「월간클로드」 2기 · 기초 이론서

만들고, 올린다

내 사이트는 인터넷에,
내 앱은 **구글 플레이에.**

다 읽고 나면
카톡으로 보낼 수 있는 내 웹사이트 주소와
플레이스토어에서 받아 설치되는 내 앱이 남습니다.
그리고 그 사이에 무슨 일이 있었는지 설명할 수 있습니다.

전석환 지음

v0.1 초고 · 2026년 9월 14일

작가의 말

> 초고 v0.2 — **저자 목소리로 다듬으실 자리입니다.** 저자가 직접 짚어 준 핵심(딸깍 이후 · 만든 사람의 생각을 기록해 유지보수)을 뼈대로 삼고, 나머지는 2기 강의 기획서와 강의 사이트에서 확인되는 사실만으로 채웠습니다.

유튜브에 이런 말이 돌아다닙니다.

> **"딸깍 하면 웹사이트가 만들어진다."**

틀린 말이 아닙니다. AI에게 "내 공방 소개 페이지 만들어줘" 라고 하면 정말 금방 나옵니다. 20분이면 그럴듯한 사이트가 화면에 뜹니다. 그걸 보면 누구나 이렇게 생각합니다.

> "와, 나도 이제 AI로 웹이든 앱이든 그냥 쉽게 만들 수 있겠네."

그래서 강의를 기획하면서 가장 오래 붙잡고 있던 질문이 이것이었습니다.

> **"Claude한테 시키면 20분이면 나오는데, 이걸 왜 배워요?"**

답은 **만들고 난 다음**에 있습니다.

사이트가 한 번 나오고 나면, 반드시 고칠 일이 생깁니다. 사진을 바꾸고 싶고, 가격이 바뀌었고, 맨 위 큰 화면이 좀 답답해 보입니다.

그런데 여기서 막힙니다. **내가 직접 만든 게 아니기 때문입니다.**

AI가 만들어 줬으니, 어디를 어떻게 건드려야 하는지 모릅니다. 사이트 맨 위에 큼직하게 들어가는 첫 화면을 **히어로(Hero)** 영역이라고 부르는데, 이 단어조차 모르는 게 당연합니다. 부를 이름을 모르니 고쳐 달라는 말도 흐릿해집니다.

그래서 이렇게만 말하게 됩니다.

> "수정해줘."

AI는 고쳐 줍니다. 문제는 **어떻게** 고치느냐입니다.

AI는 이 사이트를 **처음에 왜 이렇게 만들었는지** 모릅니다. 왜 이 색을 골랐는지, 왜 메뉴를 세 개로 줄였는지, 왜 예약 버튼을 맨 위에 올렸는지 — 만든 사람의 생각이 어디에도 남아 있지 않으니까요. 그러면 AI는 **자기 생각대로** 고칩니다. 그럴듯하게, 하지만 다르게.

몇 번 "수정해줘" 를 반복하고 나면, 처음 사이트와는 **완전히 다른 느낌의 사이트**가 되어 있습니다. 고친 게 아니라 새로 만든 것에 가깝습니다.

딸깍으로 만든 사이트는, 딸깍으로 고칠 수 없습니다.

이 책과 강의가 가려는 방향은 하나입니다.

> **만든 사람의 생각을 기록해 두고, 그 기록을 지키면서 고친다.**

무엇을 만들고 **무엇을 안 만들기로 했는지**, 왜 이 화면을 골랐는지, 무엇이 **왜 바뀌었는지.** 이것들을 글로 남겨 둡니다. 개발자들은 이런 문서를 설계 문서라고 부르고, 그 안에 담긴 것을 **철학**이라고 부르기도 합니다.

기록이 있으면 AI가 달라집니다. "수정해줘" 라고만 해도, AI는 먼저 그 기록을 읽고 **원래 생각에 맞춰서** 고칩니다. 석 달 뒤에 다시 열어도, 다른 사람이 맡아도, 사이트는 처음 느낌을 잃지 않습니다. 이것을 **유지보수**라고 합니다. 만든 것을 망가뜨리지 않고 계속 쓸 수 있게 돌보는 일입니다.

그래서 2기 첫 주는 코드를 한 줄도 쓰지 않습니다. 두 시간 내내 **정하고, 기록합니다.** 뭘 만들지, 뭘 안 만들지, 어떻게 생겼는지. 화면 시안을 보고 나서는 설계 문서를 다시 고쳐 쓰고, **뭐가 왜 바뀌었는지** 남깁니다. 느리게 보이지만, 이게 둘째 주부터의 모든 수정을 빠르게 만듭니다.

이 부분이 이 책의 핵심입니다.

그리고 하나 더 있습니다. **AI가 대신 해주지 못하는 구간**입니다.

개발자 계정을 만들고 신분증으로 본인 확인을 받는 일, 사이트를 인터넷 주소에 연결하는 일, 스토어 등록 양식을 한 칸씩 채우는 일. 이건 말로 시켜서 끝나지 않습니다. **사람이 직접 가입하고, 직접 누르고, 직접 기다려야 합니다.** 딸깍 영상에는 이 부분이 나오지 않습니다.

구글 플레이에는 혼자 넘기 어려운 벽도 하나 있습니다. 개인 계정으로 앱을 정식 공개하려면 테스터 12명이 14일 동안 참여해야 합니다. 월간클로드는 매달 사람이 모이는 곳이라, 서로의 앱에 테스터가 되어줄 수 있습니다. **혼자서는 못 넘는 벽을 기수가 같이 넘습니다.**

그럼, 0장에서 뵙겠습니다.

전석환

목차

작가의 말 2

0부 — 준비

00 이 책은 무엇을 약속하는가 6

01 설치, 한 시간이면 끝난다 13

02 첫 대화, 그리고 권한 창 35

1부 — Claude Code라는 도구

03 AI에도 종류가 있다 45

04 챗봇과 무엇이 다른가 51

2부 — 설계와 시안 (Week 1)

05 작업 공간 만들기, CLAUDE.md와 Skill 58

06 1주차 지도, 왜 코드보다 설계가 먼저인가 70

07 무엇을 만들고, 무엇을 안 만들까 77

08 모으고, 견주고, 갈림길을 고른다 88

09 PRD, 설계 문서로 굳히기 98

10 화면 시안 고르기, 웹 넷 앱 넷 105

11 PRD 고쳐 쓰기, 왜 바뀌었는지 남긴다 121

쪽수는 이 PDF의 실제 쪽 번호입니다. 본문의 「N.N절」 표기는 해당 장 안의 절 번호입니다.

00장 — 이 책은 무엇을 약속하는가

> 초고 v0.1 (2026-09-14)

0.1 먼저, 이 책이 생긴 이유

「월간클로드」2기 강의는 4주 동안 이렇게 진행됩니다.

```
Week 0   같이 설치하기 (개강 전, 온라인 1시간)
Week 1   작업 환경 만들고, 설계·시안까지
Week 2   안드로이드 앱 만들기
Week 3   배포하기 — 웹은 Vercel, 앱은 구글 플레이
Week 4   공공데이터 활용하기
```

매주 월요일 밤 9시. 방식은 내내 같습니다. **준비된 프롬프트를 붙여넣고, Claude와 말로 주고받으며, 그 자리에서 만듭니다.**

그런데 두 시간 안에 뭔가를 끝까지 만들려면 **설명은 건너뛸 수밖에 없습니다.** 강의는 원래 그런 자리입니다. 그러면 이런 것이 남습니다.

> "화면에 창이 하나 떴는데 그냥 '허용' 눌렀어요. 그게 뭐였죠?"
>
> "Node.js는 왜 깔았어요? 한 번도 안 열어봤는데."
>
> "개발자 계정을 왜 그렇게 미리 하라고 하셨어요?"

손은 움직였는데, **머릿속에 이름이 안 붙은 겁니다.** 이 책은 그 빈자리를 채웁니다.

```
강의   손을 움직였다.  "이 상자를 붙여넣으세요" → 붙여넣었다 → 됐다
          ↓  남은 것:  "그런데 방금 그게 뭐였지?"
이 책   그 빈자리를 채운다
```

강의에서 그냥 넘어간 자리마다, 이 책은 세 가지를 채웁니다.

	채우는 것	예를 들면
①	**왜 그런가**	그 창은 AI가 내 컴퓨터의 파일을 건드리려 할 때 뜹니다
②	**뭐라고 부르는가**	그 창의 이름은 **권한 창**입니다
③	**언제 안 되는가**	이 방식이 통하지 않는 경우와, 그때 뭘 해야 하는지

②가 중요한 이유는 — **이름을 알아야 검색할 수 있기 때문입니다.** "그 창"은 검색이 안 되지만 "권한 창"은 검색이 됩니다.

0.2 강의를 안 들으셨어도 됩니다

이 책은 2기 수강생을 위해 시작했지만, **강의를 안 들은 분도 읽을 수 있게** 썼습니다. 실습 절차는 요약해서 실어 두었고, 무게는 설명 쪽에 두었습니다.

그리고 **코딩을 한 번도 안 해보셨어도 됩니다.** 이 책은 다음을 전부 모른다고 가정하고 씁니다.

- 프로그래밍 언어 문법 — HTML(에이치티엠엘)도, 자바스크립트도
- 터미널 (검은 화면에 글자 치는 그것)
- Git(깃)과 GitHub(깃허브)
- API(에이피아이)

전부 나오긴 나옵니다. 다만 **나올 때마다 처음부터 설명합니다.** "당연히 아시겠지만" 이라는 말은 이 책에 나오지 않습니다.

0.3 다 읽으면 무엇이 남는가

이 책의 약속은 한 문장입니다.

> **내 사이트는 인터넷에, 내 앱은 구글 플레이에.**

손에 남는 것은 두 가지입니다.

	무엇	언제
🌐	**카톡으로 보낼 수 있는 내 웹사이트 주소**	3주차
📱	**플레이스토어에서 받아 내 폰에 설치되는 내 앱**	3주차

그런데 이 책이 정말로 드리려는 것은 결과물이 아닙니다. Claude에게 "내 공방 소개 페이지 만들어줘" 라고 하면 **20분이면 나옵니다.** 그건 이 책 없이도 됩니다.

이 책이 드리는 것은 그 20분 **바깥**에 있는 두 가지입니다.

① 만든 것을 망가뜨리지 않고 고치는 법

AI가 만들어 준 사이트는 내가 짠 게 아니라서, 어디를 어떻게 고쳐야 할지 모릅니다. "수정해줘" 라고만 하면 AI는 **처음에 왜 그렇게 만들었는지 모르고** 자기 생각대로 고칩니다. 몇 번 반복하면 처음과 완전히 다른 사이트가 됩니다.

그래서 이 책은 **무엇을 만들고 무엇을 안 만들기로 했는지, 무엇이 왜 바뀌었는지**를 먼저 글로 남기게 합니다. 그 기록이 있으면 AI는 원래 생각에 맞춰 고칩니다. 한 번에 뽑는 건 시작이고, **값어치는 그다음 — 계속 고쳐 쓸 수 있게 하는 데 있습니다.**

② Claude가 대신 해주지 못하는 구간 ⭐

이쪽이 훨씬 큽니다.

🌐 웹사이트 쪽	📱 앱 쪽
코드를 GitHub에 올리고 Vercel에 연결	구글 개발자 계정 · 신원 확인
내 주소 정하기	스토어 등록정보 · 콘텐츠 등급
카톡 공유 미리보기 · 개인정보처리방침	심사 반려 대응 · 테스트 트랙

이건 말로 시켜서 끝나지 않습니다. **사람이 직접 가입하고, 직접 누르고, 직접 기다려야 합니다.** 3주차가 강의의 본체인 이유이고, 이 책이 그 앞의 장들을 공들여 준비하는 이유입니다.

0.4 이 책이 약속하지 않는 것

기술 책은 보통 할 수 있는 것부터 말합니다. 이 책은 **못 하는 것을 먼저 말합니다.** 기대가 어긋난 채로 시작하면 중간에 책을 덮게 되기 때문입니다.

① 4주 안에 앱이 "정식 출시" 되지는 않습니다

가장 조심해서 말해야 하는 부분입니다. 정확한 약속은 **"올리는 데까지"** 입니다.

단계	4주 안에?
🌐 인터넷에 올려서 주소 갖기 (아무나 접속)	✅ 3주차 · **조건 없음**
📱 앱 파일을 만들어 스토어에 올리기	✅ 3주차
📱 스토어를 통해 **내 폰에** 설치하기	✅ 3주차 목표
📱 **아무나 검색해서** 받기 (정식 공개)	개인 계정은 **테스터 12명 × 14일** 뒤

웹사이트는 조건이 없습니다. 주소를 받는 그날부터 누구나 들어올 수 있습니다. 앱은 다릅니다. 이 차이는 3주차에서 자세히 다룹니다.

② 아이폰 앱은 만들지 않습니다

안드로이드만 다룹니다. 애플 개발자 계정은 해마다 $99가 들고 심사도 까다롭습니다. 아이폰만 있는 분도 2주차 실습은 따라올 수 있지만, 스토어를 통해 설치하는 것은 안드로이드 폰에서만 됩니다.

③ 개발자가 되지는 않습니다

목표는 프로그래머가 아니라 **AI에게 만들 것을 정확하게 시킬 줄 아는 사람**입니다. 다른 직업입니다.

④ "말만 하면 다 되는" 마법은 없습니다

애매하게 시키면 애매한 결과가 나옵니다. 그래서 2기 1주차는 코드를 한 줄도 쓰지 않고 **무엇을 만들지 · 무엇을 안 만들지 · 어떻게 생겼는지** 정하는 데 두 시간을 통째로 씁니다.

⑤ AI는 틀립니다. 자주 틀립니다

없는 기능을 있다고 하고, 숫자를 그럴듯하게 지어냅니다. 왜 그런지는 3장에서 봅니다.

⑥ 돈이 듭니다

항목	금액	
Claude Pro 구독	월 $20부터	필수
구글 플레이 개발자 등록	**$25 (평생 1회)**	필수
GitHub · Vercel · 공공데이터포털	무료	—

유료 도메인(`내가게.com` 같은 주소)은 쓰지 않습니다. 무료로 주는 `oo.vercel.app` 주소로 진행합니다.

0.5 이 책에서 다루지 않는 것

분량 때문에 의도적으로 뺀 것들입니다.

안 다루는 것	왜 뺐나	어디로
아이폰(iOS) 앱	비용과 심사 난이도	—
결제 · 문자 발송 · 지도 · 1:1 채팅	별도 가입·심사·요금이 붙습니다	1주차에서 이유를 봅니다
현업 개발자용 설계·아키텍처	이 책의 목표가 아닙니다	—
유료 도메인 연결	무료 주소로 충분합니다	—

> 결제·문자·지도를 뺀 것은 "어려워서" 가 아닙니다. **만들 수는 있지만, 오늘 당장 쓸 수 있게 되기까지 사람이 거쳐야 할 절차가 너무 많기 때문입니다.** 1주차의 판단 기준 한 줄 — **사람 · 시간 · 신청 · 명단** 네 가지로 설명되면 만들 수 있습니다.

0.6 이 책을 읽는 법

구성

부	무엇	강의
0부	준비 — 설치하고 첫 대화까지	Week 0
1부	Claude Code라는 도구	Week 1

순서대로 읽으세요

앞 장에서 만든 것을 뒷 장에서 계속 씁니다. 특히 2기는 **1주차에 정한 것으로 2주차에 만들고, 3주차에 올립니다.** 한 주를 건너뛰면 다음 주에 올릴 물건이 없습니다.

본문에 나오는 상자

표시	뜻	읽는 법
📖	**용어**	새 낱말이 나올 때. 두세 줄
💡	**왜 그런가**	이 책의 핵심. 실습 옆에 붙어서 이유를 설명합니다
⚠️	**여기서 막힙니다**	실제로 자주 걸리는 지점입니다
🔄	**혼자 해보기**	답을 안 알려줍니다. 여기서 진짜 실력이 붙습니다

0.7 시작하기 전 준비물

	무엇	확인
1	**노트북** — Windows 또는 Mac	사양이 낮아도 됩니다. 무거운 작업은 클라우드가 합니다
2	**안드로이드 폰**	없으면 아이폰·PC로 대부분 됩니다(1장)
3	**Claude Pro 이상 구독**	무료로는 실습 중간에 한도에 막힙니다
4	⭐ **구글 플레이 개발자 계정**	**지금 신청하세요.** 1장에서 이유를 설명합니다
5	**만들고 싶은 것 하나**	가게 · 모임 · 수업 · 신청받는 일 무엇이든

4번은 이 책에서 **유일하게 "미리" 해야 하는 일**입니다. 신원 확인에 보통 하루, 서류가 어긋나면 **2주**까지 걸립니다. 3주차 당일에 계정이 없으면 그 회차를 통째로 놓칩니다.

5번은 지금 정하지 않으셔도 됩니다. 1주차에 Claude가 **물어봐 주면서** 같이 정합니다.

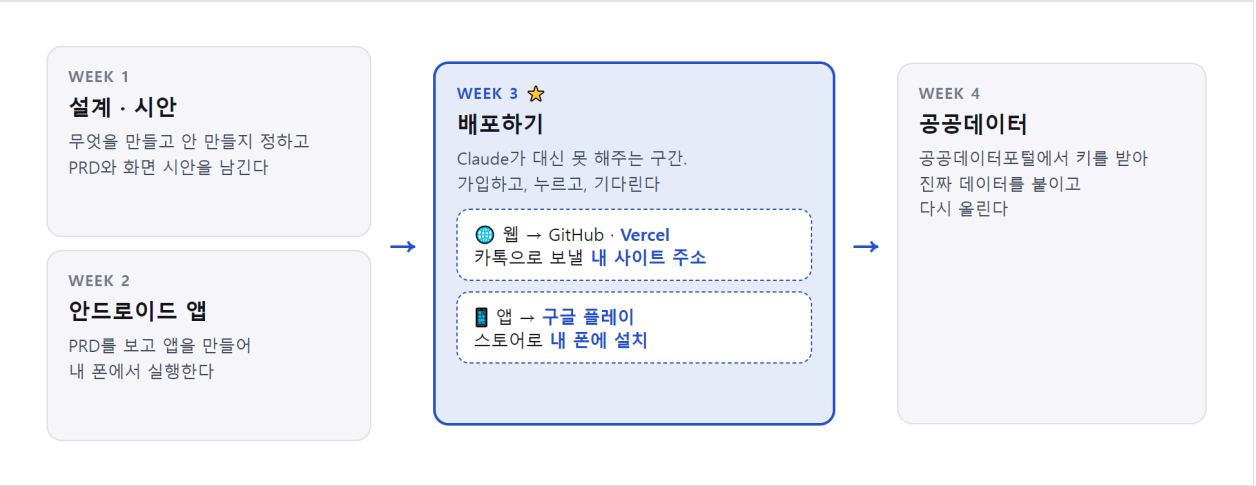

그림 0-1 4주의 흐름. 1주차 설계·시안과 2주차 앱이 3주차에서 만나 웹은 Vercel로, 앱은 구글 플레이로 나간다. 4주차에는 공공데이터를 붙여 다시 올린다

0.8 마지막으로

2기 강의 사이트 FAQ에 이런 질문이 있습니다.

> **"코딩을 전혀 몰라도 따라갈 수 있나요?"**
>
> 네. 코드는 Claude가 씁니다. 여러분은 무엇을 만들지 정하고, 마음에 안 드는 부분을 말로 고칩니다.

이 답은 사실입니다. 다만 한 가지가 빠져 있습니다. **"무엇을 만들지 정하는 것"이 생각보다 어렵습니다.** 그리고 **"말로 고치는 것"에도 요령이 있습니다.** 이 책의 대부분은 그 두 가지에 쓰입니다.

준비되셨으면, 1장에서 설치부터 시작합니다.

01장 — 설치, 한 시간이면 끝난다

> 초고 v0.1 (2026-09-14)

1.1 이 장을 시작하기 전에

솔직히 말씀드리면, **이 장이 이 책에서 제일 재미없습니다.** 만드는 것도 아니고 배우는 것도 아닌, 그냥 설치니까요.

그래서 2기 강의에서는 설치를 **혼자 하지 않게** 했습니다. 개강 전에 온라인으로 한 시간 모여서 **같이** 합니다. 이걸 Week 0이라고 부릅니다.

> 혼자 설치하다 막히면 하루가 날아가지만, 같이 하면 한 시간이면 끝납니다.

이 장은 그 한 시간을 글로 옮긴 것입니다. 그리고 다른 설치 안내와 한 가지가 다릅니다. **깔라고만 하지 않고, 그게 왜 필요한지 먼저 말합니다.** 이유를 모르고 깔면 나중에 뭔가 고장 났을 때 어디를 봐야 할지 알 수 없기 때문입니다.

단, 딱 하나는 같이 할 수 없습니다. 구글 플레이 개발자 계정입니다. 이건 이 장을 읽는 **지금** 시작하셔야 합니다. 1.3절에서 이유를 봅니다.

1.2 무엇을 준비하나 — 전체 그림

준비할 것은 **설치 네 가지, 계정 다섯 개, 폰 하나**입니다. 한꺼번에 보면 많아 보이지만, **언제 쓰는지**를 같이 보면 왜 필요한지 보입니다.

	무엇	하는 일	언제 쓰나
설치	**VS Code + Claude Code 확장**	작업실 + AI 본체	1주차부터 매번
설치	**Node.js**	앱을 만들 때 필요한 실행기	2주차
설치	**Python**	Claude가 쓴 작업용 코드를 돌리는 실행기	1주차부터
설치	**Git**	파일이 바뀌기 전 상태를 저장하는 도구	3주차
계정	**Claude Pro 구독**	AI를 넉넉히 쓰기 위한 요금제	1주차부터 매번
계정	⭐ **구글 플레이 개발자**	앱을 스토어에 올리는 자격	3주차
계정	**GitHub**	만든 코드를 인터넷에 보관	3주차
계정	**Vercel**	보관한 코드를 가져가 **내 사이트 주소**로 띄움	3주차
계정	**공공데이터포털**	공공 데이터를 받아 쓰는 자격	4주차
장비	**안드로이드 폰**	만든 앱을 직접 돌려보기	2·3주차

표를 보면 한 가지가 눈에 띕니다. **개발자 계정은 3주차에 쓰는데, 제일 먼저 하라고 합니다.** 순서가 이상해 보이지만 이유가 있습니다.

1.3 ⭐ 가장 먼저 — 구글 플레이 개발자 계정

왜 제일 먼저인가

다른 것은 전부 **그 자리에서** 끝납니다. 가입 버튼을 누르면 바로 됩니다. 개발자 계정만 **기다려야** 합니다.

> 📖 **개발자 계정(Google Play Console)**
>
> 구글 플레이스토어에 앱을 올릴 수 있는 **자격증.** 등록비 $25(평생 1회)를 내고, **구글이 내가 누구인지 확인한 뒤에야** 앱을 올릴 수 있습니다. 앱을 올리고 관리하는 화면의 이름이 **Play Console(플레이 콘솔)** 입니다.

구글이 신분을 확인하는 데 **보통 하루, 서류가 어긋나면 2주**까지 걸립니다. 날짜를 거꾸로 세어 보면 이렇습니다.

```
9/28 (3주차)  스토어에 올리는 날. 계정이 반드시 있어야 한다
9/14 (개강)   이때는 이미 신청이 접수돼 있어야 안전하다
Week 0        신청 상태 점검. 반려된 사람은 여기서 같이 고친다
지금          신청 ⭐
```

3주차 당일에 계정이 없으면 **그 회차 실습을 통째로 놓칩니다.** Week 0은 진행 상황을 **점검하는** 자리지, 시작하는 자리가 아닙니다.

결제로 끝이 아닙니다

가장 흔한 오해입니다. $25를 내면 계정은 만들어집니다. 그런데 **세 가지가 더 남습니다.**

```
$25 결제 → 계정 생성
              ↓
    ① 본인 확인 (신분증 올리기)   ← 여기가 며칠 걸린다
    ② 안드로이드 기기 확인        ← 1~2분
    ③ 전화번호 인증               ← ①이 끝나야 열린다
              ↓
          앱을 올릴 수 있다
```

셋이 다 끝나야 합니다. 그리고 **③은 ①이 승인돼야 열립니다.** 순서가 걸려 있어서, ①을 미루면 뒤가 전부 밀립니다.

등록 순서

시작 주소는 https://play.google.com/console/signup 입니다.

> 단계별 화면은 「**Week 0 · 같이 설치하기**」 페이지에 모두 있습니다. 같이 펼쳐 놓고 따라 하세요. **https://monthly-claude-2-class.vercel.app/week-0**

> ⚠ **어떤 구글 계정으로 로그인했는지 먼저 보세요**
>
> 지금 로그인된 구글 계정이 이 개발자 계정의 **주인**이 됩니다. 만든 뒤에는 바꿀 수 없습니다. 앞으로 계속 쓸 계정인지 확인하고 시작하세요.

STEP 1 — 계정 유형: 「개인」을 고릅니다

두 갈래가 나옵니다. **한 번 정하면 못 바꿉니다.**

	개인	기관/단체
필요 서류	신분증	사업자등록증 + **D-U-N-S 번호**
확인 기간	보통 하루~며칠	번호 발급 + 조직 인증, 두 번
정식 공개 조건	테스터 12명 × 14일	면제

사업자등록증이 있어도 **이번에는 개인을 권합니다.** 기관 계정에는 D-U-N-S라는 9자리 사업체 번호가 필요한데, 국내 소규모 사업자는 대부분 갖고 있지 않습니다. 발급에만 며칠~2주가 걸리고, 그 뒤에 구글이 조직 확인을 또 합니다. 개강까지 빠듯합니다.

기관 계정의 이점은 "테스터 12명" 조건이 면제되는 것 하나인데, 이건 **기수끼리 서로 테스터가 되어주면** 됩니다.

그림 1-1 Play Console 계정 유형 선택 화면. 위쪽 「조직 유형 선택」은 건드리지 않고, 아래쪽 「개인 → 시작하기」를 누른다

STEP 2 — 개발자 이름

플레이스토어에서 앱 아래에 표시되는 이름입니다. 본명이 아니어도 됩니다. 가게 이름이나 활동명도 괜찮습니다.

STEP 3 — 결제 프로필: 국가를 반드시 확인

결제 프로필의 **국가는 나중에 못 바꿉니다.** 세금 처리와 직결되기 때문입니다. 한국에 계시면 **대한민국** 프로필을 고르세요. 예전에 다른 구글 서비스를 쓰다가 미국 프로필이 만들어져 있는 경우가 있으니 목록에서 국가를 보고 고르세요.

주소는 **신분증에 적힌 주소와 똑같이** 적습니다. 어긋나면 본인 확인에서 반려됩니다.

STEP 4 — 연락처: 공개되는 것과 아닌 것

칸	공개?
개발자 프로필 이메일	**플레이스토어에 공개됩니다**
Google 연락처	공개 안 됨. 구글이 연락할 때만 씁니다

전화번호는 **국제 형식**으로 넣습니다.

```
010-1234-5678   →   +82 10-1234-5678
                        ↑ 맨 앞의 0을 뺀다
```

STEP 5 — 약관 동의 후 $25 결제

한 번만 내는 돈이고 갱신은 없습니다. 해외결제가 되는 카드가 필요합니다.

> ⚠ **본인 확인이 안 되면 환불되지 않습니다**
>
> 약관에 적혀 있습니다. 결제 전에 신분증(주민등록증 · 운전면허증 · 여권)을 준비하고, 결제 프로필의 이름·주소가 신분증과 같은지 한 번 더 보세요.

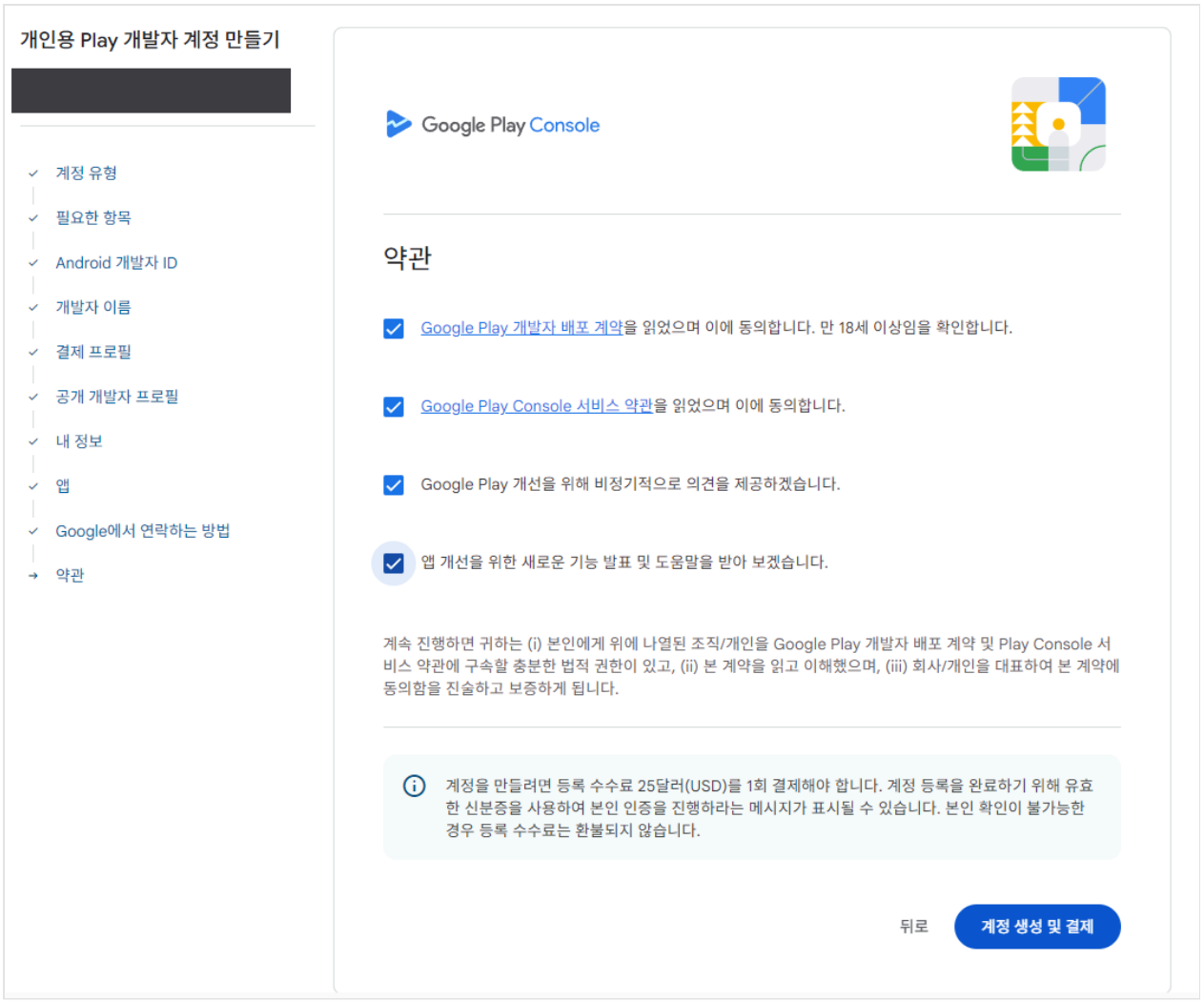

그림 1-2 약관 동의 화면. 위 두 항목(개발자 배포 계약 · Console 서비스 약관)이 필수이고, 아래 안내에 "본인 확인 불가 시 환불되지 않음" 이 적혀 있다

STEP 6 ☆ — 본인 확인 (여기가 병목)

Play Console 홈에 「**개발자 계정 설정 완료**」 카드가 뜹니다. 본인 확인 → 시작하기를 눌러 신분증을 올립니다. 올리면 "Google에서 신원 확인 중입니다" 로 바뀌고, 끝나면 메일이 옵니다.

