

The Sonnet 5.5 Prompting Guide

Mastering Effective, Efficient, and
Reliable Prompting for Claude Sonnet 5.5



AKIRA MORIYAMA

The Sonnet 5.5 Prompting Guide

Mastering Effective, Efficient, and Reliable Prompting for Claude Sonnet 5.5

Steve Publications

This book is available at <https://leanpub.com/thesonnet55promptingguide>

This version was published on 2026-09-29



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2026 Steve Publications

Contents

Mastering Effective, Efficient, and Reliable Prompting for Claude Sonnet 5.5	1
Introduction: Why Sonnet 5.5 Deserves Its Own Prompting Guide . . .	2
The problem with generic prompt advice	2
What this book covers	3
What this book is not	4
How to use this book	5
A note on versions and change	6
Chapter 1: Why Sonnet 5.5 Is Different	7
What “Sonnet 5.5” means: the model family and its positioning	7
Key behavioral changes compared to earlier Claude versions	8
What “prompting” actually controls versus what API configuration controls	9
Why generic prompt-engineering advice often misses the mark	11
How this book is structured for progressive mastery	12
Chapter 2: The Prompting Landscape: Instructions, Configuration, and Safeguards	14
The dual levers: what you write in the prompt versus what you set in the API call	14
Effort levels: what they are, what they control, and how to choose . .	15
Thinking configuration: upfront thinking versus between_tools and adaptive reasoning	17
Token budgets and max_tokens as design decisions	18
Where application-level safeguards matter more than prompt instructions	19
Summary	20

Chapter 3: Reading the Official Documentation: What Anthropic Says

About Sonnet 5.5 22

 The “Prompting Claude Sonnet 5.5” document: structure and intent 22

 Categories of official guidance: documented behavior, recommendations, observations, beta features 22

 How to interpret empirical claims from the documentation 22

 Where the documentation is silent and what that means 22

 Navigating version-specific and evolving behavior 22

 Summary 23

Chapter 4: The Anatomy of an Effective Prompt 24

 Role and context: when it helps, when it is noise 24

 Objectives and success criteria: stating what “good” looks like 24

 Task decomposition: breaking complex work into model-friendly units 24

 Constraints and priorities: guiding tradeoffs without overconstraining 24

 Input organization: structuring information for clarity and retrieval 24

 Examples and output specifications: when they are essential versus optional 25

 Evaluation criteria: giving the model a standard for self-checking 25

 Stopping conditions: telling Claude when it is done 25

 Summary 25

Chapter 5: Concise Versus Detailed: Finding the Right Level of Instruction 26

 The cost of verbosity: token overhead, attention dilution, and contradictory instructions 26

 The cost of brevity: ambiguity, missed constraints, and inconsistent output 26

 When concise prompts succeed: routine tasks, constrained domains, expert users 26

 When detailed prompts are necessary: novel tasks, ambiguous domains, production reliability 26

 How to compress prompts without losing essential guidance 26

 Testing prompt length as a variable 27

 Summary 27

Chapter 6: Ambiguity, Conflicting Requirements, and Robustness 28

 Why ambiguous prompts produce unreliable results 28

 Resolving conflicting requirements: prioritization strategies 28

CONTENTS

Explicit ambiguity-handling instructions: what to do when unsure . .	28
Designing for changing inputs: prompts that adapt rather than break	28
Failure recovery: asking the model to flag problems instead of hallucinating solutions	28
Summary	29
Chapter 7: Effort Calibration: Quality, Latency, Cost, and the Tradeoff	
Triangle	30
What “effort” means for Sonnet 5.5: official definition and practical effects	30
Low effort: when to use it, what you gain, what you risk	30
Medium effort: the default operating point for agentic work	30
High effort: the API default for excellent results	30
Xhigh and Max: complex reasoning, high-stakes tasks, and when it earns its cost	30
Measuring the impact: quality vs latency vs token usage vs cost	31
How prompt complexity interacts with effort levels	31
Decision framework: choosing the right effort for your task	31
Summary	31
Chapter 8: Steering Initiative and Scope: Controlling How Far Claude Goes	
Goes	32
The initiative problem: why Claude adds extra steps, explanations, and recommendations	32
When extra initiative is valuable and when it is wasteful	32
Explicit scope-limiting instructions: patterns that work	32
Preventing unnecessary additions: concise task-bounding language .	32
Handling open-ended requests: when broadness is a feature versus a bug	32
User-facing progress updates: prompting Claude to report back during long tasks	33
Controlling follow-up questions and clarifications	33
Summary	33
Chapter 9: Thinking Configuration: Adaptive Reasoning and Strategic Depth	
Depth	34
The three modes: upfront thinking, between_tools, and adaptive thinking	34
When upfront thinking helps: complex reasoning before action	34

CONTENTS

When between_tools helps: iterative planning during tool use 34

When adaptive thinking helps: letting the model choose 34

Reasoning tasks that produce structured output: special considerations 34

How thinking affects latency, token cost, and output quality 35

Common misconceptions about thinking and “smarter” responses . . 35

Reading responses by block type 35

Summary 35

Chapter 10: Structured Outputs and JSON Reliability 36

Why structured output matters for production workflows 36

Prompt-level techniques: schema specification, examples, and constraints 36

API-level structured outputs: the dedicated feature and how it differs from prompting 36

JSON reliability: factors that improve or degrade format adherence . 36

Handling schema violations: tolerance, error recovery, and retries . . 36

Complex schemas: nested objects, arrays, unions, and optional fields 37

When to use structured outputs versus parsing free-form responses . 37

Summary 37

Chapter 11: Tool Use, Research, and Current Information 38

How tool use integrates with prompting: definitions, instructions, and constraints 38

