A FREE GUIDE FROM KRISTEN PARISE

Get More From Claude

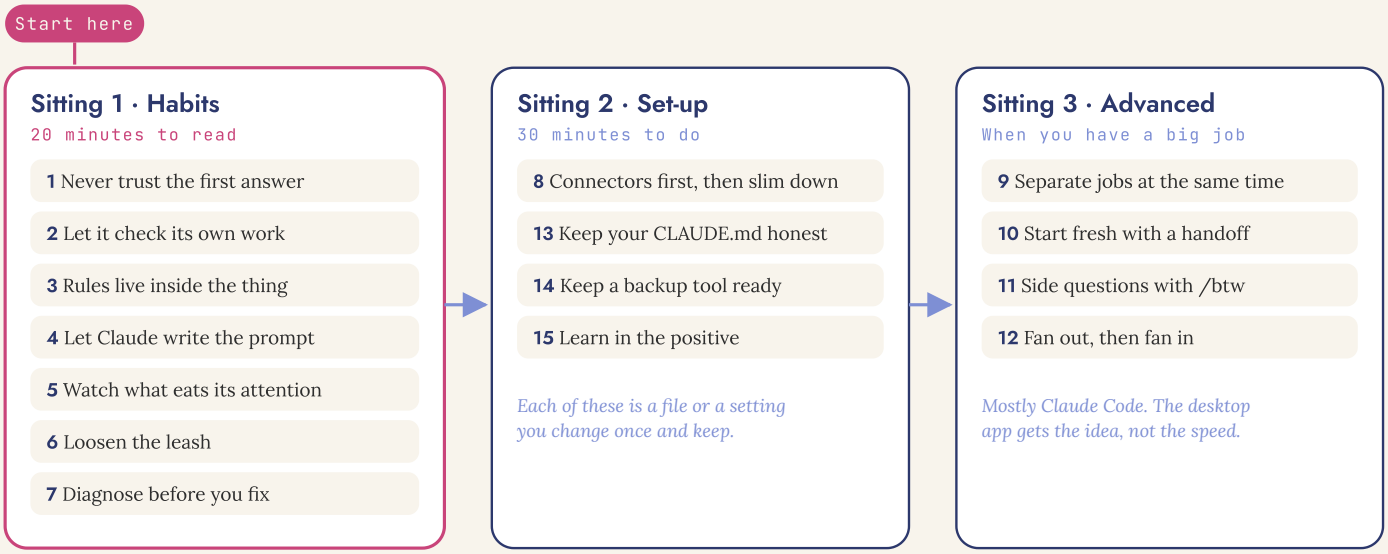
Fifty moves from a $30,000 education, in plain words. Fifteen principles, six prompts, three sittings, and a steadier Claude by Friday.

FOR BUSINESS OWNERS WHO USE CLAUDE AND DO NOT WRITE CODE

TIME THREE SITTINGS, ABOUT AN HOUR UPDATED 26 SEPTEMBER 2026

On the 26th of September I asked Claude to show me what it was carrying before I typed a word. One hundred plugins. Eighty-one of them I had installed in one click back in June, and the list included legal research, Kubernetes and financial-modelling tools my pelvic health clinic will never use. Every one of them had been eating Claude's attention on every chat, all summer. Turning off 77 of them took one chat. That was the fifth point on Nick Saraev's list, and it was the biggest fix on it.

Nick Saraev has spent more than $30,000 and about a thousand hours on Claude Code. His video, *I've spent over $30,000 on Claude Code*, is the fifteen things he wishes he had known at the start. The ideas here are his. The plain words and the mentoring frame are mine, because most of the people I coach do not write code and still need every one of these. I have kept his order, simplest to most technical, and numbered the moves 1 to 50 myself, so you can say "I'm on move 23" and we both know where you are.



The sections below run in Nick's order, 1 to 15. Follow the map and jump. Every chip is a link.

The route. Three sittings, fifteen principles, fifty numbered moves.

You do not need to be technical. I am a pelvic health physiotherapist who learned this by paying for it, one chat at a time. You need a Claude you can open, one real task you do every week, and the patience to run a prompt more than once.

