

목차

프롤로그: 두 사람의 이야기

부모님 집에서 시작한 청년

27번 실패한 개발자

그들과 당신의 차이

이 책이 도와드릴 것

시작하기 전에

Chapter 1. AI 시대, 코딩의 새로운 패러다임

학습 목표

1. 전통적인 코딩 vs 바이브 코딩

2. AI가 코딩을 어떻게 바꾸고 있는가

3. 왜 지금이 시작하기 좋은 시기인가

실생활 비유: 요리로 이해하는 바이브 코딩

실제 사례: Pieter Levels 이야기

핵심 포인트

다음 챕터 미리보기

Chapter 2. 바이브 코딩 vs 전통적 코딩

학습 목표

1. 전통적 코딩의 어려움

2. 바이브 코딩의 장점

실생활 비유: 자동차 운전 vs 자율주행

실제 사례: Marc Lou의 27번의 시도

핵심 포인트

다음 챕터 미리보기

Chapter 3. 당신도 할 수 있는 이유

학습 목표

1. 나이는 숫자에 불과합니다

2. 경험이 곧 아이디어

3. 작게 시작하는 용기

실생활 비유: 씨앗 심기

실제 사례: 늦게 시작해도 성공할 수 있습니다

핵심 포인트

다음 챕터 미리보기

Part 1 마무리: 바이브 코딩 마인드셋 점검

Part 1 핵심 요약

실습 과제: 내가 만들고 싶은 것 3가지 적어보기

자가 진단 체크리스트

당신도 할 수 있습니다

Part 2 미리보기: 본격적으로 시작하기

Chapter 4. AI 도구 설치하고 첫 대화 나누기

학습 목표

1. Claude 가입하기

2. Cursor 설치하기

3. AI와 첫 대화 나누기

실생활 비유: 새 스마트폰 처음 켜기

실제 사례: 처음 AI를 써본 사람들의 반응

핵심 포인트

다음 챕터 미리보기

Chapter 5. 10분 만에 첫 웹페이지 만들기

학습 목표

1. AI에게 웹페이지 요청하기

2. 수정 요청하기

3. 더 멋진 웹페이지 만들기

실생활 비유: 레고 블록으로 집 만들기

실제 사례: 바이브 코딩 10분 챌린지

핵심 포인트

다음 챕터 미리보기

Chapter 06. 내 웹페이지를 세상에 공개하기

학습 목표

1. 배포란 무엇인가요?

2. Netlify로 배포하기

3. 도메인 이야기

실생활 비유: 가게 문 열기

실제 사례: Pieter Levels의 첫 배포 이야기

핵심 포인트

다음 챕터 미리보기

Part 2 마무리: 첫 웹사이트 만들기 프로젝트

Part 2 핵심 요약

실습 과제: 나만의 자기소개 웹페이지를 만들고 배포하기

자가 진단 체크리스트

당신도 할 수 있습니다

Part 3 미리보기: 본격 프로젝트 시작하기

Chapter 07. 실전 프로젝트 기획하기

학습 목표

1. 아이디어는 불편함에서 시작합니다

2. MVP: 핵심 기능만 담은 첫 버전

실생활 비유: 식당 메뉴판

실제 사례: Marc Lou의 ShipFast

3. 내 프로젝트 기획하기

핵심 포인트

다음 챕터 미리보기

Chapter 08. 기능 하나씩 추가하기

학습 목표

1. 좋은 프롬프트 작성법

2. 기능 추가 순서

3. 에러 해결하기

실생활 비유: 집 인테리어

실제 사례: NomadList 기능 추가 히스토리

핵심 포인트

다음 챕터 미리보기

Chapter 09. 디자인 다듬기

학습 목표

1. 색상과 폰트 바꾸기
2. 레이아웃 개선하기
3. 모바일 대응 (반응형 디자인)

