Prompt Engineering

How to talk to AI so it actually listens

José María Vázquez

What we're covering today

01 What IS a prompt, really?

5 min

02 The anatomy of a great prompt

12 min

03 Prompting patterns that work

18 min

04 Common mistakes (& how to fix them)

12 min

05 Live examples — real code & tasks

7 min

06 Q&A + your prompts, roasted

5 min

01 / What IS a prompt?

A prompt is a conversation, not a command

❌ Old mindset

"Type keywords into a search box and hope for the best"

✅ New mindset

"Brief a smart colleague who knows nothing about your context"

🎯 Why it matters

The quality of your output is directly proportional to the quality of your input

02 / Anatomy of a great prompt

The 4 ingredients

🎭 Role

Who should the AI be?

"Act as a senior PHP developer..."

🎯 Task

What exactly do you need?

Be specific, use verbs.

📋 Context

What does it need to know?

Code, constraints, audience.

📐 Format

How should it answer?

JSON, bullet list, 3 paragraphs...

💡 Memory tip: R.T.C.F. — "Really Try Crafting First" before sending a sloppy prompt.

02 / Context & Persona deep dive

Why context is everything

📋 Context depth

Don't just paste code. Give the AI:

- Your stack & version (PHP 8.2, WP 6.5)
- What you've already tried
- Constraints (no library changes)
- The actual error, not your interpretation

🎭 Persona precision

Vague: "Act as a developer"

Better: "Act as a senior WP engineer at a WordPress agency. You care deeply about security, backward compat, and WP coding standards." Precision = consistent, expert-level answers.

⚡ Combined power

Role + Context = surgical answers

"Act as a WP security expert. We run multisite on PHP 8.2. A client plugin does direct DB queries without $wpdb->prepare(). Audit it and list vulnerabilities by severity."

💡 Pro tip: The more specific your persona, the more consistent and expert the output across a long conversation.

02 / A structured framework for complex prompts

The ACHIEVE Framework

A	C	H	I	E	V	E
Action	**Context**	**Holistic**	**Input**	**Expectation**	**Validate**	**Evaluate**
What to do	Background info	Full picture	Data / examples	Format & length	Self-check stop	Iterate & refine

Example — code review prompt:

- Action: Review this PHP plugin for security issues
- Context: WordPress 6.5 multisite, agency client, code attached
- Holistic: Consider WP standards + OWASP
- Input: [paste code]
- Expectation: Numbered list by severity (Critical/High/Medium)
- Validate: Flag anything you're uncertain about
- Evaluate: After your list, suggest which issue to fix first and why

02 / See the difference

Same task. Wildly different results.

❌ BEFORE

"fix my code"

Result: a generic answer that probably doesn't apply to your specific bug, framework, or codebase. Thanks, useless.

✅ AFTER

"Act as a senior PHP 8.2 developer. I'm getting a fatal error on line 42 of my WordPress plugin…
[paste code + error].
Explain the bug and fix it without changing the API."

Result: surgical fix with explanation. Actually useful.

03 / Prompting patterns – overview

5 foundational + 3 advanced patterns

🧠 **Chain of Thought**	*"Think step by step before answering"*
🎭 **Persona**	*"Act as a skeptical security engineer..."*
📌 **Few-Shot Examples**	*Show 2–3 examples before your real question*
🔁 **Self-Critique**	*"Now review your answer and improve it"*
🧩 **Structured Output**	*"Respond ONLY as valid JSON: {key: ...}"*

03 / Deep dive: Chain of Thought

"Think step by step"

Those 4 words can transform a wrong answer into a correct one.

📝 Your prompt

"How should I architect this caching layer?

Think step by step."

🔄 AI reasons out loud

"First, I'll consider read vs write patterns.
Then latency requirements.
Then..."

✅ Better answer

A reasoned recommendation tailored to your actual constraints — not generic.

💡 Pro tip: CoT = one linear reasoning path. For big decisions with real tradeoffs, combine with Tree of Thought (next slide) — generate 3 CoT branches and compare. Use "Think step by step" for logic; use ToT for open-ended design decisions.

03 / Advanced pattern

Tree of Thought (ToT)

When one path of reasoning isn't enough — explore multiple branches in parallel

🔗 Chain of Thought

A → B → C → Answer

One linear path.
Great for: step-by-step reasoning, math, logic, debugging.