Tool use for research: prompting Claude to gather and synthesize current information 38

Prompting for appropriate tool selection: avoiding overuse and underuse 38

Tolerant tool-call handling: parsing errors, retries, and fallback behavior 38

Long-running agentic workflows: structuring multi-step tool interactions 38

Combining tool results with reasoning: synthesis and verification patterns 39

Summary 39

Chapter 12: Visual Inputs: Charts, Diagrams, and Technical Drawings . 40

What kinds of visual inputs Sonnet 5.5 handles well 40

Prompting for chart and graph analysis: reading values, trends, and patterns 40

CONTENTS

Technical drawings and schematics: structured extraction from diagrams	40
Combining visual and textual prompts: coordinated instructions . . .	40
Limitations and failure modes with complex visuals	40
Crop tool: improving accuracy on dense charts and drawings	41
Summary	41
Chapter 13: Long-Context Tasks and Massive Inputs	42
Context window capabilities and practical limits	42
Organizing large inputs: summaries, indexes, and structured access .	42
Prompting strategies that degrade with context length and alternatives	42
Multi-pass processing: chunking, summarizing, and integrating	42
Maintaining coherence across long conversations	42
When long context helps and when smaller focused contexts work better	43
Summary	43
Chapter 14: Research, Writing, and Content Workflows	44
Research prompts: scoping, sourcing, synthesizing, and citing	44
Writing prompts: drafting, voice, structure, and completeness	44
Editing prompts: critique, revision, tone adjustment, and style enforcement	44
Summarization: faithful compression without loss of meaning	44
Analysis prompts: comparing sources, extracting insights, and drawing conclusions	44
Chain-of-prompts for complex content projects	45
Summary	45
Chapter 15: Coding, Debugging, and Software Engineering	46
Coding task scoping: when to be specific and when to trust the model	46
Debugging prompts: providing context, error messages, and expected behavior	46
Software engineering: architecture, refactoring, and design review . .	46
Verification requirements: testing, edge cases, and self-check patterns	46
Incremental development: building codebase changes step by step . .	46
When additional tools or human review are non-negotiable	47
Summary	47
Chapter 16: Data Extraction and Document Processing	48
Field extraction: schemas, mappings, and handling ambiguity	48

CONTENTS

Document classification and categorization	48
Entity recognition and relationship extraction	48
Dealing with poor-quality, noisy, or inconsistent source documents	48
Batch processing patterns and maintaining consistency	48
Validation and spot-checking strategies	49
Summary	49
Chapter 17: Planning, Agentic Workflows, and Multi-Step Tasks	50
Task planning: prompting Claude to generate executable plans	50
Step-by-step execution: maintaining state and progress	50
Agentic loops: when Claude should decide what to do next	50
Mid-turn user messages: handling interruptions and new information	50
Error recovery in long workflows: backtracking and correction	50
Designing workflows that are robust to individual step failures	51
Summary	51
Chapter 18: Prompt Injection, Safety, and Refusals	52
Prompt injection fundamentals: what it is and how it works	52
System prompt versus user message: trust boundaries	52
Mid-turn prompt-injection considerations with user messages during long tasks	52
Tool-result handling: why untrusted data needs sanitization	52
Refusal behavior: categories, triggers, and workarounds (where legit- imate)	52
Reasoning-extraction requests: what is possible and what is restricted	53
When application-level safeguards are required beyond prompts	53
Summary	53
Chapter 19: Reliability Engineering: Making Prompts Production-Ready	54
What “reliable” means for LLM prompts: consistency, correctness, and recoverability	54
Versioning prompts: tracking changes and impact	54
A/B testing prompts: comparing quality and efficiency	54
Monitoring output quality: metrics and alerting	54
Handling model updates: when behavior changes and how to adapt	54
Building fallback strategies for when prompts fail	54
Summary	55
Chapter 20: Prompt Optimization: Improving Without Bloating	56
Diagnose-first optimization: identifying the real bottleneck	56

