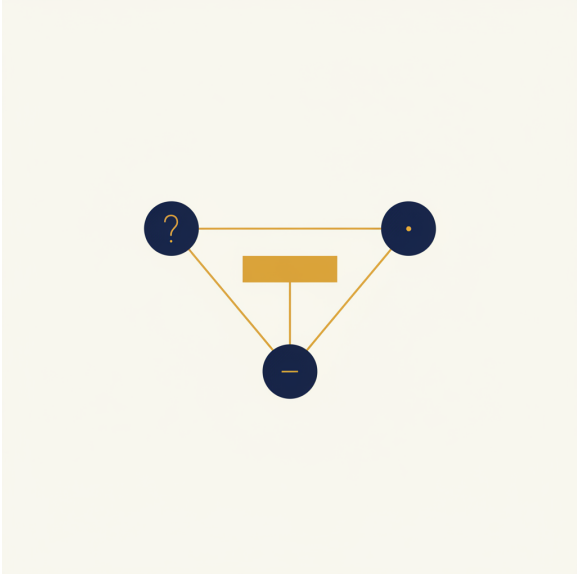
TIER 1 - FOUNDATIONS * V1.0 -- MAY 2026

HE 3-QUESTION PROMPT FRAMEWOR

The shortest reliable structure for getting 80%-usable output the first time. Three questions. No templates to memorize. No prompt engineering certifications.



BY Alex Jahn / Agent Logic

v1.0 -- May 2026 **EDITION**

Anyone whose AI prompts work some of the time and they can't tell you why **AUDIENCE**

15-20 minutes **READ TIME**

Free. Forever. **COST**

Prepared by Agent Logic / alexanderjahn79@icloud.com / theaiguywi.com

CONTENTS

What's in here

SECTION 1

Why most prompts fail

The 4-out-of-5 rule

Watch a small-business owner use ChatGPT for ten minutes and you'll see a pattern. They type a one-line prompt. They get a generic answer. They sigh. They paste a different one-line prompt. They get another generic answer. They give up and conclude "this isn't ready for real work."

The model isn't broken. The prompts are. Roughly four out of every five prompts I see typed by an AI beginner are missing at least one of three pieces of information the model can't function without. The result is a generic, hedged, useless answer -- and the user blames the technology.

This module is the fix. It's not a list of magic phrases. It's not "act as a senior copywriter with 20 years of experience." It's three questions you ask yourself *before* you
the prompt, consistently take you from "generic answer that doesn't help" to "specific answer I can actually use."

If you read the first module -- *What An L*
prediction machine; the quality of the prediction is conditioned entirely on what you give it. This module is the *how*: what

What this is not

It's not "the only prompt structure that works." It's the simplest reliable one. There are more elaborate frameworks -- chain-of-thought, role-prompting, few-shot examples, system prompts -- and they all have their place. But none of them matter if you can't get the basics right first. Most people skip the basics, get burned, then assume they need to learn the elaborate stuff. They don't. They need this one.

You can write a great prompt with these three questions in 30 seconds. You can write a great prompt without thinking about them only if you've already internalized them. Either way, the three questions are the ground truth.

> ***The model can't read your mind, can't see your screen, and can't remember last Tuesday. Whatever it doesn't know, you have to put in the prompt. That's the whole game.***

SECTION 2

The three questions

Here they are. That's it. The whole framework.

> **1. What do I want?**
> The actual output. A draft email. A pricing breakdown. A list of 5 ideas. A revised paragraph. A plan. State the deliverable, in plain words, before anything else.

2. What context do I have? The information the model would need to do this for *me specifi*
for an imaginary average user. Your industry. Your customer. The constraints of your situation. The thing that makes the answer different than the generic version.

3. What constraint? The boundaries the answer has to fit inside. Length. Tone. Format. Audience. What to avoid. The shape the output has to take to actually be useful.

That's the framework. Question, context, constraint. Three answers, three sentences, every prompt that matters. Read on for what each one really means and what happens when you skip it.

Question 1: What do I want?

This sounds obvious until you start watching what people actually type. "Tell me about marketing for plumbers." Tell you *what* abou
grant proposal? You don't know, and the model definitely doesn't know, so it produces a hedged, all-purpose blob that's useless for everything.