빠르면 당일에 승인됩니다. 강사는 신청한 날 바로 됐습니다. 다만 서류가 어긋나면 반려되고 며칠이 더 가므로 **미루지 마세요.**

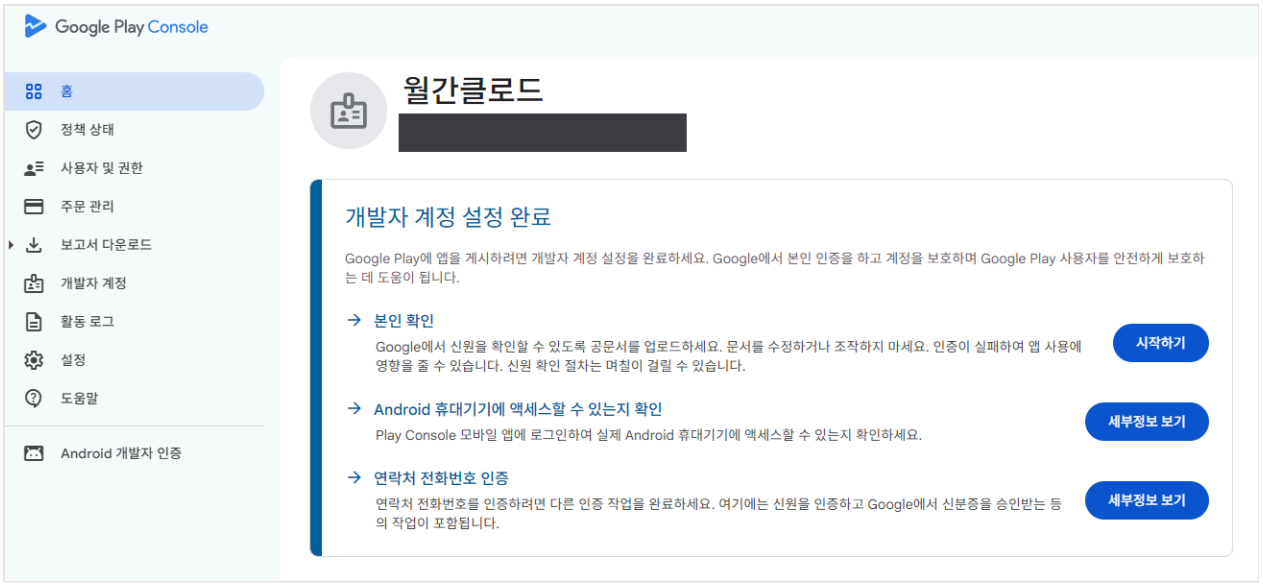

그림 1-3 결제 직후의 Play Console 홈. 결제가 끝나도 본인 확인 · 기기 확인 · 전화번호 인증 세 가지가 남아 있다

STEP 7 — 기기 확인 + 전화번호 인증

- **기기 확인** — 안드로이드 폰에서 Play 스토어 →「Google Play Console」앱 설치 → **같은 구글 계정**으로 로그인. 화면의 QR 코드를 폰으로 찍으면 바로 넘어갑니다. 본인 확인과 상관없이 **지금 바로** 할 수 있습니다
- **전화번호 인증** — 본인 확인이 승인된 **뒤에** 열립니다. 먼저 누르면 "다른 인증 작업을 완료하세요" 만 나옵니다

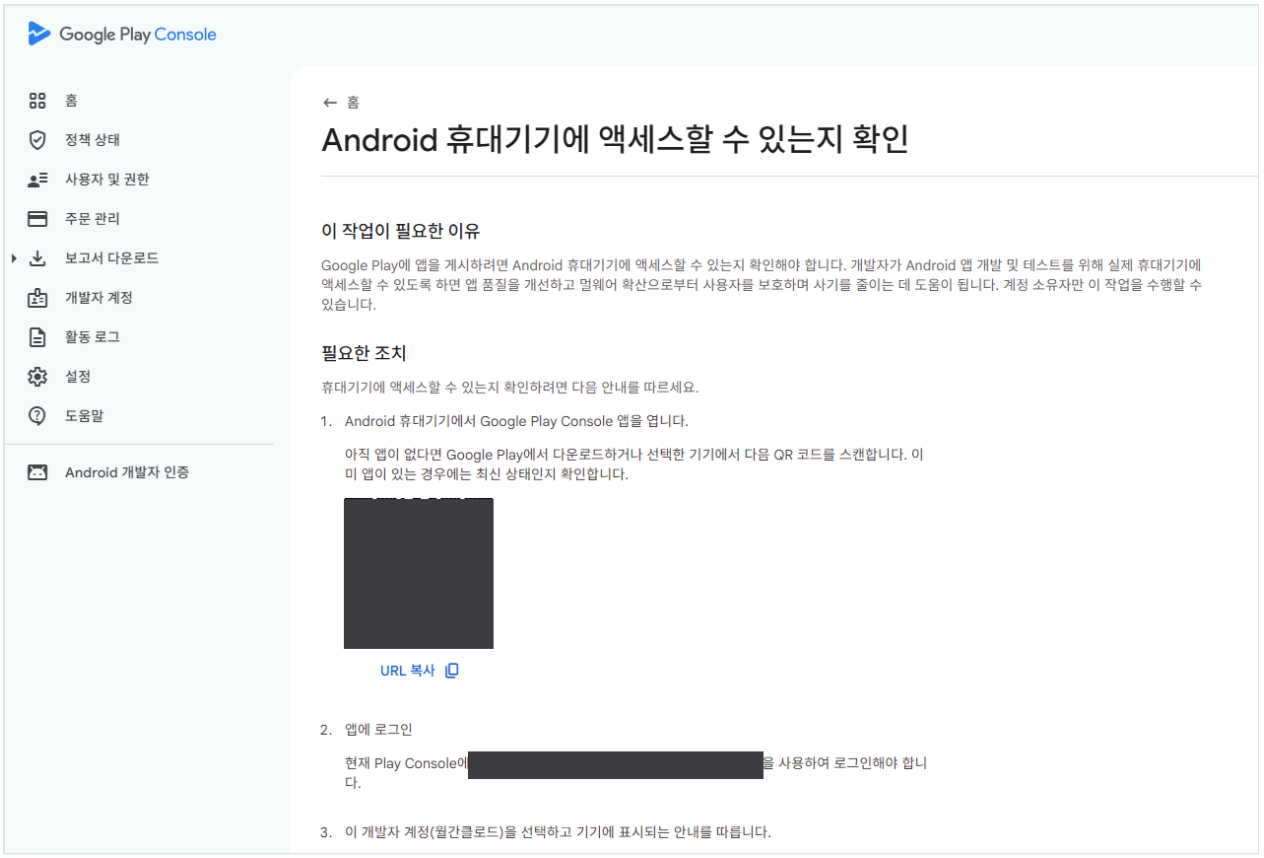

그림 1-4 Android 기기 확인 화면. 화면의 QR 코드를 폰으로 찍어 Play Console 앱을 설치하고, 개발자 계정과 같은 구글 계정으로 로그인한다

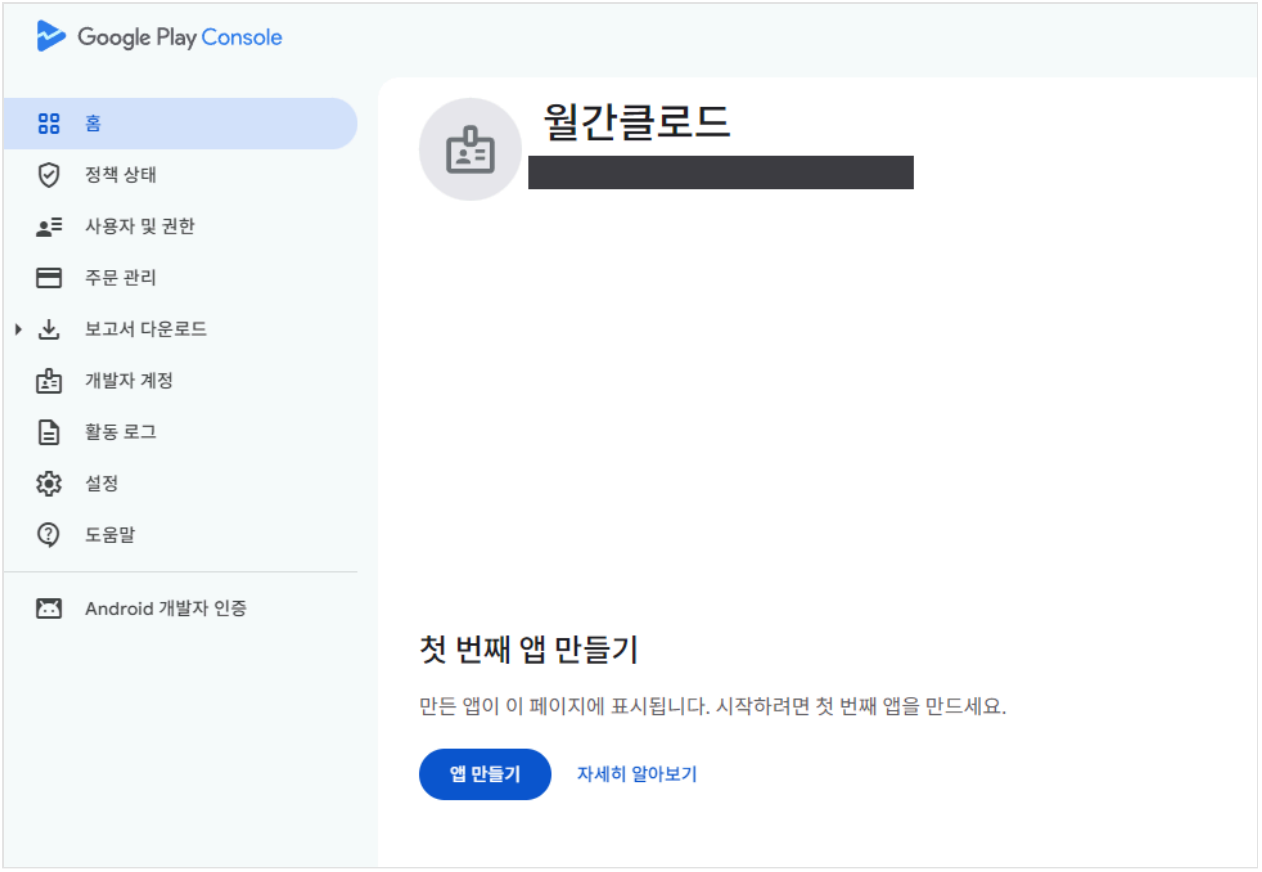

그림 1-5 세 가지가 다 끝난 Play Console 홈. 경고 배너가 사라지고 「앱 만들기」 버튼이 열린다 — 여기까지가 개강 전 목표

세 가지가 다 끝나면 상단의 파란 경고 배너가 사라지고 **「앱 만들기」 버튼이 열립니다.**
여기까지가 개강 전 목표입니다. 실제 앱 등록은 3주차에 같이 합니다.

여기서 막힙니다

증상	원인과 대처
인증 문자가 안 온다	`+82 010…` 처럼 **0을 남긴 경우**입니다. `+82 10…` 으로 다시
D-U-N-S 번호를 요구한다	**기관/단체**를 골랐습니다. 뒤로 가서 **개인**으로
본인 확인이 반려됐다	신분증과 결제 프로필의 **이름·주소가 다른** 경우가 대부분입니다
결제했는데 앱을 못 올린다	세 가지(본인·기기·전화번호)가 **다** 끝나야 합니다

1.4 VS Code (작업실)

이제 같이 하는 설치입니다.

VS Code(브이에스 코드)는 Microsoft가 만든 무료 프로그램입니다. 메모장이라고 생각하시면 절반은 맞습니다. 글자를 쓰고 저장하는 프로그램이니까요. 다만 **여기에 다른 프로그램을 붙일 수 있다**는 점이 다릅니다. Claude를 붙일 자리가 여기입니다.

Windows

1. https://code.visualstudio.com/ 접속
2. **Download for Windows** → 받은 `.exe` 실행
3. 설치 옵션은 **전부 기본값**으로 두고 Next

Mac

1. 같은 주소에서 **Download for Mac** (안 보이면 버튼 아래 **other platforms** → macOS)
2. 받은 `.zip` 을 풀고 `Visual Studio Code` 앱을 **응용 프로그램 폴더로 드래그**
3. 처음 실행할 때 "확인되지 않은 개발자" 경고가 뜨면 — **시스템 설정 → 개인정보 보호 및 보안**에서 **"확인 후 열기"**

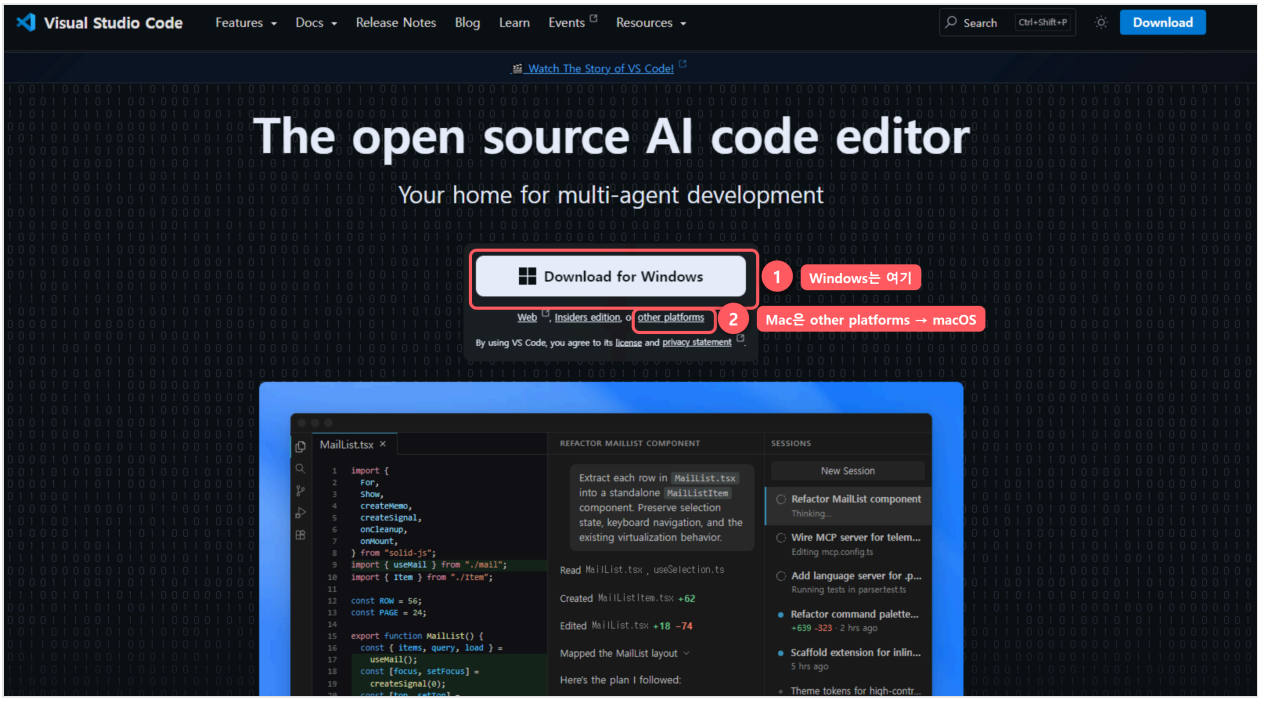

그림 1-6 VS Code 다운로드 페이지(code.visualstudio.com). ① Windows는 가운데「Download for Windows」, ② Mac은 그 아래「other platforms」를 눌러 macOS용을 받는다. 접속한 컴퓨터에 따라 ①의 글자가「Download for Mac」으로 바뀌어 보이기도 한다

1.5 Claude Code 확장 (AI 본체)

드디어 주인공입니다.

> 📖 **확장(extension, 익스텐션)**
>
> VS Code에 **기능을 하나 더 붙이는 부품.** 스마트폰에 앱을 하나 더 까는 것과 같습니다.
> 우리가 붙일 확장의 이름이 **Claude Code** 입니다.

설치

Windows와 Mac이 같습니다.

1. VS Code 실행
2. 왼쪽 세로 막대에서 **네모 블록 모양 아이콘**(확장) 클릭
   - 단축키: Windows `Ctrl + Shift + X` / Mac `Cmd + Shift + X`
3. 검색창에 `claude` 입력
4. **Claude Code for VS Code** — 만든 곳이 **Anthropic**(파란 인증 표시)인지 확인하고 **Install**
   - ⚠ 비슷한 이름의 다른 확장이 함께 나옵니다. **만든 곳을 꼭 보세요**
5. 왼쪽 세로 막대에 **Claude 아이콘**이 새로 생기면 완료

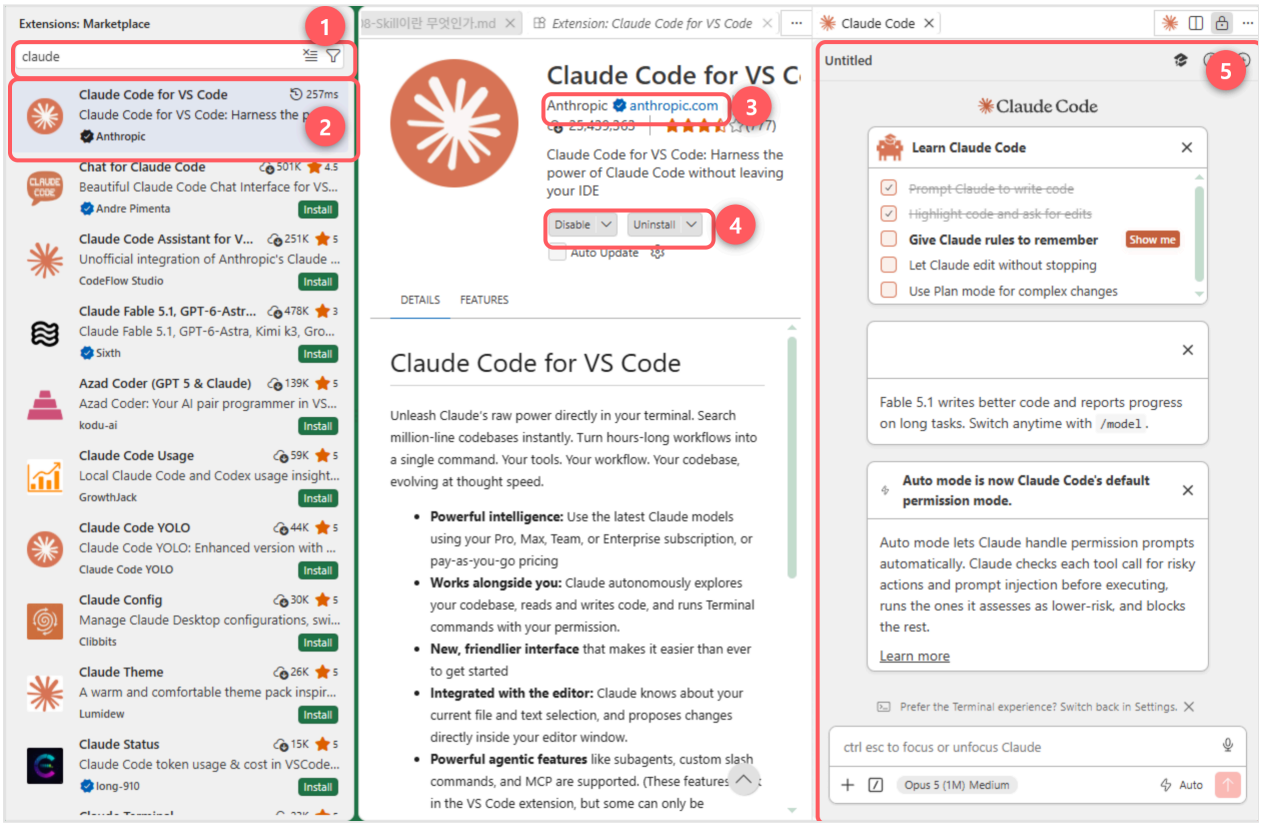

그림 1-7 VS Code 확장 화면. ① 검색창에 claude 입력 ② 목록 맨 위「Claude Code for VS Code」— 만든 곳이 Anthropic ③ 파란 인증 표시와 anthropic.com 확인 ④ 설치 전에는 Install, 설치가 끝나면 이 자리가 Disable · Uninstall 로 바뀐다 ⑤ 설치 후 열리는 Claude Code 패널. 아래로 비슷한 이름의 다른 확장이 줄줄이 있다

로그인과 구독

1. 왼쪽의 **Claude 아이콘** 클릭 → 패널이 열립니다
2. **어떻게 로그인할지 먼저 묻습니다.** 셋 중 맨 위 **Claude.ai Subscription** 을 고르세요
3. **브라우저가 자동으로 열립니다** → claude.ai 계정으로 로그인
4. 인증이 끝나면 자동으로 VS Code로 돌아옵니다

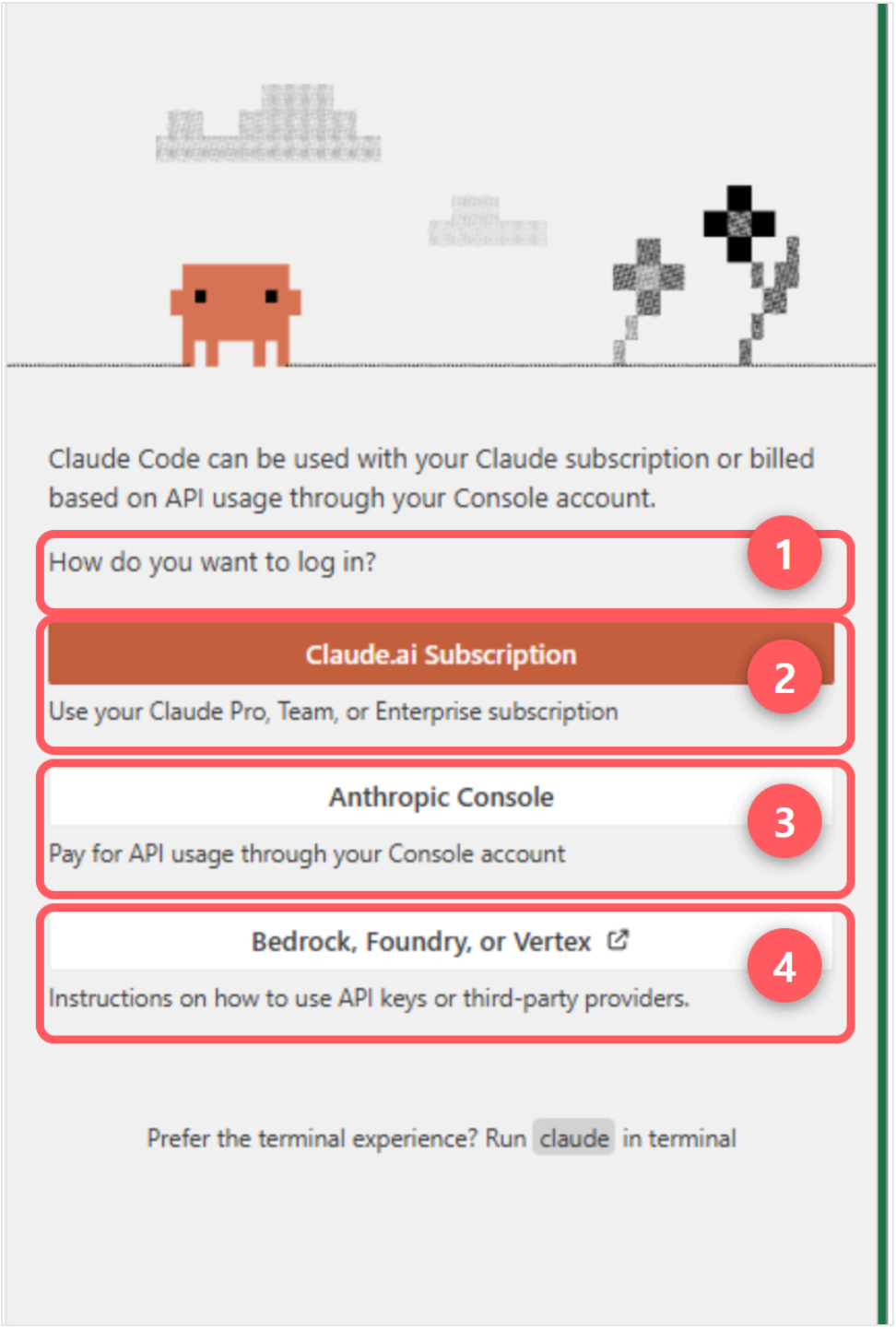

그림 1-8 확장을 깔고 Claude 패널을 처음 열면 나오는 화면. ①「어떻게 로그인할까요?」② Claude.ai Subscription — Pro 구독으로 쓰는 길, **우리가 고를 것** ③ Anthropic Console — 쓴 만큼 요금을 내는 길 ④ Bedrock · Foundry · Vertex — 회사에서 다른 서비스를 거쳐 쓰는 길

> ⚠ **세 갈래 중에 어디로?** > > 이 책은 **Claude.ai Subscription**(맨 위) 하나만 씁니다. Pro 구독을 결제하고 > 그 계정으로 로그인하는 길입니다. > > 아래 둘은 **쓴 만큼 요금이 매겨지는** 길입니다(4주차에 나오는 API 방식). > 잘못 고르면 결제 수단을 따로 등록하라고 하거나, 구독이 있는데도 요금이 나갑니다. > **맨 위를 고르세요.**

Pro 이상 구독(월 $20)이 필요합니다. 무료로도 로그인은 됩니다. 다만 실습 분량이 많아서 **중간에 사용량 한도에 막힙니다.** 한창 화면을 고르다가 멈추는 게 제일 답답합니다. claude.ai 에서 미리 결제하고, VS Code에서도 **같은 계정**으로 로그인하세요.

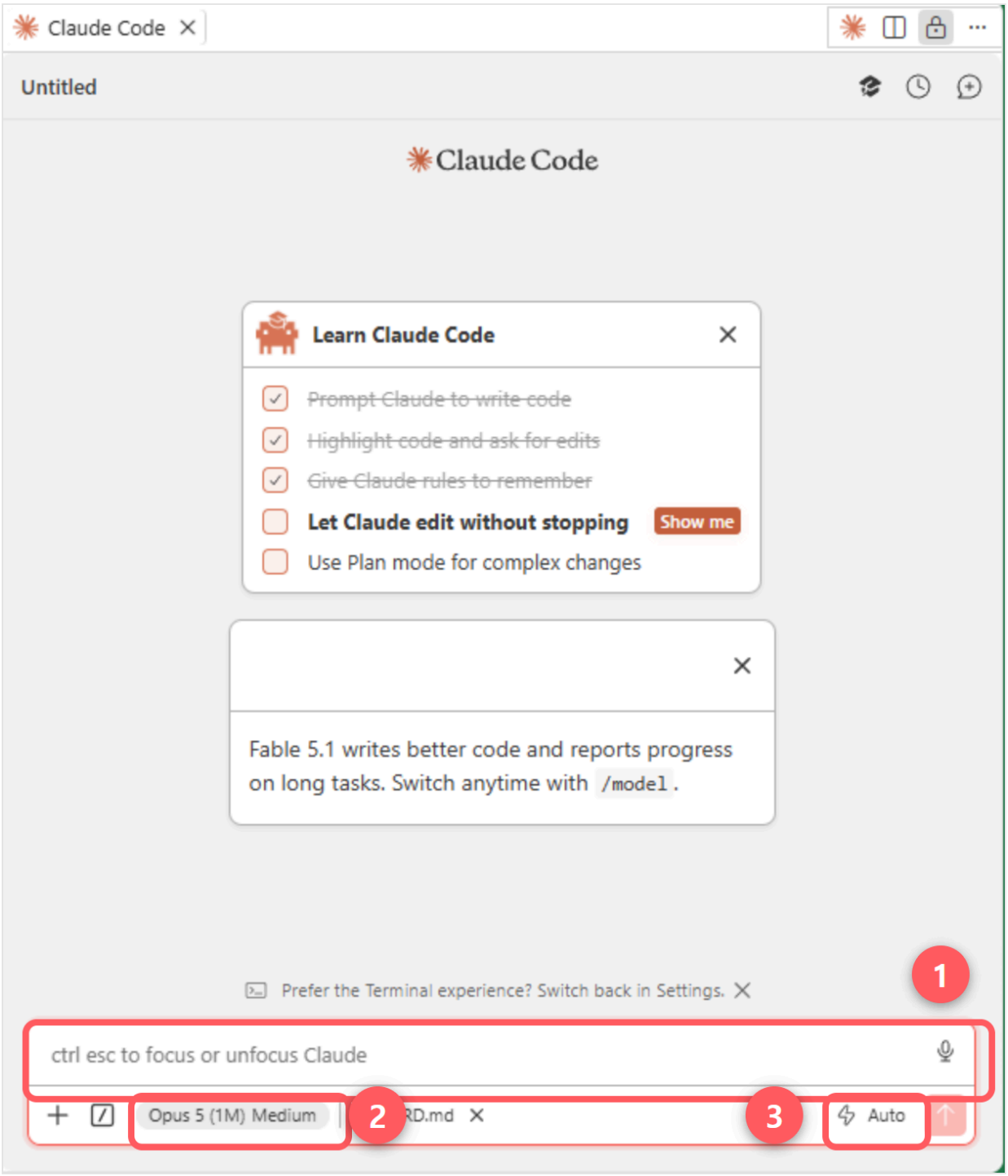

그림 1-9 로그인이 끝난 Claude 패널. ① 입력칸 ② 쓰는 모델 ③ 권한 모드. 로그인 방식을 고르는 화면 대신 이 입력칸이 보이면 로그인된 것

1.6 Node.js (앱을 만드는 실행기)

왜 필요한가

1주차에는 안 씁니다. **2주차에 앱을 만들 때** 씁니다.

Claude에게 "앱 만들어줘" 라고 하면, Claude는 앱의 코드를 씁니다. 그런데 코드는 글자일 뿐입니다. 누군가 그 글자를 읽고 실제로 **돌려야** 합니다. 앱을 만드는 도구들이 바로 이 Node.js 위에서 돌아갑니다.

> 📖 **Node.js(노드제이에스)**
>
> **자바스크립트**(웹과 앱에서 가장 많이 쓰는 프로그래밍 언어)로 쓴 코드를 내 컴퓨터에서 **실행해 주는 프로그램.** 2주차에 쓰는 앱 제작 도구가 이 위에서 돌아갑니다. 없으면 도구 자체가 켜지지 않습니다.

자바스크립트를 배울 필요는 없습니다. 코드는 Claude가 씁니다. 우리는 그 코드를 돌려줄 실행기만 깔아두면 됩니다.

설치

1. https://nodejs.org/ 접속 → 내려받기
2. 받은 설치 파일 실행 → **기본값**으로 진행

> ⚠️ **Windows — PATH 항목을 확인하세요**
>
> 설치 중간에 **Custom Setup** 화면이 나오고, 목록 맨 아래에 **Add to PATH** 가 있습니다. 앞의 아이콘이 다른 항목들과 같은 **디스크 모양**이면 설치됩니다(기본값). 아이콘에 **빨간 X** 가 붙어 있다면 눌러서 설치하도록 바꾸세요. 그대로 **Next** 를 누르면 됩니다.
>
> 이게 무슨 뜻이냐면 — 컴퓨터에게 **"Node.js를 여기 깔았으니, 누가 찾으면 알려줘"** 라고 등록하는 것입니다. 꺼져 있으면 Node.js는 깔려 있는데 다른 프로그램이 못 찾습니다. "분명 깔았는데 없다고 나와요" 의 정체가 이겁니다.

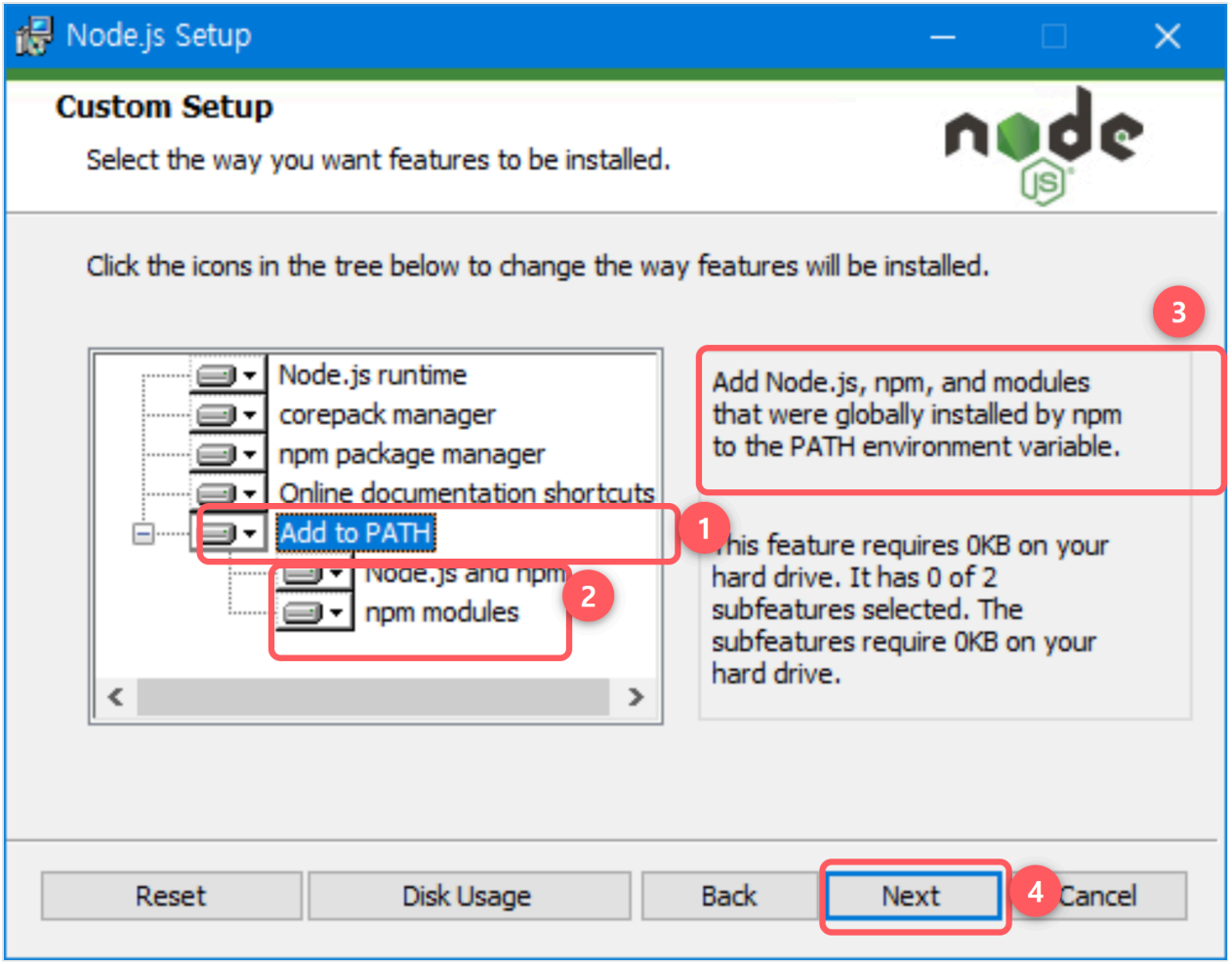

그림 1-10 Windows Node.js 설치 중 Custom Setup 화면. ① Add to PATH — 이 항목이 켜져 있어야 다른 프로그램이 Node.js 를 찾는다 ② 그 아래 딸린 두 항목(Node.js and npm · npm modules) ③ 오른쪽 설명 — "PATH 환경 변수에 추가한다" ④ 아이콘을 확인했으면 Next

잘 깔렸는지 확인

확인은 **터미널**이라는 창에서 합니다. 마우스 대신 **글자를 쳐서 컴퓨터에게 명령하는 창**입니다.
지금은 딱 한 줄만 칩니다.

- **Windows** — 시작 메뉴에서 `cmd` 검색 → 명령 프롬프트 실행
- **Mac** — `Cmd + Space` → `터미널` 검색 → 실행

아래를 치고 Enter를 누르세요.

```
node --version
```

`v` 로 시작하는 번호(예: `v22.x.x`)가 나오면 됐습니다. "찾을 수 없습니다" 가 뜨면 1.12절로 가세요.

1.7 Python (작업용 코드를 돌리는 실행기)

왜 필요한가

Claude에게 "이 폴더 사진 크기 줄여줘", "엑셀을 표로 바꿔줘" 같은 일을 시키면, Claude는 그 일을 하는 **짧은 코드를 써서 돌립니다.** 그 코드가 대부분 **Python**(파이썬)입니다. Node.js가 앱 도구를 돌린다면, Python은 이런 **작업용 코드**를 돌립니다.

Python 문법은 배우지 않습니다. 실행기만 깔아 둡니다.

설치

설치할 버전은 **3.11.9** 입니다. 최신 버전이 아닙니다.

1. https://www.python.org/downloads/release/python-3119/ 접속
2. 페이지 **맨 아래 Files 표**로 내려갑니다

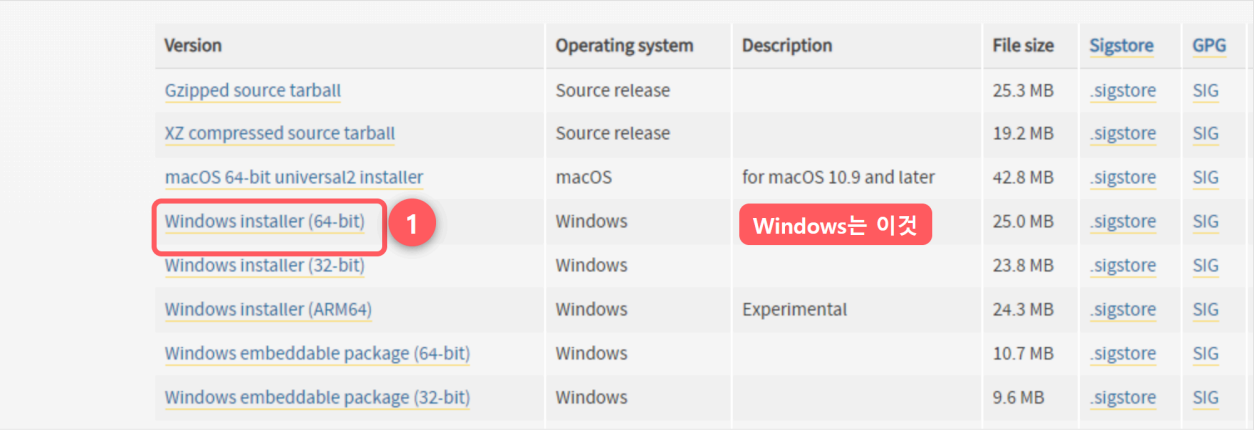

그림 1-11 Python 3.11.9 내려받기 페이지 맨 아래의 Files 표. ① Windows는「Windows installer (64-bit)」. Mac은 그 세 줄 위「macOS 64-bit universal2 installer」

Windows

3. **Windows installer (64-bit)** 를 받아 실행합니다

> ⚠ **첫 화면 아래쪽 체크박스를 반드시 켜세요**
>
> 설치 첫 화면 **아래쪽**에 `Add python.exe to PATH` 체크박스가 있습니다. 기본값이 **꺼짐**이라 그냥 **Install Now** 를 누르면 놓칩니다.
>
> 켜지 않으면 Python은 깔렸는데 Claude가 **찾지 못합니다.** Node.js의 Add to PATH와 같은 이야기입니다. 켰으면 **Install Now**.