실생활 비유: 옷 입히기

실제 사례: 심플한 디자인으로 성공한 서비스들

핵심 포인트

다음 챕터 미리보기

Chapter 10. 사용자 피드백 받고 개선하기

학습 목표

1. 주변 5명에게 보여주기
2. 피드백 정리하기: 중요/나중에/무시
3. 우선순위 정해서 개선하기

실생활 비유: 시식 코너

실제 사례: Pieter Levels의 트위터 피드백 전략

핵심 포인트

다음 챕터 미리보기

Part 3 마무리: 나만의 앱 아이디어 구체화

Part 3 핵심 요약

실습 과제: 내 프로젝트 기획서 1장 작성하기

자가 진단 체크리스트

당신도 할 수 있습니다

Part 4 미리보기: 수익화 시작하기

Chapter 11. 수익화 전략 세우기

학습 목표

1. 수익 모델의 종류

2. 내 프로젝트에 맞는 모델 찾기

3. 가격 정하기

실생활 비유: 과일 가게

실제 사례: Pieter Levels의 수익 모델

핵심 포인트

다음 챕터 미리보기

Chapter 12. 첫 수익 만들기

학습 목표

1. 결제 시스템 연동하기

2. 첫 고객 찾기

3. 가격 테스트하기

실생활 비유: 시장에서 첫 손님 맞이하기

실제 사례: Marc Lou의 첫 100달러

핵심 포인트

다음 챕터 미리보기

Chapter 13. 마케팅의 기초

학습 목표

1. SNS 활용하기

2. 콘텐츠 마케팅

3. 커뮤니티 참여하기

실생활 비유: 동네 입소문

실제 사례: Marc Lou의 트위터 마케팅

핵심 포인트

다음 챕터 미리보기

Chapter 14. 수익 확장하기

학습 목표

1. 두 번째 제품 만들기

2. 기존 제품 업그레이드

실생활 비유: 과수원 키우기

실제 사례: Pieter Levels의 다중 프로젝트 전략

핵심 포인트

다음 챕터 미리보기

Part 4 마무리: 수익화 계획 세우기

Part 4 핵심 요약

실습 과제: 내 프로젝트의 수익 모델 캔버스 작성하기

자가 진단 체크리스트

당신도 할 수 있습니다

Part 5 미리보기: 지속 가능한 성장

Chapter 15. 자동화로 시간 벌기

학습 목표

1. 반복 작업 자동화하기
2. AI 챗봇으로 고객 응대하기
3. 모니터링 자동 설정

실생활 비유: 세탁기와 식기세척기

실제 사례: 1인 운영자의 하루

핵심 포인트

다음 챕터 미리보기

Chapter 16. 나만의 커뮤니티 만들기

학습 목표

1. 사용자 커뮤니티의 힘
2. 커뮤니티 시작하기
3. 커뮤니티 운영 노하우

실생활 비유: 동네 모임

실제 사례: NomadList 커뮤니티

핵심 포인트

다음 챕터 미리보기

Chapter 17. 1인 사업가의 장기 비전

학습 목표

1. 지속 가능한 사업

2. 건강과 균형

3. AI 기술의 미래와 기회

실생활 비유: 마라톤

실제 사례: Pieter Levels 10년의 여정

핵심 포인트

마무리

Part 5 마무리: 지속 가능한 1인 사업 로드맵

Part 5 핵심 요약

실습 과제: 나의 12개월 로드맵 만들기

자가 진단 체크리스트

당신도 할 수 있습니다

에필로그 안내

에필로그: 당신의 이야기는 이제 시작입니다

여기까지 오신 당신에게

여러분이 얻은 것들

한 가지 부탁

완벽하지 않아도 시작하세요

Pieter Levels가 당신에게 하고 싶은 말

부록

A. 유용한 도구 목록

B. 자주 묻는 질문 (FAQ)

C. 용어 사전

D. 이 책에서 소개한 인물

바이브 코딩으로 1인 사업가 되기

코딩 몰라도 괜찮아요, AI가 함께 합니다



Author: J.M

프롤로그: 두 사람의 이야기

부모님 집에서 시작한 청년

2014년, 네덜란드의 한 청년이 있었습니다.

이름은 Pieter Levels(피터 레벨스).

그는 부모님 집에서 지내고 있었습니다.

우울증을 겪고 있었습니다.

특별한 학위도, 직장도 없었습니다.

그가 한 일은 단순했습니다.

“12개월 안에 12개 프로젝트를 만들자.”

노트북 하나로 시작했습니다.

대단한 기술이 있던 것은 아닙니다.

간단한 코딩을 독학으로 배웠을 뿐입니다.

12개 중 대부분은 실패했습니다.

하지만 몇 개가 살아남았습니다.

그중 하나가 NomadList(노매드리스트)입니다.

디지털 노마드를 위한 도시 정보 사이트입니다.

2025년 현재, 그의 연 수입은 약 38억 원입니다.

