CLAUDE · 8 min read · August 9, 2026

He Deleted 80% of His AI Prompt. It Got Smarter.

The Anthropic engineer who built Claude Code cuts 80% of his own system prompt every time a new model ships. Here's the exact 3-question test to find out what you can cut from yours.

His name is Boris Cherny. He’s an engineer at Anthropic, and he built Claude Code. Every time a new model ships, his team doesn’t add more rules to its system prompt. They delete about 80% of it. Then they only add a line back if removing it causes an actual, observed failure. The model gets smarter, not worse.

That should bother you a little. Nearly every AI tip you’ve ever read says the opposite: more context, more detail, more rules. Here’s the real, usable version of what Cherny’s team does, turned into something you can run on your very next prompt.

> TIP
> This isn't about writing shorter prompts for the sake of it. It's about finding out which parts of your prompt are actually doing work, and cutting the rest. Some prompts genuinely need length. Most don't.

PART 1 Why More Instructions Can Make AI Worse

This isn’t magic, and it isn’t Claude being temperamental. It comes down to how a model actually handles instructions.

When you hand Claude (or ChatGPT) a prompt, it isn't reading your instructions in order and picking the important ones. It's trying to satisfy all of them at once, at the same time, with equal weight. Every extra instruction is one more constraint competing for attention. Pile on enough of them, especially vague or redundant ones, and the instructions that actually matter get diluted instead of reinforced.

There's a second problem, and it's the one Cherny's team is really solving for. Instructions written for an older, weaker model often exist to patch a weakness that model had. A newer, smarter model doesn't have that weakness anymore, but the patch is still sitting in the prompt, doing nothing useful except adding noise and, sometimes, actively conflicting with what the new model would have done correctly on its own.

That's the whole logic behind deleting 80% of a prompt on every model upgrade: stop assuming old instructions still earn their place. Prove it, or cut it.

PART 2 The 3-Question Cut Test

This is the method, adapted from Cherny's actual workflow (subtract first, add back only on a proven failure) so you can run it on a single prompt in under a minute, not a whole engineering team's system prompt.

Before you send your next real prompt, go line by line and ask:

1. **Read it one sentence at a time.** Not the whole paragraph, one instruction at a time.
2. **For each line, ask: "If I deleted only this, would the answer get measurably worse, not different, worse?"** Different isn't the bar. Different just means Claude wrote it another way. Worse means it actually stopped doing the thing you needed.
3. **If you can't name the specific way it would get worse, cut it.** Not "soften it," not "shorten it." Delete the line completely.

Send the trimmed version. If the answer comes back worse in a way you can point to, add that exact line back in. Nothing else. Now you know precisely what that line was doing, instead of guessing.

That's it. That's the whole test. Run it enough times and you'll start noticing the same categories of line failing the test over and over, which is the next part.

PART 3 What Almost Always Survives the Cut (and What Doesn't)

Cut this on sight:

- **Politeness padding.** "Please," "I would really appreciate it if," "thank you in advance." Claude isn't a person you need to be polite to for a better result. It doesn't change output quality.
- **Vague quality asks.** "Be thorough." "Make it good." "Really think about this." Claude already defaults to trying its best. These lines feel like they're doing something. They almost never change the actual output.
- **Hedge language.** "Try to," "if possible," "if you can." This weakens an instruction instead of strengthening it. If you want it, say it plainly.
- **Restating the same instruction twice, worded differently, "for emphasis."** Claude doesn't need to be told a thing twice to take it seriously. This just adds length without adding a new constraint.
- **Old patches for a mistake the model doesn't make anymore.** If you added a line months ago to stop a specific bad habit, test whether the current model still has that habit before assuming the patch still earns its place.

Keep this every time:

- **The actual outcome you want.** What you're asking for, stated plainly. This is the one line that's never optional.

- **Real hard constraints only you would know.** Length, audience, tone, deadline, format. Claude has no way to guess these.
- **Facts the model can't infer.** Your actual numbers, your actual client's situation, your actual product details.
- **One clear example, if the format is unusual.** A single example of the exact output shape you want does more work than three paragraphs describing it.

BEFORE AND AFTER

Before (bloated): "Hi! I would really appreciate it if you could please write me a caption for an Instagram post about a productivity tip. Please try to make it engaging and really think about the audience, who are busy professionals. Make sure it's not too long, but also make sure it covers everything important. Please be thorough and give a great, high-quality caption. Thanks so much in advance!"

After (cut): "Write an Instagram caption for busy professionals about a productivity tip. Under 3 sentences, no hashtags."

Everything cut from that first version failed the test. "I would really appreciate it" and "thanks so much in advance" are pure politeness padding. "Try to make it engaging" and "really think about the audience" are vague quality asks Claude already defaults to. "Be thorough" and "high-quality" don't specify anything actionable. What survived is the only thing that was ever load-bearing: the actual outcome, the actual audience, and a real constraint (under 3 sentences) Claude had no way to guess on its own.

PART 4 When to Add a Line Back

Only after you've seen it fail. Not before. Not "just in case." Not because it feels safer to leave it in.

If you cut a line and the next answer is missing something specific and real, that's your signal, not a guess. Add that one line back, worded plainly, and move on. This is the actual discipline behind Cherny's 80% number: nothing survives on the assumption that it might be needed someday. It survives because it already proved it was needed, once, for real.

The takeaway

The guy who built Claude Code doesn't write longer prompts when a smarter model ships. He deletes most of what's already there and only adds a line back when cutting it actually breaks something. Run the same 3-question test on your next prompt before you send it, not just the ones that already aren't working. You'll probably find more than half of it was never doing anything at all.