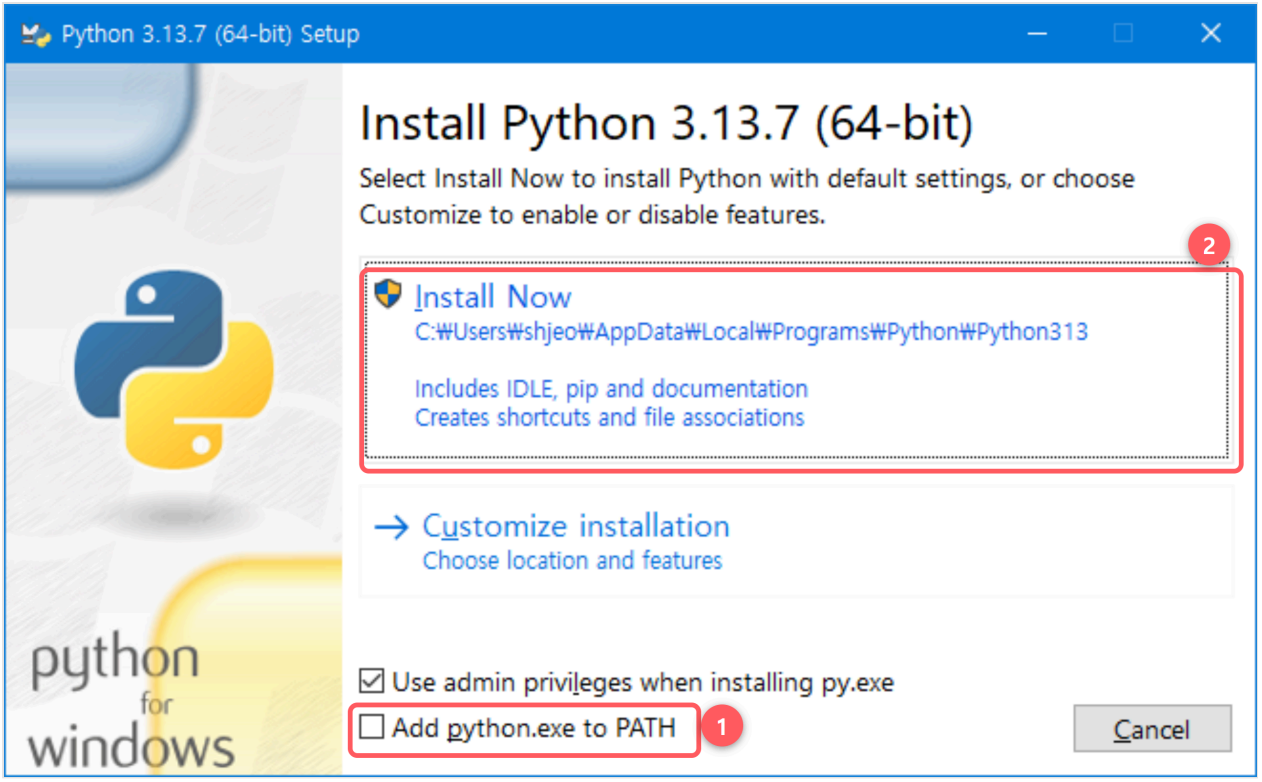

그림 1-12 Python 설치 첫 화면. ① 맨 아래「Add python.exe to PATH」— 기본값은 꺼져 있다. 먼저 체크한다 ② 그다음 Install Now

Mac

3. **macOS 64-bit universal2 installer** 를 받아 실행 → **전부 기본값**으로 진행

잘 깔렸는지 확인

터미널에서 아래를 칩니다. Windows는 `python`, Mac은 `python3` 입니다.

```
python --version
```

`Python 3.11.9` 가 나오면 됐습니다.

1.8 Git (바뀌기 전 상태를 저장하는 도구)

왜 필요한가

2장에서 "되돌리기" 를 배울 때, 마음에 드는 상태에서 **사본을 떠 두라**고 합니다. 그 일을 자동으로, 매번 해주는 도구가 **Git**(깃)입니다.

3주차에 만든 사이트를 GitHub에 올릴 때 이 Git이 쓰입니다. Windows에서는 Claude Code도 Git을 씁니다.

설치

Windows

1. https://git-scm.com/install/windows 접속
2. **Git for Windows/x64 Setup** 을 받아 실행

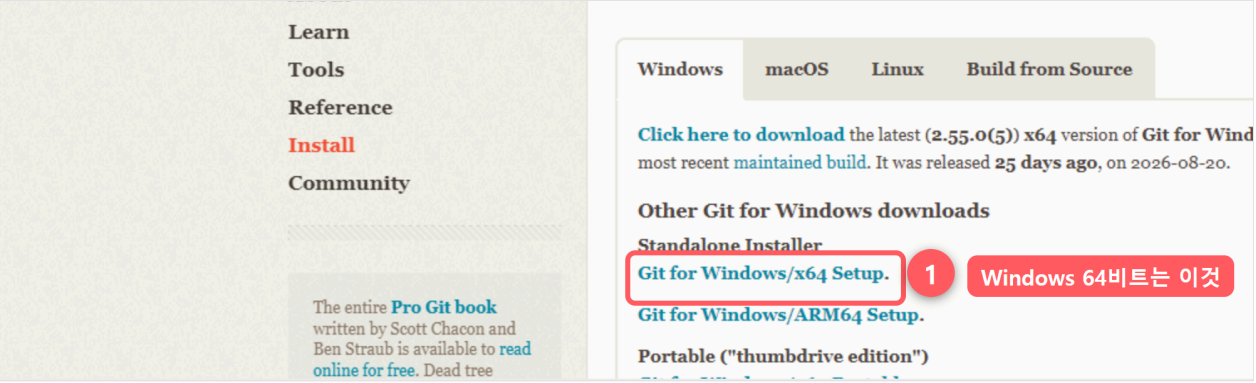

그림 1-13 Git 내려받기 페이지. ①「Git for Windows/x64 Setup」

3. 설치 화면이 **여러 장** 나옵니다. **전부 기본값으로 Next** 를 누르세요

Mac

Mac에는 Git이 대부분 들어 있습니다. 터미널에서 `git --version` 을 쳐서 `git version 2.x.x` 가 나오면 넘어갑니다. 없으면 설치하라는 창이 자동으로 뜹니다.

잘 깔렸는지 확인

```
git --version
```

`git version` 으로 시작하는 줄이 나오면 됐습니다.

1.9 계정 세 개 더 — 가입만 해두기

지금은 **가입만** 합니다. 쓰는 법은 그 주차에 배웁니다.

계정	주소	무엇에 쓰나
GitHub(깃허브)	https://github.com	3주차. 만든 코드를 인터넷에 보관하는 창고
Vercel(버셀)	https://vercel.com	3주차. GitHub 창고의 코드를 가져가 `oo.vercel.app` 주소를 붙여 띄워 줍니다
공공데이터포털	https://www.data.go.kr	4주차. 날씨·미세먼지 같은 공공 데이터를 받아 내 앱에 붙입니다

셋 다 무료입니다. Vercel은 **GitHub 계정으로 가입**하면 3주차에 둘을 잇기가 수월합니다.
공공데이터포털은 3주차 과제로 **쓸 데이터를 정해 신청**까지 넣습니다. 승인에 하루 이틀 걸리는 데이터가 있어서, 4주차 당일에 신청하면 늦습니다. **여기서도 "기다리는 것" 이 순서를 정합니다.**

1.10 폰과 컴퓨터 — 장비 질문

강의 안내 페이지의「장비 관련 질문」을 옮겼습니다.

질문	답
맥을 씁니다	전혀 문제없습니다. 모든 도구가 똑같이 돌아가고, 앱 파일은 클라우드가 만들어 줍니다
안드로이드 폰이 없습니다	끝까지 완주할 수 있습니다. 실물이 필요한 곳은 **3주차 마지막 "스토어로 내 폰에 설치"** 한 곳뿐입니다. 지인 폰을 잠깐 빌리면 1분이면 끝납니다
아이폰만 있습니다	2주차는 그대로 됩니다. 3주차도 스토어에 올리는 것까지는 같습니다. **아이폰에 설치하는 것만** 안 됩니다(플레이스토어가 없으므로)
컴퓨터 사양이 낮습니다	괜찮습니다. 무거운 작업(앱 파일 만들기)은 **클라우드가** 대신 합니다. 그래서 10GB가 넘는 Android Studio를 설치하지 않습니다

1.11 마지막 — 폴더 열고 말 걸어보기

설치는 끝났습니다. 이제 **실제로 되는지** 확인합니다.

작업 폴더 만들기

Claude Code는 **폴더 하나를 열어놓고** 그 안에서 일합니다. 작업 폴더는 전부 `workspace` 아래에 둡니다.

- **Windows**: `C:\workspace\`
- **Mac**: `~/workspace/`

그 안에 만들 것마다 폴더를 하나씩 만듭니다. 1주차는 강의 자료 폴더입니다.

```
C:\workspace\
└── month-claude-2-week1\
```

> ⚠ **폴더 위치를 아무 데나 잡지 마세요**
>
> | 피할 것 | 왜 |
> |---|---|
> | **한글이 들어간 경로** | `C:\내문서\작업` 같은 경로에서 도구가 오작동합니다 |
> | **공백이 들어간 경로** | 이름 중간의 빈칸을 경계로 착각하는 도구가 있습니다 |
> | **OneDrive·iCloud·구글드라이브 폴더** | 동기화가 파일을 계속 만지면서 충돌합니다 |
>
> `C:\workspace\` 처럼 **짧고, 영어고, 드라이브 바로 아래.** 이게 제일 안전합니다. 2주차에 앱 도구를 돌리기 시작하면 이 차이가 크게 납니다.

폴더 열기

VS Code에서 **File → Open Folder** (Mac은 **File → Open**) 로 `C:\workspace\month-claude-2-week1` 을 엽니다.

- 단축키: `Ctrl + K` 를 누르고 이어서 `Ctrl + O` (Mac은 `Cmd`)

"이 폴더의 작성자를 신뢰합니까?" 가 뜨면 **신뢰함(Yes, I trust)** 을 고르세요.

첫 마디

Claude 패널 아래쪽 입력창에 이렇게 치고 Enter를 누르세요.

```
이 폴더 뭐 있어?
```

Claude가 폴더 안의 파일을 하나씩 열어 보고, 무엇이 들어 있는지 알려 줍니다. 중요한 건 답의 내용이 아니라, **Claude가 내 컴퓨터의 폴더를 실제로 들여다봤다**는 사실입니다.

그림 1-14 폴더를 열고 처음 말을 건 화면. ① 왼쪽 탐색기 — 강의 자료 파일 세 개와 images 폴더 ②「이 폴더 뭐 있어?」③ Claude의 답 — 파일마다 무엇인지 표로 정리하고, 지침서.md 의 주요 규칙까지 읽어 알려 준다

1.12 안 될 때

증상으로 찾으세요.

① 확장 검색에 Claude Code가 안 나온다

VS Code 버전이 낮습니다. **Help → Check for Updates** 로 업데이트한 뒤 다시 검색하세요. 회사 PC라면 방화벽이 막고 있을 수도 있습니다.

② Sign in을 눌러도 브라우저가 안 열린다

10초쯤 기다려 보세요. 그래도 안 되면 기본 브라우저를 Chrome이나 Edge로 바꿔 보세요.

③ (Windows) `node` 를 찾을 수 없다고 나온다

1.6절의 **PATH 항목**이 꺼져 있었을 가능성이 큽니다. 이미 켜진 채로 깔았다면 **터미널 창을 닫고 새로 여세요.** 설치 전에 열어둔 창은 새로 깐 프로그램을 모릅니다. 그래도 안 되면 Node.js를 지우고 다시 깝니다.

④ (Windows) `python` 을 찾을 수 없다고 나온다

1.7절의 **Add python.exe to PATH** 체크박스를 놓친 경우가 거의 전부입니다. Python을 지우고 다시 깔면서 **이번엔 체크박스를 켭니다.**

⑤ 메시지를 보냈는데 답이 없다

1. **로그인 상태** — 패널에 입력칸이 보이나요? 로그인 방식을 고르는 화면(그림 1-8)이 다시 떠 있다면 로그인이 풀린 것입니다
2. **인터넷 연결**
3. **VS Code를 껐다 켜기** — 놀랍게도 이걸로 자주 해결됩니다

⑥ 회사 PC라 설치가 막힌다

회사 보안 정책으로 설치가 안 되는 경우가 있습니다. **개인 PC를 쓰시는 편**이 빠릅니다. claude.ai 웹으로 대신할 수는 **없습니다.** 웹 Claude는 내 컴퓨터의 파일을 만들지 못합니다. 그 차이는 4장에서 직접 봅니다.

1.13 정리

준비한 것	하는 일	쓰는 때
⭐ 구글 플레이 개발자 계정	앱을 스토어에 올리는 자격	3주차 — **그러나 제일 먼저**
VS Code + Claude Code 확장	작업실 + AI 본체	매번
Claude Pro	넉넉히 쓰기 위한 요금제	매번
Node.js	앱 도구를 돌리는 실행기	2주차
Python 3.11.9	작업용 코드를 돌리는 실행기	1주차부터
Git	바뀌기 전 상태를 저장	3주차
GitHub · Vercel · 공공데이터포털	코드 보관 · 사이트 띄우기 · 공공 데이터	3·4주차

이 장에서 기억할 것은 하나입니다.

> **순서는 "언제 쓰나" 가 아니라 "얼마나 기다려야 하나" 로 정한다.**

다음 장에서는 방금 연 그 폴더에서 **본격적으로 첫 대화**를 합니다. 그리고 곧 화면에 창이 하나 뜹니다. 이번엔 그게 뭔지 알고 누르게 됩니다.

02장 — 첫 대화, 그리고 권한 창

> 초고 v0.1 (2026-09-14)

2.1 Claude는 폴더 하나를 본다

1장 마지막에 폴더를 하나 열고 "이 폴더 뭐 있어?" 라고 물었습니다. 그게 우연이 아닙니다. **Claude Code는 반드시 폴더 하나를 열어놓고 시작합니다.**

> 📖 **작업 폴더(working directory)**
>
> Claude가 **들여다볼 수 있는 범위.** VS Code에서 연 그 폴더 하나와, 그 안의 하위 폴더들이 전부입니다.
>
> 회사로 치면 **배정받은 책상**입니다. 그 책상 위 서류는 마음대로 보지만, 옆 팀 책상은 못 봅니다.

초보자가 가장 자주 겪는 두 가지 증상이 전부 여기서 나옵니다.

증상	진짜 원인
"그 파일 없다는데요?"	그 파일이 **작업 폴더 밖**에 있습니다
"엉뚱한 파일을 고쳤어요"	폴더를 **너무 넓게** 열었습니다

그래서 규칙이 하나 생깁니다.

> **만들 것 하나에 폴더 하나.**
>
> 바탕화면 전체나 `C:\` 전체를 열지 마세요. 범위가 넓으면 Claude가 헤매고, 엉뚱한 걸 건드립니다.

2기 1주차 수업이 **새 폴더를 작업 공간으로 꾸미는 일**로 시작하는 이유가 이것입니다. 폴더를 열어주는 것이 곧 **"여기서 일해라"** 라는 뜻이기 때문입니다.

2.2 첫 마디는 인사부터

상자를 붙여넣는 수업

2기 강의는 여러분이 문장을 지어내게 하지 않습니다. **준비된 상자를 복사해서 붙여넣게** 합니다. 강의 자료 `제작prompt.md` 에 📋 표시가 붙은 상자들이 있고, 그걸 순서대로 붙여넣으며 진행합니다.

> 📖 **프롬프트(prompt)**
>
> AI에게 보내는 **요청 글.** 입력창에 치는 문장 하나하나가 전부 프롬프트입니다. 강의 자료 이름의 `setup-prompt` , `제작prompt` 가 "미리 써둔 요청 글" 이라는 뜻입니다.

붙여넣는 법은 이렇습니다.

```
붙여넣는 곳 → Claude Code 화면 맨 아래, 커서가 깜빡이는 입력칸
붙여넣기    → Ctrl + V    (맥은 ⌘ + V)
보내기      → 엔터
```

0단계 — 말이 통하는지 확인하기

1주차 첫 상자는 인사입니다.

```
안녕? 나 오늘 처음 왔어.
이 폴더에 지침서.md 가 있을 거야. 그거 먼저 읽고,
내가 지금 어떤 폴더에 있는지 알려주면서 한 줄로 인사해줘.
```

Claude가 폴더 이름을 한 줄로 알려주고, 짧게 인사하고, **다음 상자를 붙여넣으라고** 안내하면 성공입니다. 30초면 끝납니다.

그런데 이 짧은 상자 안에 **Claude에게 파일을 읽으라는 요청**이 들어 있습니다. `지침서.md 먼저 읽고` . 그래서 여기서 **화면에 창이 하나 뜹니다.** 잠깐 멈추고 2.4절로 갑니다.

> 💡 **왜 직접 쓰지 않고 상자를 붙여넣나**
>
> `지침서.md` 는 수강생이 읽는 문서가 아닙니다. **Claude가 읽고 따르는 진행 규칙**입니다. 질문 문구, 보기, 답변별 처리, 저장 위치가 전부 거기 정해져 있습니다.
>
> 목적은 하나 — **수강생이 어떤 답을 하든 강사 화면과 같은 결과가 나오게.** 여덟 명이 각자 다르게 물으면 여덟 개의 다른 대화가 됩니다. 두 시간 안에 같이 끝까지 가려면, 요청 글을

> 맞춰 두는 게 제일 확실합니다.

2.3 Claude가 하는 일은 세 가지뿐이다

앞으로 뭘 시키든, Claude가 실제로 하는 동작은 **세 가지 중 하나**입니다.

```
① 읽기 (read)    파일을 열어 내용을 본다
② 쓰기 (write)   파일을 만들거나 고친다
③ 실행 (run)     프로그램을 돌린다
```

2기 4주 동안 할 일을 이 셋으로 뜯어보면 이렇습니다.

할 일	뜯어보면
지침서를 따라 진행하기	**읽기**
정한 것을 `브레인스토밍.md` 로 정리	**쓰기**
화면 시안을 만들어 브라우저로 열기	**쓰기 → 실행**
2주차에 앱을 내 폰에 띄우기	**쓰기 → 실행**

"웹사이트 만들어줘" 도, "앱 만들어줘" 도, 뜯어보면 이 셋의 조합입니다.

2.4 그 창의 이름은 권한 창이다

왜 뜨는가

이유는 아주 단순합니다.

> **Claude가 지금 내 컴퓨터의 파일을 건드리려 하기 때문입니다.**

ChatGPT나 웹 Claude에는 이 창이 없습니다. 건드릴 파일이 없으니까요. 글자만 주고받습니다.
그런데 Claude Code는 진짜로 내 파일을 엽니다. 그러니 **묻고 시작합니다.**

> 📖 **권한 창 / 권한 모드(permission mode)**

> Claude가 파일을 읽거나·쓰거나·실행하기 **직전에 멈춰서 허락을 구하는 장치.** 무엇을 하려는지 보여주고, 사용자가 **허용(Allow)** 또는 **거부(Deny)** 를 고릅니다.
>
> 어디까지 묻고 어디부터 자동으로 넘어갈지를 정해둔 설정을 **권한 모드**라고 부릅니다.

무엇을 보고 눌러야 하나

창에서 볼 것은 두 가지입니다.

볼 것	확인할 내용
동작	읽기인가, 쓰기인가, 실행인가
대상	**어느 파일**인가

읽기는 마음 편히 허용해도 됩니다. 파일이 바뀌지 않으니까요. **주의해야 할 건 쓰기와 실행입니다.**

> ⚠ **파일 이름을 꼭 보세요**
>
> 내가 방금 만들라고 한 파일인가요? 아니면 **이미 있던 파일을 덮어쓰려는** 건가요? 원본을 날리는 사고는 거의 전부 여기서 납니다. 이름만 한 번 읽으면 막을 수 있습니다.

왜 이렇게 귀찮게 만들었나

처음에는 짜증납니다. 몇 번이고 계속 물으니까요. 그런데 이게 **불편이 아니라 실수 방지 장치**입니다.

AI는 틀립니다(3장). 그 틀림이 **내 파일에 그대로 반영되기 전에** 사람이 한 번 끼어드는 자리가 권한 창입니다.

권한 모드는 네 가지다

얼마나 자주 물을지는 입력칸 오른쪽 아래 버튼에서 고릅니다. 누르면 목록이 뜨고, `Shift + Tab` 을 누를 때마다 차례로 바뀝니다.

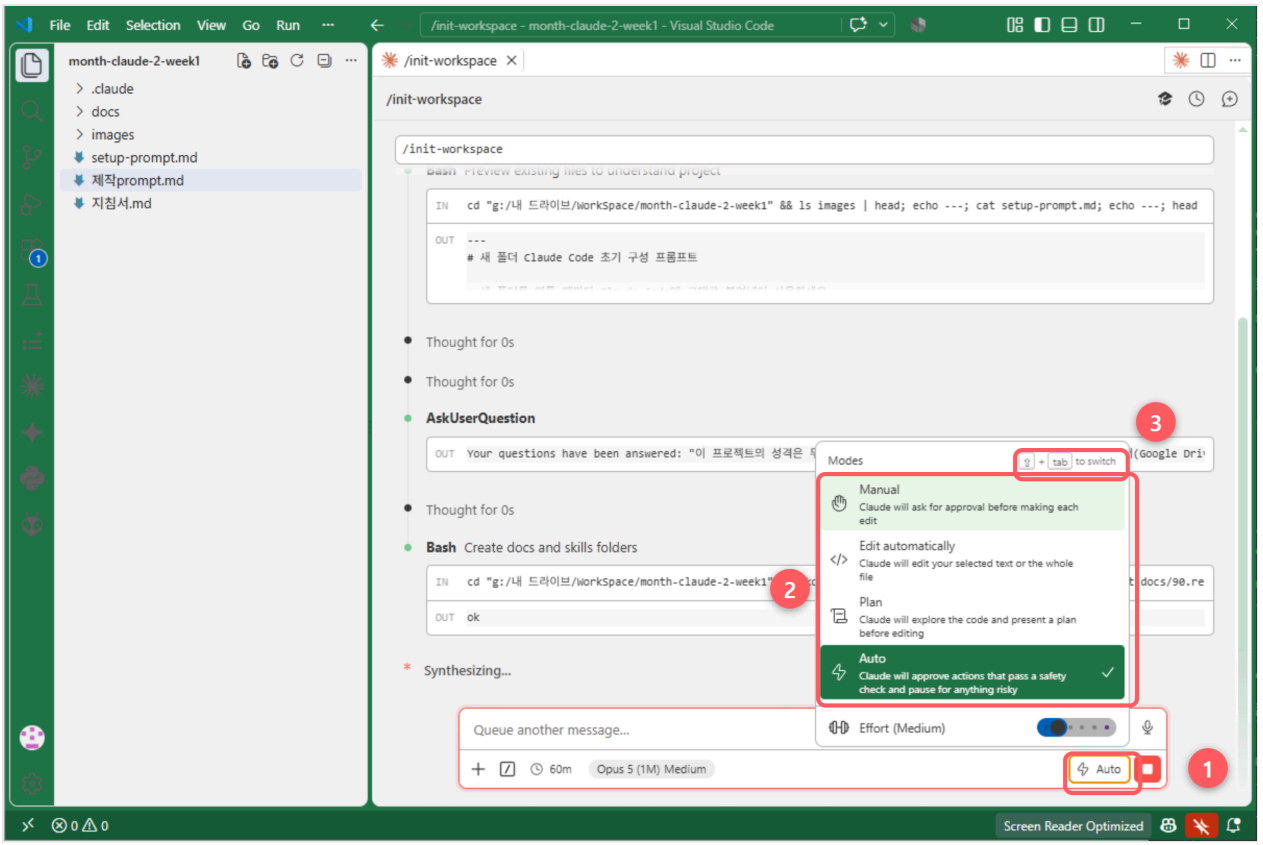

그림 2-1 권한 모드 고르기. ① 입력칸 오른쪽 아래의 모드 버튼(지금은 Auto) ② 네 가지 모드 — Manual · Edit automatically · Plan · Auto ③ Shift+Tab 으로도 바꿀 수 있다는 안내

| 모드 | 쉽게 말하면 | |---|---| | **Manual** | 고치기 전마다 **매번 묻는다.** 처음 연습할 때 | | **Edit automatically** | 파일 고치기는 묻지 않는다 | | **Plan** | 둘러본 뒤 손대기 전에 **"이렇게 하겠다" 계획부터** | | **Auto** | 안전해 보이는 건 알아서, **위험할 때만 멈추고** 묻는다 |

> ⚠ **권한 창이 안 뜨는데요?**
>
> 요즘 Claude Code는 **Auto** 가 기본으로 켜져 있을 수 있습니다. 그러면 읽기처럼 위험하지 않은 동작은 창 없이 바로 지나갑니다. 고장이 아닙니다.
>
> 이 장에서 권한 창을 **직접 보고 누르는 연습**을 하려면 모드를 **Manual** 로 바꾸세요. 익숙해진 뒤에 Auto로 돌아가면 됩니다.

지금은 Manual로 두고, 다 읽고 누르는 연습을 하세요.

자, 0단계 인사에서 뜬 창을 보세요. 동작은 **읽기**, 대상은 `지침서.md` 일 겁니다. **허용**을 누르세요.

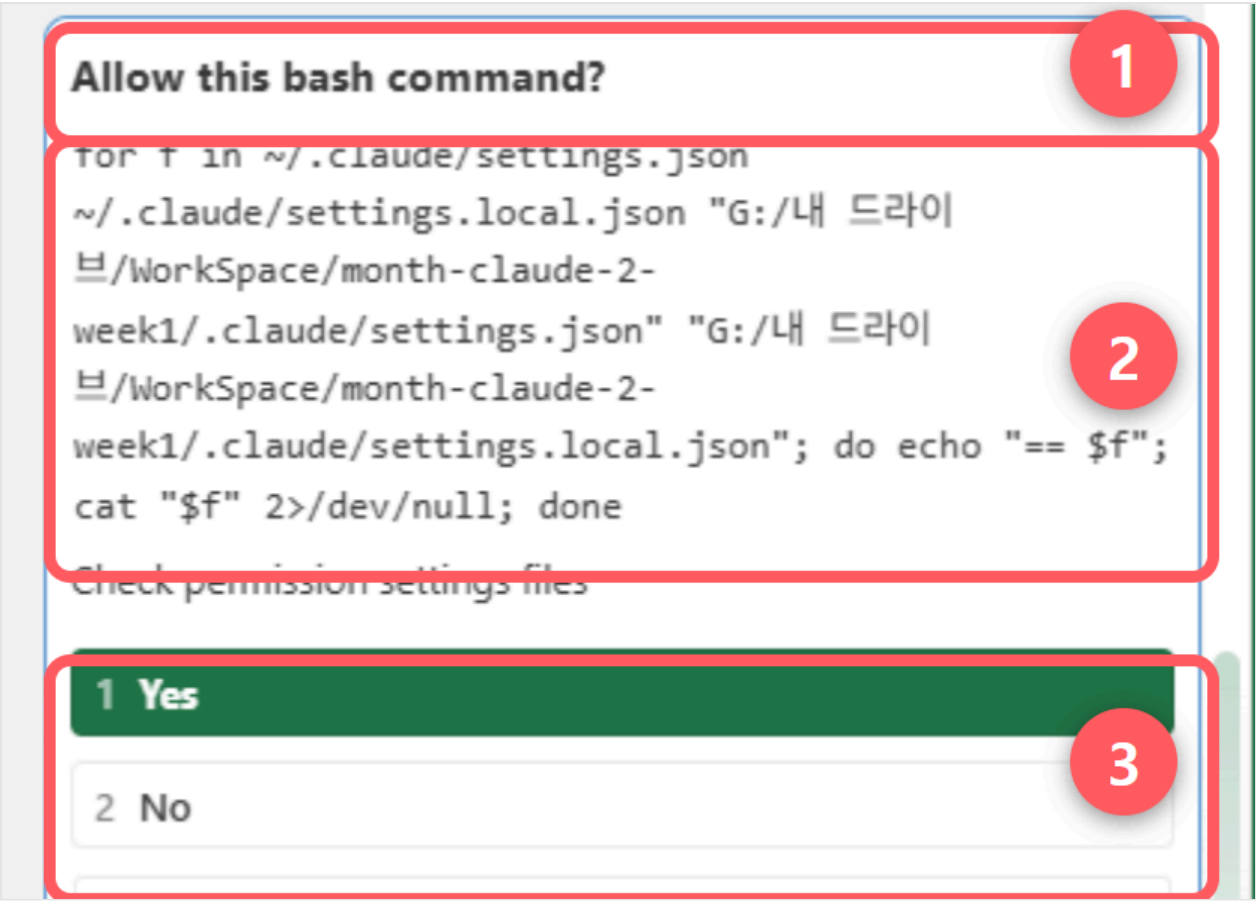

그림 2-2 권한 창. ① 무엇에 대한 허락인지 — 여기서는 「이 bash 명령을 허용할까요?」, 곧 실행입니다 ② 정확히 무엇을 하려는지 — 명령 내용과 한 줄 설명(설정 파일을 확인한다) ③ 1 Yes · 2 No

---

2.5 쓰기와 실행 — 파일 하나 만들어 열어보기

이번엔 Claude에게 파일을 **만들고, 열라고** 시켜 봅니다. 연습이라 결과물은 버려도 됩니다.

```
이 폴더에 hello.html 이라는 파일을 만들어줘.
화면 가운데에 '안녕하세요' 라고 크게 나오는 페이지면 돼.
```

권한 창이 또 뜹니다. 이번엔 **쓰기**입니다. 만들려는 파일 이름이 `hello.html` 로 적혀 있는지 보고 허용하세요.

허용하면 **VS Code 왼쪽 파일 목록에 `hello.html` 이 새로 생깁니다.**

잠깐 이 장면을 보세요. 방금 **AI가 내 컴퓨터에 파일을 만들었습니다.** 웹 챗봇이라면 "아래 코드를 복사해서 hello.html 로 저장하세요" 라고 했을 자리입니다. 복사도, 붙여넣기도, 저장도 없었습니다.

이어서 칩니다.

```
방금 만든 hello.html 을 브라우저로 열어줘
```

권한 창이 세 번째로 뜹니다. **실행**입니다. 허용하면 브라우저에 "안녕하세요" 가 뜹니다.

그림 2-3 쓰기와 실행. ① 쓰기 — 작업 폴더에 hello.html 이 새로 생긴다 ② 실행 — "브라우저로 열어줘" 한마디에 그 파일이 브라우저에 열려「안녕하세요」가 뜬다. 복사도 붙여넣기도 저장도 없었다

이게 1주차 8단계의 축소판입니다. 8단계에서 Claude는 화면 시안을 만들어 **아티팩트**라는 링크로 띄우고, 그게 안 되면 **HTML 파일로 만들어 브라우저로 열어 줍니다.** 방금 한 일과 같습니다. 크기만 다릅니다.

2.6 되돌리기 — 알아야 과감하게 시킨다

왜 지금 배우나

초보자가 Claude에게 과감하게 못 시키는 이유는 대부분 하나입니다. **"망가지면 어떡하지."**

> 💡 **되돌릴 수 있다는 걸 알아야 과감하게 시킨다**
>
> 되돌리는 법을 모르면 요청이 소심해집니다. "조금만 바꿔줘" 만 반복하다 시간이 갑니다.
> 되돌릴 줄 알면 **"완전히 다르게 해봐"** 가 가능해집니다. 마음에 안 들면 돌아오면 되니까요.
>
> 2기 강의 소개에 "망쳤을 때 되돌리는 법" 이 들어 있는 이유입니다.

말로 되돌리기

제일 간단합니다. **바로 다음 요청에서** 이렇게 말하세요.

```
방금 건 취소하고 다시 하자
```

Claude는 방금 뭘 바꿨는지 기억하고 있습니다. 다만 **오래 지나면 안 됩니다.** 스무 번쯤 주고받은 뒤에 "처음 거로 돌려줘" 하면 잘 안 됩니다. 왜 그런지는 3장「AI가 기억하는 양에는 끝이 있다」에서 봅니다. 규칙은 하나 — **되돌리려면 바로 다음에.**

VS Code로 되돌리기

Claude가 파일을 고친 직후라면, 그 파일을 열어두고 `Ctrl + Z` (Mac: `Cmd + Z`)가 먹힙니다.

진짜 안전장치 — 사본 떠두기

가장 확실한 방법은 촌스럽지만 이겁니다. **마음에 드는 상태가 나오면, 그때 사본을 하나 떠두세요.**

```
지금 hello.html 을 hello_백업1.html 로 복사해줘
```

이 일을 프로그램이 자동으로, 매번 해주는 것이 **Git**(깃)입니다. 3주차에 만든 것을 인터넷에 올릴 때 같이 만납니다. 그때까지는 손으로 합니다.

2.7 막혔을 때 — 그대로 치세요

1주차 자료 `제작prompt.md` 맨 끝에 있는 표입니다. 2기 1주차에서 실제로 쓰는 문장들입니다.

이런 일이 생기면	이렇게 치세요
질문을 한꺼번에 여러 개 한다	`한 번에 하나씩만 물어봐`
답이 너무 길어서 못 읽겠다	`짧게`
자꾸 코드를 짜려고 한다	`아직 코드 쓰지 마`
무슨 말인지 모르겠다	`그게 무슨 뜻이야? 쉽게 설명해줘`
뭘 해야 할지 모르겠다	`내가 지금 뭘 하면 돼?`
되돌리고 싶다	`방금 건 취소하고 다시 하자`
시안이 두 개 다 마음에 든다	`둘 다 좋은데 합칠 수 있어?`
시안이 다 별로다	`셋 다 별로야. 다시 만들어줘`

그리고 하나 더.

- 오타 나도 괜찮습니다. **말하듯이** 쓰시면 알아듣습니다
- 모르면 `모르겠어` 라고 답해도 됩니다
- 잘못 말해도 안 망가집니다. 다시 말하면 고쳐 줍니다

표에 없는 문제는 **그냥 물어보세요.** `이거 왜 이래?` 도 충분한 질문입니다.

2.8 안 될 때

메시지를 보냈는데 아무 반응이 없다

1. **패널에 입력칸이 보이나요?** 로그인 방식을 고르는 화면이 떠 있으면 로그인이 풀린 겁니다 → **Claude.ai Subscription** 으로 다시 로그인
2. **인터넷 연결**을 확인하세요
3. **VS Code를 껐다 켜세요.** 정말로 이걸로 자주 해결됩니다

"지침서.md 가 없다" 고 한다

2.1절입니다. `지침서.md` 가 **작업 폴더 밖**에 있습니다. 강의 자료 압축을 푼 파일들이 **지금 연 폴더 안에** 있는지 보세요. `제작prompt.md` 와 `지침서.md` 는 같은 폴더에 둡니다.

권한 창에서 실수로 거부를 눌렀다

아무 일도 안 일어납니다. 그냥 **다시 시키면** 됩니다. 거부는 되돌릴 수 없는 사고가 아닙니다. 오히려 **애매하면 거부하는 게 맞습니다.**

인사만 하라고 했는데 자꾸 뭘 만들자고 한다

`지침서.md` 를 안 읽었을 가능성이 큽니다. `지침서.md 먼저 읽고 0단계만 해줘` 라고 다시 치세요.
그래도 앞서가면 `내가 다음 상자 붙여넣을 때까지 기다려` 라고 하세요.

2.9 정리

오늘 한 일은 세 줄입니다.

- 파일을 **읽게** 했다 — `지침서.md`
- 파일을 **쓰게** 했다 — `hello.html`
- **실행**하게 했다 — 브라우저로 열기

그때마다 권한 창이 떴고, 이제 그게 뭔지 압니다. 그리고 망쳐도 **되돌릴 수 있다**는 것도 압니다.