State the output. Specifically. *Write a 20*
subject-line variations. Make a checklist. The model is great at producing what you ask for. It's terrible at guessing what you wanted to ask for.

If you can't articulate what you want, that's not the model's failure. That's a sign you haven't decided yet, and the prompt isn't the right tool for that step. Decide first. Then prompt.

Question 2: What context do I have?

This is the one that separates a generic answer from a useful one.

The model has read a lot. It has not read *your* busin
constraints, your last six months of work, or your specific industry's quirks. If you don't tell it, it makes things up that sound plausible-on-average. Plausible-on-average is exactly the trap.

The fix isn't to write a novel. It's to write the three to five sentences of context that distinguish your situation from the generic case.

Bad: "Write me a sales pitch." Better: "Write me a sales pitch. I run a residential carpentry business in a small Midwest town. My customers are homeowners 40-65, mostly word-of-mouth referrals. They care more about reliability and finish quality than price. The pitch is for a follow-up email after a kitchen remodel quote."

Both prompts produce a sales pitch. Only the second one produces a sales pitch that sounds like *your* business.

> **Heuristic for context:** If the answer would be the same for you and your competitor down the street, you didn't give the model enough context. The whole point is to get an answer that *isn't* gener

Question 3: What constraint?

The third question is what most people skip, and skipping it is why "this AI thing keeps giving me five paragraphs when I needed two sentences."

Constraints are anything that shapes the output to fit your actual use:

- **Length.** "200 words." "Three bullet points." "One sentence."
- **Format.** "As a numbered list." "As a table." "As a single paragraph, no bullets." "In the voice of a customer service email."
- **Tone.** "Plain, direct, no marketing language." "Warm but professional." "Slightly self-deprecating."
- **Audience.** "Written for a homeowner who's never done a renovation before." "For someone in my industry with five years of experience."
- **What to avoid.** "Don't use the word 'leverage.'" "No emojis." "Don't apologize at the start."

You can stack constraints. The more you stack, the more the model gets pinned into producing exactly what fits. Don't worry about over-constraining -- that's a problem you discover by *trying*, not by hedging upfront.

SECTION 3

Worked examples

Three pairs. Same task, generic prompt versus 3-question prompt. Read them side by side. The pattern will install itself.

Example 1 -- Marketing email after a quote

Generic:

> *"Write me a follow-up email after I send a quote."*

With the framework:

> *"**Want:** Write a 150-word follow-up email I can send three days after sending a quote.* ***Context:*** *I*
> *a small residential carpentry business in Wisconsin. The customer asked for a quote on a $14,000 deck rebuild, didn't reply for three days. They were warm during the on-site walkthrough -- engaged, asked good questions. No red flags.* ***Constrain***
> *quote (insulting). One light call-to-action -- invite them to ask any follow-up questions or schedule a call. No marketing language."*

The first one gets you a generic email any business could send. The second one gets you a draft you might actually mail. Same model. Same task. The work was in the prompt.

Example 2 -- Pricing decision help

Generic:

> *"How much should I charge for a kitchen remodel?"*

With the framework:

> *"**Want:** Help me think through my pricing on a specific kitchen remodel quote. Specifically, give me 3 questions I should ask myself before finalizing the number.* ***Context:*** *I*
> *Customer wants new cabinets (we install, customer buys), tile backsplash, two pendant lights, paint, no flooring. Estimated 75 hours labor, $4,800 materials. My standard hourly rate is $85. Customer is referral-of-a-referral, no prior relationship.* ***Constrain***
> *method. I want a quick second-pass -- 3 questions -- to make sure I'm not missing something obvious. Bullet list, no preamble."*

The generic prompt gets you a Wikipedia-level explainer about kitchen pricing. The framework prompt gets you a fast pre-quote sanity check from something that read every business-advice book

ever written.

Example 3 -- Personal life-admin

Generic:

> *"Help me plan my week."*

With the framework:

> *"**Want:** Sort the task list below into a 3-day plan.* ***Context:** I two job sites running, and a sick parent I need to call this week. Tuesday morning is locked (job site). Thursday afternoon is locked (kid pickup). Everything else is flex. Tasks: file Q1 sales tax, return Home Depot order, call lumberyard about delivery delay, draft proposal for the Schmidt project, schedule annual physical, reply to Tom about Saturday softball, fix the sticky cabinet door at home. **Constraint:** Cluster errands geographically. Put admin/calls in afternoons (mornings are job-site time). Mark anything you'd push to next week instead of cramming."*

This is not "let AI run your life." This is "use AI to do the boring sorting so you can focus on the thinking." The model is great at this. But not without context.

