How to Optimize Prompts for ChatGPT, Claude, and Gemini

fromscratch editorial • 3 min read

How to Optimize Prompts for ChatGPT, Claude, and Gemini Most people don't have a model problem. They have a prompt problem. You ask for "a landing page," get mush, and then assume the model is weak. Usually it isn't. It just had to guess what you meant. And it guessed wrong. If you want consistently good outputs, use a repeatable structure instead of rewriting from scratch every time. You can do this manually, or run your draft through the prompt optimizer first and then tune from there. If you want a fast QA layer after optimization, run this prompt engineering checklist before shipping. The 5-part prompt structure that actually works Keep this simple. You need five parts. 1. **Role**: who the model should be. 2. **Context**: what it's working with. 3. **Constraints**: what it must avoid or include. 4. **Output format**: exact shape of the answer. 5. **Quality bar**: what "good" means for this task. No magic words. No mystical framework names. Just less ambiguity. Raw prompt vs optimized prompt Bad prompt: Optimized prompt: Same task. Different result quality. Night and day. What changes by model (and what doesn't) The backbone stays the same for all three major assistants. That's the useful part. You don't need three totally separate prompt systems. What tends to change: - **ChatGPT**: usually responds well to explicit output sections and concise constraints. - **Claude**: often benefits from clearer tone guidance and examples of what to avoid. - **Gemini**: often follows structured bullet instructions cleanly when format is strict. What should not change: - role clarity - context depth - output shape - acceptance criteria If you want a fast baseline, draft once and run it through the prompt optimizer, then test model-specific edits from there. A concrete optimization workflow (15 minutes) Minute 1-3: Write the ugly first draft Don't overthink it. Just capture intent. Example: Minute 4-6: Add hard constraints Now force precision. - number of posts - format per post - forbidden language - target reader - action to take Minute 7-10: Lock the output schema If output shape is fuzzy, you'll get uneven quality. Tell the model exactly how to return content. Minute 11-15: Evaluate and rerun one variable Don't rewrite everything at once. Pick one thing: - tighten tone - shorten length - strengthen CTA Then rerun. This is where most teams lose time. They change six variables and can't tell what improved the result. Common mistakes that kill output quality 1) Asking for quality without defining quality "Make it better" means nothing. Say what better means: - fewer adjectives - shorter sentences - one concrete example - zero passive voice in headline 2) Missing audience context "Write blog intro" for who? CTO? solo founder? student? The model can't infer your funnel stage reliably. 3) No output format If you skip format, you invite randomness. That's fine for brainstorming. It's bad for production workflows. 4) Prompt stacking without cleanup People keep appending instructions until the prompt is a junk drawer. If it's bloated, reset and rewrite from clean structure. Or use the prompt optimizer to compress and reorganize it. Quick model-ready templates Template: feature announcement Template: support doc rewrite Final pass checklist Before you ship any prompt to production, check three things. 1. Could a stranger run this prompt and produce a usable output? 2. Does the output format match your publishing workflow? 3. Did you define what to avoid, not just what to include? If any answer is no, tighten it. Then test in your target model. If you're moving fast and don't want to tune by hand every time, use the prompt optimizer as your first pass, then layer model-specific edits. Clean prompt in. Cleaner output out. That's the whole game.