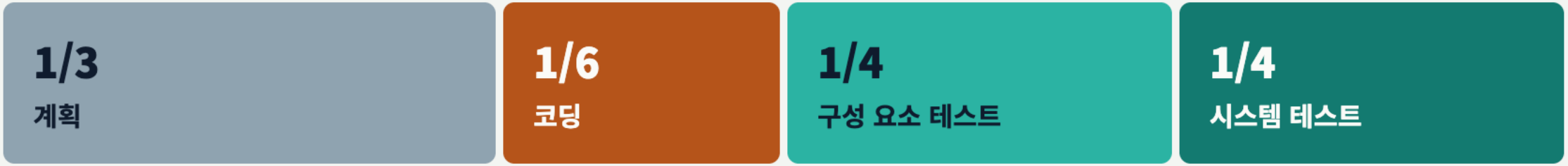
손을 떼는 것은 문법뿐이다.
설계, 명세, 장벽은 여전히 사람의 몫이다

출처: 백명석, 「개발자는 코딩을 해야 하나?」, 브런치 @cleancode (2026.10.27)

원래부터 그랬다

코딩은 개발 일정의 6분의 1이었다

프레드 브룩스가 제시한 개발 일정 배분 (『The Mythical Man-Month』, 1975)



테스트가 절반

만드는 일보다 맞게 만들었는지 확인하는 일이 더 컸다

그 1/6도 줄어 왔다

IDE 자동 완성, 이제는 AI 에이전트가 코딩을 맡는다

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10) · Frederick P. Brooks, 『The Mythical Man-Month』 (1975)

질문을 둘로 나누면

할 줄은 알아야 하고, 직접 칠 필요는 없다

코딩을 할 줄 알아야 하는가?

그렇다

개념을 알아야 AI에게 정확히 시킬 수 있다. tmux의 pane 간 메시지 전달을 알면 “에이전트끼리 메시지를 주고받게 해 줘”라고 요청할 수 있다

코드를 직접 작성해야 하는가?

아니다

전략을 짜는 장수가 매일 검술을 연습하진 않아도, 필요한 순간에 이기려고 홀로 수련은 한다

주말 바이브코딩으로 프로토타입은 나온다. 하지만 운영 가능한 제품으로 만드는 데는 약 9배의 비용이 든다 (브룩스)

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10) · 마틴 파울러, 「Is Design Dead?」

추상화의 역사

우리는 늘 아래 단계를 보지 않게 됐다

1단계
종이테이프
구멍으로 바이너리 입력

2단계
어셈블러
니모닉 텍스트

3단계
고급 언어
FORTRAN, COBOL, C

4단계
트랜스파일러
TypeScript → JS

5단계 · 지금
텍스트 명세
AI가 만든 코드는 컴파일러 출력

바뀐 것은 하나, 의도를 표현하는 수단이다. 구멍 → 어셈블리 → C → 시나리오

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10) · 5단계는 엉클 밥(Robert C. Martin)의 주장으로 소개됨

나도 코딩이 즐겁다, 그런데

즐거움은 타이핑이 아니라 풀이에 있었다

점점 지루해진 것

제조

방향이 정해진 뒤 뻔한 구현을 끝까지 쳐 내는 일

→

여전히 즐거운 것

이해 · 분해 · 해결

문제를 이해하고, 쪼개고, 해결책을 세우는 일. 구현은 AI에게 넘겼다

AI에게 구현을 맡기는 두 조건

1. 필요할 때 그 코드를 이해할 수 있을 것
2. 기능 검증에 문제가 없을 것

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10) · 브룩스, 『The Mythical Man-Month』 1장 “The Tar Pit”

손을 떼는 것은 문법뿐이다

줄은 놓고, 코드의 형태는 쥔다

손을 떼도 되는 것

문법

함수 바디 안의 한 줄 한 줄. AI가 사람보다 잘한다

계속 지켜야 하는 것 · 코드의 형태(shape)

- 설계, 아키텍처, 모듈 구조와 배치
- 함수의 목록, 이름, 크기, 역할
- 함수와 컴포넌트 사이의 호출 관계
- 생성자와 함수의 인자

지혜 하지 말아야 할 것을 아는 경험

야망 잘하고 싶은 내적 동기

통찰 지식이 아닌 경험에서 온 판단

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10) · 엉클 밥 “DISENGAGE FROM CODE! MANAGE THE AGENTS INSTEAD!” · 하단 세 가지는 저자가 정리한 사람의 강점

장벽은 프롬프트가 아니라 도구여야 한다

AI는 규칙은 어겨도 벽은 못 넘는다

프롬프트나 CLAUDE.md의 규칙은 '지침'으로 취급돼 자주 어겨지고, 컨텍스트가 압축되면 빠진다

물리적 장벽	하는 일
단위 테스트 + 높은 커버리지	실패하면 통과할 때까지 다음 단계로 못 넘어감
인수 테스트 (Gherkin)	사람이 소유하는 명세. AI는 허가 없이 수정 불가
돌연변이 테스트	커버리지가 놓치는 assertion의 구멍을 찾아냄
CRAP, 의존성 검사, lint	함수 크기, 의존 방향, 기본 품질을 측정

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10) · 엉클 밥, 「Clean AI: Agentic Discipline」 강의

사람은 여전히 루프 안에

에이전트 여섯이 두 시간, 남은 버그 열 개

~10

엉클 밥이 에이전트 여섯으로 두 시간 동안 만든 게임에 남은 버그 수

약 5개 · 명세 검토 실패

에이전트가 쓴 명세를 사람이 제대로 검토하지 않아서 생겼다

나머지 · 기대를 몰라서

에이전트는 화면을 보지도, 게임을 해 보지도 않았다. 사람이 직접 플레이하며 찾았다

테스트는 명세에 적힌 것만 지킨다. 명세에 없는 기대는 사람만 안다

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10)에 소개된 엉클 밥의 사례

‘개발을 잘한다’의 의미가 바뀐다

10분짜리 결과물과 두 시간짜리 결과물

에이전트 하나에게 맡겼다면

10분

빨리 끝난다. 하지만 단위 테스트, 인수 테스트, QA, 하드닝, 리팩터링이 없다

장벽을 세운 멀티 에이전트

2시간

완전하진 않지만, 모든 측면에서 테스트된 엔지니어링 결과물

예전: 코드를 잘 쓰는 사람 → **이제: 문제를 이해하고, 해결책을 찾고, 분해하고, 팀이 풀도록 돕는 사람**

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10) · 10분은 저자가 단일 에이전트일 때를 가정해 든 비교

결론

즐거움의 위치가 한 단계 올라간다

01 원리는 배운다
직접 치진 않아도, 개념을 알아야 AI에게 정확히 시킨다

02 의도는 명세로
Given-When-Then 시나리오를 먼저 쓰고, AI가 못 고치게 한다

03 규칙 대신 장벽
프롬프트로 부탁하지 말고 테스트와 도구로 막는다

04 형태를 보고 직접 써 본다
줄 대신 구조를 읽고, 결과물은 사람이 직접 돌려 본다

문법은 AI에게, 문제 이해·분해·장벽 설계는 사람에게

출처: 백명석, 「개발자는 코딩을 해야 하나?」 (2026.10)의 결론을 강의용 실천 항목으로 정리