30

Seconds.

That's how long the framework takes to apply once you've used it five or ten times. You don't write three sections -- you mentally check the three questions and the answers come out as one well-shaped sentence each. It becomes reflex.

SECTION 4

Where the framework breaks

The 3-question framework is the foundation, not the ceiling. There are categories of work where it's not enough on its own. Naming them so you don't blame the framework when the issue is somewhere else.

When you need an example more than instructions

For tasks like "match my tone" or "produce something in this exact format," telling the model what you want is less effective than *showing* it
biggest upgrade past the 3 questions. Paste two examples of the desired output before your prompt. The model will pattern-match to them.

The framework still applies -- you'd describe what you want, give context, set constraint -- and then add: "Here are two examples of the output style I want. Match the tone and structure." Then paste them.

When the task is too big for one prompt

If your "want" is *"Write a 5*
proposal that doesn't have the depth you need anywhere. The fix isn't a better single prompt. It's breaking the work into stages: outline first, then draft each section, then refine.

This is a workflow problem, not a prompting problem. Tier 1 won't fully solve it; Tier 2's "Multi-step task chains" module will.

When you need it to use tools

For anything that requires browsing the web, looking up live prices, calling an API, sending an email -- you've graduated from prompting a chat model to running an *agent*. Dif
module. (Tier 2's "Picking the right model" and Tier 3's "Software integration" cover where to go from here.)

When the answer needs to be verified externally

The framework gets you a great-looking answer. It doesn't get you a *correct* an
being wrong is high, you still have to verify. That's the next module -- *Reading A*
and it's not optional.

SECTION 5

The pocket version

For when you don't want to read 12 pages every time you open a chat window.

The pocket version of the framework:

Before you type, ask yourself:

- **WANT** -- What's the deliverable?
- **CONTEXT** -- What does the model need to know about my situation that it can't guess?
- **CONSTRAINT** -- How does the answer need to be shaped?

Type all three answers in any order. The model doesn't care about structure. It cares about information.

That's the whole thing. Tape it to your monitor for two weeks. After that, it'll be muscle memory and you won't need the tape.

The before/after gut check

If you want a quick way to test whether your prompt is good before you hit enter, ask yourself one question:

If I sent this same prompt to a competent freelancer with no context about my business -- would they have enough to produce something useful?

If yes: send it. If no: you know what's missing. Add it.

The freelancer test works because both a good freelancer and a good model need the same things to do the work -- clarity on the deliverable, the situational context, and the shape of the output. The model is faster and cheaper. The information requirements are the same.

SECTION 6

Where to go from here

You now have the second piece of the Tier 1 foundation. The first module gave you the mental model -- *how to talk*
the series closes the loop:

- **Reading AI output critically** -- once the model gives you a great-looking answer, how do you tell whether it's actually right? The five tells the model is bullshitting you and the verification moves that catch them before you act on bad info.

After that, you're past the foundations and into Tier 2 -- applying this stuff to your actual job, role-by-role. Templates, model selection, privacy, multi-step workflows, the whole stack.

Get the next module the day it drops: theaiguywi.com/training

One short email per release. No drip sequence. No upsells. Opt out anytime.

If you want this same framework taught directly to your team -- the prompts, the context-building habits, the muscle memory that turns "AI is interesting" into "AI is making us 30% faster" -- that's the consulting offer. I run it the same way I run it for myself, on my own carpentry business, with my own ops.

Reach out: alexanderjahn79@icloud.com

A short call. Honest scope. We figure out together if it's a fit.

Closing -- the lock-in line

If you forget everything else in this primer, remember this:

3

Three questions, every prompt that matters: Want, Context, Constraint.

The model is great at producing what you ask for, in the situation you describe, in the shape you specify. It's terrible at guessing any of the three. Stop guessing. Tell it.

You have the operating procedure. The next module makes sure you don't get burned by the answers.

Agent Logic -- *theaiguyw*

Fond du Lac, WI. This is module 2 of 6 in Tier 1 (Personal).