다음 장은 잠깐 손을 쉽니다. 지금까지 "Claude" 라고 불러온 이것이 **대체 무엇인지**, 왜 틀리는지, 왜 어떤 건 잘하고 어떤 건 못하는지 — 짧게 짚고 갑니다.

03장 — AI에도 종류가 있다

> 초고 v0.1 (2026-09-14)

3.1 이 장은 손을 쉽니다

2장까지 손을 움직였습니다. 이 장은 실습이 없습니다. **읽기만 하는 장**입니다.

지금부터 4주 동안 "AI에게 만들 것을 시키는" 일을 계속할 텐데, **상대가 뭔지 모르고 시키면 계속 헛다리를 짚기 때문**입니다.

이 장에서 답할 질문은 셋입니다.

1. AI라고 부르는 것들이 다 같은 건가?
2. Claude는 대체 어떤 원리로 답하나?
3. 왜 "수정해줘" 라고 하면 **엉뚱하게** 고쳐 오나?

세 번째 질문이 이 책의 핵심과 닿아 있습니다. 짧게 갑니다.

3.2 AI는 한 종류가 아니다

"AI 써봤어요?" 라는 질문은 사실 "탈것 타봤어요?" 만큼 막연합니다. 자전거인지 비행기인지에 따라 완전히 다른 이야기니까요.

지금 우리가 AI라고 부르는 것들은 크게 이렇게 나뉩니다.

종류	하는 일	예
💬 **대화형** (챗봇)	질문에 답하고, 글을 쓰고, 요약한다	ChatGPT · Claude · Gemini
🎨 **이미지 생성**	글을 받아 그림을 만든다	Midjourney · DALL-E · Stable Diffusion
⌨️ **코딩** ⭐	코드를 쓰고, **파일을 만들고**, 실행한다	**Claude Code** · GitHub Copilot · OpenAI Codex
🎯 **특화**	번역·음성·영상 한 분야에 맞춤	DeepL · Whisper · Sora

우리가 쓰는 건 코딩 칸입니다.

여기서 첫 번째 오해가 풀립니다. Claude Code에게 "가게 사진을 그려줘" 라고 해도 안 됩니다. 종류가 다릅니다. 2기 1주차에서 앱 첫 화면에 넣을 사진을 `images/` 폴더에 **샘플로 미리 넣어 두는** 이유가 이것입니다. 사진은 사람이 준비합니다.

다만 **코딩 칸은 대화형 칸을 포함**합니다. Claude Code는 대화형이 하는 일(글 읽고 글로 답하기)을 다 하면서, 거기에 두 가지가 더해진 것입니다.

```
대화형 AI          글 → 글
    +
파일 접근          내 컴퓨터의 파일을 읽고 쓴다
    +
코드 실행          프로그램을 돌려본다
──────────────────────────────────────
= Claude Code
```

한 줄로 줄이면 — **일반 챗봇은 "답변"만 합니다. 코딩 AI는 "실행"까지 합니다.**

3.3 Claude는 어떤 원리로 답하는가

다음 낱말을 고르는 기계

Claude 같은 대화형 AI의 정식 이름은 이렇습니다.

> 📖 **LLM(Large Language Model, 거대 언어 모델)**
>
> 엄청난 양의 글을 미리 읽어두고, **"이 다음에 올 낱말로 가장 그럴듯한 것"** 을 계속 골라 문장을 만들어 내는 프로그램.
>
> **거대(Large)** — 읽은 글의 양이 어마어마하다 **언어(Language)** — 다루는 재료가 글이다 **모델(Model)** — 그렇게 만들어진 프로그램 덩어리

너무 단순해 보이지만 정말 이게 전부입니다.

"오늘 날씨가 참" 다음에 뭐가 올 것 같으세요? "좋네요" 가 떠오르셨다면, 방금 머릿속에서 일어난 일이 LLM이 하는 일과 같습니다. 다만 LLM은 그걸 **사람이 평생 읽을 양의 수백만 배**를 읽고 나서 합니다.

코드도 글입니다. 웹사이트의 코드도 결국 글자의 나열이라, LLM은 "이 다음에 올 가장 그럴듯한 코드" 를 고르는 방식으로 사이트를 만듭니다.

그래서 생기는 두 가지 성질

이 구조 하나로, 여러분이 겪을 거의 모든 현상이 설명됩니다.

① 아주 그럴듯한 것을 만듭니다

매끄럽게 잇는 게 이 기계의 본업입니다. 그래서 **"사이트처럼 보이는 것"**, **"앱 화면처럼 보이는 것"** 을 만드는 데 압도적으로 강합니다. 딸깍 하면 뭔가 나오는 이유입니다.

② 모르는 것도 그럴듯하게 만듭니다 ⚠️

여기가 중요합니다. LLM은 **"가장 그럴듯한 다음 낱말"** 을 고를 뿐, **"이게 사실인가"** 를 확인하지 않습니다.

> 📖 **환각(hallucination)**
>
> AI가 **사실이 아닌 것을 사실인 것처럼 말하는 현상.** 거짓말과 다릅니다. 거짓말은 진실을 알면서 속이는 것이지만, AI는 **애초에 참·거짓을 구분하지 않습니다.** 그럴듯함만 봅니다.

웹과 앱에서는 이런 모습으로 나타납니다. 1주차 8단계 화면 시안에는 **예시 데이터**가 채워져 나옵니다. `김하나 · 이두리 · 박세라` 같은 이름, 그럴듯한 시간표, 그럴듯한 가격. 시안이니까 괜찮습니다. 문제는 **그대로 인터넷에 올라갈 때**입니다.

> 💡 **틀리는 방식에 규칙이 있습니다**
>
> 아무렇게나 틀리는 게 아닙니다.
>
잘 맞히는 것	잘 틀리는 것
화면 구조·배치·색 조합	**구체적인 숫자** — 가격, 영업시간, 전화번호
문장 다듬기·소개글	**고유명사** — 가게 이름, 주소, 사람 이름
흔한 화면 모양	**최신 정보** — 요즘 바뀐 규정, 새로 나온 기능
"어떻게 만들지?"	**"그게 사실이야?"**
>
> 그래서 사이트를 올리기 전에 볼 곳도 정해져 있습니다. **숫자와 이름을 보세요.** 배치가 예쁜지 보는 데 시간 쓰지 마세요.

3.4 AI가 기억하는 양에는 끝이 있다

두 번째로 알아야 할 성질입니다. 그리고 **"수정해줘" 가 엉뚱하게 가는 이유**의 절반이 여기 있습니다.

AI와 오래 대화하다 보면 이런 일이 생깁니다.

> "아까 메뉴는 세 개로 줄이기로 했잖아요." "죄송합니다. 메뉴를 다섯 개로 늘려 드릴까요?"

치매가 아닙니다. **구조상 당연한 일**입니다.

> 📖 **컨텍스트(context, 맥락)**
>
> AI가 **지금 이 순간 눈앞에 펼쳐놓고 보고 있는 글 전체.** 지금까지 나눈 대화 + 읽은 파일 내용이 전부 여기에 쌓입니다.
>
> **공책 한 권**이라고 생각하세요. 대화가 쌓일수록 공책이 채워집니다. 그리고 공책은 **분량이 정해져 있습니다.** 다 차면, 앞장부터 찢어냅니다.

앞장이 찢겨 나간 부분이 "아까 정한 그거" 입니다.

이 공책이 얼마나 찼는지 세는 단위를 **토큰(token)** 이라고 합니다. 글자 수도 낱말 수도 아닌 그 중간쯤입니다. 숫자를 외울 필요는 없습니다. **"길어지면 찬다"** 는 감각 하나면 충분합니다.

새 대화를 열면 공책도 새것이다

더 중요한 사실이 있습니다. **대화를 새로 열면, AI는 이전 대화를 전혀 모릅니다.** 어제 두 시간 동안 같이 정한 것도, 새 대화에서는 백지입니다.

그러면 이런 일이 생깁니다.

```
월요일    두 시간 대화하며 사이트를 만든다
          (색은 차분하게, 메뉴는 셋, 예약 버튼은 맨 위 – 이유까지 다 얘기함)
              ↓
목요일    새 대화를 열고 "첫 화면 좀 수정해줘"
              ↓
AI        월요일의 이유를 하나도 모른다
          → 자기가 보기에 그럴듯한 쪽으로 고친다
          → 색이 화려해지고, 메뉴가 늘어난다
```

AI가 게으르거나 고집을 부리는 게 아닙니다. **3.3절의 LLM과 3.4절의 공책이 만나면 이렇게 될 수밖에 없습니다.** 모르면, 그럴듯한 쪽을 고릅니다.

> 💡 **그래서 "만든 사람의 생각" 을 파일에 적어 둔다**
>
> 공책은 찢기지만 **파일은 남습니다.** 새 대화를 열어도 Claude는 폴더 안의 파일을 읽을 수 있습니다(2장).
>
> 무엇을 만들고 **무엇을 안 만들기로 했는지**, 왜 이 화면을 골랐는지, 무엇이 **왜 바뀌었는지.** 이걸 파일로 남겨 두면, 목요일의 "수정해줘" 도 월요일의 생각에 맞춰 고쳐집니다.
>
> 2기 1주차가 코드 없이 두 시간 내내 **정하고 문서로 남기는** 이유가 이것입니다. 그 문서의 이름이 **PRD(설계 문서)** 이고, 2주차에 자세히 봅니다.

그래서 이렇게 하세요

상황	할 일
대화가 길어져 답이 이상해진다	**새 대화를 시작**하세요. 공책을 새로 꺼내는 겁니다
만들 것이 바뀌었다	이어 하지 말고 **새 대화**
새 대화에서 예전 결정을 잊는다	결정을 **파일에 적어 두고** "그 파일 먼저 읽어"
매번 같은 말을 반복하게 된다	그 말을 **파일에 적어 둘 때가 된 것**입니다

3.5 정리 — 이 장에서 기억할 세 줄

용어	한 줄
LLM	다음 낱말을 고르는 기계. 그래서 잘 만들고, 그래서 지어냅니다
환각	모르는 걸 그럴듯하게 지어내는 현상. **숫자와 이름을 의심하세요**
컨텍스트	AI의 공책. 차면 앞장이 찢기고, 새 대화면 백지입니다

그리고 여기서 따라 나오는 실천 두 가지 —

- **올리기 전에 숫자와 이름은 사람이 본다**
- **남겨야 할 생각은 대화가 아니라 파일에 적는다**

다음 장에서는 다시 손을 움직입니다. **웹 챗봇과 Claude Code에게 똑같이 "웹페이지 만들어줘" 를 시켜서**, 방금 배운 차이를 눈으로 확인합니다.

04장 — 챗봇과 무엇이 다른가

> 초고 v0.1 (2026-09-14)

4.1 말로 하면 안 와닿습니다

"Claude Code는 내 파일을 직접 만진다."

3장까지 이 말을 여러 번 했습니다. 그런데 이 문장은 읽어도 잘 안 와닿습니다. **직접 해보기 전까지는 그냥 문장일 뿐입니다.**

2기 강의 안내에 이런 사람이 나옵니다.

> **"ChatGPT는 써봤는데 대화만 남고 손에 쥔 게 없어 허전했던 분."**

이 장은 그 허전함의 정체를 확인하는 장입니다. 딱 하나만 합니다.

> **똑같은 일을 웹 챗봇과 Claude Code에게 각각 시켜본다.**

같은 일을 두 번 하니까 비효율적으로 보이지만, 이 30분이 나중에 **"이건 어디서 할까"** 를 판단하는 기준이 됩니다.

4.2 시켜볼 일 — 가게 소개 페이지 한 장

시킬 일은 이것입니다. 가게 이름과 내용은 아무거나 좋습니다.

> **"동네 꽃집 소개 웹페이지를 만들어줘. 가게 이름, 한 줄 소개, 영업시간, 전화번호가 들어가게. 그리고 그걸 파일로 저장해줘."**

마지막 문장이 핵심입니다. **"파일로 저장해줘."** 여기서 둘이 갈립니다.

그 전에 낱말 하나만 짚고 갑니다.

> 📖 **HTML(에이치티엠엘)**
>
> **웹페이지를 이루는 글.** 브라우저(크롬·엣지·사파리)는 이 글을 읽어서 화면에 제목·문단·버튼을 그려 줍니다. 메모장으로 열면 글자가 보이고, 브라우저로 열면 페이지가 보입니다. 파일 이름이 `.html` 로 끝납니다.

4.3 먼저 웹 챗봇에서

브라우저에서 claude.ai(또는 ChatGPT)에 들어가 새 대화를 열고, 위 요청을 그대로 씁니다.

결과: 페이지는 아주 잘 만들어 줍니다. 옆에 미리보기가 뜨고, 파일로 받는 버튼도 붙습니다.

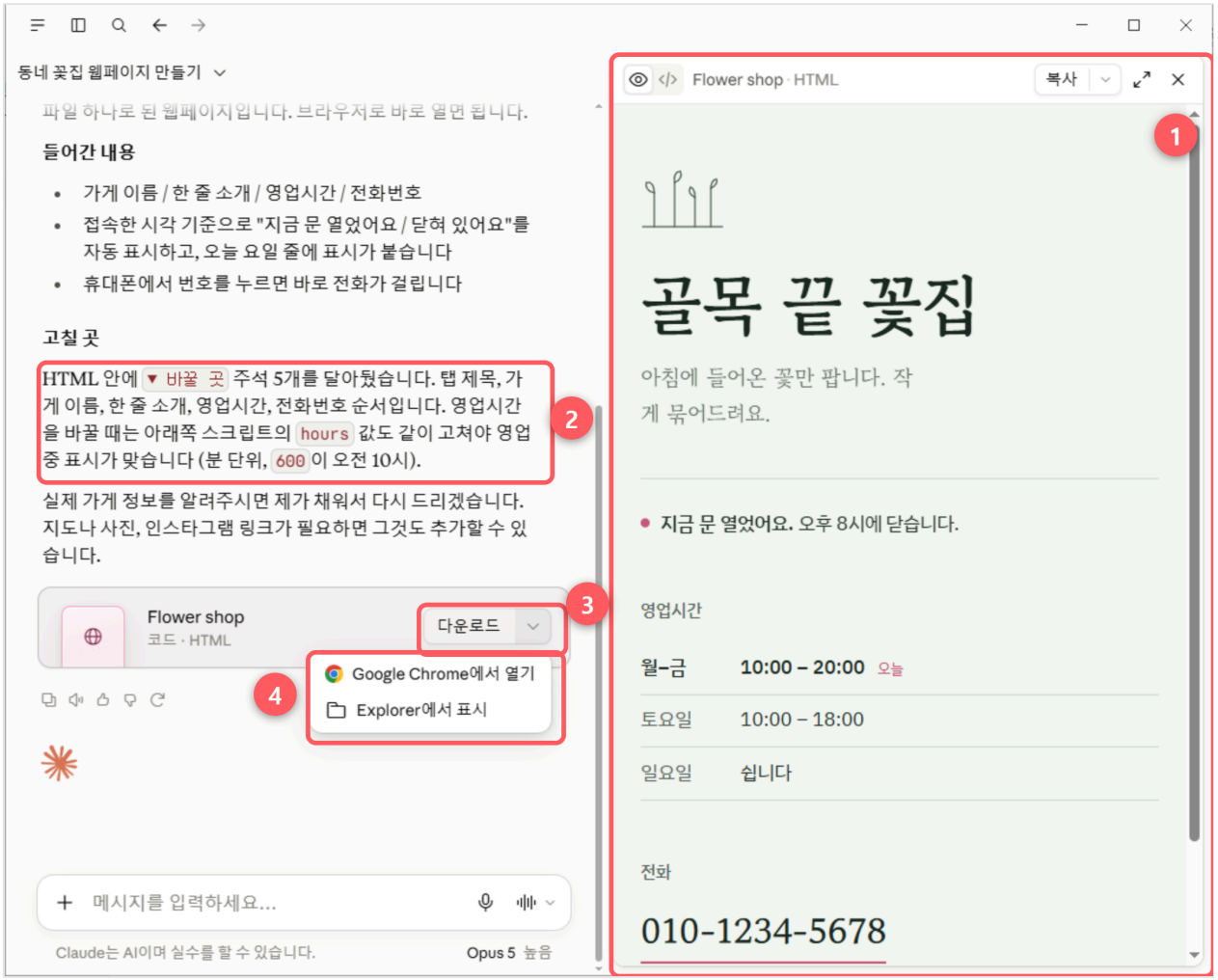

그림 4-1 웹 챗봇에 꽃집 페이지를 시킨 화면. ① 오른쪽 미리보기 ② "고칠 곳" 안내 — 영업시간을 바꾸려면 코드 안의 hours 값도 같이 고치라고 한다 ③ 다운로드 버튼 ④ 받은 파일을 브라우저로 열거나 폴더에서 찾기

그런데 **파일은 내 작업 폴더에 생기지 않습니다.** 그다음은 여러분 몫입니다.

1. **다운로드** 를 누른다 — 파일은 PC의 「다운로드」 폴더로 간다

2. 다운로드 폴더를 열어 파일을 찾는다
3. 작업 폴더로 옮기고, 이름이 겹치면 덮어쓸지 정한다

진짜 문제는 고칠 때 옵니다

한 번이면 참을 만합니다. **진짜 문제는 그다음입니다.**

> "영업시간을 바꿔줘. 그리고 맨 위를 좀 더 크게."

②를 다시 보세요. 웹 챗봇은 **어디를 고치면 되는지 알려 줍니다.** 코드를 직접 고치거나, 새로 만든 파일을 **다시** 받아서 위의 세 단계를 **또** 해야 합니다. 열 번 고치면 열 번 합니다. 그러다 옛날 파일을 옮기고, 어느 게 최신인지 헷갈립니다.

챗봇은 조언자입니다. 무엇을 어떻게 하면 되는지 말해 줍니다. **하는 건 사람입니다.**

4.4 이번엔 Claude Code에서

VS Code로 돌아옵니다. 2장에서 연 작업 폴더가 열려 있어야 합니다.

Claude 패널에 같은 요청을 칩니다. **다운로드도, 옮기기도 없습니다.**

```
동네 꽃집 소개 웹페이지를 만들어줘.
가게 이름, 한 줄 소개, 영업시간, 전화번호가 들어가게.
index.html 파일로 저장하고 브라우저로 열어줘.
```

- 권한 창 ① **쓰기** → 파일 이름이 `index.html` 인지 보고 허용
- 권한 창 ② **실행** → 허용

왼쪽 파일 목록에 `index.html` 이 생기고, 브라우저에 페이지가 뜹니다.

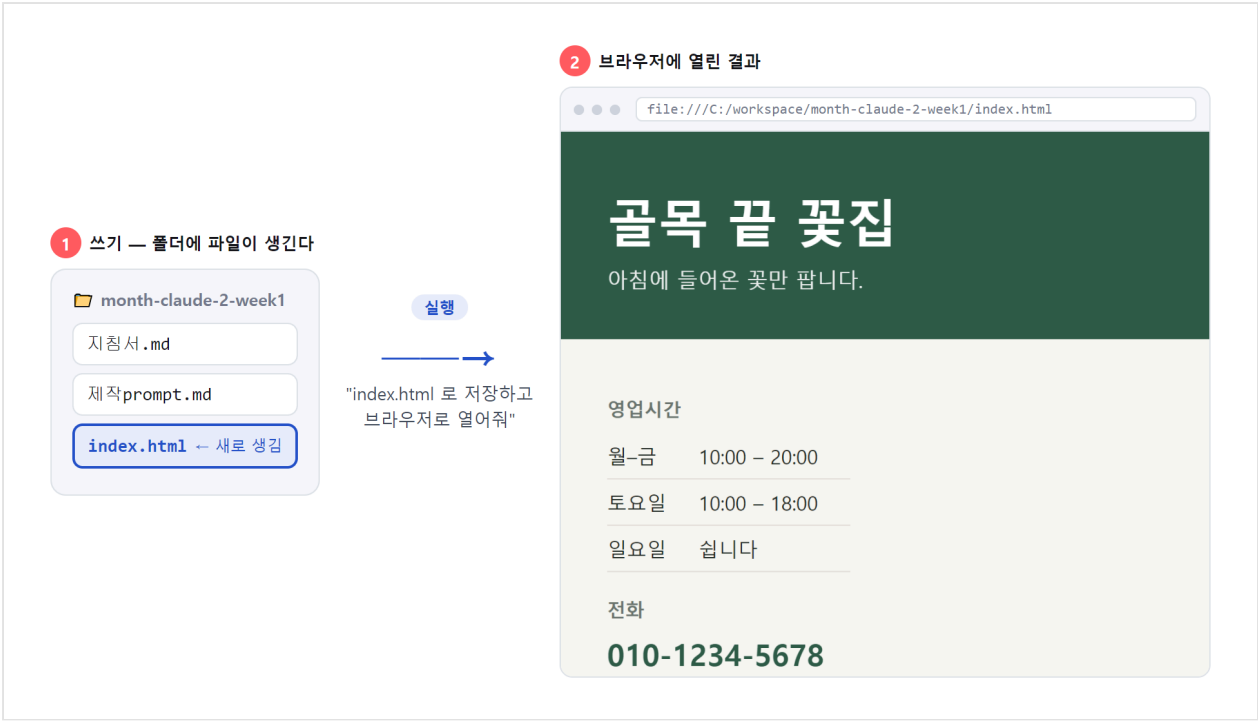

그림 4-2 Claude Code에서 같은 일을 시킨 결과. ① 작업 폴더에 index.html 이 바로 생긴다 ② 브라우저에 열린 꽃집 페이지

이번엔 고쳐 봅니다.

```
영업시간을 오전 10시~오후 8시로 바꾸고, 맨 위를 좀 더 크게 해줘.
```

Claude는 **그 파일을 직접 열어서 그 부분만** 고칩니다. 새 코드를 받을 일이 없습니다. 브라우저를 새로고침하면 바뀐 페이지가 보입니다.

Claude Code는 실행자입니다. 말하면 **해 줍니다.**

4.5 한 장으로 보면

	🌐 웹 챗봇 (claude.ai · ChatGPT)	⌨ Claude Code
파일 접근	파일을 직접 **올려야** 한다	내 컴퓨터의 파일을 **직접 읽고 쓴다**
실행	코드를 **보여줄 뿐**	코드를 **직접 실행**하고 결과를 만든다
고치기	고칠 곳을 **알려 주고**, 새 파일을 **다시** 받는다	**그 파일의 그 부분만** 고친다
결과물	**다운로드 폴더**로 받는 파일	**작업 폴더**에 바로 생기는 파일

일하는 흐름으로 그리면 차이가 더 선명합니다.

```
웹 챗봇      요청 → AI 답변 → 사람 다운로드 → 사람 찾기 → 사람 옮기기

Claude Code  요청 → AI: 읽기 → 만들기 → 저장 → 결과 파일
```

위 줄은 사람이 세 번 끼고, 아래 줄은 **사람이 끼지 않습니다.** 사람이 하는 일은 권한 창에서 확인하는 것뿐입니다.

4.6 파일로 남아야 하는 진짜 이유

"다운로드 몇 번 하는 게 뭐 대수냐" 싶을 수 있습니다. 2기에서는 대수입니다. 이유가 두 가지 있습니다.

> 💡 **파일이 있어야 인터넷에 올릴 수 있다**
>
> 3주차에 사이트를 인터넷에 올립니다. 올리는 과정은 이렇습니다 — **내 폴더의 파일**을 GitHub에 올리고, Vercel이 그 파일을 가져가 주소를 붙여 띄웁니다.
>
> 출발점이 **폴더 안의 파일**입니다. 대화창의 글은 올릴 수 없습니다. 2주차의 앱도 같습니다. 폰에서 돌아가는 앱은 폴더에 쌓인 파일 뭉치입니다.

> 💡 **파일이 있어야 "만든 사람의 생각" 이 같이 남는다**
>
> 3장에서 봤듯, AI는 새 대화를 열면 이전 대화를 모릅니다. 웹 챗봇에서 두 시간 동안 정한 것은 **그 대화창 안에만** 있습니다.
>
> Claude Code는 폴더에 사이트 파일과 **나란히 설계 문서를 둘 수 있습니다.** 무엇을 만들고 무엇을 안 만들기로 했는지, 무엇이 왜 바뀌었는지. 다음에 "수정해줘" 라고 하면 Claude는 **그 문서를 먼저 읽고** 원래 생각에 맞춰 고칩니다. 대화는 사라져도 생각은 폴더에 남습니다.

아티팩트는 무엇인가

2기 1주차 8단계에서 화면 시안을 **아티팩트**라는 링크로 띄워 봅니다. 헷갈리기 쉬운 부분이라 짚고 갑니다.

아티팩트는 **만든 화면을 링크로 열어 보는 미리보기 창**입니다. 여러 안을 나란히 보고 고르기에 편합니다. 그런데 미리보기가 안 뜰 때가 있습니다. 그러면 Claude는 붙잡고 씨름하지 않고 **바로 HTML 파일로 만들어** 브라우저로 열어 줍니다.

```
docs/30.output/시안-8-1-웹-로그인명단.html
```

어느 쪽으로 보든, **고르는 것은 사람이고 남는 것은 폴더의 기록**입니다. 고른 결과는 설계 문서에 적힙니다.

4.7 그럼 웹 챗봇은 이제 안 쓰나

아닙니다. **웹 챗봇이 더 나은 경우가 분명히 있습니다.**

상황	어디서	왜
그냥 물어보고 싶다	**웹**	폴더 열 것도 없습니다. 브라우저가 빠릅니다
아이디어를 가볍게 굴려보고 싶다	**웹**	결과물이 파일로 나올 필요가 없습니다
휴대폰으로	**웹**	Claude Code는 PC에서 씁니다
사이트·앱을 **만든다**	**Code**	결과가 파일로 남아야 올릴 수 있습니다
만든 것을 **계속 고친다**	**Code**	그 부분만 고치고, 기록을 읽고 고칩니다
인터넷에 올린다	**Code**	출발점이 폴더의 파일입니다

한 줄로 줄이면 이렇습니다.

> **묻고 끝나면 웹. 만들고 고치고 올릴 거면 Claude Code.**

4.8 Claude Code도 못하는 것

못하는 것도 말씀드립니다.

못하는 것	설명
사진·그림 만들기	종류가 다릅니다(3.2절). 사진은 사람이 준비합니다
사실 확인	3.3절의 환각. 가격·영업시간·주소는 사람이 봐야 합니다
계정 가입·본인 확인	개발자 계정, GitHub, Vercel 가입은 **사람이 직접** 합니다
기다리기	구글의 신원 확인, 공공데이터 승인은 AI가 앞당길 수 없습니다
판단이 필요한 결정	"이 기능을 넣을까 뺄까" 는 만드는 사람의 일입니다

셋째·넷째 줄이 2기에서 특히 중요합니다. 0장에서 말한 **"Claude가 대신 해주지 못하는 구간"** 이 바로 여기입니다.

> ⚠ **개인정보를 다룰 때**
>
> Claude에게 읽힌 파일 내용은 **Anthropic(Claude를 만든 회사) 서버로 전송됩니다.** 내 컴퓨터 안에서만 처리되는 게 아닙니다.
>
> 손님 명단·연락처 같은 실제 개인정보를 넣기 전에 한 번 더 생각하세요. 시안 단계에서는 `김하나 · 이두리` 같은 **가짜 이름**으로 충분합니다.

4.9 정리

같은 일을 두 번 시켜봤습니다. 결론은 한 줄입니다.

> **웹 챗봇은 말해주고, Claude Code는 해준다.**

그리고 2기에서 그 차이가 만드는 결과는 이것입니다 — **웹 챗봇의 결과는 대화창과 다운로드 폴더에 남고, Claude Code의 결과는 작업 폴더에 남는다. 폴더에 남아야 고칠 수 있고, 올릴 수 있다.**

여기까지가 준비입니다. 다음 장부터 2기 1주차 실습 — **새 폴더를 작업 공간으로 꾸미고, 무엇을 만들지 정하는 일** — 로 들어갑니다.

05장 — 작업 공간 만들기, CLAUDE.md와 Skill

> 초고 v0.1 (2026-09-14)

5.1 1주차는 폴더를 꾸미는 일로 시작한다

2장에서 이런 규칙을 봤습니다. **만들 것 하나에 폴더 하나.** Claude는 열어준 폴더 하나만 보기 때문입니다.

그런데 폴더를 새로 만들기만 하면 빈 방입니다. 빈 방에서 일을 시작하면 금방 어질러집니다. 받은 자료, 쓰다 만 초안, 완성본, 참고한 링크가 한데 섞입니다. 그러면 Claude도 헷갈리고, 사람도 헷갈립니다.

그래서 2기 1주차의 1부는 **빈 폴더를 "일할 수 있는 방" 으로 꾸미는 일**입니다. 만드는 것은 세 가지입니다.

	만드는 것	한 줄로
①	`CLAUDE.md`	Claude가 **대화를 시작할 때마다 먼저 읽는** 규칙 문서
②	`.claude/skills/`	**Skill**(반복 작업 절차)을 넣어 두는 폴더
③	`docs/` 아래 작업 폴더 4개	원본 · 작업 중 · 결과물 · 참고 — 자리를 나눠 둔다

이 장은 셋이 각각 **왜** 필요한지 봅니다. 그리고 이 셋이 작가의 말에서 말한 **"만든 사람의 생각을 기록해 두는 자리"** 의 첫 번째 모습이라는 것도 봅니다.

5.2 따라 하기 — 프롬프트 한 번이면 끝난다

⌨️ 따라 하기

1. 새 폴더를 만듭니다 (1장 1.11절 규칙대로 — 짧고, 영어고, 드라이브 바로 아래)
2. VS Code에서 그 폴더를 엽니다
3. 강의 자료 `setup-prompt.md` 의 📋 **복사용 프롬프트** 상자를 통째로 복사해 Claude 패널에 붙여넣고 Enter

프롬프트의 첫 줄은 이렇게 시작합니다.

```
이 폴더를 Claude Code 작업 공간으로 초기 구성해줘.
다음 세 가지를 만들어줘.
```

그 아래에 무엇을, 어떤 모양으로 만들지가 자세히 적혀 있습니다. 권한 창이 몇 번 뜹니다. 전부 **쓰기**이고, 대상은 `CLAUDE.md` 와 폴더들입니다. 이름을 보고 허용하세요.

끝나면 Claude가 무엇이 생겼는지 **파일 트리**로 보여줍니다. 5.5절의 그림 5-2 ②와 같은 모양이면 성공입니다.

폴더 안의 내용을 나뭇가지 모양으로 그린 목록을 **파일 트리(file tree)** 라고 부릅니다. 줄기(폴더)에서 가지(하위 폴더)가 뻗고 끝에 잎(파일)이 달린 모양이라 트리(나무)입니다. `├──` 와 `└──` 는 "이 폴더 안에 들어 있다" 는 뜻의 선입니다.

> ⚠ **`.claude` 폴더가 안 보여요**
>
> 이름이 점(`.`)으로 시작하는 폴더는 **숨김 폴더**로 취급될 때가 있습니다. VS Code 탐색기에서는 보통 보이지만, Windows 파일 탐색기나 Mac Finder에서는 안 보일 수 있습니다. **없어진 게 아닙니다.** VS Code에서 확인하세요.

5.3 docs/ — 자리를 나누는 네 개의 폴더

번호가 붙은 네 폴더

```
docs/
  10.input/      ← 외부에서 받은 원본 자료 (수정하지 않음)
  20.working/    ← 작업 중인 초안·중간 산출물
  30.output/     ← 최종 결과물 (공유·배포용)
  90.reference/  ← 참고 문서·링크·캡처
```

부엌으로 비유하면 이렇습니다.

폴더	부엌에서	1주차에 들어갈 것
`10.input`	장 봐 온 **재료**	가게 사진, 받은 자료
`20.working`	손질 중인 **도마 위**	`브레인스토밍.md` · `PRD.md`
`30.output`	손님에게 나갈 **접시**	`시안-8-1-….html` 네 장
`90.reference`	벽에 붙인 **레시피**	`참고링크.md`

> 💡 **왜 폴더 이름 앞에 번호를 붙이나**
>
> 두 가지 이유입니다.
>
> ① **순서가 눈에 보입니다.** 폴더는 이름순으로 정렬됩니다. 번호를 붙이면 재료 → 작업 → 결과 순서대로 늘어섭니다. 흐름이 곧 목록 순서가 됩니다.
>
> ② **원본을 지킵니다.** `10.input` 에는 "수정하지 않음" 이라는 규칙이 붙어 있습니다. Claude가 받은 사진을 줄이거나 자료를 고쳐야 할 때, **원본은 두고 `20.working` 에 사본을 만들어** 고칩니다. 망쳐도 원본이 남습니다. 2장의 "되돌리기" 를 폴더 구조로 미리 막아두는 셈입니다.
>
> 10 · 20 · 30 사이를 비워 둔 것은 나중에 `15.` 같은 폴더를 끼워 넣을 자리입니다. 90은 "맨 뒤에 두는 참고" 라는 뜻입니다.

Claude가 만들고, Claude가 쓴다

이 네 폴더는 **사람보다 Claude를 위한 것**에 가깝습니다. 1주차 내내 Claude는 정한 것을 `20.working/` 에, 시안을 `30.output/` 에, 찾아본 링크를 `90.reference/` 에 알아서 저장합니다. 폴더가 없으면 Claude가 만듭니다.

그리고 폴더마다 **무엇을 넣는 곳인지**는 따로 설명 파일을 두지 않고 `CLAUDE.md` 한 곳에 적어 둡니다. 다음 절이 그 이야기입니다.

5.4 CLAUDE.md — 매번 설명하지 않는 법

매번 같은 말을 하게 된다

3장에서 봤듯이, **새 대화를 열면 Claude는 이전 대화를 모릅니다.** 공책이 새것이 됩니다.

그러면 대화를 열 때마다 이런 말을 반복하게 됩니다.

> "답은 한국어로 해줘. 원본 폴더는 건드리지 마. 결과물은 30.output 에 저장해. 어려운 말 쓰지 마..."

매번 하기 번거롭고, 한 번 빼먹으면 Claude는 그 규칙을 모르고 일합니다.

한 번 적어두면 매번 읽는다

> 📖 **CLAUDE.md(클로드 엠디)**
>
> 작업 폴더 맨 위에 두는 **규칙 문서.** Claude Code는 **대화를 시작할 때마다 이 파일을 먼저 읽고** 시작합니다. 여기 적어 둔 것은 말하지 않아도 지킵니다.
>
> 회사로 치면 **신입에게 첫날 건네는 업무 안내서**입니다. 매번 말로 가르치지 않아도, 들어올 때마다 그걸 읽고 자리에 앉습니다.

파일 이름 끝의 `.md` 는 **마크다운**이라는 글 형식이라는 뜻입니다.

> 📖 **마크다운(Markdown, .md)**
>
> 메모장으로 쓰는 글에 **간단한 기호로 모양을 붙이는 약속.** 줄 맨 앞에 `#` 을 쓰면 제목, `-` 를 쓰면 목록, 글자를 `**` 로 감싸면 굵게 됩니다.
>
> 사람은 메모장으로 열어도 읽을 수 있고, AI는 구조를 정확히 알아봅니다. 그래서 **사람과 AI가 같이 읽는 문서**를 쓸 때 가장 많이 씁니다. 1주차에 만드는 `브레인스토밍.md` · `PRD.md` 도 전부 마크다운입니다.

setup-prompt가 만든 CLAUDE.md 안에는

열어 보면 섹션이 네 개 있습니다.

```
# (폴더 이름)
TODO: 프로젝트 설명

## 프로젝트 개요            ← 비어 있음. 나중에 채운다
## 폴더 구조               ← 5.3절의 docs/ 네 폴더
## 작업 규칙 (Claude가 따라야 할 규칙)
### Skill 구성 규칙        ← 다음 절
## 자주 쓰는 명령           ← 비어 있음
```

맨 아래에는 **"이 문서는 Claude가 매 대화 시작 시 자동으로 읽는다"** 는 안내가 한 줄 붙습니다. 이 한 줄은 Claude보다 **사람**을 위한 것입니다. 석 달 뒤 이 파일을 연 내가 "이게 뭐였지?" 하지 않게요.

