APPENDIX B

Cheat Sheets and Quick-Reference Cards

These cards gather the frameworks you'll reach for most, one per topic, with the chapter that teaches each. Copy or print the ones you use; the companion page has printable versions.

B.1 The SCOPE prompt (Chapter 6)

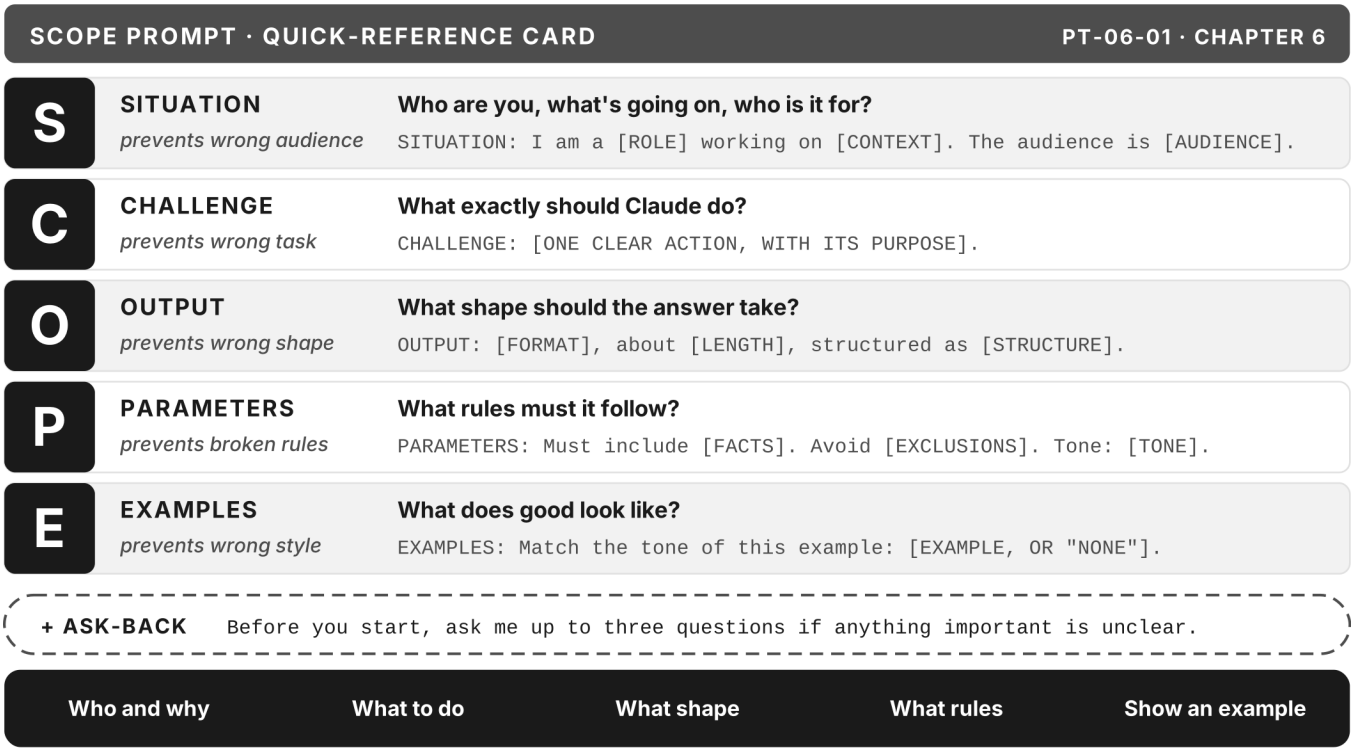

Figure B.1 Answer five questions before Claude starts, and add the ask-back line to let it ask its own.

For a quick question, a strong first message is enough: what it's for, who it's for, and the facts (Chapter 3). Claude only sees what's on the page, so anything you leave out, it has to guess.

B.2 Technique selector (Chapters 7 and 8)

If the problem is…	Try this	Template
The style or quality isn't right	Two or three examples ("shots") in `<example>` tags	PT-07-01
A long prompt; Claude mixes up the parts	XML-style tags; long material first, the request last	PT-07-02

If the problem is…	Try this	Template
You need to compare or scan the answer	A table; name the columns in order	PT-07-03
It's going into another tool	JSON with the exact keys and allowed values	PT-07-04
It's too long or too short	Count sentences or items, not just words	PT-07-05
It needs reasoning, not recall	Step by step; for harder tasks, `<thinking>` apart from `<answer>`	PT-08-01
It's too big for one prompt	Decompose backward from the output, then chain one job per step	PT-08-02, PT-08-03
You need confidence in the result	A self-check against a rubric; a self-check isn't a fact-check	PT-08-04, PT-08-05

Say this for the shape you want: "Give me 5 numbered points, one sentence each" (list) · "Columns: Theme | Count | Action. One row per theme" (table) · "Return only valid JSON with the keys …" (JSON) · "Three sentences, under 60 words" (length) · "Write flowing paragraphs, no bullets or headings" (prose) · "No introduction or closing remarks" (just the output).

