Prompt Chain Library

Five reusable AI workflow patterns for consultants, documentation teams, and operations leads.

About this library

A prompt chain is a structured sequence of inputs designed to produce a specific, reproducible output from a language model. Unlike single-shot prompts, chains break complex tasks into stages, with each stage building on the last, and with deliberate decision points along the way.

The chains in this library are drawn from real consulting engagements. Each one includes the prompt sequence, the use case it addresses, and a practitioner note explaining the design decisions behind it.

These patterns are meant to be adapted, not copied verbatim. The logic matters more than the exact wording.

How to use this library

Each chain in this library is a sequence of prompts designed to be run in a single conversation with Claude. To use a chain:

1. Open a new conversation in Claude.
2. Find the chain that matches your task.
3. Replace any bracketed placeholders with your actual content, either by typing it directly or uploading a file.
4. Run the prompts in order, one at a time. Each prompt is a separate message: send it, read Claude's response, then send the next one. Don't paste the whole sequence into a single message.
5. At any decision point marked in the chain, review Claude's output before continuing.

A note on adapting these chains

The exact wording of each prompt matters less than the structure. Feel free to rephrase steps in your own voice. What you should keep consistent is the sequence and decision points – those are where the chain does its real work.

Chain 01: Style Guide Onboarding

Use case: Teaching Claude your organization's house style using layered references and live examples

Goal: Install a persistent editorial voice that reflects your brand's standards

Style onboarding works best in layers rather than as a single dump. Each pass adds specificity: the first establishes vocabulary, the second adds structure, the third adds rhythm. The gold standard example does more work than any rule, because it shows the target rather than describing it. The calibration test is not optional: it surfaces misunderstandings before they grow.

Prompt sequence

1. **Share your primary style reference**: "Here is our style guide. Read it carefully and confirm you understand it."
2. **Layer in structural frameworks**: "We also follow the Diátaxis framework for documentation. Here are the core principles."
3. **Add language discipline**: "For sentence construction, apply ASD Simplified Technical English: one instruction per sentence, 25-word maximum for descriptions."
4. **Anchor with a gold standard example:** "Here is a document that represents ideal output. Use it as your benchmark."
5. **Run a calibration test**: "Review this rough draft using the style guide we just covered. Flag every violation."
6. **Iterate on edge cases**: "The guide is silent on [X]. Here is the rule we want you to follow going forward."

Chain 02: Document Review with a Trained Persona

Use case: Submitting engineer-written documentation to a trained editorial persona for structured critique and rewrite

Goal: Produce a consistent editorial pass without a human editor in the loop

Separating critique from rewrite is intentional; it forces the model to reason before it produces, and it gives you a decision point before the output is generated. The flagging step at the end is how the style guide grows: every ambiguity that surfaces in a real document is an opportunity to sharpen the rules.

Prompt sequence

1. **Activate the persona**: "You are FERN, Foxfire Works' editorial reviewer. Apply the full FERN style guide."
2. **Submit the document**: "Review the following document and flag every issue you find."
3. **Request a structured critique**: "Organize your findings by category: clarity, structure, consistency, and precision."
4. **Request a rewrite**: "Now rewrite the document applying all corrections. Flag any decisions you made where the guide was ambiguous."
5. **Review flagged decisions**: "For item [X], the correct rule is [Y]. Update accordingly."
6. **Confirm and close**: "This version is approved. Save the rule clarification for future reviews."

Chain 03: Template-to-Skill Conversion

Use case: Taking an existing document template and encoding it as a reusable Claude skill

Goal: Make a template self-executing so that Claude applies it automatically without being reminded

Templates carry implicit knowledge that their authors never wrote down. The rule-extraction step (step 3) makes that knowledge explicit, often for the first time. It’s worth doing even if you never build a skill, because it reveals assumptions the template had always made silently.

Prompt sequence

1. **Provide the template**: "Here is our standard tutorial template. Read it and confirm you understand its structure."
2. **Explain the intent**: "This template implements the Diátaxis tutorial framework. The goal is that a learner can do one thing correctly by the end."
3. **Identify the rules**: "List every structural rule embedded in this template — what must always appear, what must never appear."
4. **Test against a real document**: "Apply this template to the following rough tutorial draft."
5. **Surface gaps**: "Note any situations the template doesn't cover and propose a rule for each."
6. **Install as a skill**: "Encode everything we've established into a skill file I can install in my Claude settings."

Chain 04: Business Setup Scoping

Use case: Using a structured clarifying-questions approach to scope a complex deliverable before generating it

Goal: Produce a right-sized deliverable on the first pass, without over-building or under-specifying

The GO signal is a deliberate gate. It prevents the model from generating a large deliverable based on a misread of the task, which wastes time and creates friction. The confirmation step (step 3) is the cheapest quality check in the workflow: it costs one message and catches scope errors before they become output errors.

Prompt sequence

1. **State the task broadly**: "I need to set up [business/system/process]. Help me think through what's required."
2. **Answer scoping questions**: Claude asks targeted questions; you answer each one specifically.
3. **Confirm the scope**: "Based on my answers, confirm your understanding of what we're building before you start."
4. **Issue the GO signal**: "That's correct. GO."
5. **Review the deliverable**: "Flag anything that needs adjustment before we finalize."
6. **Save to your knowledge base**: "Save this to [Notion/Drive/etc.] under [location]."

Chain 05: Mid-Task Style Correction

Use case: Catching a style guide violation in live output, correcting it, and updating the working rule in place

Goal: Keep the style guide current without interrupting the workflow

Mid-task corrections are not interruptions; they’re the primary mechanism for style guide evolution. Every correction is data. The best style guides are written backward from real violations, not forward from imagined ones. Capturing corrections as they happen, rather than at a scheduled review, keeps the guide alive and grounded in actual use.

Prompt sequence

1. **Identify the violation**: "In the output above, you used [X]. That's inconsistent with our style guide."
2. **State the correct rule**: "The correct form is [Y]. Here's why: [brief rationale]."
3. **Request a correction**: "Apply this correction to the affected passage."
4. **Confirm the update**: "Treat this as a standing rule for the rest of this session."
5. **(Optional) Log for the style guide**: "Add this rule to the open issues list so we can formalize it in the next style guide update."

Using these chains

Each chain is designed to be run in a single Claude conversation. Start a new conversation for each chain, or separate chains with a clear context reset if combining them.

Bracketed text like [X] or [Y] indicates a placeholder: replace it with your specific content before running the chain.

Chains 01 and 03 produce persistent artifacts (style guides, skill files) that improve over time. The others are designed for repeatable use on new inputs.