비어 있는 칸은 **비어 있는 게 정상**입니다. 지금은 뭘 만들지도 안 정했습니다. 1주차가 끝나고 PRD가 생기면 그때 채울 게 생깁니다.

> ⚠️ **CLAUDE.md에 다 적으면 안 되나요?**
>
> 안 됩니다. `CLAUDE.md` 는 **매번** 읽힙니다. 3장의 공책(컨텍스트)을 매번 그만큼 차지한다는 뜻입니다. 길어질수록 정작 대화할 공간이 줄어듭니다.
>
> 그래서 `CLAUDE.md` 에는 **항상 지켜야 하는 짧은 규칙**만 둡니다. 만들 것에 대한 긴 내용은 `PRD.md` 같은 **따로 된 문서**에 두고, 필요할 때 읽게 합니다.

5.5 Skill — 반복 작업에 이름 붙여 저장하기

붙여넣기가 귀찮아지는 순간

방금 붙여넣은 `setup-prompt` 는 꽤 깁니다. 새 폴더를 만들 때마다 이걸 찾아서 복사하고 붙여넣어야 합니다. 두세 번 하면 귀찮아집니다.

이럴 때 쓰는 것이 **Skill**입니다.

> 📖 **Skill(스킬)**
>
> 자주 시키는 **작업의 순서를 글로 적어 저장해 두고, 이름만 불러 쓰는 것.** 입력창에 `/` 와 이름을 치면(`/init-workspace`) 적어 둔 절차대로 일합니다.
>
> 회사로 치면 **업무 매뉴얼**입니다. "새 프로젝트 폴더 만들기" 매뉴얼을 한 번 써 두면, 다음부터는 "매뉴얼대로 해" 한마디면 됩니다.

`setup-prompt.md` 에도 이렇게 적혀 있습니다.

> 더 편하게 쓰려면 `/init-workspace` Skill로 등록되어 있습니다.

같은 프롬프트를 Skill로 만들어 둔 것이 `/init-workspace` 입니다. 긴 상자를 붙여넣는 대신 **`/init-workspace` 한 줄**이면 같은 작업 공간이 생깁니다.

실제로 돌려 보면

강의 자료 폴더(`month-claude-2-week1`)를 열고 `/init-workspace` 를 친 화면입니다. Skill은 붙여넣기 프롬프트보다 **한 걸음 더** 갑니다. 폴더 안의 파일을 먼저 훑어본 뒤, **이 폴더가 무슨 일을 하는 곳인지 물어봅니다.**

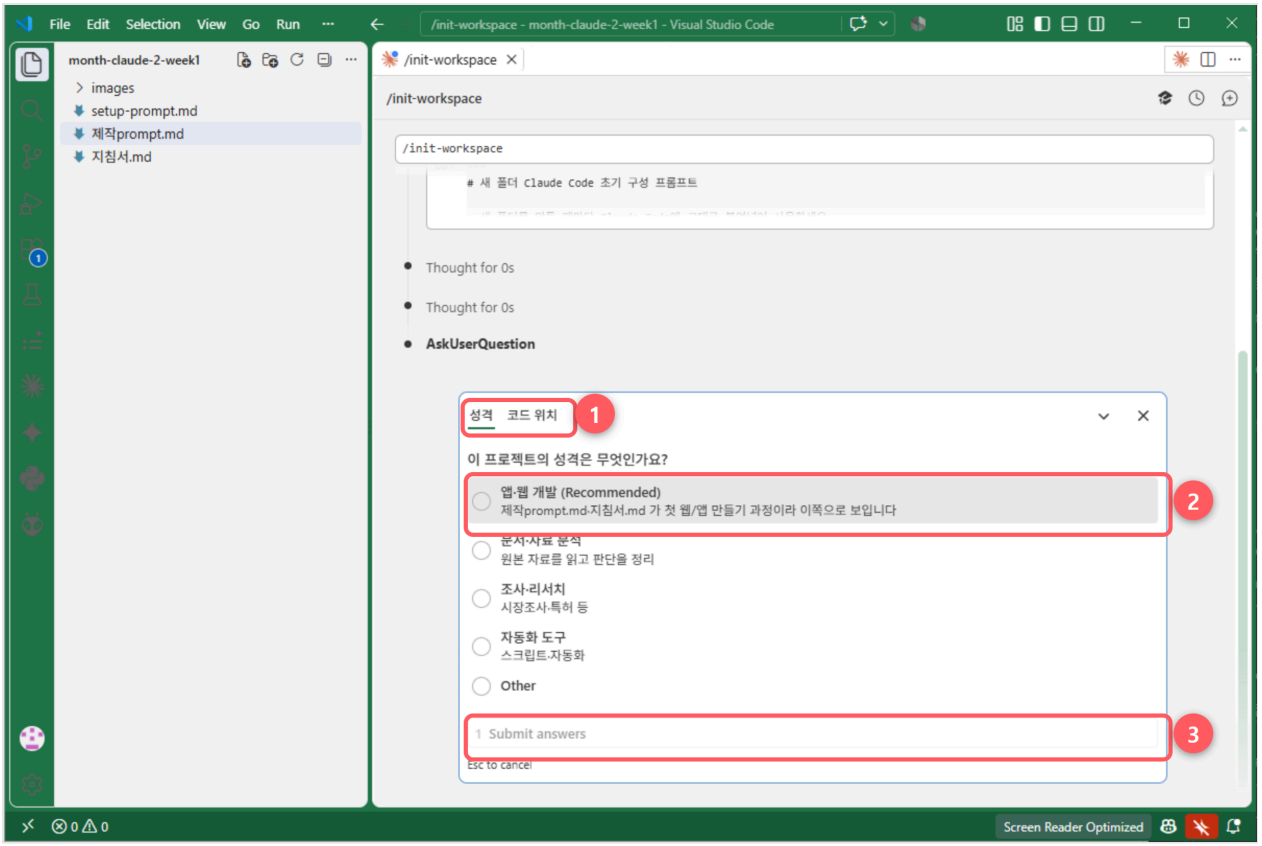

그림 5-1 /init-workspace 가 시작하자마자 묻는 질문. ① 질문이 두 개(성격 · 코드 위치)라 탭으로 나뉜다 ② 폴더 안의 제작prompt.md · 지침서.md 를 보고 「앱·웹 개발」을 추천으로 올려 두었다 ③ 고른 뒤 Submit answers

질문에 **보기가 붙어 있고, 추천(Recommended)이 하나 짚혀 있다**는 점을 보세요. 6장에서 볼 "열린 질문 대신 보기를 준다" 는 규칙이 이 Skill에도 들어 있습니다.
답하고 나면 폴더를 만들고, 무엇이 생겼는지와 **CLAUDE.md에 무엇을 적었는지** 알려 줍니다.

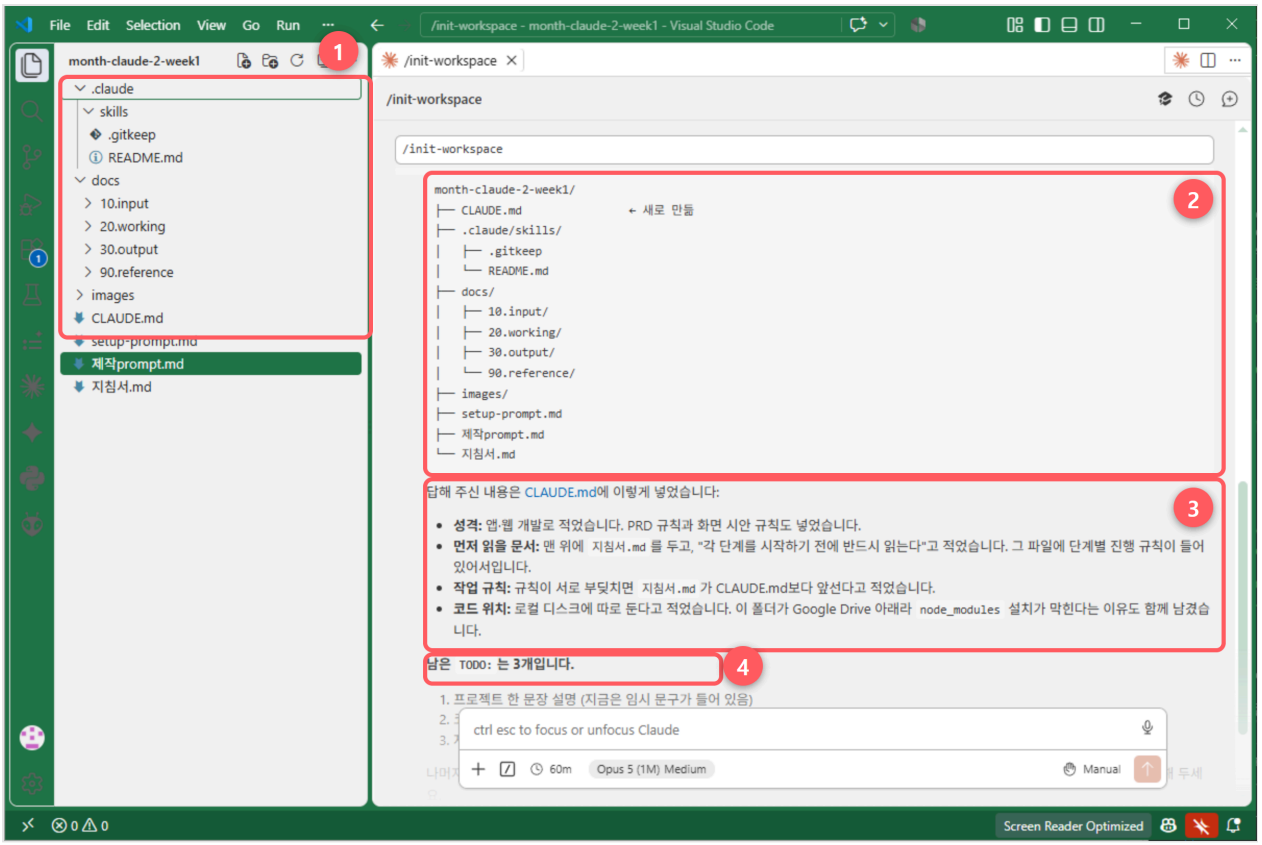

그림 5-2 /init-workspace 가 끝난 화면. ① 왼쪽 탐색기에 .claude/skills · docs 네 폴더 · CLAUDE.md 가 새로 생겼다 ② 만든 것의 파일 트리 — 원래 있던 강의 자료는 그대로 ③ 답한 내용을 CLAUDE.md 의 성격 · 먼저 읽을 문서 · 작업 규칙 · 코드 위치에 적었다는 요약 ④ 아직 채우지 못한 TODO 개수

③을 한 줄씩 보면, 5.4절에서 말한 "CLAUDE.md에 적는 것" 이 실제로 무엇인지 보입니다.

CLAUDE.md 에 적힌 것	왜 적었나
성격 — 앱·웹 개발	대화를 열 때마다 "이 폴더가 뭐 하는 곳인지" 다시 설명하지 않으려고
먼저 읽을 문서 — `지침서.md`	단계를 시작하기 전에 반드시 읽게 하려고
작업 규칙 — 부딪히면 `지침서.md` 가 앞선다	규칙이 두 곳에 있을 때 무엇을 따를지 정해 두려고
코드 위치 — 로컬 디스크에 따로	이 폴더가 구글드라이브 안이라 앱 도구 설치가 막히기 때문

마지막 줄이 특히 좋은 예입니다. **"코드는 로컬 디스크에 둔다"** 는 결정만이 아니라 **"구글드라이브 안이라서"** 라는 **이유**까지 남겼습니다. 석 달 뒤의 나도, 새 대화의 Claude도 이 한 줄을 읽으면 같은 실수를 하지 않습니다. 이 책이 계속 말하는 **"생각을 기록한다"** 가 작업 공간을 꾸미는 첫날부터 시작됩니다.

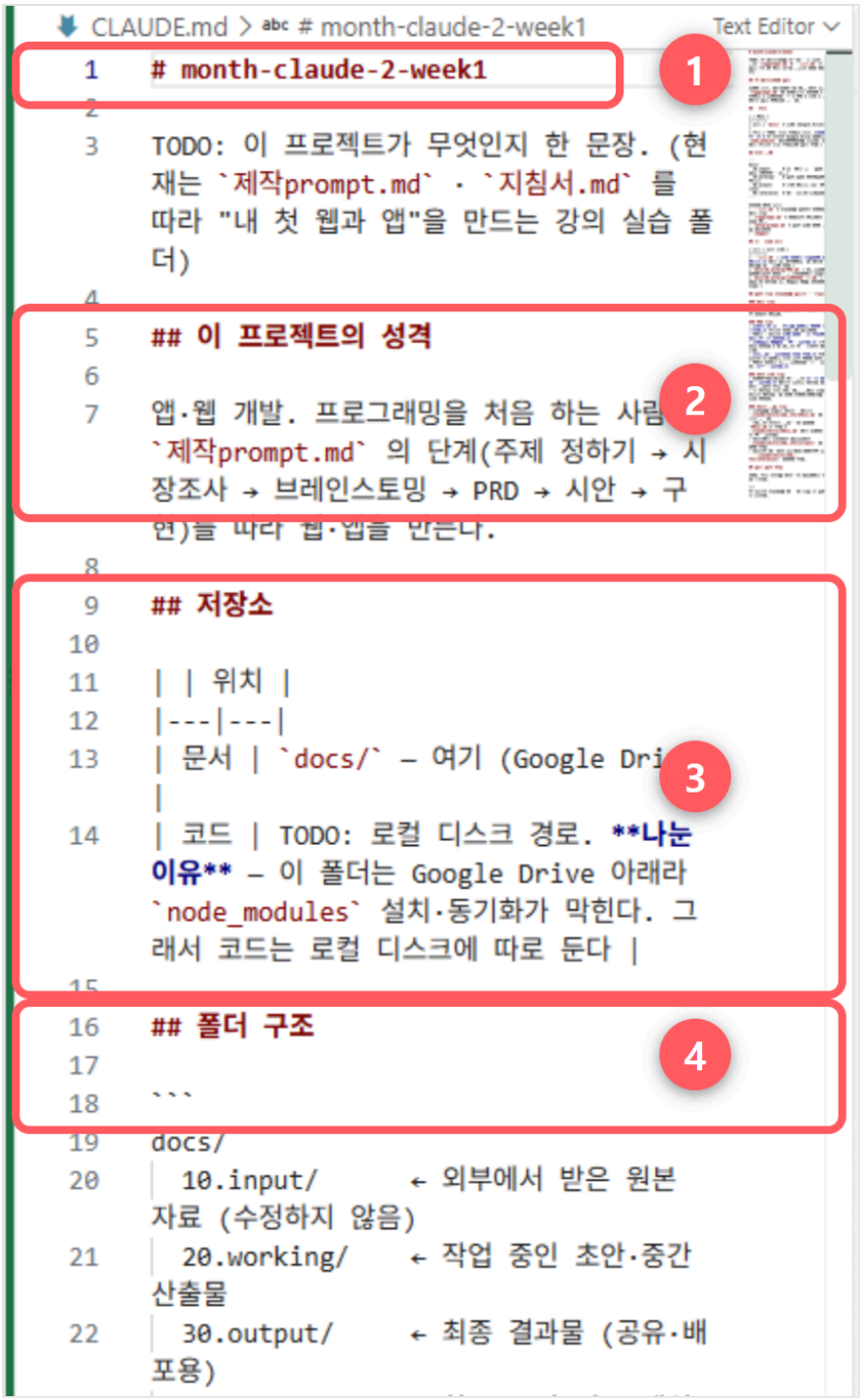

그림 5-3 /init-workspace 가 쓴 CLAUDE.md 실물. ① 맨 위 제목은 폴더 이름 ②「이 프로젝트의 성격」— 답한 내용이 문장으로 ③「저장소」— 문서는 여기, 코드는 로컬 디스크. **나눈 이유**까지 적혀 있다 ④「폴더 구조」— docs 네 폴더와 각각의 쓰임

③을 보면 표 한 칸에 **TODO** 가 그대로 남아 있습니다. 로컬 디스크의 실제 경로는 아직 안 정했기 때문입니다. **정하지 않은 것을 정한 척 채우지 않는 것** — 이것도 기록의 일부입니다. 남은 TODO는 **비어 있는 게 정상**입니다. 프로젝트 한 줄 설명처럼, 아직 정하지 않은 것은 정한 척 채우지 않고 TODO로 남겨 둡니다.

Skill은 어디에 사나

Skill은 **폴더 하나 + 그 안의 `SKILL.md` 파일 하나**로 이루어집니다. `CLAUDE.md` 의「Skill 구성 규칙」이 바로 이 모양을 정해 둔 것입니다.

```
.claude/skills/
└── init-workspace/          ← 스킬 이름으로 된 폴더
    ├── SKILL.md             ← 절차를 적은 본문 (필수)
    ├── scripts/             ← 스킬이 쓰는 프로그램 (있으면)
    └── references/          ← 스킬이 참고하는 문서·예시 (있으면)
```

> ⚠ **`SKILL.md` 를 폴더 없이 두면 인식이 안 됩니다**
>
> `.claude/skills/SKILL.md` 처럼 **평평하게** 두면 안 됩니다. 반드시 `.claude/skills/{스킬이름}/SKILL.md` — **이름 폴더 안에** 둡니다. 규칙에 이 한 줄이 굳이 들어가 있는 이유는, 실제로 이렇게 만들어 놓고 "Skill이 안 불려요" 하는 일이 잦기 때문입니다.

Skill을 두는 곳은 두 군데입니다.

위치	어디서 쓰나
작업 폴더의 `.claude/skills/`	**그 폴더에서만**
내 PC의 사용자 폴더 `~/.claude/skills/`	**어느 폴더에서나**

`/init-workspace` 는 **새 폴더에서** 쓰는 Skill이니, 사용자 폴더에 둬야 어디서든 불립니다. 설치는 강의 자료 폴더에서 복사 한 번이면 됩니다.

Windows (PowerShell)

```
New-Item -ItemType Directory -Force "$env:USERPROFILE\.claude\skills"
Copy-Item -Recurse -Force .claude\skills\init-workspace `
  "$env:USERPROFILE\.claude\skills\"
```

Mac

```
mkdir -p ~/.claude/skills
cp -r .claude/skills/init-workspace ~/.claude/skills/
```

복사한 뒤에는 **Claude Code를 껐다 켭니다.** Skill 목록은 시작할 때 한 번만 읽기 때문입니다. 새 폴더를 열고 `/init-workspace` 를 쳐서 `CLAUDE.md` 가 생기면 성공입니다.

CLAUDE.md와 Skill은 무엇이 다른가

둘 다 "글로 적어 두면 Claude가 따른다" 는 점이 같아서 헷갈립니다. 한 줄로 가르면 이렇습니다.

	`CLAUDE.md`	Skill
언제	**항상** — 대화를 열 때마다	**부를 때만** — `/이름` 을 쳤을 때
무엇	배경 규칙	작업 절차
비유	사내 규정 ("복장은 단정하게")	업무 매뉴얼 ("견적서 작성법")
예	"원본 폴더는 수정하지 않는다"	"새 폴더를 작업 공간으로 꾸민다"

> 💡 **1주차의 `지침서.md` 는 무엇에 가까울까**
>
> 2장에서 본 `지침서.md` 는 **"이 파일 먼저 읽어"** 라고 해야 읽힙니다. 항상 읽히는 `CLAUDE.md` 도 아니고, `/이름` 으로 부르는 Skill도 아닙니다.
>
> 강의에서 여덟 명이 **어느 폴더에서 시작하든** 똑같이 진행되게 하려고, 일부러 한 파일에 담아 첫 상자에서 읽히는 방식을 골랐습니다. 같은 내용을 Skill로 만들어도 됩니다. **어디에 적느냐보다, 적어 두었느냐가 먼저입니다.**

5.6 이 세 가지가 "생각을 기록하는 자리" 다

작가의 말에서 이 책의 핵심을 이렇게 말했습니다.

> **만든 사람의 생각을 기록해 두고, 그 기록을 지키면서 고친다.**

오늘 만든 세 가지가 그 기록이 들어갈 자리입니다.

```
CLAUDE.md          항상 지킬 규칙        "원본은 건드리지 않는다"
docs/20.working/   만들 것에 대한 생각    브레인스토밍.md · PRD.md (7·9장)
.claude/skills/    반복할 절차           /init-workspace
```

아직 셋 다 거의 비어 있습니다. 1주차가 끝날 때쯤 `20.working/` 에 **뭘 만들지 · 뭘 안 만들지 · 어떻게 생겼는지** 가 채워집니다. 그러면 2주차에 "만들어줘" 라고 할 때, 3주차에 "수정해줘" 라고 할 때, Claude는 **그걸 먼저 읽고** 일합니다.

5.7 안 될 때

파일 트리가 프롬프트와 다르게 생겼다

Claude가 규칙을 조금 다르게 알아들었을 수 있습니다. 이렇게 치세요.

```
setup-prompt.md 의 1~3번 내용이랑 지금 폴더를 비교해서
다른 부분만 고쳐줘
```

`/init-workspace` 를 쳤는데 목록에 안 나온다

1. Skill 폴더가 **사용자 폴더**에 들어갔는지 — `~/.claude/skills/init-workspace/SKILL.md`
2. `SKILL.md` 가 **이름 폴더 안에** 있는지 (5.5절 ⚠️)
3. **Claude Code를 껐다 켰는지** — 켜져 있는 동안 넣은 Skill은 모릅니다

docs/ 아래 폴더에 README가 잔뜩 생겼다

프롬프트에 **"각 폴더에는 README.md를 만들지 않는다"** 고 적혀 있습니다. 폴더 용도는 `CLAUDE.md` 에 이미 있으니 겹칠 필요가 없어서입니다. 생겼다면 `docs 아래 README 파일은 지워줘` 라고 하면 됩니다. 문제가 되는 건 아닙니다.

5.8 정리

만든 것	하는 일	비유
`docs/10·20·30·90`	원본 · 작업 · 결과 · 참고의 자리를 나눈다	재료 · 도마 · 접시 · 레시피
`CLAUDE.md`	**항상** 읽히는 규칙	신입 업무 안내서
`.claude/skills/`	**부를 때** 실행되는 절차	업무 매뉴얼

이 장에서 나온 말

말	쉬운 뜻
작업 공간(workspace)	Claude와 일할 수 있게 꾸며 둔 폴더 한 개
파일 트리	폴더 안의 내용을 나뭇가지 모양으로 그린 목록
숨김 폴더	이름이 `.` 으로 시작해 탐색기에서 안 보일 수 있는 폴더
CLAUDE.md	대화 시작마다 Claude가 먼저 읽는 규칙 문서
마크다운(.md)	`#` · `-` · `**` 같은 기호로 모양을 붙이는 글 형식
Skill	작업 절차를 적어 두고 `/이름` 으로 부르는 것
슬래시 명령	입력창에 `/` 로 시작해 치는 명령. Skill을 부를 때 쓴다
사용자 폴더	내 PC 계정의 기본 폴더. `~` 또는 `%USERPROFILE%` 로 적는다

방이 준비됐습니다. 다음 장에서는 이 방에서 **1주차 두 시간 동안 무엇을 어떤 순서로 하는지** 지도를 먼저 펼칩니다.

06장 — 1주차 지도, 왜 코드보다 설계가 먼저인가

> 초고 v0.1 (2026-09-14)

6.1 오늘은 코드를 한 줄도 쓰지 않는다

2기 1주차 강의 안내는 이렇게 시작합니다.

> 오늘은 웹과 앱을 실제로 코딩하지 않고 **시안까지만** 만듭니다. 끝날 때면 **뭘 만들지 · 뭘 안 만들지 · 어떻게 생겼는지**가 전부 정해져 있습니다. 다음 주에는 그 문서를 보면서 그대로 만듭니다.

두 시간짜리 수업에서 코드를 한 줄도 안 쓴다니, 처음 들으면 이상합니다. Claude에게 시키면 20분이면 사이트가 나온다고 했는데요.

이 장은 **왜 그렇게 하는지**, 그리고 **두 시간 동안 무엇을 어떤 순서로 하는지** 지도를 먼저 펼칩니다. 7장부터 11장까지가 이 지도의 한 칸씩입니다.

6.2 두 가지 낱말부터

1주차 내내 나오는 낱말 두 개를 먼저 잡고 갑니다.

> 📖 **설계(design)**
>
> 만들기 전에 **무엇을, 누구를 위해, 어디까지 만들지 정해서 적어 두는 일.** 집을 짓기 전에 그리는 **도면**과 같습니다. 도면 없이 벽돌부터 쌓으면, 다 쌓고 나서 "문이 어디 있지?" 가 됩니다.
>
> 그렇게 적어 둔 문서를 **설계 문서**라고 부릅니다. 1주차에 만드는 설계 문서의 이름이 **PRD**입니다(9장).

> 📖 **웹과 앱 — 관리하는 쪽과 신청하는 쪽**
>
> 2기에서 만드는 것은 한 벌입니다. **같은 일을 두 사람이 서로 다른 화면으로 봅니다.**

> **웹(web)** — 컴퓨터 브라우저로 여는 화면. 여기서는 **관리하는 사람**(사장님, 강사, 모임장)이 명단을 보고 공지를 올립니다. **앱(app)** — 폰에 설치해서 여는 화면. 여기서는 **신청하는 사람**(손님, 수강생, 회원)이 시간표를 보고 신청합니다.
>
> 한쪽에서 누르면 다른 쪽에 바로 뜹니다. 손님이 앱에서 신청하면 사장님 웹 명단에 이름이 올라갑니다.

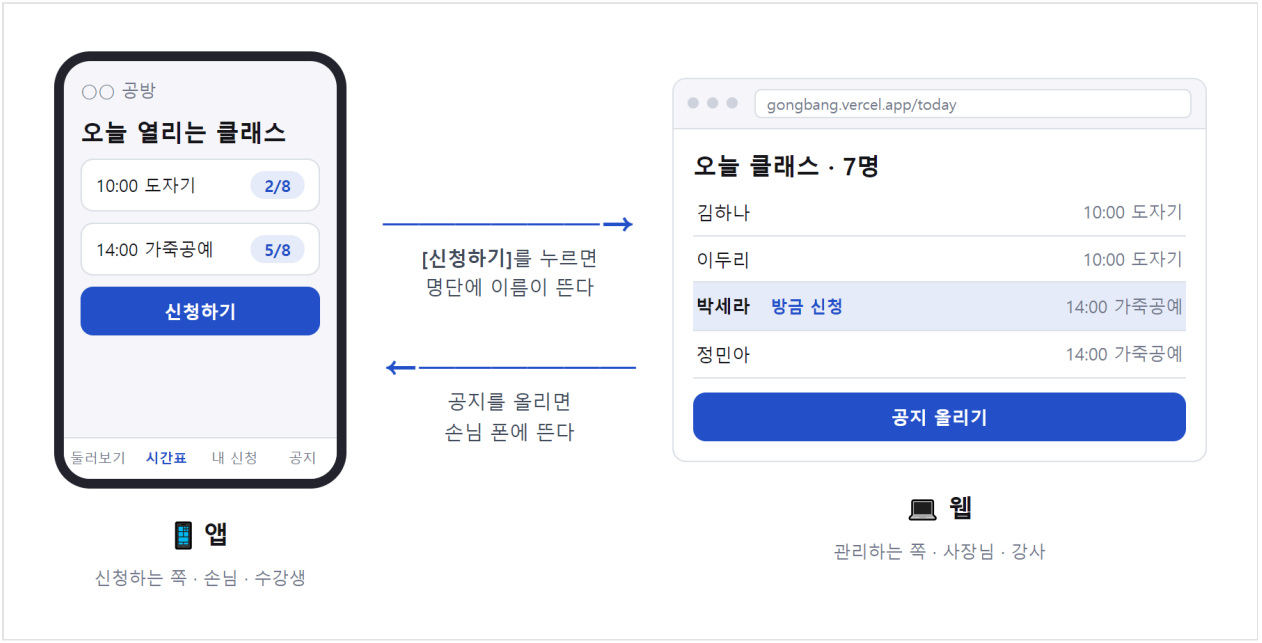

그림 6-1 같은 일을 두 화면으로 본다. 손님이 앱에서 [신청하기]를 누르면 사장님 웹 명단에 이름이 뜨고, 사장님이 웹에서 공지를 올리면 손님 폰에 뜬다

6.3 지도 — 0단계부터 9단계까지

1주차 자료 `제작prompt.md` 에는 📋 상자가 단계별로 들어 있습니다. 한 상자를 붙여넣으면 한 단계가 돌아가고, 끝나면 Claude가 **다음 상자를 붙여넣으라고** 알려 줍니다.

덩어리	단계	하는 일	시간	끝나면 남는 것
① **정하기** 내 얘기	0	인사하기	30초	—
	1	무엇을 만들지 정하기	3분	브레인스토밍.md
	2	어떻게 쓰이는지 이야기하기	5분	
	3	**무엇을 안 만들지 정하기**	7분	
	4	정한 것을 한 장으로 모으기	5분	
② **견주기** 남의 것	5	비슷한 서비스 3개의 기능표 보기	10분	참고링크.md
③ **갈림길** 둘 중 하나	6	갈림길 네 개 고르기	7분	브레인스토밍.md 에 덧붙임
④ **굳히기**	7	설계 문서(PRD) 쓰기	7분	PRD.md v0.1
⑤ **보고 고쳐 쓰기**	8	**화면 시안 고르기 — 네 번**	40분	시안 네 장 (아티팩트)
	9	설계 문서 고쳐 쓰기	7분	PRD.md v0.2 + 변경 이력

화면(8단계)은 맨 나중이다. 그 앞 40분 넘게 **말로 정하고 글로 남긴다.** 순서는 바꾸지 않는다.

그림 6-2 1주차 지도. 열 단계를 다섯 덩어리로 묶었다. 8단계 화면 고르기가 40분으로 가장 길지만, 그 앞에서 40분 넘게 말로 정하고 글로 남긴다

시간을 더해 보면 한 가지가 보입니다. **8단계 화면 고르기가 40분으로 제일 깁니다.** 그런데 거기 가기까지 **여섯 단계, 40분 넘게** 말로 정하는 데 씁니다. 화면을 보기 전에 정할 것이 그만큼 많다는 뜻입니다.

6.4 왜 화면이 맨 나중인가

1단계에서 수강생이 "화면 좀 먼저 보여줘" 라고 하면, Claude는 이렇게 답하도록 정해져 있습니다.

> "화면은 나중에 3가지 안으로 만들어 보여드릴게요. 지금 그리면 그 그림에 갇혀버려서, 먼저 정할 것부터 정하겠습니다."

> 💡 **그림이 먼저 나오면, 그림이 정답 행세를 한다**
>
> 화면을 먼저 보면 사람은 그 화면을 **기준으로** 생각하기 시작합니다. "여기에 버튼 하나 더", "이 칸은 좀 넓게". 그런데 정작 **그 화면이 필요한지**, **그 칸에 뭐가 들어가야 하는지**는 아무도 정하지 않았습니다.
>
> 정하는 과정이 통째로 날아가고, **처음 나온 그림이 답이 됩니다.** 그 그림은 AI가 "그럴듯하게" 고른 것일 뿐인데요(3장).

> 코드도 같습니다. 코드가 먼저 나가면 **그 코드가 답 노릇을 해버립니다.** 그래서 1~7단계에서는 화면도, 표도, 코드도 꺼내지 않습니다.

작가의 말의 "딸깍" 이 바로 이것입니다. 딸깍 하면 화면이 먼저 나옵니다. 그리고 그 화면에는 **만든 사람의 생각이 없습니다.** 생각은 나중에 붙일 수가 없습니다. 먼저 정해야 화면에 담깁니다.

6.5 왜 이 순서인가

단계 순서에도 이유가 있습니다.

```
내 것을 정해 모으고(1~4) → 남들과 견주고(5) → 갈림길을 펼쳐 고르고(6) → 굳힌다(7)
```

만약	생기는 일
조사(5)를 먼저 하면	남의 기능 목록을 **베낍니다.** 경쟁 서비스에 있는 건 다 필요해 보입니다
갈림길(6)을 조사 앞에 두면	무엇이 갈림길인지 **안 보입니다.** 남들이 어떻게 갈렸는지 봐야 내 갈림길이 보입니다
PRD(7)를 화면(8) 뒤에 쓰면	화면이 PRD를 대신합니다. 6.4절의 "그림이 정답 행세"
화면(8)을 보고도 PRD를 안 고치면(9)	문서와 화면이 **어긋난 채로** 2주차에 들어갑니다

그래서 **순서를 바꾸지 않습니다.**

6.6 Claude가 지키는 약속들

1주차의 Claude는 평소보다 **조심스럽게** 움직입니다. 2장에서 본 `지침서.md` 에 그렇게 정해 두었기 때문입니다. 알고 있으면 "왜 이렇게 답답하게 굴지?" 싶은 순간이 이해됩니다.

한 번에 한 단계만

붙여넣은 상자가 몇 단계인지 확인하고, **그 단계만** 합니다. 결과가 나오면 바로 멈추고 다음 상자를 기다립니다. "그다음은?" 이라고 물어도 **다음 상자를 붙여넣으라고만** 합니다.

질문 수에 상한이 있다

수강생은 **답을 많이 못 합니다.** 다섯 번만 물어도 지칩니다. 그래서 단계마다 물을 수 있는 수가 정해져 있습니다.

단계	0	1	2	3	4	5	6	7	8	9
질문 최대	0	1	3	3	**0**	3	4	2	회차마다 1	**0**

4단계와 9단계는 **질문이 0개**입니다. 수강생이 쉬어가는 자리입니다. 상한에 닿으면 남은 건 Claude가 정하고 이렇게 말합니다.

> "나머지는 제가 무난한 쪽으로 정해뒀습니다. 하다가 아니다 싶으면 그때 바꿔요."

질문에는 반드시 보기가 붙는다

"어떻게 하고 싶으세요?" 같은 **열린 질문**은 처음 하는 사람을 얼어붙게 합니다. 그래서 모든 질문에 **번호 보기 3~5개**를 붙이고, 마지막 번호는 항상 **직접 말하기** 입니다. 그리고 하나를 짚어 줍니다 — "잘 모르시겠으면 2번이 제일 무난합니다."

엉뚱하게 답해도 지적하지 않는다

이렇게 답하면	Claude는
질문과 다른 얘기	쓸 조각 하나를 집어 인정하고, 보기를 다시 준다
너무 큰 것	거절하지 않고 **잘라 준다.** 잘라낸 쪽이 핵심이라고 말해 준다
못 알아들음	같은 말을 반복하지 않고 **더 쉽게** 묻는다
대답이 없음	하나를 골라 두고 "나중에 바꿀 수 있어요" 하고 넘어간다

세 번 물어도 안 풀리면 붙잡지 않습니다. Claude가 정하고 진행합니다.

> 💡 **모르겠으면 "모르겠어" 가 정답입니다**
>
> 1주차의 모든 질문에는 **"모르겠다" 를 골랐을 때 어디로 가는지**가 미리 정해져 있습니다. 대부분 **제일 단순하고 무난한 쪽**입니다.
>
> 억지로 짜내서 답하면 그 답이 설계 문서에 들어갑니다. 모르면 모른다고 하는 게 문서를 더 정확하게 만듭니다. 그리고 전부 **나중에 바꿀 수 있습니다.**

6.7 만들 수 있는 것의 선

1주차에 무엇을 고르든 **넘지 않는 선**이 있습니다.

