바이브코딩: 말하면 이루어진다

코딩 없이 AI로 나만의 시스템을 만드는 법

원샷 지음

2026년 3월

이 책은 코딩을 모르는 사람이 쓴, 코딩을 모르는 사람을 위한 책입니다.

목차

프롤로그 — 나는 코딩을 모른다

나는 코딩을 모른다.

HTML이 무엇인지는 대충 안다. 엑셀에서 SUM 함수 정도는 쓸 수 있다. 그러나 프로그래머들이 쓰는 Python, JavaScript, 그 어떤 언어도 배운 적이 없다. 배우려고 시도한 적도 없다. 그게 내 일이 아니라고 생각했고, 그 생각은 틀리지 않았다.

그런데 지금, 내 노트북에는 내가 만든 시스템이 돌아가고 있다.

매일 아침, 날짜와 요일에 맞는 할일 목록이 자동으로 만들어진다. 꿈을 말하면 여섯 가지 각도로 분석되어 날짜별로 저장된다. 녹음한 대화가 주제별로 정리된다. 핸드폰에서 명령을 보내면 집 노트북이 실행된다. 주제를 말하면 수십 페이지짜리 전자책이 5분 안에 완성된다.

코드 한 줄 직접 쓰지 않고.

이것이 **바이브코딩**이다.

이 책은 그 경험을 담은 기록이다. 어떻게 시작했는지, 어떻게 말했는지, 무엇이 만들어졌는지, 그리고 그 과정에서 내 하루가 어떻게 달라졌는지.

코딩을 모르는 당신을 위한 이야기다.

1장. 바이브코딩이란 무엇인가

1.1 바이브코딩의 탄생

2025년 초, Andrej Karpathy라는 사람이 짧은 글을 하나 썼다.

그는 테슬라의 전 AI 총괄이자 OpenAI 공동창업자다. 수백만 명의 팔로워를 가진 AI 분야의 권위자다. 그런 사람이 이런 말을 했다.

> “요즘 나는 ’바이브코딩’이라는 걸 하고 있다. 완전히 흐름에 몸을 맡기고, 코드가 존재한다는 것조차 잊어버리는 것이다. AI에게 원하는

> 것을 말하고, AI가 만들어준 것을 실행하고, 오류가 나면 오류를 붙여넣고, 다시 말한다."

처음 이 말을 들었을 때, 많은 사람들이 비웃었다. "그게 개발이야?" "결국 AI가 만들고 사람은 검사만 한다는 거잖아." "그러면 개발자가 필요 없어지는 거야?"

그러나 Karpathy의 핵심은 다른 데 있었다. 바이브코딩은 개발자가 더 빠르게 일하는 방법이기도 하지만, 동시에 **비개발자가 처음으로 뭔가를 만들 수 있게 되는 방법**이기도 하다.

프로그래밍 언어를 배우는 데는 수년이 걸린다. 문법을 익히고, 오류를 이해하고, 라이브러리를 배우고, 프로젝트 구조를 잡는 법을 터득하는 데. 그런데 바이브코딩은 이 모든 과정을 건너뛴다. 원하는 것을 말하면 된다.

이것이 혁명이다.

1.2 기존 코딩과 무엇이 다른가

기존 코딩은 이렇게 생겼다.

```python
def create_daily_note(date, day_of_week):
    filename = f"{date} 오늘의 할일.md"
    content = f"# {date} 오늘의 할일\n\n"
    if day_of_week == "금요일":
        content += "- [ ] 주차 예약\n"
    return content
```

이것을 이해하려면 Python 문법을 알아야 하고, 함수가 무엇인지 알아야 하고, 조건문이 어떻게 작동하는지 알아야 한다.

바이브코딩은 이렇게 생겼다.

"매일 아침 오늘의 할일을 만들어줘. 금요일에는 자동으로 주차 예약 항목을 추가해줘. 목요일과 일요일에는 쓰레기 버리기를 추가해줘. 파일명은 YYYY-MM-DD 형식으로, 폴더는 여기에 저장해줘."

이게 전부다. AI가 나머지를 한다.