직원 없이 혼자서 운영합니다.

27번 실패한 개발자

또 한 사람이 있습니다.

Marc Lou(마크 루).

그는 프랑스 출신의 1인 개발자입니다.

그에게도 화려한 시작은 없었습니다.

만든 제품이 27번이나 실패했습니다.

27번이라는 숫자를 생각해 보세요.

보통 사람이라면 5번쯤에서 포기했을 것입니다.

하지만 그는 계속했습니다.

전략을 바꿨습니다.

“작은 제품을 빠르게 만들고,

반응이 있는 것만 키우자.”

28번째 시도가 성공했습니다.

그리고 29번째, 30번째도 성공했습니다.

지금 그는 월 수천만 원을 벌고 있습니다.

역시 혼자서 운영합니다.

그들과 당신의 차이

이 두 사람의 이야기를 들으면 이런 생각이 드실 수 있습니다.

“그 사람들은 특별한 거 아니야?”

“젊으니까 가능했던 거 아닌가?”

“나는 코딩을 모르는데…”

한 가지 중요한 사실이 있습니다.

Pieter와 Marc가 시작할 때는 AI가 없었습니다.

그들은 코딩을 직접 배워야 했습니다.
에러가 나면 혼자 해결해야 했습니다.
모든 것을 손으로 만들어야 했습니다.

2025년 지금, 여러분에게는 AI가 있습니다.

- 코드를 몰라도 AI가 만들어 줍니다
- 에러가 나면 AI가 고쳐 줍니다
- 디자인도 AI가 도와줍니다
- 마케팅 문구도 AI가 써줍니다

그들보다 훨씬 유리한 조건에서 시작하는 것입니다.

이 책이 도와드릴 것

이 책은 여러분을 위해 쓰였습니다.

코딩을 전혀 모르는 분.

컴퓨터가 조금 어려운 분.

하지만 **“나도 뭔가 만들어 보고 싶다”**는 마음이 있는 분.

이 책을 다 읽고 나면 이런 것들을 할 수 있게 됩니다.

1. AI 도구를 자유롭게 사용할 수 있습니다
2. 나만의 웹사이트나 서비스를 만들 수 있습니다
3. 만든 것을 인터넷에 공개할 수 있습니다
4. 첫 수익을 만드는 방법을 알게 됩니다
5. 지속적으로 성장하는 방법을 배웁니다

어렵지 않습니다.

천천히, 한 걸음씩 함께 가겠습니다.

시작하기 전에

한 가지만 약속해 주세요.

“완벽하지 않아도 괜찮다.”

이 책을 읽으면서 막히는 부분이 있을 수 있습니다.

이해가 안 되는 부분도 있을 수 있습니다.

괜찮습니다.

다시 읽으면 됩니다.

AI에게 물어보면 됩니다.

주변에 도움을 요청하면 됩니다.

중요한 건 **멈추지 않는 것**입니다.

Pieter Levels도 처음에는 아무것도 몰랐습니다.

Marc Lou도 27번이나 실패했습니다.

그들이 성공한 비결은 단 하나입니다.

멈추지 않았다는 것.

자, 이제 시작해 볼까요?

첫 페이지를 넘겨 주세요.

Chapter 1. AI 시대, 코딩의 새로운 패러다임

학습 목표

이 챕터를 읽고 나면 이런 것들을 알게 됩니다.

1. 전통적인 코딩과 바이브 코딩이 어떻게 다른지 이해합니다.
 2. AI가 코딩을 어떻게 바꾸고 있는지 큰 그림을 그릴 수 있습니다.
 3. 지금이 시작하기에 가장 좋은 시기인 이유를 설명할 수 있습니다.
-

1. 전통적인 코딩 vs 바이브 코딩

코딩, 꼭 어렵게 배워야 할까?

“코딩을 배우려면 영어부터 해야 하나요?”

많은 분들이 이렇게 물어봅니다.

솔직히 말하면, 예전에는 맞는 말이었습니다.

전통적인 코딩은 이런 과정이었습니다.

프로그래밍 언어를 배운다 →
문법을 외운다 →
에러를 찾는다 →
또 에러를 찾는다 →
몇 달이 지나서야 뭔가 만들어진다

대학교에서 4년을 배워도 막상 실무에서는 또 새로 배워야 했습니다.

바이브 코딩은 다릅니다

바이브 코딩은 완전히 다른 방식입니다.