만든다	안 만든다 (이유)
관리하는 사람이 보는 **웹 화면**	결제 · 정산 — 사업자 등록과 심사가 필요
신청하는 사람이 쓰는 **앱 화면**	문자 · 카톡 발송 — 승인 절차와 건당 요금
로그인 · 목록 · 신청 · 취소 · 출석 · 공지	지도 · 위치 · 배달 추적 — 별도 가입과 요금
한쪽에서 누르면 **다른 쪽에 바로 뜨는 것**	1:1 채팅 — 만들면 끝이 없다
	카카오 · 네이버 로그인 — 심사 필요

판단 기준은 한 줄입니다.

> **사람 · 시간 · 신청 · 명단 — 이 네 가지로 설명되면 만들 수 있다.**

선 밖의 것을 원해도 **통째로 거절하지 않습니다.** 되는 부분만 잘라서 제안합니다. 배달 앱을 만들고 싶다고 하면 "주문 받고 픽업 시간 정하기" 로, 쇼핑몰이면 "상품 보고 신청 받기" 로 자릅니다.
7장에서 자세히 봅니다.

6.8 정리

- 1주차는 **코드 없이** 정하고, 기록하고, 화면을 고릅니다
- **화면이 먼저 나오면 정답 행세를 합니다.** 그래서 화면은 8단계입니다
- 순서는 **정하기 → 견주기 → 갈림길 → 굳히기 → 보고 고쳐 쓰기.** 바꾸지 않습니다
- 모르면 **"모르겠어"** — 무난한 쪽으로 정해지고, 나중에 바꿀 수 있습니다

이 장에서 나온 말

말	쉬운 뜻
설계	만들기 전에 무엇을·누구를 위해·어디까지 만들지 정해 적는 일. 집의 도면
설계 문서	설계를 적어 둔 문서. 1주차에서는 `PRD.md`
웹(관리하는 쪽)	컴퓨터 브라우저로 여는 화면. 사장님·강사가 명단을 본다
앱(신청하는 쪽)	폰으로 여는 화면. 손님·수강생이 시간표를 보고 신청한다
산출물	한 단계가 끝나면 남는 결과 파일. `브레인스토밍.md` · `PRD.md` 등
시안	실제로 만들기 전에 "이렇게 생길 것" 을 그려 본 화면. 10장
열린 질문 / 보기 질문	답을 자유롭게 쓰는 질문 / 번호 중에 고르는 질문

다음 장부터 지도의 첫 칸, **무엇을 만들고 무엇을 안 만들지** 정하는 1~3단계로 들어갑니다.

07장 — 무엇을 만들고, 무엇을 안 만들까

> 초고 v0.1 (2026-09-14) · 1주차 1~3단계

7.1 이 장에서 하는 일

지도(6장)의 첫 덩어리, **정하기**입니다. 세 단계를 거칩니다.

단계	정하는 것	질문 수
1단계	**주제 한 줄** — 누구를 위한 무엇	1개
2단계	**어떻게 쓰이는지** — 지금 방식 · 불편한 순간 · 사람 수	3개
3단계	**무엇을 안 만들지** ⭐	3개

세 단계에서 정한 것은 전부 한 파일에 쌓입니다.

> 📖 **브레인스토밍(brainstorming)**
>
> 정답을 찾기 전에 **생각을 일단 다 꺼내 늘어놓는 일.** 머릿속 폭풍(storm)이라는 뜻입니다.
> 좋은 생각과 엉뚱한 생각을 가리지 않고 꺼낸 다음, 그중에서 고릅니다.
>
> 1주차에서는 이렇게 꺼내고 고른 과정을 `docs/20.working/브레인스토밍.md` 에 적습니다. **아직 안 정한 것까지** 담는 메모장입니다.

7.2 1단계 — 무엇을 만들지 정하기 (3분)

따라 하기

⌨️ `제작prompt.md` 의 **1단계 상자**를 붙여넣습니다.

```
나는 프로그래밍을 처음 해. 개발자가 아니야.
웹이랑 앱을 하나 만들어보고 싶은데, 뭘 만들지는 아직 안 정했어.

혼자서는 뭐가 가능한지도 모르겠으니까,
니가 물어봐 주면서 같이 정해줘.
```

상자에 **"아직 안 정했어"** 가 들어 있어서, Claude는 되묻지 않고 바로 예시를 보여줍니다.

```
1  공방 원데이 클래스 신청받기
2  필라테스·PT 수업 예약
3  과외·스터디 출석 관리
4  독서모임·동호회 정기모임 참석 확인
5  네일·헤어 예약 잡기
6  반려동물 미용 예약
7  사내 회의실·장비 빌려주기
8  학원 상담·보강 신청
9  공동구매 신청받기
10 행사·모임 참가 접수
11 수리·AS 접수받기
12 체험단·설문 참가 신청
13 위에 없음 – 직접 말하기
```

번호만 말해도 되고 말로 해도 됩니다. 잘 모르겠으면 **1번이 제일 만들기 쉽습니다.**

고르면 Claude가 **딱 두 줄**로 정리합니다.

```
나는 (누구)를 위한 (무엇)을 만든다
관리하는 쪽 – (누구) / 신청하는 쪽 – (누구)
```

"맞아" 라고 하면 `docs/20.working/브레인스토밍.md` 가 생깁니다. 탐색기에 이 파일이 보이면 성공입니다.

> ⚠ **1단계에서 이것저것 묻지 않아요?**

> 안 묻습니다. 1단계의 질문은 **"어떤 웹과 앱을 만들어 보고 싶으신가요?"** 하나뿐입니다. 몇 명인지, 뭐가 불편한지는 전부 2단계입니다. 처음부터 캐물으면 지쳐서 2단계까지 못 오기 때문입니다. **큰 틀만 잡으면 1단계는 끝입니다.**

열두 개는 이름만 다르고 뼈대가 같다

예시 열두 개는 달라 보이지만, **만드는 것은 똑같습니다.** 전부 네 덩어리로 설명됩니다.

뼈대	1 공방 원데이 클래스	5 네일·헤어	9 공동구매
사람 신청하는 쪽 명단	수강생	손님	신청자
열리는 칸 신청이 들어갈 자리 ← 여기만 다르다	클래스 2/8	예약 시간 예약 가능	공구 건 수량
신청 누가 어느 자리에	김하나 → 10시 도자기	이두리 → 14:00	박세라 → 사과 2박스
알림판 여럿에게 한 번에	재료비 안내	9/20 휴무	입금 마감 안내

그래서 서로 다른 주제를 골라도 1~7단계는 똑같이 간다. 화면 모양이 갈리는 건 8단계부터.

그림 7-1 열두 주제는 이름만 다르고 뼈대가 같다. 공방·네일·공동구매 모두 사람·열리는 칸·신청·알림판 네 덩어리로 설명되고, 주제마다 다른 건 「열리는 칸」 뿐이다

그래서 강의에서 여덟 명이 서로 다른 주제를 골라도 **진도가 갈라지지 않습니다.** 화면 모양이 달라지는 건 **8단계부터**입니다(10장).

> 💡 **"한 칸에 몇 명" 을 부르는 말 — 정원**
>
> 한 자리에 들어갈 수 있는 최대 인원을 **정원**이라고 합니다. 공방 클래스는 정원이 여러 명이라 `2/8` (8자리 중 2자리 참) 처럼 숫자로 보이고, 네일 예약은 정원이 **1명**이라 `예약 가능 / 마감` 으로 보입니다. 이 차이 하나가 나중에 시간표 화면의 모양을 가릅니다.

목록에 없는 걸 말하면 — 잘라서 받는다

13번(직접 말하기)으로 큰 것을 말해도 거절하지 않습니다. **되는 부분만 잘라서** 제안합니다. 자주 나오는 것은 자르는 법이 정해져 있습니다.

말한 것	잘라서 이렇게	빠지는 것
배달 앱	주문 받고 **픽업 시간** 정하기	지도 · 배달 추적 · 결제
쇼핑몰	상품 보고 **신청 받기**	결제 · 재고 · 배송
중고거래	물건 올리고 **문의 받기**	채팅 · 결제
SNS·커뮤니티	공지 올리고 **댓글 받기**	팔로우 · 피드 · 알림
매출·통계 대시보드	**명단과 출석 기록**	통계는 기록이 쌓인 다음 일
자동 문자 알림	**화면에 바로 뜨게**	문자는 승인·요금이 필요
회원가입·결제	미리 넣어둔 계정으로 **로그인**	가입·결제는 범위 밖

잘라낸 쪽이 **핵심**입니다. 배달 앱의 핵심은 지도가 아니라 "주문이 들어오고 사장님이 그걸 본다" 입니다.

7.3 2단계 — 어떻게 쓰이는지 이야기하기 (5분)

⌨ **2단계 상자**를 붙여넣으면 Claude가 세 개만 묻습니다. 전부 보기가 붙어 있습니다.

```
Q1. 지금은 그걸 어떻게 처리하고 계세요?
    1) 카톡으로  2) 전화로  3) 종이 노트나 달력
    4) 엑셀  5) 네이버 예약 같은 걸 쓴다  6) 직접 말하기

Q2. 그것 때문에 제일 불편한 순간은 언제인가요?
    1) 늦은 밤에도 연락이 온다
    2) 누가 오는지 매번 세어봐야 한다
    3) 갑자기 못 온다는 연락이 와서 자리가 빈다
    4) 같은 걸 여러 사람에게 따로따로 알려야 한다
    5) 직접 말하기

Q3. 관계된 사람이 몇 명쯤 되나요?
    1) 10명 아래  2) 수십 명  3) 수백 명 이상  4) 아직 모르겠다
```

평범한 질문 같지만, **답 하나하나가 설계를 바꿉니다.**

그림 7-2 2단계 질문 셋은 각각 설계 하나를 정한다. Q1은 왜 만드는지, Q2는 제일 먼저 만들 화면, Q3는 명단 모양(목록인지 표+검색인지)

Q1 — 왜 만드는지의 근거

지금 방식은 설계를 바꾸지 않습니다. 대신 **"왜 만드나"** 를 한 줄로 남깁니다.

답	브레인스토밍.md 에 남는 말
카톡	신청이 대화에 섞여 흘러간다
전화	받는 사람이 있어야만 접수된다
종이	찾아보려면 넘겨야 한다
엑셀	신청자가 직접 넣지 못한다
기존 서비스	남는 기능이 많고 월 요금이 나간다

나중에 "이걸 왜 만들었더라?" 할 때 이 한 줄이 답입니다. **만든 사람의 생각**의 첫 줄입니다.

Q2 — 제일 먼저 만들 화면이 정해진다 ⭐

제일 불편한 순간	제일 먼저 만들 화면
늦은 밤에도 연락이 온다	앱에서 **신청 받기**
누가 오는지 매번 세어봐야 한다	웹에서 **오늘 명단 보기**
갑자기 못 온다고 해서 자리가 빈다	**신청과 취소**
같은 걸 따로따로 알려야 한다	**공지 한 번에 올리기**

제일 아픈 곳부터 고치는 겁니다. 여기서 정하는 건 **순서**뿐입니다. 나머지를 빼는 게 아닙니다.
무엇을 뺄지는 3단계에서만 정합니다.

Q3 — 화면 모양이 갈린다

사람 수	명단 화면이 이렇게 된다
10명 아래	**목록** — 한 줄에 한 사람씩 쭉. 검색은 안 만든다
수십 명	**표 + 이름 검색** 하나
수백 명 이상	표 + 검색까지만. 나눠 보기는 나중에
모르겠다	수십 명으로 두고 넘어간다

> 💡 **목록과 표는 무엇이 다른가**
>
> **목록**은 한 사람을 한 덩어리로 보여줍니다. `김하나 · 010-… · 10시 클래스`. 사람이 적을 때 제일 읽기 쉽습니다.
>
> **표**는 칸을 세로로 맞춰 줄 세웁니다. 이름 칸, 연락처 칸, 시간 칸. 사람이 많아지면 **칸을 따라 훑어야** 원하는 걸 찾습니다. 그래서 수십 명부터는 표와 **검색**(이름을 쳐서 찾기)이 같이 붙습니다.
>
> 10명한테 검색창을 붙이면 쓸 일 없는 칸만 늘어납니다. **사람 수가 화면을 정합니다.**

7.4 3단계 — 무엇을 안 만들지 정하기 (7분) ⭐

이 단계가 1주차의 본론이다

`지침서.md` 는 3단계를 이렇게 적어 둡니다.

> **이 단계가 이 강의의 본론이다.** 만들 것을 늘리는 게 아니라 **막는 것**이 완성을 만든다.

> 📖 **범위(scope, 스코프)**
>
> **이번에 어디까지 만들지 그어 둔 선.** 선 안은 만들고, 선 밖은 이번엔 안 만듭니다.
>
> 범위를 정하지 않으면 만들다가 "이것도 있으면 좋겠다" 가 끝없이 붙습니다. 개발하는 사람들은 이걸 **"범위가 샌다"** 고 부릅니다. 샌 범위는 완성을 막습니다.

따라 하기 — 하고 싶은 걸 다 꺼내기

⌨ **3단계 상자**를 붙여넣으면 Claude가 **하고 싶은 것 열두 가지**를 보기로 늘어놓고 다 고르라고 합니다. 여러 개 골라도 됩니다.

그리고 가릅니다. **1~5는 전부 만들고, 6~11은 전부 안 만듭니다.**

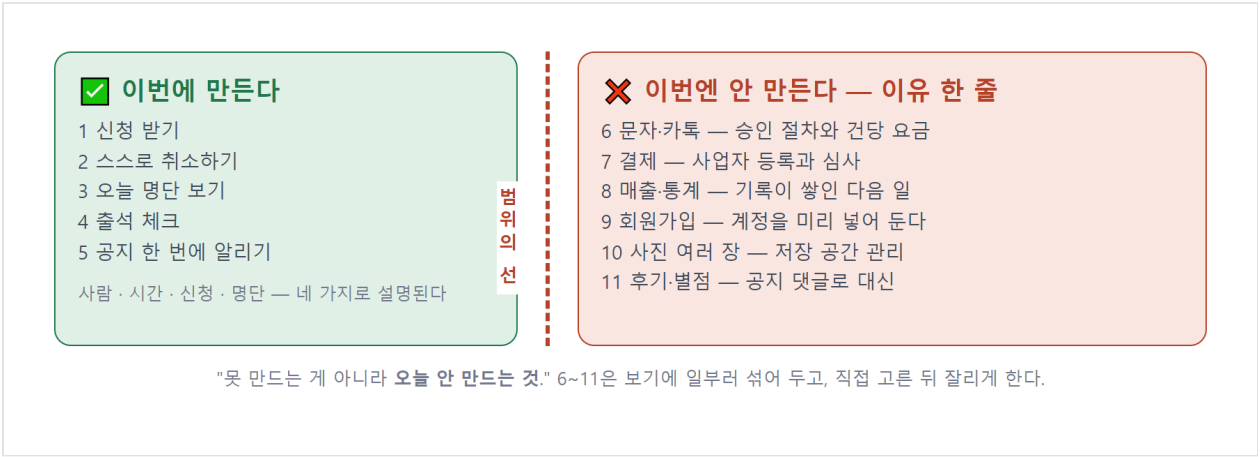

그림 7-3 3단계에서 긋는 범위의 선. 선 안(1~5)은 이번에 만들고, 선 밖(6~11)은 이유 한 줄과 함께 이번엔 안 만든다

> 💡 **왜 못 만들 걸 보기에 일부러 섞어 두나** > > 6~11번은 **일부러** 섞어 둔 것입니다. 고르기 전에 "이건 안 됩니다" 라고 > 미리 말해 주지도 않습니다. > > **직접 고르고 → 잘리는 걸 겪어야** 범위가 몸에 남기 때문입니다. 남이 그어 준 > 선은 금방 잊고, 내가 넘었다가 돌아온 선은 기억합니다.
> 잘릴 때 Claude는 반드시 이 말을 덧붙입니다.

> "못 만드는 게 아니라 **오늘 안 만드는 겁니다.** 신청하고 명단 보는 게 먼저 돌아가야, 저것들을 붙일 자리가 생깁니다."

서운한 게 정상입니다. 1단계에서 미뤄 둔 것들 — 필라테스의 **10회권 남은 횟수**, 네일숍의 **시술 종류·걸리는 시간** 같은 것도 여기서 「안 만들 것」으로 들어갑니다.

두 개 더 묻는다 — 관리자와 취소

```
Q4. 관리하는 사람은 몇 명인가요?
    1) 나 혼자다  2) 두세 명이 나눠 본다  3) 아직 모르겠다

Q5. 신청한 사람이 스스로 취소할 수 있어야 하나요?
    1) 스스로 취소할 수 있게  2) 나한테 연락하게  3) 아직 모르겠다
```

Q4 — 관리자 수

> 📖 **계정(account)과 권한 나누기**
>
> **계정**은 로그인할 때 쓰는 **아이디 한 벌**입니다. **관리자 계정**은 명단을 보고 공지를 올릴 수 있는 아이디입니다.
>
> **권한 나누기**는 계정마다 할 수 있는 일을 다르게 주는 것입니다. "원장님은 다 보고, 선생님은 자기 반만 본다" 같은 것. 편리하지만 만들 게 크게 늘어납니다.

답	이렇게 간다
혼자	관리자 계정 **하나**
두세 명	**그래도 계정 하나를 같이 씁니다.** 권한 나누기는 범위 밖
모르겠다	혼자로 두고 넘어간다

Q5 — 취소 — 앱에 화면 하나가 더 필요한지가 갈립니다.

답	이렇게 간다
스스로 취소	앱에 「**내 신청**」**화면**이 생기고 취소 버튼이 붙는다
나한테 연락	앱은 **신청까지만.** 취소는 웹에서 관리자가
모르겠다	스스로 취소로 둔다. 시연이 더 잘 보인다

★ 앞으로 돌아가 다시 보기

3단계 끝에 Claude가 **한 번만** 되묻습니다.

> "아까 **(2단계에서 정한 화면)** 부터 만들기로 했었는데, 방금 범위를 줄여보니 그게 아직도 제일 먼저일까요?
>
> 1. 그대로 간다 2) 바꾸고 싶다"

범위를 좁히면 앞에서 정한 것도 다시 봐야 합니다. 이건 강사가 실제로 겪은 가장 큰 일에서 나온 규칙입니다. 범위를 늦게 말하는 바람에, 앞에서 내린 결론을 전부 다시 봐야 했습니다.

> ⚠ **범위는 뒤에서 정할수록 비싸다**

> 1단계에서 "결제는 안 한다" 를 정하면 한 줄입니다. 화면을 다 고른 뒤에 정하면 결제 버튼이 들어간 화면을 **다시 골라야** 합니다. 코드까지 짠 뒤라면 **다시 만들어야** 합니다. 그래서 1주차는 범위를 **화면보다 먼저** 정합니다.

저장

`브레인스토밍.md` 에 네 덩어리가 덧붙습니다.

```
## 이번에 만들 것
- 신청 받기 / 스스로 취소 / 오늘 명단 / 출석 / 공지

## 이번엔 안 만들 것
- 결제 — 사업자 등록과 심사가 필요합니다
- …

## 이렇게 정했다
- 사람 수 — 10명 아래 → 목록
- 관리자 — 혼자 → 계정 하나
- 취소 — 스스로 → 앱에서

## 아직 안 정한 것
- …
```

Claude가 한마디 짚습니다. **"저 「안 만들 것」 목록이 오늘 끝낼 수 있게 해주는 겁니다."**

7.5 안 될 때

증상	이렇게 치세요
1단계부터 화면을 그리려 한다	`아직 화면 그리지 마. 1단계만 해줘`
질문을 한꺼번에 여러 개 한다	`한 번에 하나씩만 물어봐`
내가 말한 게 통째로 안 된다고 한다	`그중에 오늘 되는 부분만 잘라서 알려줘`
안 만들 것으로 간 게 너무 아쉽다	`이건 안 만들 것 말고 아직 안 정한 것에 적어줘`
브레인스토밍.md 가 안 생겼다	`지금까지 정한 거 docs/20.working/브레인스토밍.md 로 저장해줘`

7.6 정리

단계	정한 것	설계에 미치는 영향
1	누구를 위한 무엇	부르는 이름 (수강생 / 손님 / 신청자)
2 Q1	지금 방식	**왜 만드나** 한 줄
2 Q2	제일 불편한 순간	**제일 먼저 만들 화면**
2 Q3	사람 수	**목록 / 표+검색**
3	만들 것 · 안 만들 것	**범위**
3 Q4·Q5	관리자 · 취소	계정 하나 · 「내 신청」 화면 유무

이 장에서 나온 말

말	쉬운 뜻
브레인스토밍	생각을 일단 다 꺼내 늘어놓고 고르는 일. 안 정한 것까지 담는다
뼈대 네 덩어리	사람 · 열리는 칸 · 신청 · 알림판. 주제가 달라도 똑같다
정원	한 자리에 들어갈 수 있는 최대 인원
목록 / 표	한 사람씩 덩어리로 / 칸을 세로로 맞춰 줄 세운 것
검색	이름 같은 걸 쳐서 원하는 줄만 찾는 기능
범위(scope)	이번에 어디까지 만들지 그어 둔 선. 넘치면 "범위가 샌다"
계정	로그인에 쓰는 아이디 한 벌. 관리자 계정 · 사용자 계정
권한 나누기	계정마다 할 수 있는 일을 다르게 주는 것

다음 장에서는 지금까지의 **내 얘기**를 한 장으로 모으고, 처음으로 **남들은 어떻게 하는지** 봅니다.

08장 — 모으고, 견주고, 갈림길을 고른다

> 초고 v0.1 (2026-09-14) · 1주차 4~6단계

8.1 이 장에서 하는 일

7장에서 **내 얘기**를 했습니다. 이 장은 세 걸음입니다.

단계	하는 일	질문 수	성격
4단계	흩어진 결정을 **한 장으로 모으기**	**0개**	쉬어가기
5단계	**남들은 어떻게 하나** 보기	3개	견주기
6단계	남은 **갈림길 네 개** 고르기	4개	고르기

8.2 4단계 — 정한 것을 문서로 모으기 (5분)

대화는 흩어지고, 문서는 모인다

1~3단계의 답은 **대화 속에 흩어져** 있습니다. 스크롤을 올려야 보이고, 새 대화를 열면 사라집니다(3장). 4단계는 그걸 **읽을 수 있는 한 장**으로 모읍니다.

⌨️ **4단계 상자**를 붙여넣습니다.

```
지금까지 정한 게 여기저기 흩어져 있는 것 같아.
나중에 다시 봐도 알 수 있게 한 장으로 정리해줘.
```

이번엔 질문이 없습니다. 1~3단계 답으로 다 채울 수 있기 때문입니다. 여기서 쉬어가세요.

목차는 일곱 장, 고정이다

`브레인스토밍.md` 는 누가 만들든 **같은 일곱 장**으로 나옵니다.

```
1. 무엇을 만드나              ← 1단계 한 줄 정의
2. 누가 쓰나                  ← 관리하는 쪽 / 신청하는 쪽
3. 지금은 어떻게 하고 있나     ← 2단계 Q1
4. 제일 불편한 것             ← 2단계 Q2 + 제일 먼저 만들 화면
5. 이번에 만들 것             ← 3단계에서 만들기로 한 것
6. 이번엔 안 만들 것          ← 3단계에서 뺀 것 (이유 한 줄씩)
7. 아직 안 정한 것
```

Claude는 **문서 전체를 대화창에 쏟지 않습니다.** 목차 일곱 줄과, 7장「아직 안 정한 것」만 펼쳐 보여줍니다. 나머지는 **파일을 열어서** 보라고 합니다.

> 💡 **왜 전부 보여주지 않고 파일을 열라고 하나**
>
> 대화창에 긴 문서를 쏟으면 **읽지 않습니다.** 스크롤해서 넘깁니다. 그리고 그 긴 글이 공책(컨텍스트)을 차지합니다.
>
> 파일을 열면 **내가 쓴 설계를 내 눈으로** 한 번 보게 됩니다. 이 습관이 중요합니다. 2주차부터 Claude가 이 문서를 읽고 일할 텐데, **사람이 한 번도 안 읽은 문서**를 AI가 따르게 되면 안 되니까요.

> ⚠️ **아직 PRD가 아닙니다**
>
> 4단계 문서는 **내가 정한 것을 모아 둔 메모**입니다. 아직 남들이 어떻게 하는지 안 봤으니 확정할 수 없습니다. 확정 문서(PRD)는 7단계에서 따로 만듭니다(9장).

8.3 5단계 — 남들은 어떻게 하나 보기 (10분)

수강생은 검색하지 않는다

⌨️ **5단계 상자**를 붙여넣습니다.

```
지금까지 내 얘기만 한 것 같아.
비슷한 걸 이미 만들어 쓰는 데가 있을 텐데,
니가 좀 찾아서 보여줘. 보고 내가 고를게.
```

> 📖 **시장조사 · 벤치마킹(benchmarking)**
>
> 내가 만들려는 것과 **비슷한 걸 이미 하고 있는 곳을 찾아보는 일.** 무엇을 제공하는지, 사람들이 무엇에 불평하는지 봅니다.
>
> 똑같이 따라 하려는 게 아닙니다. **남과 견줘 봐야 내 것이 보이기** 때문에 합니다. 기준 삼아 견준다(benchmark)는 뜻입니다.

"찾아보세요" 라고 시키지 않습니다. **Claude가 찾아 와서 고르게만** 합니다. 찾는 법은 정해져 있습니다.

1. 내 주제와 비슷한 **한국 서비스 3개**를 찾는다
2. **기능만** 뽑아 표 하나로 만든다 — **가격은 적지 않는다**
3. 기능은 **6~8줄**까지

그림 8-1 5단계. 남들 기능표를 보고 "나한텐 안 쓴다" 를 고르면, 그 말이 이유가 되어 브레인스토밍.md「안 만들 것」에 그대로 적힌다

| 하지 않는 것 | 이유 | |---|---| | **가격 비교** | 월 얼마인지 따지기 시작하면 기능이 안 보입니다. **기능과 구조만** 봅니다 | | **해외 서비스** | 결제·문자 환경이 달라서 섞이면 헷갈립니다. 한국 것만 봅니다 | | **Claude 혼자 결론** | 무엇을 뺄지는 **수강생이** 고릅니다 |

찾은 곳은 반드시 파일로 남긴다

찾은 서비스 주소는 `docs/90.reference/참고링크.md` 에 저장합니다. 5장의 **레시피 벽**(90.reference)이 처음 쓰이는 자리입니다.

> 📖 **링크 · URL(유알엘)**

> 인터넷에서 **어떤 페이지가 있는 곳의 주소.** `https://` 로 시작하는 그 글자줄입니다. 누르면 그 페이지로 가서 **링크**라고도 부릅니다.
>
> 3주차에 내 사이트를 올리면 **내 사이트도 URL을 하나 갖게** 됩니다.

> ⚠️ **파일로 남기라고 말하지 않으면 대화로만 끝납니다**
>
> 좋은 조사를 해 놓고 대화창에만 두면, **스크롤과 함께 사라집니다.** 다음 주에 "그때 본 그 서비스 뭐였지?" 가 됩니다. 1주차 규칙에 **"반드시 참고링크.md 로 남긴다"** 가 박혀 있는 이유입니다.

질문 세 개 — 한 번에 하나씩

```
Q8.  위 표에서, 나한테는 있어도 안 쓸 것 같은 게 있나요?
Q9.  반대로, 저 중에 나한테 꼭 필요한 건 뭔가요?
Q10. 저기 없는데 나한테는 꼭 있었으면 하는 게 있나요?
```

질문	답은 어디로 가나
Q8 안 쓸 것	`브레인스토밍.md` **6장「안 만들 것」.** 이유는 **내가 한 말 그대로**
Q9 꼭 필요한 것	이미 5장에 있으면 확인만. 없으면 선(6장 6.7절)으로 재고 5장에 더한다
Q10 없는데 필요한 것	선으로 잰다. 되면 5장, 안 되면 6장 + 이유

Q8이 이 단계의 핵심이다

남들이 **다 가진** 기능을, 내가 **"나한텐 안 쓴다"** 고 말하는 순간 — 「안 만들 것」이 남이 그어 준 선이 아니라 **내 판단**이 됩니다.

강사가 실제로 겪은 일이 있습니다. 비슷한 서비스를 찾아보니 **전부**「직원 급여 정산」기능이 있었습니다. 다 있으니 필요해 보였습니다. 그런데 혼자 운영하는 사람에게는 **정산할 직원이 자기 자신뿐**이었습니다. 통째로 필요 없는 기능이었습니다.

1주차에서 Claude는 이걸 **먼저 알려주지 않습니다.** 수강생이 표를 보다가 스스로 발견하게 둡니다.

> 💡 **남들이 다 가졌다고, 나한테 필요한 건 아니다**
>
> 경쟁 서비스는 **수천 명의 서로 다른 사장님**을 위해 만들어졌습니다. 그래서 기능이 많습니다. 그중 대부분은 **다른 사장님의 기능**입니다.

> 5단계를 조사보다 **나중에** 두는 이유가 여기 있습니다(6장 6.5절). 내 것을 먼저 정해 두지 않으면, 이 표를 보는 순간 전부 필요해 보입니다.

쓰는 사람의 목소리가 있으면 한 줄

공식 소개 페이지는 **파는 쪽이 하고 싶은 말**입니다. 후기나 커뮤니티 글에 실제로 쓰는 사람들이 **자주 불평하는 것**이 있으면, Claude가 한 줄만 찾아서 묻습니다.

> "쓰는 분들 글을 보니 **(불평 한 가지)** 가 자주 나오네요. 이건 우리도 신경 쓸까요?"

못 찾으면 건너뜁니다. **억지로 만들어내지 않습니다.** 3장의 환각을 떠올리면 왜 그렇게 정했는지 보입니다.

8.4 6단계 — 갈림길 고르기 (7분)

묻는 게 아니라 펼치는 것

앞 단계들은 "어떠세요?" 하고 **캐물었습니다.** 6단계는 다릅니다. Claude가 **"이렇게 할 수도 있고 저렇게 할 수도 있습니다"** 하고 길을 **펼쳐 놓고**, 수강생은 **고르기만** 합니다.

⌨️ **6단계 상자**를 붙여넣습니다.

```
아직 안 정한 게 남아 있는 것 같아.
어떻게 할 수 있는지 길을 펼쳐서 보여줘.
내가 하나씩 고를게.
```

> 🔀 **갈림길 · 트레이드오프(trade-off)**
>
> **하나를 얻으면 다른 하나를 내줘야 하는 선택.** 정답이 없고 **무엇을 더 중요하게 보느냐**로 갈립니다.
>
> "아무나 신청하게 하면 새 손님이 오지만, 장난 신청도 온다" — 이런 것이 트레이드오프입니다. 그래서 보기마다 **고르면 뭐가 달라지는지**가 한 줄씩 붙습니다.

갈림길 네 개

한 번에 하나씩 나옵니다. 갈림길 1은 그림으로 보고, 나머지 셋은 이렇습니다.

그림 8-2 갈림길은 보기마다 "고르면 뭐가 달라지는지" 를 붙여 펼친다. 얻는 것(+)과 내주는 것(-)을 같이 보고, 무엇을 더 중요하게 볼지로 고른다

``` 갈림길 2. 자리가 다 차면 어떻게 할까요? A) 마감으로 막는다 → 간단하다. 오늘 안에 끝난다 B) 대기를 걸어둔다 → 빈자리가 났을 때 처리가 복잡해진다
갈림길 3. 언제까지 취소되게 할까요? A) 시작 직전까지 → 손님은 편한데, 빈자리가 그대로 뜬다 B) 2시간 전까지 → 늦은 취소를 막는다 C) 하루 전까지 → 확실한데, 손님이 답답해한다

갈림길 4. 신청할 때 뭘 받을까요? A) 이름만 → 제일 간단하다 B) 이름 + 연락처 → 연락할 수 있다 C) 이름 + 연락처 + 남길 말 → 요청사항을 받을 수 있다
```

"모르겠다" 의 행선지도 전부 정해져 있습니다.

갈림길	모르겠다고 하면	왜
1 신청 자격	**B 등록해 둔 사람만**	3단계에서 회원가입을 안 만들기로 했으니 맞물린다
2 자리가 차면	**A 마감**	제일 단순하다
3 취소 마감	**B 2시간 전**	가운데
4 받을 것	**B 이름 + 연락처**	연락은 할 수 있어야 한다

대기를 고르면 — 한 번 말린다

갈림길 2에서 **B 대기**를 고르면 Claude가 한 번 말립니다.

> "대기는 만들 수 있는데, **빈자리가 났을 때 누구한테 먼저 가는지**를 정해야 하고
> 그게 생각보다 까다롭습니다. 화면에 보이는 건 별로 없는데 일은 제일 많아요.
> 오늘은 **마감**으로 두고, 나중에 붙이시는 게 어떨까요?"

그래도 대기로 하겠다고 하면 **더 말리지 않습니다.** 대신 `브레인스토밍.md` 7장에
"대기 — 빈자리 차례를 아직 안 정함" 을 남깁니다.

> 💡 **왜 대기를 말리나 — 보이는 것과 일의 양은 다르다**
>
> 대기 기능은 화면에 **버튼 하나**입니다. 그런데 뒤에서 할 일은 많습니다. 누가
> 취소했는지 알아채고, 대기 1번에게 넘기고, 1번이 응답이 없으면 2번에게 넘기고,
> 그 사이에 또 누가 신청하면…
>
> **화면 임팩트는 작은데 규칙이 제일 까다로운 기능.** 강사도 같은 이유로 뺐습니다.
> 기능을 고를 때 "화면에 얼마나 보이나" 가 아니라 **"뒤에서 몇 가지를 정해야 하나"**
> 로 재 보세요.

저장

`브레인스토밍.md` 의 「이렇게 정했다」 에 네 줄이 붙습니다.

이렇게 정했다

- 신청 자격 — B 등록해 둔 사람만
- 자리가 찼을 때 — A 마감
- 취소 마감 — B 2시간 전
- 신청할 때 받는 것 — B 이름 + 연락처

Claude가 말합니다. **"이제 정할 건 다 정했습니다."**

8.5 안 될 때

증상	이렇게 치세요
4단계에서 또 질문을 한다	`4단계는 질문 없이 정리만 해줘`
조사 결과에 가격이 잔뜩 있다	`가격은 빼고 기능만 표로 다시 보여줘`
해외 서비스가 섞였다	`한국에서 쓰는 서비스로만 다시 찾아줘`
참고링크.md 가 안 생겼다	`찾은 곳 주소를 docs/90.reference/참고링크.md 에 저장해줘`
갈림길을 한꺼번에 네 개 보여준다	`갈림길 하나씩 보여줘`
Claude가 갈림길을 알아서 정했다	`그건 내가 고를게. 보기로 다시 보여줘`

8.6 정리

단계	한 일	남은 것
4	흩어진 결정을 **일곱 장 목차**로 모았다	`브레인스토밍.md`
5	비슷한 서비스 **3개의 기능표**를 보고, 안 쓸 것을 **내가** 골랐다	「안 만들 것」 늘어남 · `참고링크.md`
6	**갈림길 네 개**를 펼쳐 놓고 골랐다	「이렇게 정했다」 네 줄

이 장에서 나온 말

말	쉬운 뜻
목차	문서의 장 제목을 순서대로 늘어놓은 것. 문서의 뼈대
시장조사 · 벤치마킹	비슷한 걸 이미 하는 곳을 찾아 견줘 보는 일
기능(feature)	서비스가 해 주는 일 한 가지. 신청 받기 · 공지 · 출석 체크
기능표	여러 서비스의 기능을 O / X 로 나란히 놓은 표
링크 · URL	인터넷 페이지가 있는 곳의 주소. `https://` 로 시작한다
갈림길 · 트레이드오프	하나를 얻으면 하나를 내주는 선택. 정답 없이 우선순위로 고른다

```
| **확정** | 신청을 받은 뒤 관리자가 "들어오셔도 됩니다" 하고 굳히는 것 |
| **대기(웨이팅)** | 자리가 찼을 때 줄을 세워 두었다가 빈자리가 나면 넘겨주는 기능 |

정할 것은 다 정했습니다. 다음 장에서는 이걸 **설계 문서**, PRD로 굳힙니다.

---
```

09장 — PRD, 설계 문서로 굳히기

> 초고 v0.1 (2026-09-14) · 1주차 7단계

9.1 PRD라는 이름

지금까지 `브레인스토밍.md` 에 정한 것을 쌓았습니다. 7단계에서는 이걸 **확정 문서**로 옮깁니다. 그 문서의 이름이 PRD입니다.