기존 코딩	바이브코딩
문법을 배워야 한다	말만 하면 된다
오류를 직접 해결한다	오류를 말로 설명하면 된다
수시간이 걸린다	수분이 걸린다
개발 경험이 필요하다	원하는 결과를 상상할 수 있으면 된다
코드를 읽을 수 있어야 한다	결과를 볼 수 있으면 된다

두 방식의 핵심 차이는 **진입장벽**이다. 기존 코딩의 진입장벽은 언어 학습이다. 바이브코딩의 진입장벽은 '원하는 것을 명확하게 말하는 것'이다.

후자가 훨씬 낮다.

1.3 왜 지금 이 방식인가

10년 전에도 "자연어로 프로그래밍"이라는 개념은 있었다. 그러나 당시 AI는 충분히 똑똑하지 않았다. 말을 이해하는 것 같지만 엉뚱한 결과를 내놓았고, 수정을 반복하다 보면 결국 직접 코딩하는 것보다 오래 걸렸다.

지금은 다르다.

2024년 이후 나온 AI 모델들(GPT-4, Claude 3 시리즈, Gemini 등)은 복잡한 한국어 지시도 정확하게 이해한다. 오류가 나면 오류 메시지를 붙여넣는 것만으로 대부분 해결된다. 요청한 것과 다른 결과가 나오면 "이 부분이 아니라 저렇게 해줘"라고 말하면 수정된다.

AI의 능력이 임계점을 넘었다.

동시에 **코딩 도구**들도 발전했다. VS Code, Cursor, Claude Code 같은 도구들은 AI와 직접 대화하면서 파일을 만들고, 저장하고, 실행하는 것이 가능해졌다. 예전처럼 AI가 코드를 알려주면 사람이 복사해서 붙여넣는 방식이 아니라, AI가 직접 파일을 만들고 실행까지 한다.

바이브코딩이 가능해진 것은 AI가 똑똑해졌기 때문만이 아니라, **AI와 컴퓨터가 직접 연결되었기 때문**이다.

1.4 누구나 할 수 있다

바이브코딩을 하기 위해 필요한 것은 세 가지다.

첫째, 컴퓨터. 스마트폰이 아닌 노트북 또는 데스크탑이 필요하다. Claude Code나 Cursor 같은 도구는 컴퓨터 환경에서 작동한다.

둘째, AI 도구. Claude Code, Cursor, 또는 ChatGPT Plus면 충분하다. 유료 구독이 필요하지만, 월 몇 만 원 수준이다.

셋째, 원하는 것. 가장 중요한 것이다. "무언가 자동화하고 싶다", "반복하는 일이 귀찮다", "이런 걸 만들어보고 싶다" — 이런 욕구가 있으면 된다.

학력이 중요하지 않다. 나이가 중요하지 않다. 전공이 중요하지 않다. IT 경험이 중요하지 않다.

원하는 것을 한국어로 말할 수 있으면 된다.

2장. 첫걸음 — 말하는 법을 배운다

2.1 도구를 고른다

바이브코딩에 쓸 수 있는 도구는 여러 가지다. 처음 시작하는 사람에게 추천하는 순서는 다음과 같다.

1단계 — ChatGPT 또는 Claude.AI (웹) 가장 쉽게 시작할 수 있다. 브라우저에서 열고, 원하는 것을 말하고, AI가 코드를 알려주면 직접 복사해서 파일에 붙여넣는 방식이다. 진입장벽이 가장 낮다.

단점은 파일을 직접 만들어주지 않는다는 것이다. AI가 코드를 보여주면 사람이 실행해야 한다.

2단계 — Cursor VS Code를 기반으로 만들어진 AI 코딩 에디터다. 파일을 열어놓고 AI와 대화하면, AI가 직접 파일을 수정해준다. 개발자들이 가장 많이 쓰는 바이브코딩 도구다.

3단계 — Claude Code Anthropic이 만든 커맨드라인 기반 AI 도구다. 터미널에서 실행하며, 파일 생성·수정·저장·실행을 AI가 직접 한다. 가장 강력하지만 처음엔 다소 생소할 수 있다.