CONTENTS

Reducing instruction overhead: compression techniques 56

Improving accuracy without adding tokens 56

Latency optimization: prompt structure, effort, and thinking choices 56

Cost optimization: token efficiency across the request-response cycle 56

When to stop optimizing: diminishing returns and engineering judgment 57

Summary 57

Chapter 21: Common Failures and Anti-Patterns 58

Over-instructing: the paradox of too many rules 58

Under-scoping: open-ended prompts that run too far 58

Ambiguous success criteria: when “good” means nothing 58

Assuming capabilities: expecting what the model cannot reliably do . 58

Ignoring effort and thinking configuration 58

Treating prompts as static: neglecting iteration and testing 58

Confusing prompt-level fixes with system-level requirements 59

Summary 59

Chapter 22: Comparative Strategies: Matching Approach to Task Type 60

Simple requests: minimal instruction, low effort, no thinking 60

Reasoning-intensive tasks: structured thinking, higher effort, verification 60

Coding tasks: context-rich, verification-aware, incremental 60

Research tasks: tool-heavy, iterative, synthesis-focused 60

Agentic workflows: stateful, adaptive, robust to failure 60

Structured JSON generation: schema-driven, validation-heavy 61

Multimodal tasks: coordinated visual and textual instructions 61

Decision guide: matching approach to task 61

Summary 61

Chapter 23: Principles, Patterns, and What Comes Next 62

Core principles of effective Sonnet 5.5 prompting 62

The most impactful patterns reviewed 62

What prompting practice will look like as models evolve 62

Staying current: monitoring Anthropic documentation and platform changes 62

Final thoughts on treating LLMs as collaborative tools, not oracles . . 62

Conclusion: The Craft of Prompting 63

From novice to practitioner: the journey mapped 63

What makes a great prompter: skills and mindset 63

The enduring value of understanding how the model works 63

References 64

Mastering Effective, Efficient, and Reliable Prompting for Claude Sonnet 5.5

Claude Sonnet 5.5 is the latest model in Anthropic's Sonnet family: a fast, cost-efficient model that delivers quality approaching the top tier on coding, knowledge work, and agentic tasks. But getting the most from it is not about copying generic prompt templates; it is about understanding how this specific model responds to instruction, effort calibration, scope control, tool use, and structural constraints. This book is a deep, systematic reference for constructing prompts that are accurate, consistent, and production-ready. It draws primarily from Anthropic's official documentation, distinguishes documented behavior from interpretation and community practice, and covers everything from foundational prompt-design principles to advanced patterns for agentic workflows, structured output, visual analysis, and safety-critical applications. Whether you are a developer integrating the Claude API, a prompt engineer building workflows, or an advanced user pushing Sonnet 5.5 to its limits, this book will give you the framework and the specifics to do so responsibly.

Introduction: Why Sonnet 5.5 Deserves Its Own Prompting Guide

A new Claude model arrives. You paste the prompts you wrote two versions ago. Some work, some produce answers that are too terse or too chatty, and some fail in ways that feel arbitrary. You adjust the text, add a few rules, delete a few others, and hope for consistency. This is the ordinary experience of working with language models.

Claude Sonnet 5.5 changes the ordinary experience enough that you should not rely on the old habits alone. Released on September 28, 2026, it is the second model in the Claude 5.5 family after Opus 5.5, and it is positioned as the practical workhorse for everyday tasks: well-scoped coding, documents, research summaries, data extraction, and agentic workflows that need to be both capable and cost-efficient. At the same token price as Sonnet 5 (\$2 per million input tokens and \$10 per million output tokens), Anthropic reports that Sonnet 5.5 generates output more than 30 percent faster and typically costs up to 30 percent less per task due to token efficiency [1][2]. On the Terminal-Bench 4.0 agentic coding evaluation it scores 70.6 percent, compared to Sonnet 5's 10.3 percent [3]. On knowledge-work benchmarks such as GDPval-AA it scores 1844, two points behind Opus 5.5's 1846 [4]. It is the first Sonnet model to beat Pokémon Red playing only from screenshots, and early adopters including Slack and Epic Games report better accuracy and fewer steps without changing their prompts [1].

These numbers describe what the model does. They do not tell you how to prompt it.

The problem with generic prompt advice

Prompt-engineering advice is everywhere. Many guides will tell you to:

- assign a role (“You are a senior Python developer”),
- use XML tags for structure,

- add examples,
- end with “think step by step,”
- or specify output format in natural language.

Some of these techniques are sound in the abstract. Some are inherited from earlier models and earlier years. But none of them is specific to the way Sonnet 5.5 behaves under different effort settings, how it decides whether to carry a task further than you asked, how it interacts with tool use, or how its thinking configuration affects the output you get.

The most serious error you can make with Sonnet 5.5 is treating prompt text as the only lever you have. The reality is that effective use of this model depends on two coupled layers:

1. **Prompt-level instructions:** the system prompt and user messages that define the task, the scope, the constraints, and the output format.
2. **API and runtime configuration:** the effort level, thinking mode, max_tokens, tool definitions, structured-output schemas, and platform settings that shape how the model processes your prompt and what it is allowed to do.

A prompt that works beautifully at high effort may produce incomplete or overly cautious output at low effort. A prompt that relies on forced tool calls will return a 400 error because Sonnet 5.5 no longer supports them. A prompt that asks the model to “show your reasoning in your response” may trigger a refusal. Understanding these interactions is not optional if you want reliable results.

What this book covers

This book is built around a simple promise: by the time you finish it, you will know how to construct prompts for Sonnet 5.5 that are effective, efficient, and repeatable, and you will understand why each decision you make in a prompt matters.

Here is what you can expect:

- **A clear framework for prompt construction**, covering role and context, objectives, task decomposition, constraints, priorities, examples, output

specifications, evaluation criteria, tool-use instructions, verification requirements, and stopping conditions. This is not a list of tips; it is a design framework you can apply to any task.

- **A thorough explanation of Sonnet 5.5's model-specific behaviors**, based on Anthropic's official documentation: effort calibration and its impact on quality, latency, token usage, and cost; how the model steers its own initiative and scope; thinking configuration (adaptive, between_tools, and their constraints); structured output and JSON reliability; tool use; visual inputs; refusal behavior; and more.
- **Practical patterns for major use cases**: research, writing, editing, summarization, analysis, coding, debugging, software engineering, data extraction, structured output, document processing, planning, agentic workflows, multimodal analysis, long-context tasks, and complex multi-step work. Each pattern includes before-and-after examples showing weak, adequate, and improved prompts with reasoning for the changes.
- **A step-by-step prompt-engineering methodology** that takes you from defining the desired outcome through constructing the initial prompt, selecting effort and thinking configuration, specifying scope and stopping conditions, determining tool needs, defining output and verification requirements, testing, diagnosing failures, refining, and maintaining the prompt over time.
- **A dedicated treatment of prompt optimization**: how to improve accuracy, consistency, efficiency, latency, cost, tool usage, output quality, and reliability without unnecessarily inflating prompt length.
- **A catalog of common failures and anti-patterns**: why over-instructing, under-scoping, ignoring effort configuration, treating prompts as static, and making unsupported assumptions all lead to predictable problems, and how to correct them.
- **Comparative strategies across task types**: when simple prompts are sufficient and when you need structured reasoning, verification, tooling, or agentic loops.
- **Reliability, safety, and production considerations**: prompt injection, tool-result handling, mid-turn messages, refusal behavior, reasoning-extraction refusals, and when application-level safeguards are required beyond what you can control with a prompt.

