**Prompt structuring:**

1. What we are doing
2. Tag relevant files
3. How to exectute/what not to do
4. Context dump

---- (this line helps a lot, you practically tell the model that something important follows)

1. Repeat core instructions
2. Output format

**Useful Cursor prompts**

1. THE FEWER LINES OF CODE, THE BETTER The fewer lines of code, the better.

2. PROCEED LIKE A SENIOR DEVELOPER Proceed like a senior developer. (Alternate version: Proceed like a 10x engineer.)

3. DO NOT STOP UNTIL COMPLETE Do not stop working on this until you've implemented this feature fully and completely.

4. THREE REASONING PARAGRAPHS Start by writing three reasoning paragraphs analyzing what the error might be. Do not jump to conclusions.

5. ANSWER IN SHORT Answer in short.

6. DO NOT DELETE COMMENTS Do not delete comments.

7. SUMMARY OF CURRENT STATE Before we proceed, I need you to give me a summary of the current state. Summarize what we just did, which files were updated, and what didn’t work. Do not include assumptions or theories—only the facts.

8. UNBIASED 50/50 Before you answer, I want you to write two detailed paragraphs, one for each solution. Do not jump to conclusions or commit to either solution until you have fully considered both. Then tell me which solution is obviously better and why.

9. PROPERLY FORMED SEARCH QUERY Your task is to write a one-paragraph search query as if you were instructing a human researcher. Clearly state what to find and request code snippets or technical details when relevant.

10. START WITH UNCERTAINTY You should start the reasoning paragraph with lots of uncertainty and slowly gain confidence as you think about the item more.

11. BE CAREFUL WITH RED HERRINGS Give me the TL;DR of the search results, but be careful. Often the search results contain dangerous and distracting red herrings.

12. ONLY INCLUDE TRULY NECESSARY STEPS Break this large change into the required steps. Only include the truly necessary steps.

**More detailed:**

**-**-- ### \*General Prompting Techniques\*

1. \*Fewer Lines of Code\*

- \*Prompt\*: "The fewer lines of code, the better."

- \*Purpose\*: Encourages the AI to write clean, minimal code without unnecessary complexity.

2. \*Proceed Like a Senior Developer\*

- \*Prompt\*: "Proceed like a senior developer."

- \*Purpose\*: Guides the AI to think and code with the expertise of an experienced developer.

3. \*Do Not Stop Until Fully Implemented\*

- \*Prompt\*: "Do not stop working until you've implemented this feature fully and completely."

- \*Purpose\*: Reduces AI laziness by ensuring it completes multi-stage tasks without stopping prematurely.

4. \*Reasoning Paragraphs\*

- \*Prompt\*: "Start by writing three reasoning paragraphs analyzing what the error might be. Do not jump to conclusions."

- \*Purpose\*: Forces the AI to analyze errors thoroughly before attempting to fix them, improving accuracy.

5. \*Answer in Short\*

- \*Prompt\*: "Answer in short."

- \*Purpose\*: Prevents the AI from being overly verbose, saving time.

6. \*Do Not Delete Comments\*

- \*Prompt\*: "Do not delete comments."

- \*Purpose\*: Ensures the AI retains comments in the code, which are crucial for context.

7. \*Summary of Current State\*

- \*Prompt\*: "Before we proceed, I need you to give me a summary of the current state. Describe what we just did, what did not work, which files were updated, and anything else a programmer might need to work on this project. Do not include any assumptions or theories, just the facts."

- \*Purpose\*: Summarizes the current state of the project when switching to a new composer, retaining essential context.

8. \*Unbiased 50/50 Analysis\*

- \*Prompt\*: "Before you answer, I want you to write two detailed paragraphs: one arguing for each of these solutions. Do not jump to conclusions. Seriously consider both approaches. Then, after you finish, tell me which one of these solutions is obviously better than the other and why."

- \*Purpose\*: Forces the AI to objectively analyze two possible solutions before making a recommendation.

9. \*Properly Formed Search Query\*

- \*Prompt\*: "Your task is to write a one-paragraph search query as if you were telling a human researcher what to find, including all the relevant context. Format the paragraph as clear instructions commanding a researcher to find what we're looking for. Ask for code snippets or technical details when relevant."

- \*Purpose\*: Helps the AI generate effective search queries for tools like Perplexity or ChatGPT.