내가 원하는 것을 말로 설명하면, AI가 코드를 만들어 줍니다.

예를 들어 볼까요?

전통적인 코딩	바이브 코딩
HTML, CSS, JavaScript를 배운다	"예약 페이지 만들어줘"라고 말한다
코드를 한 줄 한 줄 직접 작성한다	AI가 코드를 만들어 준다
에러가 나면 직접 찾아 고친다	AI에게 "에러 고쳐줘"라고 말한다
완성까지 몇 주~몇 달	완성까지 몇 시간~며칠

전통 코딩 vs 바이브 코딩

전통 코딩



- 수개월 학습 필요



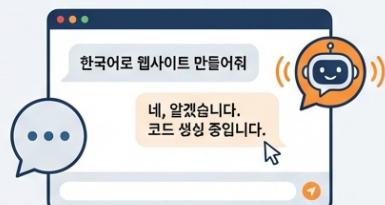
- 프로그래밍 언어 암기



- 에러 직접 디버깅



바이브 코딩



- 바로 시작 가능



- 한국어로 대화



- AI가 에러 해결



바이브 코딩에서 가장 중요한 능력은 이것입니다.

“내가 뭘 만들고 싶은지 정확히 설명하는 능력”

코드를 몰라도 됩니다.

영어를 잘 몰라도 됩니다.

한국어로 설명하면 됩니다.

2. AI가 코딩을 어떻게 바꾸고 있는가

10년 전과 지금의 차이

2015년에 간단한 웹사이트를 만들려면 이런 것들이 필요했습니다.

- HTML/CSS 기초 학습 (2~4주)
- JavaScript 기본 문법 (4~8주)
- 서버 구축 방법 (4~8주)
- 데이터베이스 설계 (2~4주)

최소 3개월은 공부해야 겨우 시작할 수 있었습니다.

2025년, 지금은 어떨까요?

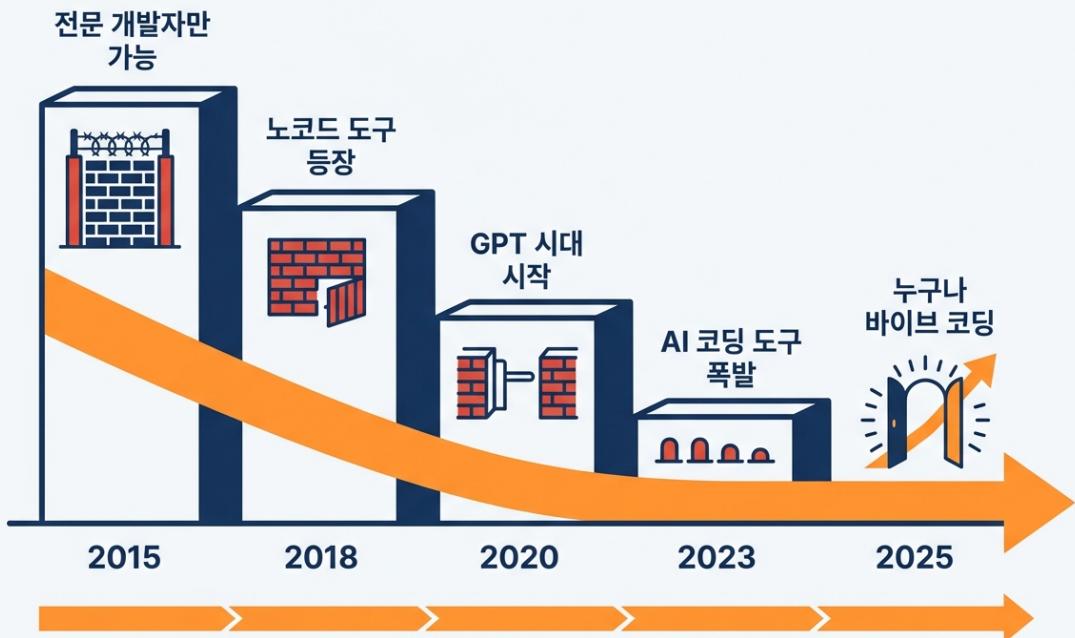
지금은 AI 도구 하나만 있으면 됩니다.