> 📖 **PRD(Product Requirements Document, 제품 요구사항 문서)**
>
> **"무엇을, 누구를 위해, 어디까지 만들 것인가"** 를 만들기 전에 적어 둔 문서.
>
> **Product(제품)** — 만들 것. 우리에겐 웹과 앱 한 벌 **Requirements(요구사항)** — 그게 반드시 해내야 하는 일들 **Document(문서)** — 그걸 글로 적어 둔 것
>
> 이름은 거창하지만, 결국 **만들기 전에 쓰는 약속서**입니다.

왜 굳이 문서로 쓰는지는 한 줄로 설명됩니다.

> **Claude는 시키는 대로 잘 만들지만, 무엇을 시킬지는 사람이 정해야 한다.**

머릿속에만 있으면 만들다가 방향이 흔들립니다. 오늘은 "간단하게" 였는데 다음 주에는 "이것도 넣자" 가 됩니다. 글로 적어 두면 **흔들릴 때 돌아올 곳**이 생깁니다.

강의 자체도 이렇게 만들어졌습니다. 2기 강의는 시작 전에 강사가 **강의용 PRD**를 먼저 썼고, 거기에 "누가 듣는가, 4주 뒤 무엇이 남는가, **무엇은 약속하지 않는가**" 를 적었습니다. 이 책의 0장 「약속하지 않는 것」이 그 문서에서 왔습니다.

9.2 브레인스토밍과 PRD는 역할이 다르다

둘 다 정한 것을 적는 문서라 헷갈립니다.

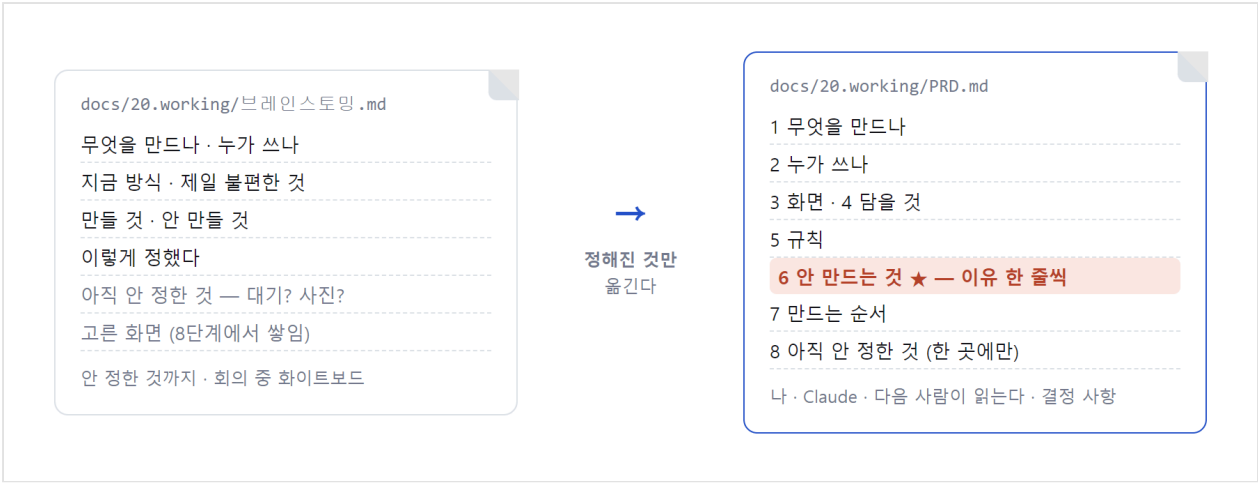

그림 9-2 브레인스토밍.md 는 안 정한 것까지 담는 과정의 기록이고, PRD.md 는 정해진 것만 옮긴 약속이다. PRD에서 제일 중요한 칸은 6장「안 만드는 것」

> 💡 **왜 두 문서를 섞지 않나** > > PRD에 "아직 모름" 이 섞여 있으면, Claude는 **그것도 정해진 것처럼** 읽습니다. > 그러면 모르는 부분을 그럴듯하게 채워서 만듭니다(3장의 환각). > > 그래서 규칙은 이렇습니다. **정해지면 브레인스토밍에서 PRD로 옮긴다.** > PRD 끝에「아직 안 정한 것」 장을 따로 두는 것도 같은 이유입니다 — 모르는 건 > **모른다고 딱 한 곳에** 모아 둡니다.

9.3 따라 하기 — 두 개만 더 묻는다

⌨ `제작prompt.md` 의 **7단계 상자**를 붙여넣습니다.

```
이제 정할 건 다 정한 것 같아.
이걸로 제대로 된 설계 문서를 만들어줘.
```

Claude가 두 개만 묻습니다.

```
Q14. 이름을 뭐라고 할까요?
     1) (주제 그대로 – 예: "oo 예약")
     2) 내 상호를 쓸게 – 상호를 말씀해 주세요
     3) 아직 모르겠다

Q15. 보통 언제 여세요?
     1) 평일 저녁  2) 평일 낮  3) 주말 위주
     4) 매일       5) 직접 말하기
```

이름을 모르겠다고 하면 「**내 예약**」으로 두고 넘어갑니다. 운영 시간을 말로 길게 해도 **요일과 시간대만** 뽑아 적습니다. 세부 시간표는 나중 일입니다.

답하면 `docs/20.working/PRD.md` 가 생깁니다.

그림 9-1 PRD.md. ① 브레인스토밍.md 와 PRD.md ② 버전 · 상태 · 근거 문서 ③ 변경점 ④ 원칙을 깬 날과 그 이유

9.4 PRD의 여덟 장

PRD도 누가 만들든 **같은 여덟 장**으로 나옵니다(그림 9-2 오른쪽). 한 장씩 **어디서 왔는지** 보면, 7단계에서 새로 묻는 게 거의 없는 이유가 보입니다. 전부 앞에서 정했습니다.

장	여기서 왔다
3 화면 이름	1단계에서 고른 주제 (오늘 클래스 / 오늘 예약 / 받은 신청)
3 명단 모양	2단계 Q3 사람 수 → 목록 / 표+검색
4 담을 것	7장의 **뼈대 네 덩어리**
5 정원	주제별 정원 (네일·반려동물·회의실은 **1**)
5 운영 시간	방금 답한 Q15
5 취소 · 관리자	3단계 Q5 · Q4
6 안 만드는 것	3단계 + **5단계 조사에서 늘어난 것**
7 만드는 순서	2단계 Q2「제일 먼저 만들 화면」

「화면」과 「담을 것」은 무엇이 다른가

처음 보면 3장과 4장이 비슷해 보입니다. 가게로 비유하면 이렇습니다.

그림 9-3 데이터는 하나, 화면은 둘. 김하나의 신청 기록 한 줄이 앱에서는「내 신청」으로, 웹에서는「오늘 클래스」 명단으로 보인다

화면은 **보여주는 곳**이고, 담을 것은 **기억해 두는 것**입니다.

> 📖 **데이터(data)**
>
> 화면 뒤에서 **기억해 두는 내용 전부.** 누가 신청했는지, 몇 시 클래스가 몇 자리 남았는지, 공지에 뭐라고 썼는지. PRD 4장「담을 것」이 이 데이터의 목록입니다.

> 2주차에 앱을 만들면, 손님 폰에서 누른 신청이 **데이터로 저장**되고, 그 데이터를 사장님 웹이 **읽어서 보여줍니다.** 한쪽에서 누르면 다른 쪽에 뜨는 비밀이 이것입니다.

9.5 제일 중요한 건 6장「안 만드는 것」

Claude는 PRD를 보여줄 때 **목차 여덟 줄**과 **6장만** 펼칩니다. 그리고 한 줄만 짚습니다.

> "이 문서에서 제일 중요한 건 **6장「안 만드는 것」**입니다. 만들 걸 적는 문서가 아니라, **안 만들 걸 지키는 문서**라서 그렇습니다."

6장은 이렇게 생겼습니다.

```
## 6. 안 만드는 것

- 결제 — 사업자 등록과 심사가 필요합니다
- 문자·카톡 자동 발송 — 승인 절차와 건당 요금이 붙습니다
- 회원가입 — 계정은 미리 몇 개 넣어두고 씁니다
- 직원 급여 정산 — 혼자 운영이라 정산할 사람이 나뿐이다
- 대기 — 빈자리 차례 규칙이 까다롭다. 오늘은 마감으로
```

> 💡 **"안 만드는 것" 이 만든 사람의 생각을 가장 잘 담는다**
>
> 작가의 말에서 "수정해줘" 라고만 하면 AI가 **자기 생각대로** 고친다고 했습니다. AI가 가장 흔하게 하는 "자기 생각" 이 뭘까요? **기능을 더 붙이는 것**입니다. "예약 앱이면 결제도 있어야죠", "알림 문자를 보내드릴까요?"
>
> 6장이 있으면 막을 수 있습니다. 2주차에 Claude가 PRD를 읽고 만들 때, 3주차에 "첫 화면 수정해줘" 라고 할 때 — Claude는 **"결제는 안 만들기로 했다, 이유는 이렇다"** 를 먼저 봅니다. 그리고 선 안에서 고칩니다.
>
> 만들 것은 누구나 비슷하게 적습니다. **안 만들 것과 그 이유**는 만든 사람만 적을 수 있습니다. 그게 이 설계의 **철학**입니다.

그래서 6장의 이유는 **짧아도 반드시** 적습니다. 이유 없이 "결제 — 안 함" 만 있으면, 석 달 뒤의 나도 AI도 **"왜? 넣으면 좋을 것 같은데"** 로 돌아갑니다.

9.6 PRD의 최소 네 가지

1주차의 PRD는 여덟 장이지만, 어떤 PRD든 **최소한 이 네 가지**는 있어야 합니다. 나중에 혼자 새로운 걸 만들 때 이 네 줄을 먼저 떠올리세요.

	물을 것	1주차 PRD에서
①	**누가 쓰는가**	2장
②	**무엇을 할 수 있어야 하는가** (화면·기능)	3·4장
③	**이번에는 하지 않을 것** ← 제일 중요	6장
④	**다 됐다고 판단하는 기준**	7장 만드는 순서

그리고 PRD를 **직접 한 글자씩 쓰지 않아도 됩니다.** "이런 걸 만들고 싶은데 PRD로 정리해 줘" 라고 시키고, **고쳐 나가는 게** 빠릅니다. 1주차에서 한 일이 정확히 그것 입니다. 다만 고칠 때 **6장만은 내 말로** 채우세요.

> ⚠ **PRD가 끝이 아닙니다 — 아직 v0.1입니다**
>
> 지금 PRD의 3장「화면」은 **글로만** 적혀 있습니다. "로그인 화면 — 상호와 비밀번호를 넣는다" 정도요. 화면을 실제로 보면 **바뀌는 게 반드시 생깁니다.**
>
> 그래서 8단계에서 화면을 고른 뒤, 9단계에서 PRD를 **다시 씁니다**(11장). **PRD는 한 번 쓰고 끝나는 문서가 아니라 계속 고쳐 쓰는 문서입니다.**

9.7 안 될 때

증상	이렇게 치세요
PRD에 화면 그림이 들어갔다	`PRD에는 화면을 글로만 적어줘. 그림은 8단계에서`
목차가 여덟 장이 아니다	`지침서에 있는 PRD 목차 여덟 장으로 다시 맞춰줘`
6장에 이유가 빠졌다	`6장 안 만드는 것마다 이유를 한 줄씩 붙여줘`
브레인스토밍에 있던 "아직 모름" 이 PRD에 섞였다	`정해진 것만 PRD에 남기고, 안 정한 건 8장으로 옮겨줘`
대화창에 PRD 전체가 쏟아졌다	`목차랑 6장만 보여줘. 나머지는 파일로 볼게`

9.8 정리

- **PRD** = 무엇을 · 누구를 위해 · 어디까지 만들지 **만들기 전에** 적은 약속서
- 브레인스토밍은 **안 정한 것까지**, PRD는 **정해진 것만**
- PRD에서 제일 중요한 것은 **6장「안 만드는 것」과 그 이유**
- 지금은 **v0.1** — 화면을 보고 나면 고쳐 씁니다

이 장에서 나온 말

말	쉬운 뜻
PRD	Product Requirements Document. 만들기 전에 쓰는 약속서
요구사항	만들 것이 반드시 해내야 하는 일
초안(draft)	처음 써 본 판. 고칠 것을 전제로 한다
데이터	화면 뒤에서 기억해 두는 내용 전부. PRD 4장의 목록
규칙	정원 · 운영 시간 · 취소 마감처럼, 화면이 따라야 할 약속
철학	무엇을 왜 만들고 **왜 안 만들기로 했는지** — 만든 사람만 적을 수 있는 생각

다음 장에서 드디어 **화면**을 봅니다. 글로 적은 여덟 화면을 실제로 그려서, 세 가지 안 중에 고릅니다.

10장 — 화면 시안 고르기, 웹 넷 앱 넷

> 초고 v0.1 (2026-09-14) · 1주차 8단계

10.1 드디어 화면이다

1단계부터 7단계까지 화면을 한 번도 안 봤습니다. 일부러 미뤘습니다(6장). 이제 PRD에 **글로 적힌 여덟 화면**을 실제로 그려서 봅니다. 1주차에서 가장 긴 40분입니다.

> 📖 **시안(試案) · 목업(mockup)**
>
> 실제로 만들기 전에 **"이렇게 생길 것" 을 그려 본 화면.** 시험 삼아 내놓는 안이라서 시안입니다. 영어로는 **목업**이라고 부릅니다 — 모형(mock)이라는 뜻입니다.
>
> 시안은 **그림**입니다. 버튼을 눌러도 아무 일도 안 일어나도 됩니다. 로그인도 진짜로 되지 않습니다. **생김새를 고르기 위한 것**이라서입니다.

글로 설명하지 않는다

8단계의 제1원칙은 이것입니다.

> 💡 **"왼쪽에 목록이 있고 오른쪽에 버튼이…" 로는 못 고른다**
>
> 화면을 말로 설명하면, 듣는 사람은 **각자 다른 그림**을 떠올립니다. 그리고 그 상상한 그림으로 고릅니다. 실제로 만들어진 걸 보면 "이게 아니었는데" 가 됩니다.
>
> 그래서 1주차의 시안은 **실제로 그려진 화면**으로만 보여줍니다. 선택지 목록도, 글자로 그린 그림도 쓰지 않습니다. **눈으로 봐야 판단이 됩니다.**

10.2 네 번에 나눠서 본다

화면은 여덟 장입니다. 한꺼번에 늘어놓으면 못 고릅니다. **두 장씩 네 번** 봅니다.

그림 10-1 8단계의 네 회차. 웹(8-1·8-2)은 브라우저 창 안에 세 안을 세로로, 앱(8-3·8-4)은 폰 틀 안에 세 안을 가로로 놓는다. A는 언제나 제일 단순한 안

두 장씩 묶으면 한 번에 보고 판단할 수 있고, **고를 때마다 뭔가 정해집니다.** 회차 사이에 Claude는 반드시 멈추고 다음 상자를 기다립니다.

8-1·8-2는 컴퓨터로 보는 웹(관리하는 쪽), **8-3·8-4는 폰으로 보는 앱**(신청하는 쪽) 입니다. 6장에서 본 그 구분입니다.

10.3 여기서만 갈라진다 — 내 줄의 상자를 쓴다

1~7단계는 누구나 같은 상자를 썼습니다. **8단계만** 1단계에서 고른 번호에 따라 상자가 셋으로 갈립니다. `제작prompt.md` 에서 **내 줄만** 쓰면 됩니다.

고른 번호	어떤 것	신청 자리가 이렇게 보인다	쓸 상자
1~4	공방 · 필라테스 · 과외 · 독서모임 — **여럿이 같이**	`아침 하타 2/8` 숫자로	㉮
5~8	네일 · 반려동물 · 회의실 · 학원 — **한 사람씩**	`14:00 예약 가능` 빈 시간 고르기	㉯
9~12	공동구매 · 행사 · 수리 · 체험단 — **받기만**	시간표가 없다. **신청서 쓰기**	㉰

13번(직접 말하기)을 골랐으면 Claude가 어느 쪽인지 알려줍니다.

여기에 **6단계 갈림길 1**(신청을 누가 하나)이 곱해집니다.

	앱에서
등록제 — 등록해 둔 사람만 / 확인 후 확정	로그인하고 신청. 「**내 신청**」 화면이 있다
열린형 — 아무나	로그인 없음. 이름·연락처만 받고, 「**신청 확인**」으로 조회

그래서 화면 이름이 조금씩 달라집니다. 예를 들면 네일숍(한 사람씩 · 열린형)은 이렇게 됩니다.

```
오늘 만들 화면은 이렇게 여덟 장입니다.

  웹 (관리하는 쪽)          앱 (신청하는 쪽)
  ① 로그인                  ⑤ 둘러보기
  ② 오늘 예약  ★메인        ⑥ 시간 고르기
  ③ 주간                    ⑦ 예약 확인
  ④ 알림판                  ⑧ 공지
```

Claude는 8-1에 들어가기 **전에** 이 목록을 먼저 보여주고, 빠진 게 있는지 묻습니다. 빠진 게 있다고 하면 **여덟 장 안에서** 바꿉니다. 아홉 장으로 늘리지 않습니다.

"저는 회원이 없는데요"

8-1에서 가장 자주 나오는 말입니다. Claude는 이렇게 답하도록 정해져 있습니다.

> "회원 명부가 없어도 됩니다. **신청할 때 이름이랑 연락처만 받으면 그게 곧 명단이 됩니다.** 가입도 비밀번호도 없습니다. 로그인은 **관리하시는 분만** 합니다."

비슷한 말도 받는 법이 정해져 있습니다.

이렇게 말하면	이렇게 된다
"앱은 필요 없는데요"	앱 대신 **손님이 폰으로 여는 웹 화면**으로. 만드는 건 같다
"신청은 제가 넣어요"	앱은 **보기만** 하는 화면. 넣는 건 웹에서
"시간표가 없는데요"	**받은 목록**이 시간표 자리에. 날짜순으로 쌓인다
"알림판은 안 쓸 것 같아요"	**공지 한 줄**만 둔다
"화면이 더 많아야 하는데요"	오늘은 **여덟 장까지.** 나머지는 다음 주에 붙인다

"이 강의는 안 맞으시네요" 라는 말은 나오지 않습니다. 뼈대(사람 · 시간 · 신청 · 명단)는 같고, **부르는 이름과 로그인 유무만** 다르기 때문입니다.

10.4 웹과 앱은 그리는 법이 다르다

	💻 웹 (8-1 · 8-2)	📱 앱 (8-3 · 8-4)
틀	**브라우저 창** 안에 — 창 점 세 개 + 주소줄	**폰 틀** 안에 — 폭 390px 고정
세 안 놓는 법	**세로로 쌓는다** (넓어서 나란히 두면 찌그러짐)	**가로로 나란히** (좁아서 셋이 한 화면에 들어감)
글자	보통	**한 단계 크게**
버튼	보통	**손가락으로 누를 크기**
맨 아래	—	**탭 막대** — 둘러보기 · 시간표 · 내 신청 · 공지

낯선 말 몇 개를 풀고 갑니다.

- **브라우저 크롬** — 크롬 브라우저가 아니라, 웹페이지를 **감싸는 창의 테두리**를 부르는 말입니다. 위쪽의 점 세 개, 주소줄, 뒤로 가기 버튼. 시안을 이 테두리 안에 넣으면 "진짜 사이트" 처럼 보입니다
- **px(픽셀)** — 화면을 이루는 **가장 작은 점 하나.** 390px은 점 390개 폭입니다. 요즘 폰 화면 폭이 대략 이만합니다
- **탭 막대(탭 바)** — 앱 **맨 아래에 붙어 있는 버튼 줄.** 누르면 화면이 바뀝니다. 카카오톡 아래쪽의 친구 · 채팅 · 더보기 줄이 탭 막대입니다

웹과 앱을 한 페이지에 섞지 않습니다. 회차가 다르면 페이지도 다릅니다.

10.5 시안 페이지 읽는 법 — 여섯 조각

시안은 화면만 덜렁 나오지 않습니다. **고를 수 있게** 여섯 조각이 위에서 아래로 붙어 나옵니다.

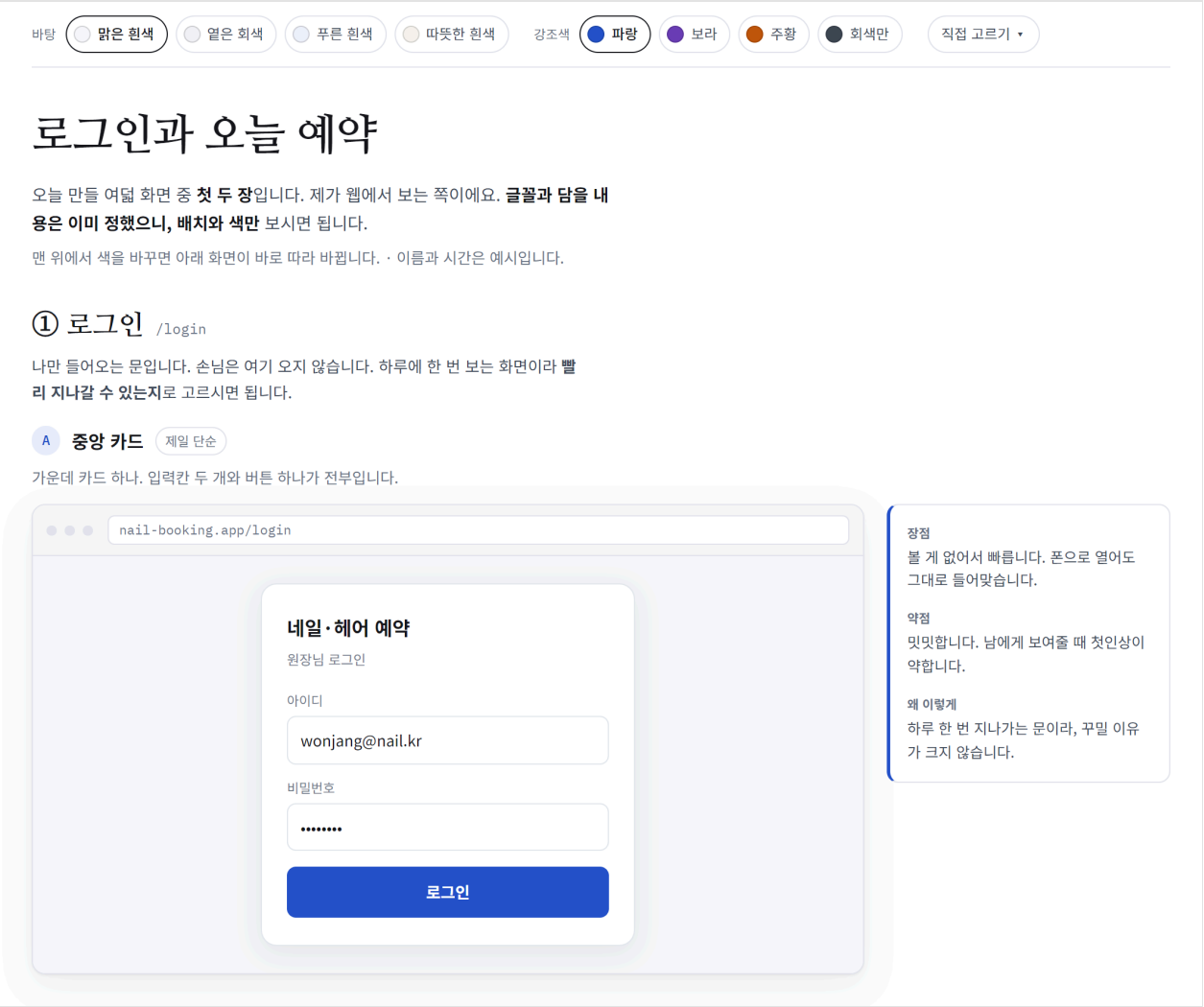

그림 10-2 8-1 시안의 윗부분. 색 바꾸는 줄 · 머리말 · 화면 이름과 주소 경로(/login) · A안 화면과 장점·약점·왜 이렇게

| 조각 | 볼 것 | |---|---| | ① 머리말 | "색과 글꼴은 이미 정했으니, **배치와 정보 구조만** 보시면 됩니다" 같은 두세 줄 | | ② 색 바꾸는 줄 | 누르면 아래 **모든 화면 색이 즉시** 바뀐다. 10.8절 | | ③ 화면 그룹 | 화면마다 A·B·C. 각 안 옆에 **장점 · 약점 · 왜 이렇게** | | ④ 비교표 | **고르는 기준** 칸이 제일 중요 — "○○를 중요하게 보신다면 이걸" | | ⑤ 짚어둘 것 | "B와 C는 겹치지 않습니다. 둘 다 원하시면 **B+C** 라고 말씀해 주세요" | | ⑥ 고르는 칸 | 맨 아래. 10.11절 |

화면 제목 옆의 `/login` 같은 글자는 **주소 경로**입니다. 사이트 주소 뒤에 붙어서 "그 사이트 안의 **어느 방**" 인지 가리킵니다. `내가게.vercel.app/login` 이면 로그인 화면입니다. 지금은 이름표로만 보면 됩니다.

> ⚠ **장점만 있는 안은 없습니다**
>
> 각 안에는 반드시 **약점**이 적혀 있습니다. 약점이 없으면 고를 이유가 없기 때문입니다. 약점 칸을 꼭 읽으세요. 고르는 건 결국 **어떤 약점을 참을지** 입니다. 8장의 트레이드오프와 같습니다.

10.6 세 안은 서로 다른 판단이다

색만 바꾼 세 개는 세 안이 아닙니다. **A · B · C는 서로 다른 생각**을 담습니다.

> 💡 **A안은 항상 「제일 단순한 것」**
>
> 세 안 중 A는 언제나 **가장 단순한 모양**입니다. 비교할 **기준점**이 있어야 B와 C가 무엇을 더했는지 읽힙니다.
>
> 그래서 "잘 모르겠어" 라고 하면 **A로 둡니다.** 제일 단순한 건 나중에 더하기 쉽고, 복잡한 건 나중에 빼기 어렵습니다.

웹 · 오늘 명단 — 무엇을 제일 먼저 보여주나

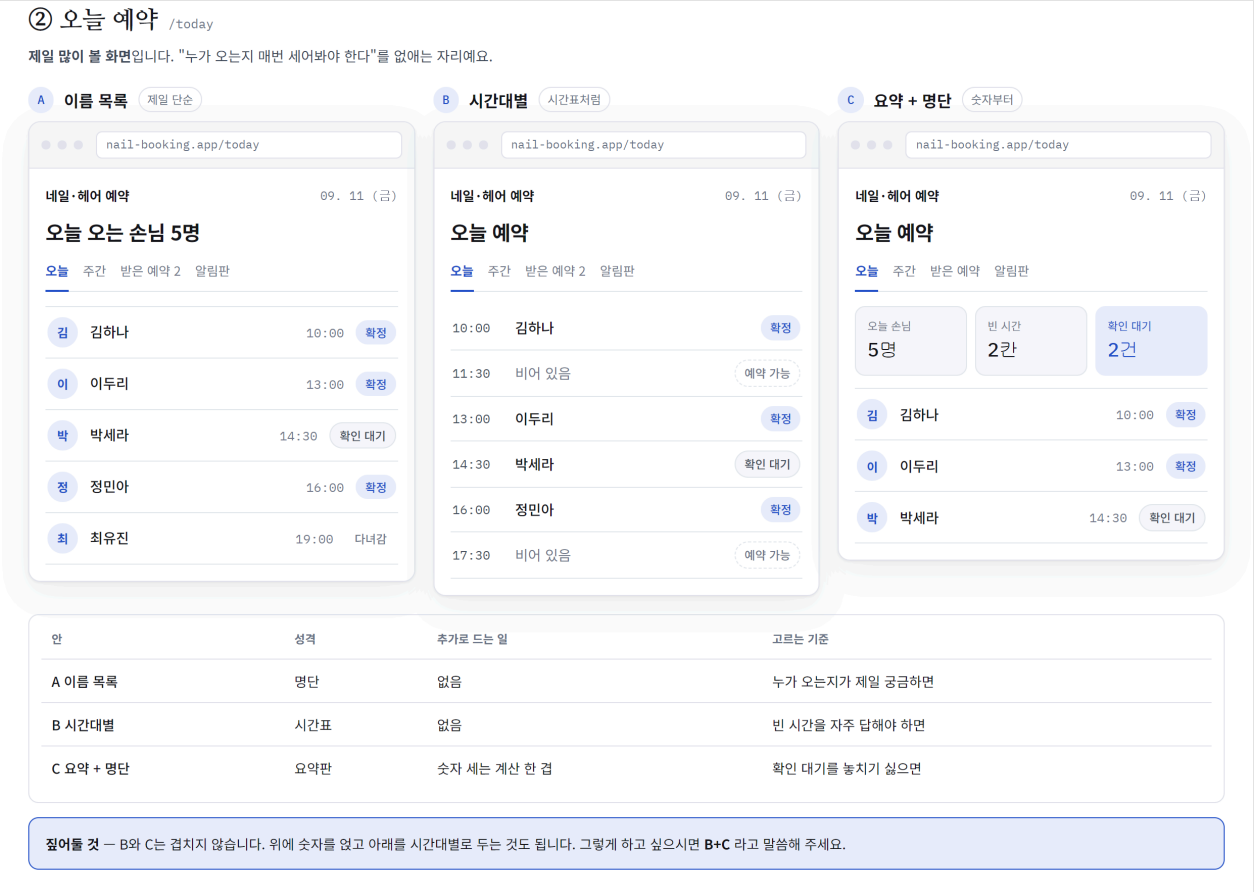

그림 10-3 「오늘 예약」 세 안과 비교표. A는 이름 목록(제일 단순), B는 빈 시간까지 보이는 시간대별, C는 숫자 요약을 얹은 안. 맨 아래 「짚어둘 것」이 B+C 합치기를 알려준다

웹 · 로그인

안	판단	꼬리표
A 중앙 카드	기능만. 입력칸 두 개	제일 단순
B 좌우 스플릿	왼쪽은 상호를 크게, 오른쪽에 입력칸	첫인상 중시
C 카드 + 빠른 입력	A에 계정 버튼 한 줄 추가	보여주기 편함

웹 · 주간

안	판단
A 시간 × 요일 표	학교 시간표 모양. 패턴이 보인다 / 칸이 좁아 글자가 작다
B 요일별 세로 카드	글자가 크다 / 시간축이 사라진다
C 오늘 화면에 탭으로 붙이기	페이지가 하나로 준다 / "여러 화면" 인상이 약해진다

앱 · 시간표

안	판단
A 날짜별 스크롤	손가락만 내리면 된다 / 먼 날짜는 오래 내려야 한다
B 위에 달력, 고른 날만	원하는 날로 바로 간다 / 화면 위쪽을 달력이 먹는다
C 이번 주 한 화면	한 주가 한눈에 / 폰이 좁아 글씨가 작아진다

다른 화면도 같은 식입니다. **"무엇을 제일 크게 보여주나"** 로 셋이 갈립니다.

낱말 두 개 — **카드**는 화면 위에 **네모난 종이 한 장**처럼 떠 있는 상자를 말합니다. **스크롤**은 화면을 **손가락이나 마우스 휠로 위아래로 미는 것**입니다.

10.7 ⭐ 앱의 첫 화면 — 「간판」, 히어로

8-3의 첫 장 **둘러보기**만 성격이 다릅니다. 나머지 화면은 **도구**인데, 여기는 **소개** 입니다. 로그인 없이 보이는 화면이고, 처음 들어온 사람이 **"여기 뭐 하는 데지"** 를 3초 안에 알아야 합니다. 그래서 `제작prompt.md` 는 이 화면을 **간판**이라고 부릅니다.

> 📖 **히어로(Hero) 영역**

> 사이트나 앱을 열었을 때 **맨 위에 크게 자리 잡은 첫 화면 한 덩어리.** 큰 사진이나 색면 위에 이름 · 한 줄 소개 · 주 버튼이 올라가 있습니다. 가게로 치면 **간판과 쇼윈도**입니다. 주인공(hero)처럼 제일 크게 나온다고 붙은 이름입니다.
>
> 이 이름을 알면 고칠 때 말이 달라집니다. "위에 좀 바꿔줘" 대신 **"히어로 영역 사진을 바꾸고 한 줄 소개를 짧게 해줘"** 라고 할 수 있습니다.