B.3 Troubleshooting card (Chapters 9 and 10)

Quote the symptom · Find the layer · Fix the cause, not just the output. Check the first three layers first; they're in your hands.

Layer	Ask yourself	First fix
1 The prompt	Did I ask for what I actually need?	Sharpen the SCOPE element; remove conflicts; rules last, stated positively
2 The material	Did Claude have the facts, clearly marked?	Supply the facts and say "use only these"; tag pasted text as data
3 The conversation	Is the chat still working from my current rules?	Drift reset (PT-10-03), or a new chat with a handoff
4 The model's limits	Is this something Claude can't know or do reliably?	Verify: sources you open, search, step by step, code

Six hallucination warning signs: exact numbers without a source · quotes and sources · recent or changing facts · niche details · gap fillers you never supplied · answers that simply agree with how you asked. The more specific and checkable a claim, the more it needs a source.

The refinement loop: evaluate (against three or four criteria written before you read), diagnose (the draft or my prompt? which follow-up fixes it?), adjust. **Precise feedback:** name the part, say the change, say what to keep; one or two changes per turn. **Fix in this order:** Draft → Right (task and facts) → Clear (order, structure) → Tight (length) → Polished (wording) → Final (your check). **Iterate, restart or hand-edit:** a word or a number, fix it yourself; a bigger change, one precise follow-up; wrong at the core, or feedback has failed twice, start a new chat with a fresh-start brief (PT-09-04). The four follow-ups are **expand, narrow, redirect, correct** (Chapter 3).

B.4 The automation ladder (Chapter 20)

Rung	What it saves you	Climb when…	Your check
1 Prompt	Nothing yet	You've typed it twice	Read the output

Rung	What it saves you	Climb when…	Your check
2 Template	The thinking	It passes your tests	Tested on real cases
3 Project	Re-explaining the background	One area of work repeats	Files current and dated
4 Skill	Deciding when to use it	You use it across chats and areas	Three requests that should trigger it, one that shouldn't
5 Connected	Copying material in	The material lives in an app	Approve actions that change things
6 Scheduled	Starting it	It runs on a timetable	Review each run; drafts, not sends

Automate what's stable, not what's still changing. Climb one rung at a time, and keep the check. Chain the rungs and you have a system (Chapter 22).

B.5 Model and tool chooser (Chapters 21, 22 and 30)

Choosing a model. Ask three questions: how hard is it? What does a mistake cost? How often, and how fast? For anything you'll repeat, let a test decide: the fastest model that passes wins. Effort is a second dial on the same model (Chapter 8). The current model names are in Chapter 30's Version Note.

The job	Best fit
Quick questions, reformatting, short replies; sorting and extracting at volume	Fast
Everyday drafting, summarizing, planning	Balanced
Long documents, analysis, tricky reasoning; high-stakes or public work	Most capable
Long agent tasks and the hardest problems	Frontier

Choosing a tool for each step. Use the lowest rung that does the step, and set how far it may run alone by what a mistake would cost.

If the step…	Use	Typical dial
Decides, approves or sends	You (always yours)	—
Is one-off thinking or writing	A prompt in a chat (Ch 6)	Suggest
Repeats the same way	A template or Project (Ch 11, Ch 5)	Suggest
Needs your know-how in any chat	A skill (Ch 20)	Suggest
Needs material from an app	A connector (Ch 20)	Read freely; approve writes
Runs on a timetable	A scheduled task (Ch 20)	Run unattended, drafts only
Clicks through websites	Claude in Chrome (Ch 21)	Ask each step at first
Works on files or code in a folder	Claude Code (Ch 21)	Ask each step at first

Numbers in any step: compute with code, don't guess (Chapter 16). **The autonomy dial** (Chapter 21): suggest · ask each step · act, pause for risk · run unattended. Start low and turn it up only when the agent has earned it. Payments, sending, deleting and anything you can't undo stay with you. Where agents act: a connector if you can, a browser if you must, the screen as a last resort.