- [Claude](#) — 코드를 대신 작성해 줍니다
- [Cursor](#) — AI가 내장된 코드 편집기입니다
- [Bolt/Lovable](#) — 말로 설명하면 웹사이트가 만들어집니다
- [Vercel/Netlify](#) — 클릭 몇 번으로 전 세계에 공개됩니다

이 도구들은 대부분 [무료](#)이거나 월 2~3만원이면 충분합니다.

진입 장벽이 사라졌습니다

코딩 진입장벽의 변화 (2015→2025)



예전에는 이런 사람만 개발자가 될 수 있었습니다.

“컴퓨터공학과 졸업”

“코딩 학원 6개월 수료”

“개발 경력 3년 이상”

지금은 다릅니다.

“내가 이런 서비스를 만들고 싶어”

이 한 마디면 시작할 수 있습니다.

3. 왜 지금이 시작하기 좋은 시기인가

세 가지 이유

첫째, AI 도구가 충분히 성숙했습니다.

2022년 ChatGPT가 처음 나왔을 때는 아직 부족했습니다.
코드를 만들어 달라고 하면 절반은 틀렸습니다.

2025년 지금은 다릅니다.

AI가 만드는 코드의 품질이 크게 좋아졌습니다.
간단한 서비스라면 거의 완벽하게 만들어 냅니다.

둘째, 성공 사례가 쌓이고 있습니다.

바이브 코딩으로 실제 돈을 버는 사람들이 나타났습니다.
이 책에서도 여러 사례를 소개하겠습니다.
"정말 되는 거구나"를 직접 확인할 수 있습니다.

셋째, 경쟁자가 아직 적습니다.

대부분의 사람들은 아직 이렇게 생각합니다.

“코딩? 나는 그런 거 못 해.”
“AI? 짧은 사람들이나 쓰는 거 아냐?”

그래서 지금이 기회입니다.

남들이 망설이는 사이에 시작하면 됩니다.

실생활 비유: 요리로 이해하는 바이브 코딩

코딩이 아직 어렵게 느껴지시나요?

요리에 비유해 보겠습니다.

전통적인 코딩 = 재료부터 직접 만들어 요리하기

밀가루를 직접 빻고, 면을 직접 뽑고, 소스를 직접 만듭니다.
맛있는 파스타 한 접시를 만들려면 몇 시간이 걸립니다.
전문 요리사만 할 수 있는 일이었습니다.

바이브 코딩 = 밀키트로 요리하기

밀키트를 주문하면 재료가 손질되어 옵니다.
레시피대로 따라하면 30분이면 완성됩니다.
요리 초보도 맛있는 한끼를 만들 수 있습니다.

AI 도구 = 똑똑한 요리 보조 셰프

옆에서 전문 셰프가 도와준다고 상상해 보세요.

“지금 불 세기를 좀 줄이세요.”
“소금은 이 정도만 넣으면 됩니다.”
“이 순서로 하면 더 맛있어요.”

AI 도구가 바로 이 역할을 합니다.
내가 방향을 정하면 AI가 실행을 도와줍니다.
중요한 건 요리사(나)의 아이디어입니다.
무엇을 만들지 결정하는 건 바로 나입니다.

실제 사례: Pieter Levels 이야기

부모님 집에서 시작한 1인 사업가

Pieter Levels(피터 레벨스)라는 사람이 있습니다.
네덜란드 출신의 1인 개발자입니다.

2014년, 그는 이런 상황이었습니다.

- 부모님 집에 얹혀살고 있었습니다
- 우울증을 겪고 있었습니다
- 특별한 기술이나 학위가 없었습니다
- 돈도 거의 없었습니다

그가 한 일은 단순했습니다.

“사람들에게 필요한 간단한 웹사이트를 만들자.”

12개 프로젝트 도전

그는 "12개월 안에 12개 프로젝트 만들기"에 도전했습니다.

대단한 기술이 필요한 게 아니었습니다.

간단한 도구를 사용해서 빠르게 만들었습니다.

실패한 것도 많았지만, 몇 개가 성공했습니다.

그중 하나가 [NomadList](#)(노매드리스트)입니다.

디지털 노마드를 위한 도시 정보 사이트입니다.

그리고 지금

2025년 현재, Pieter Levels의 연 수입은 약 38억 원입니다.

직원 없이 혼자서 운영합니다.

여기서 중요한 포인트가 있습니다.

2014년에는 간단한 사이트를 만드는 것도 꽤 어려웠습니다.

그때도 기본적인 코딩 지식은 필요했습니다.