What this book is not

This is not a collection of isolated “top tips.” You will see checklists and decision guides, but they are grounded in explanation, not slogans. You will not find exercises, quizzes, or practice activities; the examples and demonstrations serve as explanations in their own right.

This is not a generic API manual. While the book covers implementation considerations such as effort settings, thinking configuration, token budgets, streaming, structured outputs, tool calls, response block types, error handling, and prompt caching, the focus remains on how these technical controls interact with prompting. If you want a pure API reference, the Anthropic documentation is the primary source; this book builds on it.

And this is not speculative. Claims about Sonnet 5.5’s behavior are based on:

- **Official Anthropic documentation** (the primary source),
- **Anthropic announcements and system cards** (for empirical claims and benchmarks),
- **SDK and platform documentation** (for API-level behavior), with clear labeling when a claim is inferred, experimentally observed, or drawn from community practice. Where information is uncertain, undocumented, version-dependent, or subject to change, I say so explicitly.

How to use this book

If you are new to Claude or to prompt engineering in general, start at the beginning and read through the chapters sequentially. The structure is designed to build from foundations to mastery: first you learn what Sonnet 5.5 is and how it differs from earlier models, then you learn core prompt-design principles, then you explore model-specific techniques, then you apply them to specific domains, and finally you learn how to optimize, debug, and maintain prompts in production.

If you are already experienced with Claude or other models, you may want to jump to the chapters most relevant to your current problems. The chapters

on effort calibration (Chapter 7), initiative and scope (Chapter 8), thinking configuration (Chapter 9), structured outputs (Chapter 10), and refusals (Chapter 18) contain the most model-specific material that will likely change how you work with Sonnet 5.5 compared to earlier versions.

The chapter on common failures and anti-patterns (Chapter 21) is worth reading regardless of your experience level, because it describes problems that even sophisticated users encounter when they treat prompting as a one-time activity rather than an engineering discipline.

Throughout the book you will see inline citations in square brackets, such as [1][3][7]. These refer to the numbered references at the end of the book. If you want to verify a claim or explore a topic in more depth, follow the URL. All references point to sources I actually read and relied on during research.

A note on versions and change

As of the research date for this book, September 29, 2026, Claude Sonnet 5.5 was released one day prior. Anthropic has committed to keeping it active through at least September 28, 2027 [5]. The documentation it is built on is live and evolving. Some features discussed here are marked as beta, and Anthropic may change their behavior, pricing, or availability. This book labels such features clearly. If you encounter discrepancies between this book and current documentation, trust the official source and treat this book as a conceptual framework that will remain useful even as specific API details shift.

Now let us begin with the foundation: understanding what Sonnet 5.5 is, what changed compared to its predecessors, and why prompting it requires attention to more than just the words you write.

Chapter 1: Why Sonnet 5.5 Is Different

Sonnet 5.5 arrived with familiar branding but unfamiliar behavior. If you have prompted earlier Claude models, you might feel like you already know what to do. That feeling is a trap.

The difference is not just that Sonnet 5.5 is faster or more accurate. The difference is in how it interprets instructions, how it decides how far to carry a task, how it balances effort across steps, and how it responds when you combine prompt text with API-level controls. These are not superficial variations; they are design changes that matter for anyone who depends on Claude for work.

What “Sonnet 5.5” means: the model family and its positioning

Claude is not a single model. It is a family organized into tiers that Anthropic markets along two dimensions: capability and cost. The tiers are Opus (the top tier for demanding, long-horizon work), Sonnet (the mid tier optimized for speed and intelligence in balance), and Haiku (the low-cost tier for high-volume tasks). Within each tier there are numbered versions: Sonnet 4, 4.5, 4.6, 5, and now 5.5.

Sonnet 5.5 occupies a specific niche in this family. Anthropic positions it as offering “the best combination of speed and intelligence” [5]. It is designed for everyday tasks that are well-scoped but require real capability: building features, fixing bugs, writing and editing documents, performing knowledge work, and running agentic workflows that call tools and make decisions across multiple steps. For more complex, long-horizon tasks requiring careful judgment and extensive reasoning, Anthropic recommends Opus 5.5 [6].

This positioning is important for prompt design. When you prompt Sonnet 5.5, you are not prompting an oracle that can handle any level of complexity without guidance. You are prompting a model that excels when given a

clear objective, reasonable constraints, and enough context to proceed autonomously within its lane. Prompts that assume open-ended genius will underperform. Prompts that specify intent and boundaries will get more.

The model's performance justifies its position. On the Terminal-Bench 4.0 evaluation (a test of agentic coding ability), Sonnet 5.5 scores 70.6 percent, well above Sonnet 5's 10.3 percent and above Opus 5.5's 66.4 percent at Opus 5.5's xhigh effort setting [1][7]. On GDPval-AA, a knowledge-work benchmark covering multiple occupations, it scores 1844 compared to Opus 5.5's 1846 [1]. On the Artificial Analysis Intelligence Index it reaches 56, two points behind Opus 5.5 [8]. In customer testing, Box reports Sonnet 5.5 was "more accurate, 2.4x faster, and used 12 percent fewer total tokens" compared to the previous model [1]. Slack reports better results "on almost all of our offline Slackbot evals, in fewer steps and with about 14 percent fewer output tokens" [1].

