FREE GUIDE

The Founder's Guide to Prompting

Get the most out of Claude Code and Codex, even if you've never written a line of code. This is the way my co-founder Jimmy and I actually write prompts, with the engineering jargon stripped out and every example grounded in the kind of work a founder does all day.

By Preston Chin · ~12 min read

Start here

This guide is for founders and operators who use AI to get real work done but don't come from an engineering background. You will not need to write any code to use a word of it.

Quick orientation. Tools like Claude Code and Codex are AI *agents*. Unlike a normal chat window, they can read your files, run tasks, and change things on your computer. That makes them powerful, and it makes how you talk to them matter far more. A sloppy prompt in a chat costs you a minute. A sloppy prompt to an agent can send it down the wrong path for an hour and leave you cleaning up.

None of this is a rule. It's a set of habits. Read it once, use the parts that fit how you work, and steal the copy-paste blocks. If you only have two minutes, skip to the cheat sheet at the bottom. Everything above it is the why.

1. A prompt is a brief, not a wish

The single biggest shift is to stop treating the AI like a search box and start treating it like a brilliant new hire. That framing comes straight from Anthropic's own guidance: think of the model as a talented but new employee who lacks context on your norms and workflows. It's capable, but it doesn't know what you know, and it won't ask unless you set it up to.

A wish sounds like "make me a launch plan." It leaves the model guessing at your product, your audience, your timeline, and what a good plan even looks like to you. A brief tells it what you want, why you want it, and how you'll both know it's done. Same model, completely different output.

Anthropic's golden rule is the best gut-check there is: show your prompt to a colleague who has no context on the task and ask them to follow it. If they'd be confused, the AI will be too.

2. The anatomy of a strong prompt

You don't need to fill in all of these every time. But when a task matters, walking these seven lines turns a wish into a brief. Copy this and keep it somewhere:

PROMPT SKELETON Copy

```
Goal: what you want, in one or two sentences.
Context: the background, docs, numbers, and examples it needs. Paste them in.
Constraints: the rules it must follow. Tone, length, what to avoid.
Out of scope: what it should not touch or do.
Process: research first, show me a plan, wait for my go, then do it.
Done when: how we'll both know it's finished.
Output: the exact format you want back.
```

Here it is filled in for something a founder actually does, a monthly investor update:

WORKED EXAMPLE Copy

```
Goal: Draft my monthly investor update from the raw numbers and notes below.
Context: [paste this month's revenue, active users, cash balance, and 3 bullet
  points on what happened]. My last update is also pasted so you can match the
voice.
Constraints: Under 400 words. Honest and specific, no hype. Lead with the
number
  that matters most. Keep my plain, direct tone.
Out of scope: Don't invent metrics I didn't give you. Don't add a section I
don't have.
Process: First tell me the one headline you'd lead with and why. Then draft.
Done when: It reads like something I'd actually send, and every number traces
to my notes.
Output: The email, plus a one-line subject.
```

Notice the strong version doesn't just add words. It gives the model a boundary, a safety rule ("don't invent metrics"), and a clear finish line.

3. Weak vs strong, side by side

The difference is easiest to feel with a before and after. Say you're launching a new product.

Weak:

```
Write a launch email for my new product.
```

Strong:

STRONG Copy

```
Goal: Write a launch email announcing [product] to my existing newsletter list.
Context: [paste what the product does, who it's for, the price, and the one
problem
  it solves]. Here's a past email of mine that got a good response, match this
voice: [paste].
Constraints: Under 200 words. One clear call to action. No emojis. Sound like a
  person, not a press release.
Out of scope: Don't make up features or testimonials.
Done when: A reader who's never heard of this understands what it is and why to
click.
Output: Subject line, preview text, and the email body.
```

The strong prompt gives the model a target and a fence. That's the whole difference.

4. Give it a role, and tell it why

Two small moves with an outsized payoff, both from Anthropic's playbook.

Give it a role. One sentence at the top: "You're a demand-gen marketer who has launched dozens of B2B products." Anthropic notes that even a single line of role setting focuses the model's tone and judgment on your use case. It's the difference between generic advice and advice from someone who's done the thing.

Tell it why. Don't just say what to do, say the reason. Anthropic's own example: instead of "never use bullet points," write "never use bullet points, because this is going in a warm personal email and

bullets make it feel like a memo." The model generalizes from the reason and makes better calls on the edge cases you never spelled out.

5. Tell it what mode it's in

Most agent screwups are mode-confusion screwups. You wanted it to think, it started doing. You wanted a quick read, it rewrote everything. Name the mode in the first line and most of that disappears.

- **Explore:** "Just read these and tell me what's here. Don't change anything yet."
- **Plan:** "Give me a plan and the risks. Don't do it yet."
- **Do:** "Make the smallest version that works."
- **Review:** "Look at this like a skeptical expert and tell me what's wrong with it."

One habit, fewer surprises. If you catch yourself frustrated that the AI "did too much," it's almost always because you never told it which of these you wanted.

6. The three power moves

If you take three things from this whole guide, take these. Each one is a copy-paste prompt. They work in any AI chat, and they're even better inside Claude Code or Codex.

Make it interview you

PROMPT Copy

```
I want to [your goal here]. I have a rough idea but not enough detail.

Before you do anything, ask me the questions you need to turn this into a
finished plan: who it's for, the constraints, the edge cases, and the spots
where this could go wrong. Ask a few at a time, and don't start until I say go.
```

Why it works. When your intent is fuzzy, don't pretend it's crisp. This turns the AI into a thinking partner that catches your blind spot before it runs off and builds the wrong thing.