WHAT YOU END UP WITH

A prompt you can trust, because you tested it more than once and kept the version that scored higher.

A leaner Claude that starts faster and thinks clearer, because you turned off what it was carrying for no reason.

A handoff habit, so a long job never turns to mush. You close the chat, fix the summary, and start clean.

Two doors, in plain words

Every move here works in one of two places, and some only in one. I say which under each principle.

Door	What it is	What it has
The Claude desktop app (or claude.ai)	The chat window most people use. Connectors to Gmail, your calendar and your files live in Settings.	Every prompt in this guide. None of the slash commands.
Claude Code	Claude running in a terminal, inside a folder on your computer. It reads a file called `CLAUDE.md` at the start of every chat and can run several helpers at once.	Everything, including `/context`, `/compact`, `/btw` and `/insights`. To install it, ask the desktop app: "Walk me through installing Claude Code."

Before you start

- ☐ **Claude, signed in.** The desktop app covers sittings one and two. Claude Code opens the rest.
- ☐ **A preferences file,** or the willingness to make one. In Claude Code it is `CLAUDE.md` in your folder. In the desktop app it is the instructions box of a Project, or your preferences in Settings. If you have neither, principle 13 makes one.
- ☐ **One real task you do every week** to practise on. Mine is the weekly review of my clinic and shop numbers. Pick something you will do anyway.

1 of 15 Never trust the first answer

Claude is not a calculator. The same words in can give a different answer out, so one great result proves nothing.

1. **Same words in, different answer out.** Run a prompt twice and you get two answers, sometimes close, sometimes nothing alike. Never turn a prompt into a team rule because it worked once.
2. **Run it ten times and count.** Seven out of ten is your real score, not the lucky run you remember.
3. **Change one thing, run again, compare the counts.** Keep the higher number. Nick calls these test runs "evals". You are being a scientist for ten minutes.
4. **Shrink the gap between the best run and the worst.** You have to live with the worst run too, so steady beats brilliant.

Every prompt I hand a mentoring client now gets three runs on the same example before it leaves my hands. Both doors.

2 of 15 Give Claude a way to check its own work

A first draft is rough, for people and for Claude. Quality comes from the check-and-fix loop.

5. **Think in drafts.** One attempt is a first draft, so expect first-draft quality.
6. **Build in a loop.** Draft, check, fix, check again. Quality climbs every time round.
7. **Give it something concrete to compare against.** An example you love, a checklist, a screenshot of the target, a test the result has to pass.
8. **A rough checklist with a fix loop beats a perfect prompt with no loop.** Something to compare to is what changes the result.

Every email I draft with AI runs through a small checker of my own banned words before I read it. It catches what I have stopped seeing. Both doors.

3 of 15 Let the thing itself carry the instructions

Separate notes drift. Put the rules inside the file they are about.

9. **Separate notes drift.** The plan says version 1 while the real thing is on version 4, and Claude trusts the wrong one.
10. **Put the instructions inside the file they describe.** Rules in the template, the checklist in the document, comments in the code. Then Claude has to read them every time it works there.
11. **One exception: your preferences file.** `CLAUDE.md` holds how you like to work, your lessons, your rules. It does not describe what the thing does.
12. **Feed the source.** Paste the transcript, the email, the brief. Your summary of it loses the parts you did not know mattered.

4 of 15 Let Claude write the prompt

13. **Stop writing the perfect prompt.** Say "I want X for Y audience. Help me write the prompt." Claude is now better at prompting Claude than you are.
14. **Answer its questions.** Who it is for, what great looks like, whether you have an example. Then use the prompt it writes.
15. **Skip this for quick daily asks.** Use it for anything large, repeated or expensive.

Both doors. Run it in a fresh chat, and when the prompt comes back, run that in another fresh chat (move 19).

PROMPT 1 · WRITE THE PROMPT FOR ME

I want *[what you want, for example a weekly update email to my team]* for *[who it is for, for example my five-person clinic team]*. Help me write the prompt. Ask me what you need to know first.