Limitation: if step 2 is wrong, everything after it is also wrong.

🌳 Tree of Thought

A → B1, B2, B3 → best Bi → Answer

Multiple paths explored simultaneously.
Great for: architecture decisions, complex debugging, design tradeoffs.

How: ask AI to generate 3 solutions, evaluate each, then pick the best.

🔧 Prompt template: "Generate 3 different approaches to [problem]. For each: describe the solution, list pros/cons, and rate feasibility 1–10. Then recommend the best one with reasoning."

03 / Advanced pattern

Least-to-Most Prompting

Break hard problems into sub-problems, solve smallest first, build up to the complex answer

1. Decompose	Ask: "What sub-problems do I need to solve first?"	"To migrate this plugin to WP 6.5, what are the smaller problems I need to solve first?"
2. Solve small	Solve the simplest, most foundational sub-problem	"First, let's just solve: how do I check which deprecated functions we're using?"
3. Build up	Use the answer as context to solve the next sub-problem	"Great. Now using that list, let's update the DB queries one by one."
4. Integrate	Combine all sub-solutions into the final complete answer	"Now put it all together and give me a migration checklist with those fixes applied."

💡 When to use: long refactors, multi-step migrations, complex architecture problems.
Avoids the 'one giant mega-prompt' trap.

04 / Common mistakes

Things we all do (and shouldn’t)

🌫️ Vague task	→ Be specific with verbs: "Write", "Refactor", "Explain", "List"
🕳️ No context	→ Paste the relevant code, error, or background — don't make it guess
🏔️ One giant mega-prompt	→ Break complex tasks into steps. One prompt = one clear goal
🥇 Accepting first answer	→ "That's a start. Now make it more X. Add Y. Remove Z."
🎲 No format specified	→ Say exactly how you want the output: JSON, table, 3 bullets, etc.

04 / The mindset shift

Iterative Refinement

The first answer is always a draft. The model improves dramatically with targeted follow-ups.

1
Send prompt
Your initial, best-effort prompt

2
Read critically
What's missing? What's wrong? What's imprecise?

3
Targeted follow-up
"Make it more concise" / "Add error handling" / "Now write tests"

4
Lock the good
"Keep the structure, but change X to Y"

5
Repeat
2–4 cycles usually reaches production-quality output

🔁 Real cycle example:
Prompt 1: "Write a WordPress REST endpoint for user activity logs"
Follow-up: "Add authentication check — only admins"
Follow-up: "Add pagination, 50 results per page"
Follow-up: "Now write the unit test for the auth check"

05 / Bonus pattern: System Prompts

Set rules once. Never repeat them again.

🔧 What is a system prompt?

An instruction block sent before the conversation starts. It sets the AI's persona, constraints, and rules.

Think of it as your global config file — it's always in scope.

Used in: API integrations, custom GPTs, AI-powered features you build.

Example system prompt:

```
You are a code reviewer
for a WordPress agency.
Always respond in English.
Focus on: security, performance,
and WordPress coding standards.
Never suggest switching frameworks.
Format findings as a numbered list.
```

Real prompts for real dev work

🐛 Debug this

"Act as a PHP 8.2 expert. I'm getting [error] on line [N]. Here's the code: [...]. Explain why this happens and provide the minimal fix without changing function signatures."

✍️ Write a PR description

"Write a pull request description for these changes: [diff]. Format: one-line summary, what changed and why, how to test it. Keep it under 150 words."

🔍 Code review

"Review this code as a senior engineer focused on security and performance. List issues as: [SEVERITY] Issue - Why it matters - Fix. Here's the code: [...]"

The Prompt Engineering Cheatsheet

Screenshot this. Seriously.

✅ DO

- Give it a role
- Provide context & constraints
- Specify output format
- Ask it to think step by step
- Iterate — first answer is a draft
- Show examples (few-shot)

❌ DON'T

- Send vague one-liners
- Expect it to read your mind
- Accept hallucinations silently
- Write 5 questions in one prompt
- Skip format instructions
- Forget to paste the actual code

syde

Your turn.

Share a prompt you use. We'll make it better together.

Q&A 5 min | Share your prompts | Roast session 🔥

Key takeaway: every minute you spend writing a better prompt saves ten minutes of frustration.

GOLD AGENCY PARTNER WORDPRESS VIP

woo PRO Partner

BEST WORDPRESS AGENCY DESIGNRUSH.com 2019