이 책에서 소개하는 대부분의 사례는 Claude Code를 사용했다. 처음에는 낯설었지만, 익숙해지면 가장 빠르고 자유롭다.

2.2 첫 대화를 시작한다

도구를 설치했다면, 첫 대화를 시작할 때다.

많은 사람들이 첫 대화에서 막힌다. 뭘 말해야 할지 모르기 때문이다. 이때 가장 쉬운 방법은 **지금 가장 귀찮은 일 하나**를 떠올리는 것이다.

매일 아침 비슷한 형식의 메모를 만드는 것이 귀찮은가? 말하면 된다.

> "매일 아침 날짜가 들어간 메모 파일을 만들어줘. 제목은 '오늘의 할일', 내용은 빈 체크박스 5개."

폴더에 있는 파일들을 날짜별로 정리하는 것이 귀찮은가? 말하면 된다.

> "이 폴더에 있는 파일들을 날짜별로 하위 폴더에 정리해줘. 파일명에 날짜가 없는 것은 수정일 기준으로."

첫 대화에서 완벽한 결과가 나오지 않아도 된다. 결과를 보고 "이 부분은 이렇게 바꿔줘"라고 다시 말하면 된다. 그게 바이브코딩의 방식이다.

2.3 잘 말하는 것이 기술이다

바이브코딩에서 가장 중요한 기술은 코딩이 아니다. **말하는 것**이다.

잘 말한다는 것은 구체적으로 말한다는 것이다.

나쁜 예: > "할일 관리 시스템 만들어줘"

좋은 예: > "매일 아침 오늘 날짜로 파일 하나를 만들어줘. 파일명은 'YYYY-MM-DD 오늘의 할일.md' 형식으로. 내용에는 '긴급', '오늘 할 것', '완료', '메모' 섹션이 있어야 해. 금요일에는 '주차 예약' 항목을 자동으로 넣어줘. 저장 위치는 C드라이브의 Documents 폴더 안에 'daily' 폴더야."

두 번째 예시를 보면, 다음 네 가지가 포함되어 있다.

무엇을: 파일 하나
어떻게: 날짜 형식, 내용 구조
언제/조건: 금요일에는 특별 항목
어디에: 저장 위치

이 네 가지를 포함해서 말하면, AI가 원하는 결과를 만들어낼 확률이 크게 높아진다.

처음에는 어렵게 느껴진다. 그러나 몇 번 해보면 자연스러워진다. 사실 이것은 코딩이 아니라 **설명하는 능력**이다. 요리법을 설명하듯, 여행 경로를 안내하듯, 구체적으로 말하면 된다.

2.4 오류는 실패가 아니다

처음 바이브코딩을 시도하면 오류를 자주 만난다.

"파일이 저장되지 않는다", "실행이 안 된다", "결과가 내가 원하는 것과 다르다".

이것은 실패가 아니다. 바이브코딩의 정상적인 과정이다.

오류가 나면 두 가지를 한다.

오류 메시지 복사: 빨간 글씨로 나오는 오류 메시지를 그대로 복사해서 AI에게 붙여넣는다. > "이런 오류가 났어: [오류 메시지]"

원하는 것 재설명: 결과가 원하는 것과 다르면 다시 말한다. > "저장은 됐는데 파일명이 달라. YYYY-MM-DD 형식으로 만들어야 하는데 MM-DD-YYYY로 됐어."

AI는 오류 메시지를 읽고 원인을 찾아 수정해준다. 대부분의 오류는 두세 번 안에 해결된다.

바이브코딩에서 오류는 AI와 나누는 대화의 일부다. 무서워할 것이 없다.

3장. 일상을 자동화한다

3.1 오늘의 할일을 자동으로

가장 먼저 만들어볼 것을 추천하는 것은 **오늘의 할일 자동 생성**이다.

이유가 있다. 매일 쓰는 것이기 때문이다. 매일 쓰는 것이 자동화되면, 그 효과를 매일 체감할 수 있다. 동기부여가 된다.

다음과 같이 말하면 된다.

