The Prompt Engineering Cheat Sheet

By Atilla Kürük · Promptolis · 2026 edition

This cheat sheet covers the patterns that survive the model-of-the-month churn. Built from 500+ production prompts shipped to customer-facing AI features. Print it. Pin it. Reference it.

1. The 4-Block Prompt Scaffold

Every production-grade prompt has 4 blocks in this order. Skip any one and quality drops 30-50%.

```
<role>
You are a [specific role] with [years] of experience in [domain].
You have [credentials that justify quality bar].
</role>

<principles>
1. [Specific principle that shapes outputs]
2. [Another principle — usually a constraint]
3. [What you refuse to do]
</principles>

<input>
<field-1>{user-provided value}</field-1>
<field-2>{user-provided value}</field-2>
</input>

<output-format>
# [Header structure]
## [Sub-header]
[What goes here]
</output-format>
```

Why this works: role primes context, principles encode quality bars, input enforces specificity, output-format eliminates 'what should it look like' guessing.

2. The 10 Patterns That Solve 80% of Cases

Pattern	When to use	Example
Few-shot examples	Tone or format you can't describe	3-5 input/output pairs in prompt
Chain-of-thought	Multi-step reasoning	'Think step by step before answering'

Pattern	When to use	Example
Role priming	Domain expertise calibration	'You are a senior security engineer'
Output schema	Structured data extraction	'Output JSON: {field: type, ...}'
Negative space	Prevent specific failures	'Do NOT include X. Do NOT make up data.'
Constraint stack	Quality bar enforcement	'Output must: 1) cite sources 2) be <500 words'
Self-critique	Iterative refinement	'After answering, list 2 weaknesses'
Persona contrast	Avoid corporate-speak	'Direct, no fluff. Like a colleague, not a brand'
Decision tree	Branching logic	'If X, do A. If Y, do B. Else, ask.'
Auto-intake	Missing-input recovery	'If input incomplete, ask for these 3 fields'

3. Model Selection Decision Tree (2026)

Pick the model based on the task, not the brand. The right model is the one that fits the workload.

Task type	First choice	Why
Long-form writing	Claude Opus 4	Voice consistency + tone calibration
Code generation	Claude Opus 4 / GPT-5	Reasoning + correctness; both strong
Real-time chat	Claude Sonnet 4 / GPT-5 Mini	Latency + cost
Image generation	GPT-Image-1 / Midjourney v7	Quality + control
Search + cite	Perplexity / Gemini 2.5	Native web grounding
Multi-step agents	Claude Opus 4	Long-context tool use
Data extraction	Claude Sonnet 4 / GPT-5 Mini	Structured output reliability
Translation	GPT-5 / Gemini 2.5	Multilingual training
Reasoning-heavy	GPT-5 Thinking / Claude Opus 4	Chain-of-thought stability
Cost-sensitive	Claude Haiku 4 / GPT-5 Mini	10× cheaper; fine for routine tasks

4. The 20 Do's

- Use XML tags for structure — beats Markdown for nesting.
- Specify the role with credentials, not adjectives.
- Include 3-5 few-shot examples for tone you can't describe.
- Set temperature 0 for extraction, 0.7 for creative work.
- End with 'Now [verb]:' to anchor the request.
- Always specify the output format, even if 'plain text.'
- Use principles section to encode constraints, not instructions.
- For long contexts, put the question at the END, not the start.
- Use 'You refuse to recommend X' for negative-space framing.
- Include an auto-intake section to handle missing inputs gracefully.
- Test with edge-case inputs: empty, malformed, very long, very short.
- Version your prompts. v2 of a prompt is rarely the same as v1.
- Use concrete examples, not 'something like.'
- For chain-of-thought, ask the model to think BEFORE the answer.
- Cache the system prompt portion — saves 80% of cost on long sessions.
- Run the same prompt 3× to see variance. Variance reveals fragility.
- Use the prompt's own voice — don't break style mid-prompt.
- When debugging, start with simpler. Build up complexity if needed.
- Check the output for the 5 most common failure modes for this task.
- Save prompts you like. Prompt portfolios beat re-prompting from scratch.

5. The 20 Don'ts

- Don't say 'be concise' — specify a word count.
- Don't ask for 'professional tone' — describe the voice with examples.
- Don't put the question in the middle of a long prompt.
- Don't use Markdown headings for structure inside XML — confusing.
- Don't mix instructions and input in the same block.
- Don't ask for 'creative' output without describing what creative means.
- Don't ignore the output-format section — it's where 50% of bugs hide.
- Don't use temperature > 1 unless you want unhinged output.
- Don't trust 'just summarize' on legal/medical/financial content.
- Don't ship a prompt without testing 3 edge cases.
- Don't expect models to remember instructions from earlier sessions.
- Don't use jargon the model doesn't know — define it once at the top.
- Don't ask the model to lie to itself ('pretend you have access to').
- Don't trust verbatim quotes the model gives without verification.
- Don't use 'high quality' as a quality criterion. Specify what high means.
- Don't forget to test on Friday afternoon. Real users use it then.
- Don't prompt-engineer around a model limitation. Switch models.
- Don't write a 5K-token prompt for a 100-token task. Match scale.

- Don't paste 50K tokens of context if you can use 5K with the same outcome.
- Don't ship without a fallback for 'model returns garbage' edge case.

6. The 5 Mistakes That Cause 80% of Bad Outputs

1. Vague role + no principles. The model has no quality bar to hit.
2. Missing output format. The model picks one — usually not what you wanted.
3. Mixing instructions with examples. Model treats examples as instructions or vice versa.
4. Asking for too many things in one prompt. Pick the 1-2 outcomes that matter.
5. Using the wrong model. A code-gen task on a chat-tuned model produces chat-gen output.

Where Next

This cheat sheet pairs with our 588+ Originals at promptolis.com — fully crafted prompts ready to use. Each shows the patterns above in action. The newsletter delivers 2-3 new Originals every Friday.