2025년 지금은 AI가 있습니다.

그때보다 훨씬 더 쉽게 시작할 수 있습니다.

Pieter Levels도 최근 인터뷰에서 이렇게 말했습니다.

“지금 시작하는 사람들이 부럽다.
AI 덕분에 예전보다 10배는 빠르게 만들 수 있다.”

핵심 포인트

이 챕터에서 꼭 기억할 것들입니다.

1. **바이브 코딩**은 코드를 직접 작성하는 것이 아니라, AI에게 원하는 것을 설명하는 방식입니다.
2. AI 도구가 충분히 발전해서, 코딩을 몰라도 서비스를 만들 수 있는 시대가 되었습니다.
3. 가장 중요한 능력은 프로그래밍이 아니라 **“무엇을 만들고 싶은지 아는 것”**입니다.
4. 지금이 시작하기 가장 좋은 시기입니다. AI는 충분히 좋아졌고, 경쟁자는 아직 적습니다.
5. Pieter Levels처럼 특별한 배경 없이도 시작할 수 있습니다. 지금은 그때보다 훨씬 쉽습니다.

다음 챕터 미리보기

Chapter 2에서는 바이브 코딩에 사용하는 **AI 도구들을** 하나씩 살펴봅니다.

- Claude, Cursor, Bolt 같은 도구가 뭔지
- 어떤 도구를 먼저 써보면 좋은지
- 무료로 시작하는 방법

도구를 알아야 요리를 시작할 수 있겠죠?

다음 챕터에서 여러분의 주방을 세팅해 보겠습니다.

Chapter 2. 바이브 코딩 vs 전통적 코딩

학습 목표

이 챕터를 읽고 나면 이런 것들을 알게 됩니다.

1. 전통적 코딩이 왜 어려웠는지 구체적으로 이해합니다.
 2. 바이브 코딩이 가진 [핵심 장점 3가지](#)를 설명할 수 있습니다.
 3. 실패 비용이 거의 없다는 사실에 자신감을 얻습니다.
-

1. 전통적 코딩의 어려움

외국어를 배우는 것과 같았습니다

전통적 코딩은 새로운 언어를 배우는 것이었습니다.

영어도 아닌, 컴퓨터만 알아듣는 언어였습니다.

```
function calculateTotal(items) {  
    return items.reduce((sum, item) => sum + item.price, 0);  
}
```

이 코드가 무슨 뜻인지 아시나요?

"물건들의 가격을 다 합쳐라"라는 뜻입니다.

이걸 쓰려면 몇 달을 배워야 했습니다.

에러와의 끝없는 싸움

코드를 작성하면 에러가 납니다.

점 하나, 쉼표 하나 빠져도 안 돌아갑니다.

에러 메시지는 영어로 나옵니다.

무슨 뜻인지 모르겠습니다.

검색해서 찾아봐야 합니다.

찾아봐도 이해가 안 됩니다.

이 과정에서 대부분 포기합니다.

배워야 할 것이 너무 많았습니다

웹사이트 하나를 만들려면 이것들을 알아야 했습니다.

- **HTML**: 웹페이지의 뼈대
- **CSS**: 웹페이지의 디자인
- **JavaScript**: 웹페이지의 동작
- **서버**: 데이터를 저장하는 곳
- **데이터베이스**: 정보를 정리하는 곳

이 다섯 가지를 다 배우려면 최소 6개월이 걸립니다.

6개월 뒤에도 초보입니다.

2. 바이브 코딩의 장점

말로 하면 됩니다

바이브 코딩에서는 코드를 쓸 필요가 없습니다.

전통적 코딩:

코드를 직접 작성한다

바이브 코딩:

"로그인 페이지를 만들어줘"라고 말한다

이것이 전부입니다.

AI가 코드를 대신 작성해 줍니다.

속도가 다릅니다

전통적 코딩으로 로그인 페이지를 만들면?

초보자 기준 2~3일이 걸립니다.

바이브 코딩으로 같은 페이지를 만들면?

10분이면 됩니다.

작업	전통적 코딩	바이브 코딩
로그인 페이지	2~3일	10분
게시판	1~2주	30분~1시간
결제 기능	3~5일	1시간
전체 웹사이트	1~3개월	1~3일

10배 이상 빠릅니다.

개발 속도 비교



실패 비용이 거의 제로입니다

전통적으로 서비스를 만들려면 이런 비용이 들었습니다.