1. Ask me, all at once: what a great result looks like, whether I have an example I love, and anything else you need to know. Wait for my answers.
2. Write the prompt I should use, addressed to you, with my answers built in and any fill-ins left in *[square brackets]*.
3. Tell me in one line to paste it into a fresh chat, and anything I need open first (a file, a folder, a connector).

Then stop. Next: run the prompt in a fresh chat, and bring the result to Prompt 2.

What you should see. Three questions, then a prompt longer and more specific than the one you would have written, with your example folded in. If Claude skips the questions and writes the prompt straight away, say "ask me the three questions first" and let it start over.

5 of 15 Watch what eats Claude's attention

About a third of Claude's attention is used up before you say a word. This is the principle that found my hundred plugins.

16. **Type** `/context`. It shows what is loaded, tools, connectors, memories and skills, and how much room is left for you.
17. **Turn off connectors and plugins you are not using today.** Every one you load costs attention and slows the start of every chat.
18. **Longer chats cost more and get dumber, both at once.** Nick calls this context rot. It cannot be avoided, only managed.
19. **Copy the good prompt into a fresh chat.** When Claude has spent ten turns helping you shape a brief, do not run it in the same tired chat.
20. **Compact early.** `/compact` folds the chat so far into a short summary and keeps going. Nick does it long before Claude offers to.

> WHICH DOOR
>
> **Claude Code** has `/context` and `/compact`. To switch things off, type `/plugin` for plugins and `/mcp` for connectors.
>
> **The desktop app and claude.ai do not have these commands.** The nearest thing to `/context` is opening Settings, then Connectors, and counting what is switched on. The nearest thing to `/compact` is a new chat with the good prompt pasted in.

6 of 15 Loosen the leash

21. **Treat Claude like a contractor.** A contractor has their own good process from hundreds of jobs. Tell them the outcome and skip the steps.
22. **Say what done looks like.** "You are done when this is true, that is true and that is true."
23. **Drop the long "never do this" lists.** Those were for older models. Today's Claude infers most of it.
24. **One line beats ten rules.** "If you are not sure, ask me. This matters." Put it in your preferences file once.

Every repeatable job I give Claude now opens with a Done when line. Both doors.

7 of 15 Diagnose before you fix

25. **Say "list every problem you see. Do not fix anything yet."** Claude has its own idea of what counts as a problem, and it is not always yours.
26. **Cross out the ones you disagree with.** "Too casual" might be exactly the voice you wanted.
27. **Hand back only the problems you want solved.** Half the tokens, half the rework, and nothing you liked gets fixed.

Both doors. Paste a draft, or point Claude at the file.

PROMPT 2 · DIAGNOSE, DON'T FIX

List every problem you see with this. Number them. Do not fix anything yet. I will tell you which ones to fix.

[paste the draft here, or name the file if I am working in a folder]

1. One line per problem, numbered, most serious first. Say what is wrong, not how you would fix it.
2. Wait for my list of numbers.
3. Fix only the ones I name, with the smallest change that solves each. Leave everything else exactly as it was, including anything you flagged that I did not pick.

Report what you changed, by number. Then stop.

What you should see. A numbered list, a pause, then a version with only your picks changed. If Claude "tidies" a line you did not pick, say "put line X back exactly as it was" and it will.

8 of 15 Connectors first, then slim down

A connector is the one-click link to Shopify, Gmail or your calendar. A skill is a short written recipe Claude follows, in your words.

28. **Prototype with a connector.** One click, sign in, test. Prove the idea works before you build anything.
29. **Once it works and matters, turn it into a lean skill.** Only the steps you need. A connector carries every tool it has whether you need it or not, and slows every start-up.
30. **Prune the connector list.** Nick saves seconds on every launch and a lot of attention in every chat.

Connectors live in both doors. A skill is a file, so it needs Claude Code or a desktop app chat opened in a folder.