> "매일 아침 오늘의 할일 파일을 만들어주는 명령을 만들어줘. 파일명은 날짜 형식으로. 내용에는 긴급 섹션과 오늘 할 것 섹션, 완료 섹션, 메모 섹션이 있어야 해. 요일마다 자동으로 들어가는 항목이 달라: 월요일은

> 주간 목표 확인, 수요일은 중간 점검, 금요일은 주간 마무리 + 다음 주 계획.”

AI가 이것을 실행해주는 ‘스킬’ 또는 ’명령’을 만들어준다. 그 이후로는 매일 “오늘의 할일 만들어줘”라는 말 한 마디면 된다.

시간이 지나면 항목을 추가하고 싶어진다. “금요일엔 주차 예약도 추가해줘.” “일요일엔 주간 정리 작성 항목도 넣어줘.” 이렇게 말하면 AI가 수정해준다.

3개월이 지나면, 나만의 완벽한 하루 시작 루틴이 완성되어 있다.

3.2 기록 시스템을 만든다

두 번째로 자동화할 것은 **기록**이다.

사람들은 기록하고 싶어하지만 귀찮아서 안 한다. 타이핑이 귀찮고, 어디에 저장할지 고민하고, 나중에 찾기 어렵고, 결국 기록을 포기한다.

바이브코딩으로 이것을 해결할 수 있다.

꿈 기록: > “꿈 기록 시스템을 만들어줘. 꿈 내용을 말하면 자동으로 분석해서 날짜별로 저장해줘. 번호를 순서대로 붙여줘.”

일상 대화 기록: > “음성 녹음을 텍스트로 변환한 내용을 붙여넣으면, 주제별로 정리해서 날짜별 파일에 저장해줘.”

감정 일기: > “오늘의 감정을 말하면 날짜와 함께 감정 일기 파일에 저장해줘. 최신 것이 맨 위에 오도록.”

이 시스템들이 만들어지면, 기록하는 것이 놀랍도록 쉬워진다. 말하면 저장된다. 3개월 후에는 138개의 꿈 기록이, 수십 개의 대화 기록이 체계적으로 쌓여 있다.

기억은 흘러가지만, 기록은 남는다.

3.3 반복 업무를 줄인다

일상에는 반복되는 업무가 있다. 매주 같은 보고서를 만들거나, 매월 같은 양식을 작성하거나, 매일 같은 형식의 이메일을 보내거나.

이런 반복 업무를 AI에게 설명하면, AI가 템플릿을 만들어준다.

> “주간 업무 보고 템플릿을 만들어줘. 이번 주 완료 항목, 다음 주 계획, 이슈사항 세 섹션으로. 날짜가 자동으로 들어가도록.”

> “회의록 템플릿을 만들어줘. 날짜, 참석자, 안건, 결정사항, 다음 액션 포함. 파일명은 회의날짜_회의명으로.”

한 번 만들어두면 평생 쓴다. 처음 만드는 데 5분이 걸리지 않는다.

3.4 나만의 스킬을 만든다

바이브코딩에서 '스킬'은 자주 쓰는 명령을 저장해놓은 것이다. "오늘의 할일 만들어줘", "꿈 기록해줘", "주간 정리 해줘" — 이런 말들을 AI가 알아듣고 실행할 수 있도록 미리 정의해놓은 것이다.

스킬을 만들면 반복해서 말하지 않아도 된다. 짧은 말 한 마디로 긴 작업이 실행된다.

예를 들어 "꿈 기록해줘"라는 네 글자로: 1. 오늘 날짜와 요일을 확인하고 2. 꿈기록 파일을 열고 3. 현재 최고 번호를 확인하고 4. 새 번호로 꿈 내용을 정리하고 5. 6가지 각도로 분석하고 6. 파일 맨 위에 저장하는

모든 과정이 실행된다.

스킬은 AI와 함께 생활하면서 자연스럽게 쌓인다. 자주 하는 작업을 발견하면 "이걸 스킬로 만들어줘"라고 말하면 된다.

4장. 콘텐츠를 만든다

4.1 전자책 만들기

바이브코딩으로 할 수 있는 가장 인상적인 작업 중 하나가 전자책 만들기다.