These are not vanity metrics. They tell you that Sonnet 5.5 is competent enough for serious work, efficient enough to run at scale, and capable enough to handle tasks you would previously have reserved for Opus, provided you prompt it correctly.

Key behavioral changes compared to earlier Claude versions

If you migrated from Claude Sonnet 5 or an earlier version to Sonnet 5.5, you likely ran into breaking changes immediately. These are not incidental bugs; they reflect deliberate shifts in how the model is expected to be used.

Thinking is always on. Sonnet 5.5 uses adaptive thinking by default. The old pattern of disabling thinking with `thinking: {"type": "disabled"}` now returns a 400 error [9]. If you need to minimize upfront thinking, you must use `thinking: {"type": "between_tools"}` at high effort or below. At xhigh and max effort levels, even `between_tools` is rejected [9]. This means every request involves some reasoning; you cannot opt out entirely. Your prompts must be written with the assumption that the model will think before acting.

Forced tool use is gone. Earlier models allowed you to force a tool call via `tool_choice` with values like "any" or "tool." Sonnet 5.5 returns a 400 error for these [9]. You must use `tool_choice: "auto"` and guide the model through instructions instead. This forces a more conversational relationship with tools: you tell the model when and why to use tools rather than compelling

it mechanically. It also means prompts that relied on forced tools to guarantee a structured response must be rewritten.

Sampling parameters are locked. Setting temperature, top_p, or top_k to non-default values returns a 400 error [9]. There is no knob to tune randomness. This simplifies the configuration surface but removes a lever some developers relied on for controlling output variability. If you previously used temperature to make responses more creative or more deterministic, you now need prompt-level techniques instead.

Assistant prefill is rejected. Sonnet 5.5 does not support prefilled assistant messages [9]. If your workflow relied on guiding the model's response by starting it with a template, you must find another approach, such as structured outputs or explicit output-format instructions in the prompt.

Tokenization changed. Sonnet 5.5 uses approximately 30 percent more tokens than Sonnet 4.6 due to tokenizer changes [9]. This affects cost calculations, cache planning, and max_tokens budgeting. A prompt that fit comfortably within your budget on an older model may consume noticeably more tokens on Sonnet 5.5.

Refusals are categorized. Sonnet 5.5 introduces structured refusal categories: cyber, bio, frontier_llm, reasoning_extraction, and general_harms [9]. A request that was previously declined without explanation now returns a specific category in stop_details. This is valuable for building smarter fallback logic, but it also means prompts that inadvertently trigger these categories (such as asking the model to “show your reasoning in your response”) will produce refusals rather than the output you expect [10].

Each of these changes constrains the space of viable prompting strategies. Prompts that worked on Sonnet 4.6 or Sonnet 5 may need revision to account for always-on thinking, the absence of forced tool use, locked sampling parameters, and the new refusal categories.

What “prompting” actually controls versus what API configuration controls

One of the most common mistakes with Sonnet 5.5 is trying to solve configuration problems with prompt text, or vice versa.

Prompt-level instructions control the semantic content of the interaction:

- What task the model should perform
- How to interpret inputs
- What constraints and priorities apply
- What the output should look like
- How the model should handle ambiguity or failure
- Whether to use tools and when
- Whether to verify its work

When you write a system prompt saying “You are a Python developer who writes type-annotated code and runs tests before committing,” you are shaping the model’s behavior through language. The model interprets these instructions probabilistically; they are powerful but not guarantees.

API and runtime configuration controls the mechanical behavior of the model:

- How much computational effort the model invests (the effort parameter)
- When and how it thinks (thinking configuration: adaptive vs between_tools)
- How much output it can generate (max_tokens)
- Which tools it can call and whether their parameters are validated (tool definitions and strict mode)
- Whether outputs are constrained to a JSON schema (structured outputs)
- Whether caching is enabled and where cache breakpoints are placed

When you set `effort: "high"` or `thinking: {"type": "between_tools"}`, you are configuring the model’s execution behavior deterministically. These are not suggestions; they are controls.

The trap is assuming one layer can compensate for the other. For example:

- Writing “think carefully” in the prompt will not achieve the same effect as setting a higher effort level. Effort controls actual computational depth, not just the appearance of reasoning.
- Writing “never call tools unless explicitly asked” in the prompt will not override a `tool_choice` configuration that forces tool use, except that on Sonnet 5.5, forced tool use does not exist, so this particular conflict has been resolved at the API level.

- Setting a low effort level will not make the model follow complex, contradictory instructions more reliably; it may produce fewer tokens but also skip steps you intended.

Effective prompting for Sonnet 5.5 means choosing the right combination: using prompt text to define the task and constraints clearly, and using API configuration to tune effort, thinking, tool access, and output format. Neither layer is sufficient alone.

Why generic prompt-engineering advice often misses the mark

Much of the generic advice circulating about prompting language models is not wrong; it is just insufficiently specific.

The advice “use clear instructions” is correct but vague. With Sonnet 5.5, clarity means accounting for how effort level interacts with instruction complexity. A prompt packed with twenty constraints may be clear to a human reader but may confuse a model running at low effort, which has less computational budget to parse and weigh competing rules.