9 of 15 Run separate jobs at the same time

31. **Batch your asks up front.** "Fix the hero, the form and the logout button" in one message, instead of three messages waiting on each other.
32. **Say "do these in parallel as mutually exclusive scoped tasks."** Those exact words tell Claude to split the work so no two helpers touch the same thing.
33. **Add a merge step at the end.** One pass that brings the pieces together.
34. **Accept a little more error for a lot more speed.** For most work that is a good trade.

Claude Code runs the helpers side by side. In the desktop app the same words still work, but the tasks run one after another, so you save your own waiting time and nothing more.

PROMPT 3 · PARALLEL WORK (CLAUDE CODE)

Do these in parallel as mutually exclusive scoped tasks, then merge:
1. *[task, naming the file or piece it may touch]*
2. *[task, naming a different file or piece]*
3. *[task, naming a different file or piece]*
Merge means: *[what you want at the end, for example one page called merged.md that puts the three together in that order, or "check the three agree and fix any seam"]*.

Rules: no two tasks touch the same file. If a task finds something outside its own file that now contradicts it, leave that file alone and put it in the report. When all three are done, do the merge as one pass.
Report the three results and what the merge changed, in five lines or fewer. Then stop.

What you should see. Three results arriving together, then the merged result you named. If two tasks needed the same file, Claude should tell you before it starts. If the merge came back as a check instead of a file, your "Merge means" line was too vague. Name the file.

10 of 15

Start fresh with a handoff note

35. **Long chats collect contradictions.** You said "not X" on turn three and "a bit like X" on turn twelve. Claude splits the difference, and the result is mush.
36. **Ask for the handoff.** What is done, what decisions you need to make, what is next, what problems are open.
37. **Fix the summary yourself before you hand it over.** This is your one chance to correct the record.
38. **Paste it into a new chat.** Claude recognizes a handoff, asks about anything missing, and works from a short clean brief. Cheaper and sharper.

I turned this into a one-word command the day I watched the video. Both doors. If the new chat cannot open files, paste the whole summary in.

PROMPT 4 · HANDOFF

Summarize where we are: what is done, what decisions I still need to make, what is next, and what problems are open. Write it so a fresh chat can pick up without reading this one.

1. Go back to the start of this chat (or the start of the summary I pasted), not the last few turns. Use these five headings: Done (with file names or links), Decisions (two lists, Made and Open), Next (the first step concrete enough to start on with no question), Open problems (say "I assumed" wherever you guessed), Rules I stated in this chat (my corrections, and anything I said to leave alone). Where I changed my mind, keep only the final version.
2. Leave out every client or patient name and any health detail about a person.
3. Show me the five sections and ask me to correct anything wrong or missing. Wait.
4. After my corrections, save it as handoff-*[today's date, written* YYYY-MM-DD*]*.md if you can write files, and print a short opener I can paste into a new chat, under 400 words, that begins "Picking up from a handoff". If you cannot write files, say so, and make the opener carry all five sections instead, under 600 words.
Then stop. This chat is done.

What you should see. Five headings, a pause for your corrections, then a short paste-ready opener. Read the Decisions list hardest. If an early instruction you later reversed is still there, that is the contradiction move 35 warned about. Fix it now.

11 of 15 Ask side questions with /btw

39. **While Claude works, type `/btw` and ask what it is doing.** It answers without stopping the job, and without your question polluting the main task.

Claude Code only. In the desktop app, wait for the job to finish, or ask your side question in a second chat.

12 of 15 Fan out, then fan in

40. **Cheap helpers scan widely, the strong model decides.** Research is expensive when the top model does it. Let small, cheap helpers read the field and report back.
41. **Ask for it by name.** More coverage, faster, cheaper.

Claude Code, where the helpers exist. The desktop app covers the angles one at a time.

PROMPT 5 · FAN OUT, FAN IN (CLAUDE CODE)

Use a fan-out and fan-in strategy. Delegate to cheap subagents to find as much as possible about *[the question, for example ways a small clinic can cut no-shows]*, including approaches most people don't use. Then combine it all into one brief and make the recommendation.