10. \*Start with Uncertainty\*

- \*Prompt\*: "You should start the reasoning paragraph with lots of uncertainty and slowly gain confidence as you think about the item more."

- \*Purpose\*: Encourages the AI to think critically and avoid premature conclusions. --- ### \*Error Handling and Debugging\*

11. \*Debugging Instructions\*

- \*Prompt\*: "I want to debug this using Chrome DevTools. Tell me step-by-step instructions on how I can give you relevant context. If you were a senior software developer working on this project, what type of context would you need to solve this error?"

- \*Purpose\*: Guides the AI to provide detailed debugging instructions, especially for complex errors.

12. \*TLDR of Search Results\*

- \*Prompt\*: "Give me the TLDR of the search results. Be careful though; often the search results contain dangerous and distracting red herrings."

- \*Purpose\*: Summarizes web search results while avoiding irrelevant or misleading information.

--- ### \*Refactoring and Large Changes\*

13. \*Break Down Refactors\*

- \*Prompt\*: "Break this down into the required steps. Only include the truly necessary steps."

- \*Purpose\*: Helps the AI break large refactors into smaller, manageable tasks without adding unnecessary complexity.

14. \*Avoid Huge Refactors\*

- \*Prompt\*: "Avoid huge refactors. Break this down into smaller steps."

- \*Purpose\*: Prevents the AI from attempting overly complex refactors that may fail.

--- ### \*System Prompts and Rules\*

15. \*Cursor Rules File\*

- \*Prompt\*: Include sections like: - Project Overview - Personality (e.g., "Teach me like a senior developer") - Tech Stack (frontend, backend, database) - Processes (e.g., error-fixing steps) - Environment Variables - File Structure - Important Instructions (repeated for emphasis) - Comments (highlight their importance)

- \*Purpose\*: Provides the AI with context and guidelines for the project.

16. \*General AI Rules\*

- \*Prompt\*: "Keep code fast, small, and focused. Include lots of comments. Always write simple, clean, and modular code. Avoid huge refactors."

- \*Purpose\*: General best practices for any project, regardless of context.

--- ### \*Specialized Use Cases\*

17. \*Database Setup MD File\*

- \*Prompt\*: Create a markdown file (e.g., `database\_setup.md`) with: - Table structure - Fields (mandatory, nullable, defaults) - SQL queries - IRS policies

- \*Purpose\*: Provides the AI with all necessary database context when tagged.

18. \*Roadmap MD File\*

- \*Prompt\*: Create a `roadmap.md` file with: - Current focus - Next steps - Features tried but failed - General project state

- \*Purpose\*: Keeps the AI aligned with the project’s development roadmap. --- ### \*Voice Input\*

19. \*Whisper Flow\*

- \*Prompt\*: Use voice input to quickly dictate complex instructions.

- \*Example\*: "Update the AI agent Python file to call the Anthropic API, pass the JSON output, and store it in a dedicated JSON file."

- \*Purpose\*: Speeds up input for complex ideas or multi-file operations.

--- ### \*Advanced Prompt Engineering\*

20. \*Delimiters for Context\*

- \*Prompt\*: Use delimiters (e.g., "---") to separate sections like: - What you’re doing (2-4 sentences) - Tagged relevant files - How to execute (instructions) - Context dump (documentation, etc.) - Core instruction (repeated) - Output format (e.g., "Answer in short")

– \*Purpose\*: Helps the AI process long prompts more effectively by breaking them into clear sections.

--- ### \*When to Use Other Tools\*

21. \*Perplexity Web Search\*

- \*Prompt\*: Use the "Properly Formed Search Query" prompt to generate a search query for Perplexity or ChatGPT.

- \*Purpose\*: Finds up-to-date documentation or solutions to errors.

22. \*Claude as a Consultant\*

- \*Prompt\*: Use Claude for brainstorming and high-level consulting, while using Cursor for implementation.

- \*Purpose\*: Keeps project context separate and unbiased.

--- ### \*Miscellaneous\*

23. \*File Location Comments\*

- \*Prompt\*: Add comments at the top of files with their exact location (e.g., `backend/src/ai\_agent.py`).

- \*Purpose\*: Helps the AI understand file context, especially when switching between tools.

24. \*Avoid Technical Debt\*

- \*Prompt\*: "Avoid technical debt at all costs. Only implement what you understand."

- \*Purpose\*: Encourages learning and prevents over-reliance on AI for critical decisions.