The advice “add examples” is often helpful but can be wasteful if overused. Three to five well-chosen examples, wrapped in tags like `<example>`, are generally sufficient for current Claude models according to Anthropic’s prompting best practices [11]. Adding more increases token cost and cache overhead without proportional gains.

The advice “assign a role” is useful for framing the model’s perspective but becomes noise if the role does not correspond to task requirements. Telling Sonnet 5.5 “you are a world-class software architect” before asking it to summarize a short email is instruction overhead that adds nothing.

The advice “ask for step-by-step reasoning” is tricky with Sonnet 5.5 specifically. Because thinking is always on in some form, the model already reasons internally. Asking it to “show your reasoning in your response” can trigger a `reasoning_extraction` refusal [10], because Anthropic has built safeguards that prevent the model from reproducing its internal reasoning verbatim in the response text [1]. The correct approach is not to ask for reasoning in the output, but to use adaptive thinking at an appropriate effort level and let the model think in structured thinking blocks rather than in its visible response.

The advice “use XML tags for structure” is reasonable and aligns with Anthropic’s recommended practices [11], but it does not replace structured outputs when you need guaranteed JSON schemas. For production workflows that parse model output programmatically, prompting for JSON with XML-wrapped instructions is fragile; using the structured outputs feature is more reliable [12].

The gap between generic advice and Sonnet 5.5-specific reality is exactly where this book lives. The principles are real, but their application must be calibrated to this model’s specific behavior.

How this book is structured for progressive mastery

The chapters ahead are arranged to build your capability step by step.

Part I establishes the foundations: what Sonnet 5.5 is, what the documentation says, and how prompt-level instructions interact with API configuration and application safeguards.

Part II covers core prompt-design principles: the anatomy of an effective prompt, how to choose between concise and detailed instructions, how to handle ambiguity and conflicting requirements, and how to build prompts that are robust to changing inputs.

Part III dives into model-specific techniques: effort calibration, initiative and scope control, thinking configuration, structured outputs and JSON reliability, tool use, visual inputs, long-context tasks, and refusal behavior. This is where you will find the most actionable, Sonnet 5.5-specific material.

Part IV covers domain-specific prompt patterns for research, writing, coding, data extraction, planning, agentic workflows, and more, with annotated before-and-after examples.

Part V addresses reliability, safety, and production concerns: prompt injection, mid-turn messages, tool-result handling, refusal management, and engineering prompts for real-world deployment.

Part VI focuses on optimization and mastery: improving prompts without bloating them, common failures and how to correct them, and comparative strategies across task types.

The final chapter synthesizes everything into principles you can carry forward as models and platforms continue to evolve.

If you read the chapters in order, you will not just accumulate tips; you will develop a systematic understanding of how to prompt Sonnet 5.5 well. Let us begin with the next chapter: understanding the landscape of instructions, configuration, and safeguards that you must navigate.

Chapter 2: The Prompting Landscape: Instructions, Configuration, and Safeguards

Working with Sonnet 5.5 well means understanding that you are operating a system with multiple layers of control. The words you write in the prompt are only one layer. API parameters determine how the model processes those words. Application-level safeguards determine what happens when the model behaves unexpectedly or encounters content that triggers safety policies. Each layer has its own responsibilities, and confusing them leads to predictable failures.

This chapter maps the full landscape so you know what each lever does, where they interact, and where one layer cannot compensate for a weakness in another.

The dual levers: what you write in the prompt versus what you set in the API call

The fundamental distinction in Sonnet 5.5 usage is between:

- **Prompt-level instructions:** the system prompt and user messages that define intent, constraints, and output requirements.
- **API/runtime configuration:** parameters that control how the model executes, what it is allowed to produce, and how resources are allocated.

These are not interchangeable. Consider a practical example: you want Sonnet 5.5 to generate a JSON object describing a software bug fix.

Prompt-level approach only: You write: “Return the result as valid JSON with fields: file, line, description, fix.” You send this as a system prompt and the bug details as a user message. Result: The model often produces correct

JSON, but occasionally adds commentary before or after it, or invents fields, or produces malformed JSON that your parser cannot handle.

Structured outputs approach: You define a JSON schema with required properties: `file` (string), `line` (integer), `description` (string), `fix` (string). You configure the request with `output_config.format` set to that schema. You still include instructions in the prompt. Result: The model's output is guaranteed to conform to the schema, within the constraints of token limits and refusals [12].

In the first approach, you are asking the model to comply. In the second, you are constraining its output mechanically. The structured outputs feature is an API-level control that provides reliability no amount of prompt text can match.

This distinction appears throughout Sonnet 5.5 usage:

- If you want the model to reason more deeply, you set a higher effort level rather than writing “think very carefully.”
- If you want the model to use tools only when appropriate, you define tools and rely on its instruction-following rather than forcing tool calls (which Sonnet 5.5 no longer supports).
- If you want predictable output formats for programmatic parsing, you use structured outputs rather than asking for JSON in natural language.
- If you want to handle safety-related refusals gracefully, you implement application-level fallback logic rather than hoping a clever prompt will avoid the refusal.

Understanding this duality is the first step toward reliable prompting.

Effort levels: what they are, what they control, and how to choose

Effort is one of the most important configuration parameters for Sonnet 5.5, and also one of the most misunderstood.

The `output_config.effort` parameter controls how many tokens the model uses across text, tool calls, and thinking [13]. It is a tradeoff: more effort means more thorough responses but higher latency and token cost. Less effort means faster, cheaper responses but potentially less completeness.