B.6 The Workflow Canvas (Chapter 22)

Pass	Box	The question
Frame	1 Workflow	What job does it do? Name it by the result.
	2 Trigger	What starts it, and how often?
	3 Owner and reader	Who makes sure it happens? Who uses the output?
Flow	4 Inputs	What goes in? Where does it live? Who keeps it current?
	5 Steps	One job each, in order: STEP · WHO · TOOL · DIAL
	6 Output	What comes out, in what format, and where does it go?
	7 Checks	Where do you look, and what must be true to go on?
Safety net	8 If it breaks	Missing data, an error, a late input: what happens, and who's told?
	9 Measure	Minutes per run, before and after; one sign of quality
	10 Next change	One improvement to try after a few runs

Map it as it is · Design it as it should be · Test it before you trust it. Four principles: start from the output · one job per step · check where mistakes are cheapest · fix the input, not the step. Test once with a missing input on purpose.

B.7 Anatomy of an API request (Chapter 28)

Part	What it does
API key (`ANTHROPIC_API_KEY`)	Proves who's paying; keep it in an environment variable, never in your code
`model`	Which model answers
`max_tokens`	A ceiling on the reply's length, thinking included
`system` (optional)	Standing instructions: the API's version of Project instructions
`messages`	The turns, each with a role (`user` or `assistant`) and content; send everything each time
`tools`, `output_config` (optional)	Tools Claude may ask your code to run; a schema for guaranteed JSON
`content`	The reply as blocks; a thinking block can come first, so read only the `text` blocks
`stop_reason`	Why it stopped: `end_turn` (finished), `max_tokens` (cut short), `tool_use` (wants a tool)
`usage`	Input and output tokens: what you pay for

Nothing is remembered between requests. A schema guarantees the shape, not the truth. Claude asks; your code acts.

B.8 Responsible use (Chapter 29)

Level	What it covers	Examples
SHARE	General facts and your own drafts	Anonymous comments; roles and schedules without names; topics and objectives
REDACT	Useful once identifying details are gone	A bank export minus account numbers; a complaint minus name and address; work samples minus names
NEVER	Secrets and other people's records	Passwords, card and ID numbers; order histories by name; pay, health, performance, grades

Work, school or client data? Their rules come first. Context, not identifiers.

Before: ☐ Is this data share, redact or never? ☐ Do work, school or client rules apply? ☐ Have I given it the facts it needs?

During: ☐ Is pasted material tagged as data? ☐ Least access: only what the task needs. ☐ Do actions that change things need my approval?

After: ☐ Numbers from code, one checked by hand. ☐ Sources opened; quotes copied. ☐ Content about people checked for bias. ☐ Disclosed where expected. ☐ A person sends it.

B.9 Rubrics, gates and scorecards (Chapters 30 to 32)

Rubric: three to five criteria, one thing each, observable, scored **4 excellent · 3 good · 2 weak · 1 poor**; fix anything below 3. A **gate** is a criterion where only a 4 passes (facts, policy, promises). **Test set:** fixed inputs with must and must-not written before you run them: typical, edge and trap cases. Rerun the whole set after every change.

Capstone criterion	What it asks
Quality	Is it well made for its reader: clear, well structured, polished, in the right voice?
Accuracy (GATE)	Does every fact, figure and quote trace to a source you opened or to numbers computed with code?
Usefulness	Does it answer the real question, so the reader can act or decide?
Efficiency	The lowest rung that works, the right model for each step, no wasted runs, time logged?
Reusability	Could you, or someone else, run it again faster from what you saved?

A capstone **passes** when every criterion scores 3 or 4 and Accuracy scores 4; total out of 20. **Weekly scorecard** (30-day plan): one row per mission, **4 on my own · 3 with the book open · 2 part of it · 1 not yet**; any row below 3, repeat that mission before the next week. Never miss twice.

B.10 Settings and keyboard quick reference

> ▲ **Version Note: where things are (checked October 2026)**
> Menus move. If a label differs, look for the function, and check the help center (Appendix E).

To…	Look for
Choose a model, effort or thinking	The model menu beside the send button (Ch 8, 30)
Add files; turn on web search, Research or a connector for one chat	The **+** menu in the message box (Ch 4, 12, 13, 20)
Set instructions for every chat; choose a style	**Settings** (instructions for Claude); styles in the message box or Settings (Ch 2)
See, edit or pause memory	**Settings, Memory** (Ch 2, 5)
Decide whether chats help improve models; export data	**Settings, Privacy** (Ch 2, 29)
Start a chat that isn't saved to history or memory	The incognito (ghost) icon in a new chat (Ch 2)

To…	Look for
Add or switch off skills, connectors, plugins	**Customize > Skills**, **Connectors**, **Plugins** (Ch 20)
Create or review scheduled tasks	The **Scheduled** page, or `/schedule` (Ch 20)
See your usage and when limits reset	**Settings > Usage** (Ch 30)
Use a plugin's commands or a skill	Type **/** in the message box (Ch 20)
Claude Code: start, get help, start fresh, finish	`claude` · `/help` · `/clear` · `/exit`; **Shift+Tab** cycles how much it may do without asking (Ch 21)