1. Split the field into at least four angles and send one helper per angle. Each helper reports a list of approaches, one line each, with a source or example where it can find one. Mark any source it could not check as unverified.
2. Combine everything into one brief: every approach found, grouped, with the three least common marked.
3. Make the call: one recommendation, why, and what it would take to try it within a week.

Save the brief as *[filename]* and give me the recommendation in three lines. Then stop.

What you should see. A brief with more approaches than you expected, three of them ones you had not heard of, and one clear call. If every idea is one you already knew, the helpers played it safe. Say "go again, only the uncommon ones this time".

13 of 15 Keep your CLAUDE.md honest

Your preferences file is read at the start of every chat. In Claude Code it is `CLAUDE.md` in the folder. In the desktop app it is a Project's instructions. Whatever it is called, it is the one file Claude believes without checking.

42. **Old facts make bad decisions.** Nick left months-old revenue numbers in his and got advice for the smaller business he used to run.
43. **Audit it now.** Every line either earns its place or goes. A character Claude does not need is attention wasted.
44. **The shape that works:** what is where, your preferences, what it may do without asking, who you are, lessons learned.
45. **Re-read it every time the model updates.** Yesterday's rules were written for yesterday's model.

Both doors. Mine gets a dated line at the top of every block of numbers now, so Claude can see how old they are.

14 of 15 Keep a backup tool ready

46. **Have a second tool set up in the same folder.** When one service has an outage, whole companies stop. Yours does not have to.
47. **Share the rules.** A file called `AGENTS.md` that mirrors your `CLAUDE.md` means the backup knows everything Claude knew.

For Claude Code users working in a folder. In the desktop app, keep the same preferences in another tool's project too.

15 of 15 Learn from mistakes, in the positive

48. **Write the solution, not the ban.** "Batch the file reads" teaches. "Don't retry the old API" only forbids.
49. **After any build, ask "how could you have done that faster with fewer tokens?"** Then put the answer in your preferences file. Nick says this saves him three or four hours a week.
50. **Use `/insights` to see the patterns you would never notice yourself.** Small conveniences compound. "Open the file when you are done" is a rule that pays every day.

`/insights` is Claude Code. In the desktop app, end the week by asking what you made Claude redo and what one rule would have prevented it. Prompt 6 works in both doors.

PROMPT 6 · RETRO

How could you have done that faster with fewer tokens? Give me one lesson, written as what to do next time, and add it to my preferences file.

1. Look back over *[the whole build in this chat, or name the build]* with evidence: a file read twice, a wrong path tried, a check skipped, rework after I corrected you. List up to five, one line each. The lesson may be for you, or for how I asked.
2. Pick the one most likely to come up again. Write it as a single instruction in the positive, under 30 words. "Batch the file reads", never "Don't retry".
3. Add that one line under a heading called Lessons in my preferences file (*[CLAUDE.md, or the file where I keep how I like to work]*). Change nothing else in that file. If you cannot write files, print the line and tell me to paste it into my Project instructions. Report the lesson and where you put it, in two lines. Then stop.

What you should see. A short list with evidence, one lesson that starts with a verb, and a new line in your file. If the lesson starts with "Don't" or "Never", send it back: "rewrite that as what to do".

Three mistakes I made so you don't have to

1. **I trusted prompts after one good run.** Three runs on the same example is my floor now, and a prompt that scores two out of three does not go to a client.
2. **I let the tool list grow without looking.** One hundred plugins, 81 of them from a single click in June, and every chat all summer paid for them. Move 16 takes ten seconds.
3. **I pushed long chats instead of handing off.** A handoff note and a fresh chat take two minutes. I spent months not knowing that.

Pick one move today. Move 16 takes ten seconds: open Claude Code, type /context, and look at what is eating the room before you say a word. Tomorrow, hand your next big ask to Prompt 1. By Friday you will have a prompt you tested, a leaner Claude and a handoff habit, and that is a steadier week than most people get from a year of guessing.

If you want me beside you while you put these to work, start with the free audit at The Bottom Line or write to me at kristen@blueberrytherapy.ca.

© 2026 Kristen Parise BSc MSc BHScPT

Instead of whispering, I became a storm that couldn't be silenced.