Sonnet 5.5 supports five effort levels [13]:

- **max**: The deepest level of reasoning. Reserved for tasks where you have measured quality gains and the additional cost is justified.
- **xhigh**: Designed for long-horizon tasks that require extended reasoning and planning.
- **high**: The default on the Claude API. Provides excellent results for most use cases.
- **medium**: Balanced. Recommended as the starting point for agentic workflows and well-specified tasks. This is the default in Claude Code [6].
- **low**: Maximum efficiency. Best for simple tasks where completeness and depth are not critical, or for latency-sensitive applications.

Anthropic's documentation advises starting at high for most work, using medium for agentic coding and well-specified tasks, and reserving xhigh and max only where your evaluations show a quality gain that justifies the cost [10].

How effort actually affects behavior is nuanced:

At higher effort levels the model:

- Produces more thorough reasoning in its thinking blocks.
- Is more likely to carry a task further than explicitly requested, adding tests, documentation, or follow-up suggestions.
- Makes more tool calls to gather information and verify its work.
- Uses more output tokens overall.

At lower effort levels the model:

- Produces shorter, more direct responses.
- Is more likely to check in with you before proceeding beyond the explicit request.
- May skip verification steps you expect, such as running tests on code changes [10].
- Uses fewer output tokens and completes faster.

These behaviors matter for prompt design. A prompt that works well at high effort may produce incomplete output at low effort because the model skips steps it would normally take. A prompt that is too open-ended at high effort may generate excessive output because the model adds work you did not ask for. We will explore this interaction in detail in Chapter 7.

For now, the key takeaway is: effort level is not optional configuration. It is a fundamental design decision that shapes how your prompt is interpreted and executed.

Thinking configuration: upfront thinking versus `between_tools` and adaptive reasoning

Sonnet 5.5 always thinks in some form. The question is when and how.

The thinking configuration parameter determines the mode of reasoning. There are two valid modes for Sonnet 5.5:

1. **Adaptive thinking** (default): The model decides when and how much to think based on the task. This is what you get when you omit the thinking field or explicitly set `thinking: {"type": "adaptive"}`.
2. **`Between_tools`**: The model does no extended upfront thinking. It only writes short progress updates between tool calls, which appear as thinking blocks [9]. This mode is configured with `thinking: {"type": "between_tools"}`.

There are important constraints:

- `thinking: {"type": "disabled"}` returns a 400 error. You cannot fully disable thinking on Sonnet 5.5 [9].
- `between_tools` is accepted at low, medium, and high effort but returns a 400 error at xhigh and max effort [9].
- `between_tools` takes no other fields: sending `display`, `budget_tokens`, or `block_binding` with it returns a 400 error [9].

When to use each mode depends on the task:

Adaptive thinking is appropriate when:

- The task requires reasoning before taking action (e.g., solving a complex problem, analyzing data, writing code).
- You are not using tools extensively and want the model to think through the problem holistically.
- You are using `xhigh` or `max` effort, which require adaptive thinking.

Between_tools is appropriate when:

- You want the model to act immediately and only reason between tool calls (e.g., agentic workflows with many tool calls).
- You are building a tool-heavy workflow where upfront planning is unnecessary or wasteful.
- You want to minimize latency for simple tasks by avoiding extended upfront reasoning.

Reading responses by block type is critical in both modes. Because Sonnet 5.5 includes thinking blocks in its responses, code that blindly reads `content[0].text` will break [6]. You must iterate over response blocks by type: handle thinking blocks (typically passing them back unchanged in multi-turn workflows), text blocks, and `tool_use` blocks separately [6].

Thinking tokens bill as output tokens. This means that more thinking directly increases your cost. If you are optimizing for efficiency, understanding how thinking mode and effort level affect token usage is essential.

Token budgets and `max_tokens` as design decisions

The `max_tokens` parameter sets the maximum number of output tokens the model can generate in a single response. Sonnet 5.5 supports up to 128,000 output tokens natively, and up to 300,000 via the Message Batches API with a beta header [5].

Setting `max_tokens` is not just about preventing runaway generation; it is a design decision that shapes task structure.

For simple tasks, a modest `max_tokens` (such as 4,096 or 8,192) is sufficient and helps contain output. For complex, multi-step tasks, especially agentic coding, Anthropic recommends setting `max_tokens` to the model's maximum

(128,000) to give the model room to complete multi-step work without being truncated [6].

The interaction with thinking is important: thinking tokens count toward `max_tokens`. If you set a very low `max_tokens` and the model spends tokens on thinking, it may have insufficient budget for the actual response. Conversely, if you set a very high `max_tokens` and the task is simple, you are not wasting money because you pay for what the model actually generates, not for the maximum you allow. However, excessively high `max_tokens` combined with high effort on an open-ended task can lead to longer-than-expected responses.

For production workflows, a practical strategy is:

- For well-scoped single-step tasks: set `max_tokens` to a reasonable upper bound (e.g., 4,096–16,384).
- For agentic workflows and long-horizon tasks: set `max_tokens` to 128,000.
- Monitor actual token usage and adjust `max_tokens` based on empirical observations.

Token budgets also interact with cost optimization. Because Sonnet 5.5 costs up to 30 percent less per task than Sonnet 5 in Anthropic’s testing due to token efficiency [1], understanding how `max_tokens` and effort level affect actual usage is directly relevant to controlling costs.

Where application-level safeguards matter more than prompt instructions

Prompts can guide behavior but they cannot guarantee safety, correctness, or security. There are domains where relying solely on prompt-level instructions is insufficient and potentially dangerous.