- 개발자 고용: 월 수백만 원
- 외주 개발: 수천만 원
- 시간 투자: 최소 3~6개월

돈을 들이고 시간을 들여서 만들었는데 실패하면?

큰 손해입니다.

바이브 코딩은 다릅니다.

- AI 도구 비용: 무료 또는 월 2~3만 원
- 시간 투자: 며칠
- 실패하면? 다시 만들면 됩니다

실패해도 잊는 것이 거의 없습니다.

그래서 마음 편하게 도전할 수 있습니다.

실생활 비유: 자동차 운전 vs 자율주행

전통적 코딩 = 수동 변속기 운전 배우기

예전에는 운전을 배우려면 많은 것을 알아야 했습니다.

클러치를 밟고, 기어를 넣고, 악셀을 조절합니다.

타이밍이 안 맞으면 차가 덜컹거립니다.

시동이 꺼지기도 합니다.

운전면허를 따는 데만 몇 달이 걸렸습니다.

면허를 따도 도로에 나가면 무서웠습니다.

전통적 코딩이 이렇습니다.

배울 것이 많고, 실수가 잦고, 시간이 오래 걸립니다.

바이브 코딩 = 자율주행차 타기

자율주행차를 상상해 보세요.

“강남역으로 가줘.”

이 한마디면 됩니다.

클러치도 모르고, 기어도 모르고, 도로 법규도 몰라도 됩니다.

차가 알아서 갑니다.

바이브 코딩이 이렇습니다.

“로그인 페이지 만들어줘.”

이 한마디면 AI가 알아서 만들어 줍니다.

내가 해야 할 일은 목적지를 정하는 것뿐입니다.

중요한 건 어디로 갈지 아는 것

수동 운전이든 자율주행이든 공통점이 있습니다.

어디로 갈지는 내가 정해야 한다는 것입니다.

바이브 코딩도 마찬가지입니다.
무엇을 만들지 결정하는 건 나입니다.
AI는 그 결정을 실행해 주는 도구입니다.

실제 사례: Marc Lou의 27번의 시도

실패가 무서운 이유

“만들어 봤자 실패하면 어쩌지?”
이런 걱정이 가장 큰 장벽입니다.
특히 시간과 돈이 많이 들었을 때 실패가 무섭습니다.

Marc Lou는 27번 실패했습니다

Marc Lou(마크 루)라는 사람이 있습니다.
프랑스 출신의 1인 개발자입니다.
그는 무려 **27개의 제품**을 만들었습니다.
그리고 **27개 모두 실패했습니다**.

보통 사람이라면 5번쯤에서 포기했을 겁니다.
10번이면 “나는 안 되는 사람인가 보다” 생각했을 겁니다.

하지만 Marc Lou는 계속했습니다.
왜 가능했을까요?

실패 비용이 작았기 때문입니다

Marc Lou는 각 제품을 **며칠** 만에 만들었습니다.
비용도 거의 들지 않았습니다.

실패해도 잃는 것이 거의 없었습니다.
“안 되면 다음 거 만들자.”
이런 마음으로 가볍게 시도할 수 있었습니다.

전통적 코딩이었다면 27번 시도가 가능했을까요?
각각 3개월씩 걸렸다면 **7년**이 필요합니다.
현실적으로 불가능합니다.

28번째가 성공했습니다

27번 실패 후, 28번째 제품이 드디어 성공했습니다.
그리고 그 뒤로도 연달아 성공했습니다.

지금 Marc Lou는 월 수천만 원을 벌고 있습니다.
혼자서 운영합니다.

핵심 교훈은 이것입니다.

“빠르게 만들고, 빠르게 실패하고, 빠르게 다시 시작하라.”

바이브 코딩이기에 가능한 전략입니다.

핵심 포인트

이 챕터에서 꼭 기억할 것들입니다.

1. **완벽함보다 실행이 중요합니다.** 27번 실패한 Marc Lou가 증명했습니다.
2. **AI가 반복 작업을 대신합니다.** 코드 작성, 에러 수정, 디자인까지 AI가 도와줍니다.
3. **창의성에 집중할 수 있습니다.** 기술적 부분은 AI에게 맡기고, 나는 아이디어에 집중합니다.
4. **실패 비용이 거의 제로입니다.** 시간도 돈도 적게 들어서 마음 편하게 도전할 수 있습니다.