작가의 말에서 "Hero 페이지라는 단어조차 모를 수 있다" 고 했습니다. 바로 이 화면 입니다. **부를 이름을 알아야 정확히 고칠 수 있습니다.**

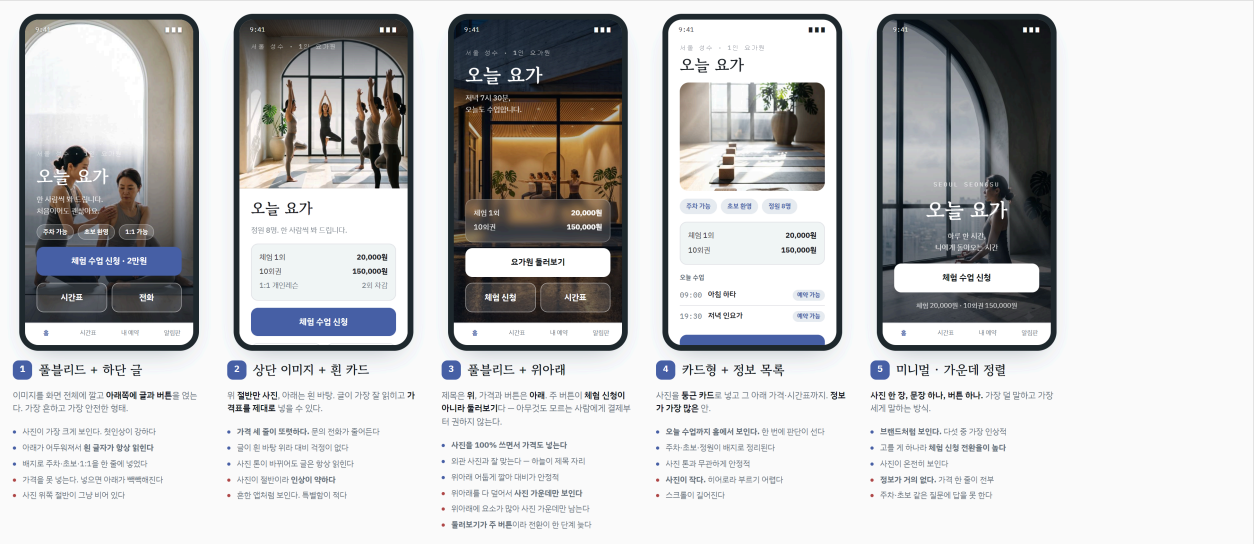

그림 10-4 히어로 시안 다섯 안. 사진을 화면 가득 까는 안(1·3·5)은 첫인상이 강하고, 사진을 줄인 안(2·4)은 가격과 정보를 또렷하게 넣을 수 있다. 첫인상과 정보량이 서로 부딪히는 축이다

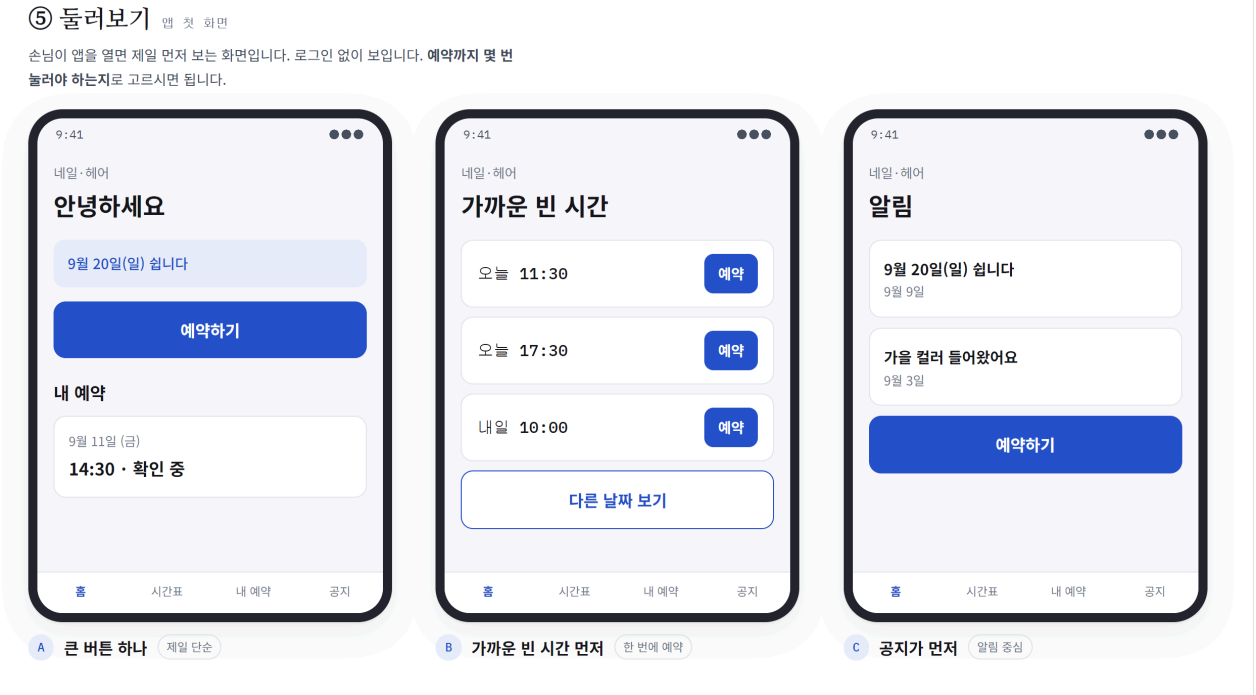

그림 10-5 손님 앱 첫 화면 세 안. A 큰 버튼 하나(제일 단순), B 가까운 빈 시간 먼저(열자마자 예약), C 공지가 먼저(휴무를 놓치지 않게)

첫 화면의 세 갈래는 이렇습니다.

안	판단	장점	약점
A 큰 표지 + 아래로 스크롤	맨 위를 **큰 색면 + 상호 + 한 줄**로 채운다	인상이 강하다	본론(오늘 열리는 것)이 한 번 스크롤 아래로 밀린다
B 표지를 반만	표지를 절반으로 줄이고 **바로 아래 오늘 것**	소개와 본론이 한 화면에	둘 다 어중간해 보일 수 있다
C 바로 본론	표지 없이 **상호 한 줄 + 오늘 열리는 것**	쓰던 사람에게 제일 빠르다	처음 온 사람은 뭐 하는 덴지 모른다

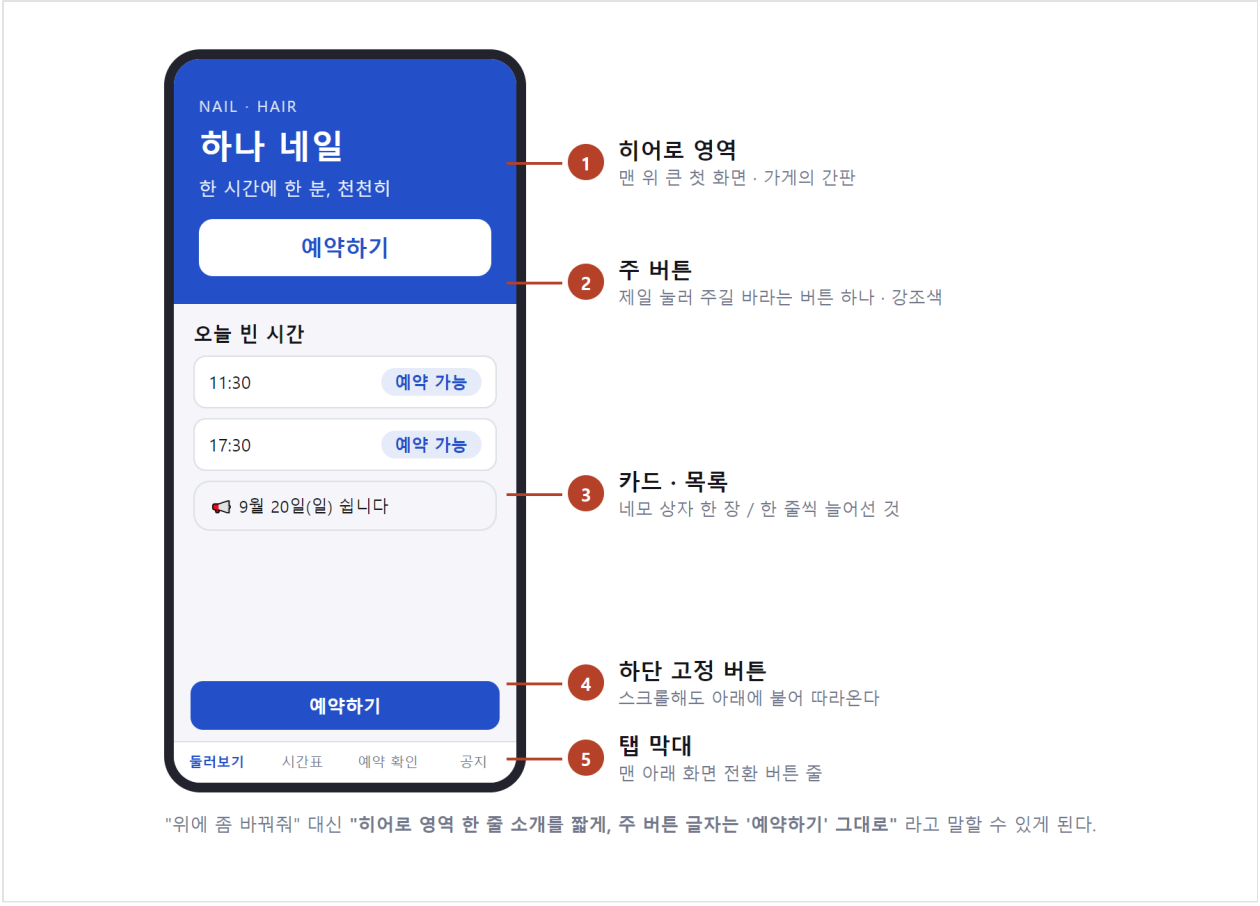

그림 10-6 앱 첫 화면의 부위별 이름. ① 히어로 영역 ② 주 버튼 ③ 카드·목록 ④ 하단 고정 버튼 ⑤ 탭 막대. 이름을 알면 고칠 곳을 정확히 말할 수 있다

사진은 폴더에 있는 것만

강의 폴더의 `images/` 에 **샘플 사진**이 들어 있습니다. 시안은 그것만 씁니다.

- **인터넷에서 사진 주소를 가져오지 않습니다.** 안 뜨거나, 나중에 깨집니다
- 주제와 딱 맞지 않아도 **분위기만 맞으면** 씁니다. 나중에 직접 찍은 걸로 바꾸면 됩니다
- 직접 찍은 사진이 있으면 `images/` 에 넣으세요. 그걸 씁니다
- 사진이 없거나 안 어울리면 **빈 네모를 두지 않고** 큰 색면 · 큰 글자 · 도형으로 채웁니다. `이미지 넣을 자리` 같은 빈칸은 **미완성처럼** 보입니다

10.8 색은 8-1에서 한 번만 고른다

8-1 시안 맨 위에는 **색 바꾸는 줄**이 붙어 있습니다. 스크롤해도 따라오고, 누르면 아래 모든 화면 색이 즉시 바뀝니다. 여기서 고른 색을 **8-2~8-4에 그대로** 씁니다.

고르는 것	보기
바탕 톤 4개	밝은 흰색 계열
강조색 4개	서로 충분히 다른 색
직접 고르기	다섯 가지 색을 하나씩 집는다 (아래)

준비된 여덟 개가 다 마음에 안 드는 사람이 반드시 있습니다. 그때는 **직접 고르기**를 누릅니다.

```
바탕색   [■]  #FBFCFD   화면 뒤쪽 전체
카드색   [■]  #FFFFFF   명단·카드 안쪽
글자색   [■]  #171D23   본문 글씨
강조색   [■]  #3E6E96   버튼 · 오늘 표시 · 선택된 것
선색     [■]  #E7ECF1   칸을 나누는 얇은 선
```

> 📖 **강조색과 색상 코드**
>
> **강조색**은 화면에서 **"여기 보세요"** 할 때만 쓰는 색입니다. 주 버튼, 오늘 날짜 표시, 선택된 칸. 적게 쓸수록 눈에 띕니다. 개발하는 사람들은 **포인트 색**, **primary(프라이머리) 색**이라고도 부릅니다.
>
> `#3E6E96` 같은 글자는 **색상 코드**입니다. `#` 뒤 여섯 글자가 빨강·초록·파랑을 얼마나 섞었는지 나타냅니다. 외울 필요는 없고, **"강조색을 #3E6E96 으로"** 라고 말하면 정확히 그 색이 된다는 것만 알면 됩니다.

[■] 네모를 누르면 색 고르는 창이 뜹니다. **처음부터 고르지 말고, 가까운 보기에서 출발해 마음에 안 드는 것만** 손보는 게 빠릅니다. 다 골랐으면 대화창에 `색 다 골랐어` 라고 한마디 치세요. Claude가 받아 적어야 8-2~8-4와 PRD에 들어갑니다.

10.9 화면 안에는 가짜지만 진짜 같은 데이터

시안은 빈 화면으로 나오지 않습니다. **예시 데이터**가 채워져 있습니다.

- 이름 5~6개, 시간 3~4개 — `김하나 · 이두리 · 박세라` 처럼 평범한 이름
- **한 회차 안의 화면들이 같은 데이터를 씁니다.** 로그인에 쓴 상호가 명단에도 그대로
- 로그인 화면은 아이디·비밀번호가 **채워진 채로**

- 한 사람씩 받는 주제(네일 · 반려동물 · 회의실)는 `2/8` 대신 **`예약 가능 / 마감`**
- 명단 모양은 PRD 3장대로 — 10명 아래면 **목록**, 수십 명이면 **표 + 검색**

빈칸만 있으면 밋밋해서 판단이 안 됩니다. 화면마다 다른 이름이 나오면 가짜처럼 보입니다. 그래서 **그럴듯하게, 일관되게** 채웁니다.

> ⚠ **예시 데이터를 진짜로 믿지 마세요**
>
> 시안의 가격 · 영업시간 · 전화번호 · 주소는 **Claude가 지어낸 것**입니다(3장 환각). 시안에서는 괜찮습니다. 그런데 2주차에 만들고 3주차에 올릴 때 **그대로 남아 있으면** 손님이 가짜 전화번호로 전화를 겁니다. **숫자와 이름은 사람이 바꿉니다.**

10.10 시안을 띄우는 두 길

아티팩트는 만든 사람의 Claude 계정에 남습니다. 나중에도 **링크로 다시 열어** 고른 안과 안 고른 안을 볼 수 있습니다.

순서	방법	이렇게 보인다
①	**아티팩트**로 띄운다	Claude가 링크를 준다. 누르면 브라우저에 시안 페이지가 열린다
②	안 되면 바로 **HTML 파일**로	`docs/30.output/` 에 파일이 생기고, Claude가 브라우저로 열어 준다

4장에서 봤듯이, **아티팩트**는 만든 화면을 링크로 열어 보는 미리보기입니다. 한 번 막히면 Claude는 붙잡고 씨름하지 않고 **바로 ②로 내려옵니다.** 강의 중이라 기다릴 시간이 없기 때문입니다.

파일로 갈 때는 이런 이름으로 네 개가 생깁니다.

```
docs/30.output/시안-8-1-웹-로그인명단.html
docs/30.output/시안-8-2-웹-주간알림판.html
docs/30.output/시안-8-3-앱-둘러보기시간표.html
docs/30.output/시안-8-4-앱-내신청알림판.html
```

A·B·C를 **파일 세 개로 쪼개지 않고 한 파일에 셋을** 넣습니다. 탭을 옮겨 다니면 비교가 안 되니까요.

10.11 고르는 법

대화창으로 돌아와서 한 줄

시안이 뜨면 Claude가 곧바로 이렇게 말합니다.

> "화면이 떴습니다. 위에서 아래로 내려가며 세 가지를 보세요. 다 보셨으면 **여기로 돌아와서 `로그인은 A, 명단은 C` 라고만 쳐주시면 됩니다.** 고민되시면 `C가 나은데 위쪽이 좀 답답해` 처럼 말로 하셔도 됩니다."

화면을 보는 동안 사람은 **대화창을 떠나 있습니다.** 돌아왔을 때 뭘 쳐야 할지 모르면 거기서 멈춥니다. 그래서 **돌아와서 칠 말**을 미리 알려 줍니다.

이렇게 치면	이렇게 된다
`로그인은 A / 1번`	그대로 확정
`C가 나은데 위쪽이 답답해`	**C로 확정하고, 그 불만 하나만 고쳐서** 다시 보여준다
`B랑 C 둘 다 좋은데 합칠 수 있어?`	합칠 수 있으면 합친다. 안 되면 뭐가 더 중요한지 하나만 묻는다
`셋 다 별로야. 다시 만들어줘`	그나마 나은 것 하나만 묻고, 고쳐서 다시
`잘 모르겠어`	**A로 두고** 넘어간다
(한참 대답이 없다)	**재촉하지 않는다.** 기다린다

페이지 맨 아래 고르는 칸

시안 페이지 맨 아래 ⑥ **고르는 칸**에서 골라도 됩니다.

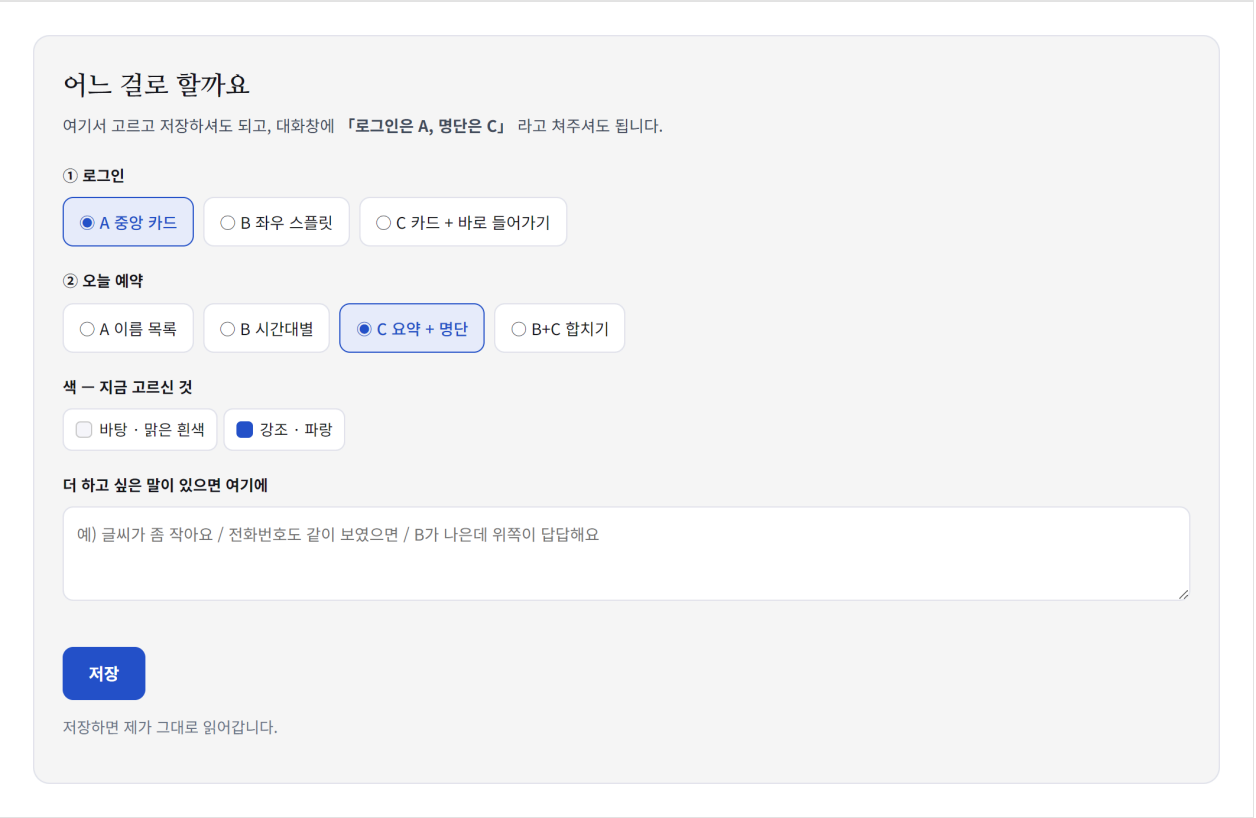

그림 10-7 시안 맨 아래의 고르는 칸. 화면마다 A·B·C 중 하나를 고르고, 지금 고른 색이 글자로 보이며, 「더 하고 싶은 말」 칸에 걸리는 점을 적은 뒤 저장한다

`( )` 와 `(•)` 는 **라디오 버튼**입니다. 옛날 라디오의 채널 버튼처럼 **하나를 누르면 나머지가 풀리는** 고르기 칸입니다.

> 💡 **왜 "더 하고 싶은 말" 칸을 꼭 두나**
>
> A · B · C 고르기만 있으면 **"이건 좋은데 이게 걸려요"** 를 말할 데가 없습니다. 사람은 셋 중 하나에 **억지로** 표를 던지고, 그 걸리는 마음은 **다음 주에 터집니다.** 만들어 놓고 나서 "사실 그때부터 글씨가 작았어요" 가 됩니다.
>
> 그래서 이 칸은 절대 빼지 않고, 적힌 건 **반드시** 받아서 처리합니다. 고칠 수 있으면 바로 고치고, 오늘 못 하는 것이면 「안 만들 것」에 이유와 함께, 뒤 단계 얘기면 「아직 안 정한 것」에 적습니다.
>
> **작은 불만을 그 자리에서 글로 남기는 것** — 이것도 만든 사람의 생각을 기록하는 일입니다.

저장을 눌렀는데 Claude가 조용하면, 대화창에 `로그인은 B, 명단은 A` 처럼 한 번 더 쳐 주세요.

10.12 회차가 끝나면

Claude는 **PRD를 아직 고치지 않습니다.** 고른 것만 `브레인스토밍.md` 에 한 줄씩 적습니다.

```
## 고른 화면
- 로그인 — A 중앙 카드
- 오늘 명단 — C 요약 + 명단
- 색 — 바탕 맑은 흰색 / 강조 파랑
```

- **고른 이유는 묻지 않습니다.** Claude가 짐작해 한 줄 붙입니다 — "숫자가 맨 위에 오는 쪽이네요"
- **안 고른 안도 이름만 적어 둡니다.** 11장에서 PRD로 옮깁니다
- **시안 파일은 지우지 않습니다.** `docs/30.output/` 에 그대로 둡니다

그리고 다음 회차 상자(8-2 → 8-3 → 8-4)를 붙여넣으라고 합니다. 네 번을 다 돌면 이렇게 말합니다.

> "여덟 화면이 다 정해졌습니다. 그런데 지금 `PRD.md` 는 **화면을 글로만** 적어둔 상태입니다.
> 방금 고르신 걸로 **PRD를 다시 써야** 합니다."

10.13 안 될 때

증상	이렇게 치세요
시안을 글이나 표로 설명한다	`글로 말고 실제 화면으로 그려서 보여줘`
아티팩트 링크가 안 열린다	`HTML 파일로 만들어서 브라우저로 열어줘`
웹 회차인데 폰 모양으로 그렸다	`8-1은 웹이야. 브라우저 창 안에 넓게 그려줘`
세 안이 색만 다르다	`색 말고 배치가 다른 세 안으로 다시 만들어줘`
사진이 안 뜬다 (엑스 박스)	`images 폴더에 있는 사진만 써줘`
두 개 다 마음에 든다	`둘 다 좋은데 합칠 수 있어?`
다 별로다	`셋 다 별로야. 다시 만들어줘`
8-1 끝났는데 8-2까지 가 버렸다	`8-1까지만. 내가 다음 상자 붙여넣을 때까지 기다려`

10.14 정리

- 시안은 **그림**이다. 생김새를 **눈으로 보고** 고르기 위한 것
- **두 장씩 네 번** — 웹 8-1·8-2, 앱 8-3·8-4
- **A는 항상 제일 단순한 안.** 모르면 A
- 앱 첫 화면은 **간판(히어로)** — 3초 안에 "여기 뭐 하는 데" 가 읽혀야 한다
- 고른 것과 **걸리는 한마디**를 남긴다. PRD는 다음 단계에서 고친다

화면을 고칠 때 부르는 이름

이 표가 이 장에서 가장 오래 쓸 표입니다. 2주차 이후에 "수정해줘" 를 **정확하게** 말하게 해 줍니다.

부르는 이름	가리키는 곳	이렇게 말한다
히어로 영역	앱·사이트 맨 위 큰 첫 화면	"히어로 영역 사진을 바꿔줘"
주 버튼	제일 눌러 주길 바라는 버튼 하나	"주 버튼 글자를 '예약하기' 로"
하단 고정 버튼	스크롤해도 아래 붙어 따라오는 버튼	"하단 고정 버튼은 그대로 둬"
탭 막대	앱 맨 아래 화면 전환 버튼 줄	"탭 막대에서 공지를 맨 오른쪽으로"
카드	화면에 떠 있는 네모 상자 한 장	"명단 카드 사이 간격을 넓혀줘"
목록 / 표	한 줄씩 덩어리 / 칸 맞춘 줄	"명단을 표로 바꾸고 검색 붙여줘"
브라우저 크롬 · 주소줄	웹페이지를 감싼 창 테두리	— (시안에서만 보인다)
경로	주소 뒤 `/login` 같은 방 이름	"/notice 화면에 글 목록 추가"
바탕색 · 강조색	뒤쪽 전체 색 · "여기 보세요" 색	"강조색을 #3E6E96 으로"
예시 데이터	시안에 채운 가짜 이름·시간	"예시 데이터를 실제 시간표로 바꿔줘"
라디오 버튼	하나만 고르는 동그라미	—
스크롤	화면을 위아래로 미는 것	"스크롤 안 해도 신청 버튼이 보이게"
px(픽셀)	화면의 가장 작은 점. 크기의 단위	"글자를 2px 키워줘"

다음 장은 1주차의 마지막 7분입니다. 화면을 보고 나서 **어긋난 PRD를 다시 쓰고, 왜 바뀌었는지 남깁니다.**

11장 — PRD 고쳐 쓰기, 왜 바뀌었는지 남긴다

> 초고 v0.1 (2026-09-14) · 1주차 9단계 · 1주차 마지막

11.1 화면을 보고 나면 문서가 안 맞는다

7단계에서 PRD를 썼을 때, 3장「화면」은 **글로만** 적혀 있었습니다.

8단계에서 실제로 화면을 보고 나니 달라졌습니다. **숫자가 맨 위에 오는 C안**을 골랐고, 색이 정해졌고, "전화번호도 같이 보였으면" 이라는 한마디가 붙었습니다. 어떤 건 **뒤집히기도** 했습니다. 글로는 목록이 좋아 보였는데, 보고 나니 표가 낫다든지요.

이대로 2주차에 들어가면, Claude는 **PRD를 읽고** 만듭니다. 고른 화면이 아니라 글만 보고요. 9단계는 이 어긋남을 되돌리는 7분입니다.

> **PRD는 한 번 쓰고 끝나는 문서가 아닙니다. 계속 고쳐 쓰는 문서입니다.**

11.2 따라 하기 — 질문은 없다

⌨ `제작prompt.md` 의 **9단계 상자**를 붙여넣습니다.

```
화면을 보고 나니까 아까 만든 설계 문서랑 안 맞는 것 같아.
고른 걸로 다시 써줘. 뭐가 왜 바뀌었는지도 같이 남겨줘.
```

질문이 0개입니다. 8단계에서 고른 것으로 다 채울 수 있습니다. 여기서는 읽기만 합니다.

Claude는 문서 전체를 쏟지 않고 **세 줄**로 말합니다.

> "3장을 고르신 시안으로 바꿨고, 5장에 색을 적었고, 7장에 만드는 순서를 매겼습니다."

그리고 **변경 이력 표만** 펼쳐 보여줍니다. 나머지는 파일을 열어서 보세요.

언제	버전	무엇	왜 ★
9/14	v0.1	처음 씀	7단계
9/14	v0.2	3장 화면을 고른 시안으로 교체, 5장에 색 추가	**화면을 실제로 보고 정했다**

"무엇" 은 파일을 비교하면 알 수 있다. **"왜" 는 적어 두지 않으면 영영 모른다.**

그림 11-1 화면을 보고 나서 PRD를 고쳐 쓴다. v0.1에서 글로만 적혀 있던「오늘 명단」이 v0.2에서는 고른 안과 안 고른 안(이유 포함)으로 바뀌고, 맨 아래 변경 이력에「왜」가 남는다

11.3 어디를 고치나

PRD	무엇이 바뀌나
3장 화면	글 설명 → **고른 안 이름 + 한 줄 설명.** 안 고른 안도 밑에 남긴다
5장 규칙	**색**(바탕 · 강조)을 적는다. 8단계에서 정해진 것
6장 안 만드는 것	시안을 보다가 "이건 빼자" 가 나왔으면 더한다
7장 만드는 순서	화면이 정해졌으니 **1번부터 8번까지** 순서를 매긴다
8장 아직 안 정한 것	풀린 건 지우고, 새로 생긴 건 더한다

3장은 이런 모양이 된다

```
### 로그인
고른 안 — A 중앙 카드
카드 하나를 화면 가운데. 상호 · 이메일 · 비밀번호 · 로그인 버튼.

안 고른 안 (남겨둠)
- B 좌우 스플릿 — 첫인상은 좋은데 10초 쓰는 화면에 과하다
- C 카드 + 빠른 입력 — 실제 손님에게는 보일 일이 없는 버튼이다
```

> 💡 **안 고른 안을 왜 지우지 않나**
>
> 지우면 **나중에 같은 제안이 또 옵니다.** 2주차에 Claude가 "로그인 화면을 좌우로 나눠서 왼쪽에 상호를 크게 넣으면 어떨까요?" 라고 합니다. 그럴듯하니까요(3장). 그러면 이미 한 고민을 **처음부터 다시** 합니다.
>
> 남겨두면 한 줄로 끝납니다. **"그건 B안이었고, 10초 쓰는 화면에 과해서 안 골랐습니다."** 사람도 AI도 이 줄을 읽고 넘어갑니다.
>
> 고른 것은 **무엇을** 만드는지 알려주고, 안 고른 것과 그 이유는 **왜 이렇게** 만드는지 알려줍니다. 뒤쪽이 만든 사람의 생각입니다.

11.4 변경 이력 — 왜 바뀌었는지 한 줄

9단계의 두 번째 일이자, 1주차에서 가장 작고 가장 오래 가는 것입니다. PRD **맨 아래**에 표 하나를 만들고 한 줄을 적습니다.

(그림 11-1 아래쪽 표가 그 모양입니다.)

> 📖 **변경 이력(change log)**
>
> 문서나 프로그램이 **언제, 무엇이, 왜 바뀌었는지** 한 줄씩 쌓아 두는 기록. 가장 중요한 칸은 「**왜**」입니다. 무엇이 바뀌었는지는 파일을 비교하면 알 수 있지만, **왜 바꿨는지는 적어 두지 않으면 영영 모릅니다.**

표의 `v0.1`, `v0.2` 는 **버전**입니다.

> 📖 **버전(version)**
>
> 같은 문서의 **몇 번째 판인지** 붙이는 번호. `v` 는 version의 첫 글자입니다.
>
> 앞자리 `0` 은 **"아직 완성 전"** 이라는 뜻으로 많이 씁니다. 뒷자리는 고칠 때마다 하나씩 올립니다. 오늘 v0.1로 태어나서 v0.2가 됐습니다. 2주차에 만들면서 또 고치면 v0.3이 됩니다. 앱이나 사이트를 정식으로 내놓을 때 **v1.0** 을 붙입니다.

변경 이력은 **매번 남깁니다.** 한 줄이면 됩니다. 나중에 **"이거 왜 이렇게 정했지?"** 가 반드시 옵니다. 그때 이 표 한 줄이 답입니다.

11.5 이것이 이 책이 말한 "유지보수" 의 시작이다

작가의 말로 돌아가 봅니다.

> AI는 이 사이트를 **처음에 왜 이렇게 만들었는지** 모릅니다. 그러면 AI는 **자기 생각대로** 고칩니다. 몇 번 "수정해줘" 를 반복하고 나면, 처음 사이트와는 완전히 다른 느낌의 사이트가 됩니다.

1주차가 끝난 지금, 폴더에는 그 "왜" 가 전부 적혀 있습니다. 같은 요청이 어떻게 달라지는지 봅시다.

그림 11-2 같은 "수정해줘" 라도, 기록이 없으면 AI가 그럴듯한 쪽으로 새로 만들고, PRD가 있으면 원래 생각(고른 안·색·안 만드는 것)을 지키며 고친다

> 📖 **유지보수(maintenance)** > > 한 번 만든 것을 **망가뜨리지 않고 계속 쓸 수 있게 돌보는 일.** 내용을 바꾸고, > 틀린 곳을 고치고, 필요한 걸 조금씩 더합니다. 집으로 치면 **수리와 관리** 입니다. > > 만드는 데 20분이 걸린 사이트도, 쓰는 동안은 **몇 달, 몇 년** 유지보수를 합니다. > 그래서 만드는 법보다 **고치는 법**이 더 오래 쓰입니다.

작가의 말에서 말한 **"만든 사람의 철학을 기록해 두고, 그 기록을 지키며 고친다"** 가 바로 이것입니다. 1주차 두 시간 동안 코드를 한 줄도 안 쓰고 **정하고 기록한** 이유가 여기서 완성됩니다.

> 💡 **기록은 AI만을 위한 게 아니다**
>
> 석 달 뒤의 **나**도 오늘의 나를 기억하지 못합니다. 왜 결제를 뺐는지, 왜 표지를 크게 했는지. 가게를 다른 사람에게 맡겨도 마찬가지입니다.
>
> PRD의 「안 만드는 것」, 「안 고른 안」, 「변경 이력」 — 이 세 곳의 **"왜"** 는 AI에게도, 미래의 나에게도, 다음 사람에게도 **같은 말**을 해 줍니다. 사람이 바뀌어도, 대화가 새로 열려도,

> **생각은 폴더에 남습니다.**

11.6 1주차 끝 — 폴더에 남은 것

Claude가 마지막으로 이렇게 말하고 **멈춥니다.** 만드는 건 2주차입니다.

> "**여기까지가 1주차입니다.** 오늘 코드는 한 줄도 안 썼는데, **뭘 만들지 · 뭘 안 만들지 · 어떻게 생겼는지**가 전부 정해졌습니다.
>
> **다음 주에 PRD를 보면서 이걸 그대로 만듭니다.**"

화면 시안 네 장은 **아티팩트로 내 Claude 계정에** 남아 있습니다. 링크로 다시 열 수 있습니다.

그림 11-3 1주차가 끝나면 남는 "만든 사람의 생각". CLAUDE.md(규칙), 브레인스토밍.md(과정), PRD.md(무엇을·왜), 참고링크.md(누구와 견줬나), 시안 아티팩트(어떻게 생겼나)

11.7 이번 주 과제

강의 안내에 적힌 1주차 과제입니다.

- 수업에서 9단계까지 못 갔다면 **이어서 끝내 오기**
- `PRD.md` 를 **한 번 읽어 보기** — 다음 주에 이 문서를 보면서 만듭니다
- 막힌 지점은 **카톡방에 남기기** — 다음 회차에서 함께 풉니다
- ⭐ **개발자 계정 신원 확인**이 아직이라면 반드시 이번 주에 마무리 (1장)

🔁 혼자 해보기

답을 알려주지 않습니다.

1. `PRD.md` 6장「안 만드는 것」을 열고, 이유 중 **하나를 내 말로** 고쳐 써 보세요. Claude가 적은 "사업자 등록과 심사가 필요합니다" 를, 내 가게 사정에 맞는 한 줄로요
2. 고쳤으면 변경 이력에 **한 줄을 직접** 더해 보세요. 버전은 몇이 될까요?
3. 새 대화를 열고 `PRD.md 먼저 읽고, 우리 첫 화면에 넣으면 안 되는 게 뭔지 알려줘` 라고 물어보세요. 방금 고친 이유가 답에 나오나요?

11.8 안 될 때

증상	이렇게 치세요
새 시안을 또 만들려 한다	`시안은 그대로. PRD만 고쳐 써줘`
안 고른 안이 지워졌다	`안 고른 안도 이름이랑 이유 한 줄씩 3장에 남겨줘`
변경 이력이 없다	`PRD 맨 아래에 변경 이력 표 만들고 v0.2 한 줄 적어줘`
변경 이력에「왜」가 비었다	`변경 이력의 왜 칸을 채워줘. 화면을 보고 바뀐 이유로`
10단계로 넘어가 코드를 쓰려 한다	`1주차는 9단계까지야. 여기서 멈춰줘`

11.9 정리

- 화면을 보고 나면 PRD가 **어긋난다.** 그래서 **다시 쓴다** (v0.1 → v0.2)
- 3장에는 **고른 안**과 **안 고른 안 + 이유**를 같이 남긴다
- 맨 아래 **변경 이력** — 언제 · 무엇 · **왜.** 매번 한 줄
- 이 기록이 2주차 이후의 **"수정해줘"** 를 원래 생각에 맞춰 준다 — **유지보수**

이 장에서 나온 말

말	쉬운 뜻
고쳐 쓰기(개정)	이미 쓴 문서를 새로 알게 된 것에 맞춰 다시 쓰는 것
변경 이력	언제 · 무엇이 · 왜 바뀌었는지 한 줄씩 쌓는 표
버전 · v0.1	문서의 몇 번째 판인지. 앞자리 0은 "완성 전"
v1.0	정식으로 내놓는 첫 판에 붙이는 번호
유지보수	만든 것을 망가뜨리지 않고 계속 쓸 수 있게 돌보는 일
철학	무엇을 왜 만들고, 왜 안 만들고, 왜 이 안을 골랐는지 — 만든 사람의 생각

1주차가 끝났습니다. 다음 주에는 이 PRD를 들고 **내 폰에서 돌아가는 앱**을 만듭니다.