주제와 구조를 말하면 AI가 내용을 생성한다. 목차를 잡아주면 챕터별로 글을 써준다. 완성된 마크다운 파일을 pandoc이라는 도구로 변환하면 EPUB이나 DOCX 파일이 만들어진다.

처음 이것을 경험했을 때의 충격을 잊을 수 없다.

"이 주제로 전자책 만들어줘"라고 말한 지 5분 만에 85페이지짜리 전자책이 완성됐다. 표지, 목차, 챕터별 내용, 에필로그까지.

물론 5분 만에 나온 결과가 완벽하지는 않다. 내용을 검토하고, 수정을 요청하고, 개인적인 경험을 추가하고, 다시 정리하는 과정이 필요하다. 하지만 빈 화면에서 시작하는 것과 85페이지 초안에서 시작하는 것의 차이는 하늘과 땅이다.

전자책을 만들 때는 다음 순서로 진행한다.

주제 확정: "AI 시대에 노인이 살아남는 법"처럼 구체적인 주제
목차 작성 요청: "이 주제로 7챕터 목차를 만들어줘"
목차 수정: 마음에 들지 않는 챕터를 수정하거나 추가
내용 생성: "이 목차로 전체 내용을 써줘"
검토 및 수정: 개인 경험 추가, 어색한 부분 수정
변환: pandoc으로 EPUB 또는 DOCX 변환

전 과정이 하루 안에 완성될 수 있다.

4.2 블로그 글 쓰기

블로그는 전자책보다 짧고, 검색엔진에 맞게 쓰는 것이 중요하다.

바이브코딩으로 블로그 글을 쓸 때는 다음을 말하면 된다.

> "이 주제로 블로그 글을 써줘. 제목에 키워드를 포함시켜줘. 도입부에서 독자의 문제를 공감해주고, 본문에서 해결책을 제시하고, 마지막에 행동을 유도하는 마무리로 끝내줘. 1,000자 내외로."

AI가 구조 잡힌 블로그 글을 써준다. 이것을 그대로 쓰기보다는, 나의 실제 경험을 추가하고 어색한 부분을 다듬으면 훨씬 자연스러운 글이 된다.

블로그에서 중요한 것은 **진짜 경험**이다. AI가 구조는 잡아주지만, 내 이야기는 내가 넣어야 한다. "내가 직접 경험한 것을 이 구조에 맞게 써줘"라고 하면 더 좋다.

4.3 전자책 vs 블로그 비교

전자책과 블로그는 같은 주제를 다루더라도 전혀 다른 형식이다.

전자책의 특징: - 길고 깊다 (수십 페이지) - 처음부터 끝까지 읽는 것을 전제로 한다 - 이야기의 흐름이 중요하다 - 독자가 돈을 내거나 이메일을 주고 받아간다 - 한 번 쓰면 오래 쓸 수 있다

블로그의 특징: - 짧고 명확하다 (1,000~3,000자) - 훑어보다가 관심 있는 부분만 읽는다 - 검색에서 찾아오도록 SEO가 중요하다 - 무료로 공개한다 - 자주 업데이트해야 한다

같은 주제로 두 가지 형식을 모두 만들면 서로 보완이 된다. 블로그로 독자를 모으고, 전자책으로 더 깊은 내용을 제공한다.

바이브코딩으로 같은 내용을 두 가지 형식으로 만드는 것은 어렵지 않다.

> "이 전자책 내용을 블로그 형식으로 요약해줘." "이 블로그 글을 전자책 챕터로 확장해줘."

두 형식을 오가는 것이 말 한 마디면 된다.

4.4 SNS 콘텐츠까지

전자책과 블로그를 넘어, 같은 내용을 SNS에 맞게 바꿀 수도 있다.

> "이 블로그 글을 인스타그램용으로 바꿔줘. 5장짜리 카드뉴스 형식으로, 각 장에 핵심 문장 하나."
>
> "이 내용을 유튜브 쇼츠 스크립트로 바꿔줘. 60초 분량."
>
> "이것을 트위터/X 스레드로 만들어줘. 10개 트윗."