Make it say your plan back

PROMPT

Copy

```
Before you do anything, say my plan back to me in your own words: what you think I'm asking for, what you're assuming, and where you think I might be wrong. Don't start until I confirm you've got it.
```

Why it works. The gap between what you meant and what it actually heard shows up instantly. Tip: word-vomit the plan out loud first with a dictation tool. You talk about three times faster than you type.

Make it show you the answer

PROMPT

Copy

```
Don't give me a wall of text. Turn your full response into one clean, self-contained web page I can read in seconds: clear headings, the key points pulled out, and any comparison as a simple table.
```

Why it works. You read a clean page in seconds instead of scrolling paragraphs. This is the fastest way to actually check the AI's work instead of skimming and hoping.

> **The first move is built into Claude Code and Codex**
>
> Both tools have a mode that makes them plan before they touch anything. In **Claude Code** it's called Plan Mode: press `Shift+Tab` to cycle the mode until the status bar reads *plan mode on* (it goes default → auto-accept → plan), or start the whole session in it with `claude --permission-mode plan`. A `/plan` prefix handles a one-off.
>
> In **Codex**, type `/plan` to draft first, or switch to read-only with `/permissions` so it can look but not change anything until you say go.

7. Make it prove it, not promise it

The AI sounds equally confident whether it's right or completely made up. Your job is to make it show its work.

Don't ask "are you sure?" It'll say yes. Ask "what are you basing this on?" Make it point to the doc, the number, or the source you gave it.

Make it separate facts from guesses. A line that saves you constantly: "Before you answer, list what you know from what I gave you, what you're assuming, and what you're unsure about." Now you know exactly which parts to trust.

Give it permission to say "I don't know." Anthropic's docs are explicit that telling the model it can express uncertainty instead of guessing cuts down on made-up answers. Add "if you don't have enough to answer, say so and tell me what you'd need."

This matters most on the stuff founders lean on AI for: summarizing customer calls, sizing a market, pulling themes from feedback. That's exactly where a confident wrong answer does the most damage.

8. Keep it in scope

Agents love to over-deliver. You ask for a small tweak and get a full rewrite. Anthropic's guidance is blunt about this: avoid over-engineering, and only make changes that are directly requested or clearly necessary. So build a fence.

SCOPE FENCE

Copy

```
Make the smallest change that does what I asked. Don't redo the parts I didn't
mention, don't "improve" things I didn't ask about, and don't add features I
didn't request. If you think something else needs changing, tell me instead of
doing it.
```

"Tell me instead of doing it" is the quiet hero of that block. You still get the AI's good ideas, you just get to approve them.

9. Show it what "good" looks like

When something's easier to show than to describe, show it. Paste one or two examples of the output you're after: a past email you loved, a competitor's landing page, the format of an investor update that worked. This is called few-shot prompting, and it's one of the most reliable ways to steer tone and structure.

Anthropic's rule of thumb: start with one example, and only add more if the output still isn't right. Three to five good, varied examples beat a paragraph of description every time. Pick ones that cover the range you care about so the model doesn't copy a pattern you didn't mean.

10. Make it automatic

You don't want to paste these habits every time. Once you know they work, you can wire them in so they apply on their own. This part is specific to Claude Code and Codex, because both read a plain instructions file before they start and treat whatever's in it as standing orders for every prompt.

Claude Code: the file is `CLAUDE.md`. Make a plain text file named `CLAUDE.md` in your project folder. Claude reads it automatically at the start of every session. Want it everywhere, not just one project? Put it at `~/.claude/CLAUDE.md`. Brand new? Run `/init` and it writes a starter for you.

Codex: the file is `AGENTS.md`. Same move. A file named `AGENTS.md` in your project folder, or `~/.codex/AGENTS.md` to make it global. Codex loads it on startup and puts it above every task.

Here's a starter block you can drop straight in:

CLAUDE.MD / AGENTS.MD

Copy

```
## How I want you to work

- Before you build on a rough request, interview me first. Ask the questions
you need to turn it into a finished plan, and don't start until I say go.
- When I give you a plan, say it back to me in your own words before you act.
Flag anything you're assuming or think I've got wrong.
- Make the smallest change that solves the stated problem. Don't touch what I
didn't ask about, and tell me instead of doing extra.
- Base claims on what I actually gave you. If you're not sure, say so instead
of guessing.
- When you're done, if the answer is long, turn it into a single clean, self-
contained web page I can scan: clear headings, key points pulled out,
comparisons as tables.
```

Why this works. These tools don't remember your preferences on their own from one session to the next. That file is their memory. It's the first thing they read, so it sits above every request and shapes all of them. That's the difference between a good habit you have to remember and one the tool just does for you.

The one-page cheat sheet

Screenshot this. Before you hit enter on anything that matters, run the list:

- ✓ Gave it a **role** and the **goal** in one line.
- ✓ Pasted the **context**, and said **why** it matters.
- ✓ Set the **constraints** and what's **out of scope**.
- ✓ Named the **mode**: explore, plan, do, or review.
- ✓ For anything fuzzy, made it **interview me** first.
- ✓ Asked for a **plan** before the work.

- ✓ Made it **flag assumptions** and cite what it's basing things on.
- ✓ Said what **done** looks like and the **format** I want back.

That's the whole craft. Not magic wording, just telling a capable assistant what you actually want, why, and what finished looks like. Do that and you'll get the answer you were after the first time, on the work that matters.