하나의 콘텐츠를 여러 채널에 맞게 변형하는 것을 **원소스 멀티유즈**라고 한다. 바이브코딩으로 이것이 가능해졌다.

한 번 좋은 내용을 만들면, 그것을 여러 형식으로 변환해서 여러 채널에 올릴 수 있다. 콘텐츠 제작의 효율이 몇 배로 높아진다.

5장. 더 높은 단계로

5.1 시스템을 연결한다

바이브코딩의 진짜 힘은 **연결**에서 나온다.

하나의 도구를 만드는 것은 1단계다. 그 도구들이 서로 연결되어 자동으로 돌아가는 것이 2단계다.

예를 들어: - 아이폰으로 녹음 → 텍스트 변환 → Claude에게 전달 → 자동 정리·저장

이 흐름이 연결되면, 말로 기록하는 것이 완전히 자동화된다. 사람이 개입하는 부분이 없다.

처음에는 각 단계를 수동으로 연결한다. 녹음하고, 전사하고, 붙여넣고. 그다음에는 각 단계를 자동화한다. 마지막에는 전체 흐름이 하나로 연결된다.

시스템을 연결할 때 중요한 원칙은 **하나씩**이다. 한 번에 모든 것을 연결하려고 하면 복잡해져서 무엇이 문제인지 알 수 없다. 하나씩 연결하고, 각 단계가 잘 작동하는 것을 확인한 다음 다음 단계로 넘어간다.

5.2 원격제어를 구현한다

바이브코딩으로 만들 수 있는 것 중 가장 인상적인 것 중 하나가 **원격제어**다.

집에 있는 노트북을 외출 중에 핸드폰으로 제어하는 것이다.

핵심은 텔레그램 봇이다. 텔레그램은 봇을 만들 수 있는 기능을 제공하고, 이 봇을 통해 컴퓨터와 통신할 수 있다.

바이브코딩으로 이것을 만드는 방법:

> “핸드폰 텔레그램에서 메시지를 보내면 집 노트북에서 실행되는 시스템을 만들어줘. 예를 들어 ’오늘의 할일 만들어줘’라고 보내면 노트북에서 할일 파일이 생성되고, 결과를 텔레그램으로 돌려줘.”

이 말 한 마디로, AI가 Python 스크립트를 만들어주고, 텔레그램 봇 설정 방법을 알려주고, 실행 방법을 설명해준다.

처음 설정에는 30분 정도 걸린다. 그 이후로는 외출 중에도 핸드폰으로 노트북을 제어할 수 있다.

5.3 자동화 파이프라인

바이브코딩이 성숙해지면, 여러 작업이 자동으로 연결되는 파이프라인이 만들어진다.

예시 파이프라인:

```
아이폰으로 대화 녹음
    ↓
PLAUD 또는 아이폰 전사로 텍스트 변환
    ↓
Claude에게 전달
    ↓
주제별 분류 및 정리
    ↓
Obsidian 날짜별 폴더에 자동 저장
    ↓
주간 정리 자동 생성
```

이 파이프라인이 완성되면, 말만 하면 된다. 녹음하면 나머지가 자동으로 처리된다.

파이프라인을 만드는 것은 시간이 걸린다. 하지만 한번 만들어두면, 매일 수십 분의 시간을 절약한다. 3개월이면 수십 시간이 절약된다.

이 시간에 나는 더 중요한 것을 한다. 생각하고, 걷고, 사람을 만나고, 새로운 것을 배운다.

5.4 나만의 AI 비서 완성

결국 바이브코딩의 목적지는 **나만의 AI 비서**다.

범용 AI 비서가 아니다. 나의 생활 방식, 나의 취향, 나의 일하는 방법에 맞게 만들어진, 나만을 위한 비서다.

나의 비서는 알고 있다: - 내가 꿈을 어떻게 분석하고 싶어하는지 - 내 할일 목록이 어떤 구조여야 하는지 - 중요한 파일이 어디에 저장되어야 하는지 - 주간 정리를 언제 어떻게 작성하는지 - 내가 자주 묻는 질문들

이 모든 것이 스킬과 설정으로 저장되어 있다. “오늘의 할일”이라고 말하면 내 방식으로 만들어진다. “꿈 기록해줘”라고 말하면 내 분석 방식으로 저장된다.

범용 도구를 쓰는 것과, 나만의 도구를 쓰는 것의 차이는 크다.

6장. 바이브코딩으로 사는 삶

6.1 시간이 생긴다

바이브코딩을 3개월 하면 가장 먼저 느끼는 것이 **시간**이다.

매일 반복하던 일들이 자동화된다. 오늘의 할일을 만드는 10분, 기록을 정리하는 20분, 반복 업무를 처리하는 30분. 이것들이 자동화되면 하루에 1시간 이상이 생긴다.

처음에는 그 시간을 AI 설정에 쓴다. 더 나은 시스템을 만들기 위해. 하지만 어느 순간부터는 그 시간을 다른 데 쓸 수 있다.

생각하는 시간. 산책하는 시간. 사람을 만나는 시간. 책 읽는 시간. 새로운 것을 배우는 시간.

기계가 반복 업무를 하는 동안, 사람은 사람이 해야 할 일을 한다.

6.2 아이디어가 실현된다

예전에는 아이디어가 생겨도 포기했다.

"이런 걸 만들면 좋겠는데... 코딩을 모르니까 안 되지." "이런 시스템이 있으면 좋겠는데... 개발자를 고용해야 하나?" "이런 도구가 있으면 편할 텐데... 그냥 포기하자."

바이브코딩 이후, 이 생각이 바뀌었다.

"이런 걸 만들면 좋겠는데... 한번 말해봐야지."

대부분의 아이디어가 실현된다. 완벽하지는 않더라도, 작동하는 무언가가 만들어진다. 그것을 쓰다 보면 더 나아진다.

아이디어를 갖고 있는 것과 그것이 실현되는 것의 차이는 엄청나다. 하나는 생각이고, 하나는 현실이다. 바이브코딩은 그 간극을 좁혀준다.

6.3 배움의 방식이 바뀐다

바이브코딩을 하면서 예상치 못한 일이 생겼다.

코딩을 배우게 된 것이다.

코드를 직접 쓰지 않는데 어떻게 배우냐고? AI가 만든 코드를 보면서 자연스럽게 익힌다. "아, 이게 이렇게 돌아가는구나." "이 부분이 이것을 하는구나."

배우려고 공부한 것이 아니라, 쓰다 보니 알게 되었다.

이것이 바이브코딩이 만드는 새로운 배움의 방식이다. 먼저 만들고, 쓰면서 이해한다. 이해하면 더 잘 말할 수 있게 된다. 더 잘 말하면 더 좋은 결과가 나온다.

학습의 순서가 뒤집혔다. 배우고 나서 쓰는 것이 아니라, 쓰면서 배운다.

6.4 앞으로의 세상

바이브코딩은 시작이다.

지금 이 방식으로 할 수 있는 것들이 1년 후에는 더 많아질 것이다. AI가 더 똑똑해지고, 도구가 더 편리해지고, 연결이 더 쉬워진다.

5년 후에는 지금 바이브코딩으로 하는 일의 대부분이 더 쉬워져 있을 것이다. 10년 후에는 이것이 당연한 것이 되어 있을 것이다.

지금 시작하는 것이 유리한 이유가 거기 있다.

처음에 배우는 사람이 나중에 가르친다. 지금 어색하게 말하면서 만들어보는 경험이, 나중에 이것을 모르는 사람들에게 전달할 수 있는 자원이 된다.

코딩을 모르는 당신이 지금 바이브코딩을 배우는 것은, 10년 뒤의 당신을 위한 투자다.

에필로그 — 당신도 할 수 있다

이 책을 쓴 나는 코딩을 모른다. 지금도 모른다. 앞으로도 코딩 문법을 배울 계획이 없다.

그래도 된다는 것을 이제 안다.

원하는 것을 말할 수 있으면 충분하다. 무엇이 필요한지 생각할 수 있으면 충분하다. 결과를 보고 맞는지 틀린지 판단할 수 있으면 충분하다.

이 세 가지는 코딩 공부와 상관없다. 살면서 자연스럽게 키워온 능력이다.

당신도 이미 그 능력을 갖고 있다.

시작은 아주 작은 것으로 하면 된다.

지금 가장 귀찮은 반복 업무 하나를 떠올려라.

그것을 AI에게 말해라.

그게 바이브코딩의 시작이다.

부록: 바이브코딩 프롬프트 50선

[일상 자동화]

> “매일 아침 날짜별 할일 파일을 만들어줘. 요일마다 자동으로 들어가는 항목을 다르게 설정해줘.”
> “이 폴더의 파일들을 날짜별로 자동 정리해줘.”
> “매주 월요일에 이번 주 목표를 작성하는 파일을 만들어줘.”
> “이 내용을 날짜별 폴더에 자동으로 저장해줘.”
> “매월 마지막 날 월간 정리 파일을 만들어줘.”

[기록 시스템]

> “꿈 내용을 말하면 분석해서 날짜순으로 저장해줘.”
> “오늘 있었던 일을 말하면 일기 파일에 추가해줘.”
> “대화 내용을 주제별로 분류해서 저장해줘.”
> “감정 상태를 기록하면 날짜와 함께 저장해줘.”

"독서 메모를 책 제목별로 분류해서 저장해줘."

[글쓰기]

"이 주제로 블로그 글을 써줘. 도입, 본문, 결론 구조로."
"이 내용을 SNS용으로 짧게 요약해줘."
"이 글을 더 친근한 말투로 바꿔줘."
"이 주제로 전자책 목차를 만들어줘. 7챕터 구성으로."
"이 목차로 전자책 내용을 써줘."
"이 긴 글을 핵심만 뽑아서 요약해줘."
"이 블로그 글을 유튜브 스크립트로 바꿔줘."
"이 내용을 카드뉴스 5장 형식으로 만들어줘."
"이 글의 제목을 SEO에 맞게 5가지 버전으로 써줘."
"이 글을 영어로 번역해줘. 자연스럽게."

[정보 처리]

"이 긴 텍스트에서 핵심 키워드만 뽑아줘."
"이 회의록을 결정사항과 액션아이템만 정리해줘."
"이 데이터를 표 형식으로 정리해줘."
"이 문서에서 날짜가 들어간 부분만 모아줘."
"이 텍스트를 날짜순으로 정렬해줘."

[이메일·문서]

"이 내용으로 정중한 이메일을 써줘."
"이 이메일에 답장을 써줘. 긍정적으로."
"회의 결과를 보고서 형식으로 정리해줘."
"이 제안서를 더 설득력 있게 다듬어줘."
"이 계약서에서 중요한 조항만 정리해줘."

[시스템 만들기]

"이 작업을 매일 자동으로 실행하는 스크립트를 만들어줘."
"이 폴더를 자동으로 백업하는 방법을 만들어줘."
"텔레그램 봇으로 이 명령을 원격으로 실행하는 법을 만들어줘."
"이 반복 작업을 자동화해줘."
"이 두 시스템을 연결하는 방법을 만들어줘."

[학습]

"이 개념을 초등학생도 이해할 수 있게 설명해줘."
"이 주제에 대해 핵심 10가지를 알려줘."
"이 내용의 찬반 양쪽 논거를 정리해줘."
"이 용어의 의미와 실제 사용 예를 알려줘."
"이 주제를 공부하는 로드맵을 만들어줘."

[분석]

"이 데이터에서 패턴을 찾아줘."
"이 상황의 원인과 해결책을 분석해줘."
"이 결정의 장단점을 비교해줘."
"이 두 가지 방법의 차이를 설명해줘."
"이 내용에서 논리적 오류를 찾아줘."

[창작]

"이 주제로 짧은 수필을 써줘."
"이 경험을 독자가 공감할 수 있게 글로 써줘."
"이 이야기의 다음 내용을 써줘."
"이 사례를 스토리텔링 형식으로 써줘."
"이 개념을 비유를 써서 설명해줘."