Prompt injection protection: You cannot prevent prompt injection through prompt text alone. If untrusted content enters a prompt (such as user-provided text or tool results), the model may interpret it as instructions rather than data. The correct approach is architectural: treat system prompts and user messages as separate trust boundaries, sanitize tool results, and never place untrusted external data directly into system messages [14].

Safety refusals: Sonnet 5.5 has built-in safety classifiers that may refuse certain requests regardless of how carefully you phrase them. The refusal categories (cyber, bio, frontier_llm, reasoning_extraction, and general_harms) are enforced by the model and cannot be bypassed through prompting [9][10]. For cyber and frontier_llm refusals, you can use server-side fallback (beta) to retry on Sonnet 5, but this is an application-level feature, not a prompt-level solution [10].

Output validation: For tasks where correctness is critical (financial calculations, medical information, legal analysis), prompt instructions such as “be accurate” or “double-check your work” are not sufficient. You need application-level validation: checking outputs against ground truth, running automated tests, or requiring human review.

Data protection: If you are processing sensitive data (PHI, PII, trade secrets), prompt instructions about confidentiality are inadequate. You must implement data isolation, zero data retention where available, and appropriate access controls at the platform and application level [12].

Tool-result handling: When the model calls tools and receives results from external systems, those results may contain unexpected or malicious content. Prompt-level instructions about how to handle tool results are fragile. You should sanitize tool results before returning them to the model, and design your tools to return bounded, well-structured data.

Rate limiting and failure handling: Prompt text cannot prevent API rate limits, network failures, or model outages. Your application must implement retry logic, timeout handling, and graceful degradation.

The lesson is clear: prompts are part of a larger system. Design your prompts to work within that system, not as if they are the entire system.

Summary

This chapter has established the landscape within which Sonnet 5.5 prompting operates. You now understand that:

- Effective prompting requires attention to both prompt-level instructions and API/runtime configuration; neither is sufficient alone.
- Effort levels (low through max) control the model’s computational depth and directly affect response quality, latency, token usage, and cost.

- Thinking configuration (adaptive vs between_tools) determines when the model reasons, with important constraints at higher effort levels.
- Token budgets (max_tokens) are a design decision that shapes task structure and must account for thinking tokens.
- Application-level safeguards are required for safety, security, correctness, and reliability in ways that prompt instructions alone cannot provide.

In the next chapter, we will examine the official Anthropic documentation more closely to understand what is documented versus inferred, and how to interpret the guidance responsibly.

Chapter 3: Reading the Official Documentation: What Anthropic Says About Sonnet 5.5

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

The “Prompting Claude Sonnet 5.5” document: structure and intent

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Categories of official guidance: documented behavior, recommendations, observations, beta features

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

How to interpret empirical claims from the documentation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Where the documentation is silent and what that means

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Navigating version-specific and evolving behavior

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 4: The Anatomy of an Effective Prompt

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Role and context: when it helps, when it is noise

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Objectives and success criteria: stating what “good” looks like

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Task decomposition: breaking complex work into model-friendly units

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Constraints and priorities: guiding tradeoffs without overconstraining

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Input organization: structuring information for clarity and retrieval

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Examples and output specifications: when they are essential versus optional

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Evaluation criteria: giving the model a standard for self-checking

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Stopping conditions: telling Claude when it is done

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 5: Concise Versus Detailed: Finding the Right Level of Instruction

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

The cost of verbosity: token overhead, attention dilution, and contradictory instructions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

The cost of brevity: ambiguity, missed constraints, and inconsistent output

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When concise prompts succeed: routine tasks, constrained domains, expert users

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When detailed prompts are necessary: novel tasks, ambiguous domains, production reliability

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

How to compress prompts without losing essential guidance

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Testing prompt length as a variable

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 6: Ambiguity, Conflicting Requirements, and Robustness

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Why ambiguous prompts produce unreliable results

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Resolving conflicting requirements: prioritization strategies

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Explicit ambiguity-handling instructions: what to do when unsure

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Designing for changing inputs: prompts that adapt rather than break

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Failure recovery: asking the model to flag problems instead of hallucinating solutions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 7: Effort Calibration: Quality, Latency, Cost, and the Tradeoff Triangle

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

What “effort” means for Sonnet 5.5: official definition and practical effects

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Low effort: when to use it, what you gain, what you risk

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Medium effort: the default operating point for agentic work

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

High effort: the API default for excellent results

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Xhigh and Max: complex reasoning, high-stakes tasks, and when it earns its cost

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Measuring the impact: quality vs latency vs token usage vs cost

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

How prompt complexity interacts with effort levels

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Decision framework: choosing the right effort for your task

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 8: Steering Initiative and Scope: Controlling How Far Claude Goes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

The initiative problem: why Claude adds extra steps, explanations, and recommendations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When extra initiative is valuable and when it is wasteful

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Explicit scope-limiting instructions: patterns that work

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Preventing unnecessary additions: concise task-bounding language

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Handling open-ended requests: when broadness is a feature versus a bug

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

User-facing progress updates: prompting Claude to report back during long tasks

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Controlling follow-up questions and clarifications

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 9: Thinking Configuration: Adaptive Reasoning and Strategic Depth

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

The three modes: upfront thinking, between_tools, and adaptive thinking

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When upfront thinking helps: complex reasoning before action

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When between_tools helps: iterative planning during tool use

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When adaptive thinking helps: letting the model choose

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Reasoning tasks that produce structured output: special considerations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

How thinking affects latency, token cost, and output quality

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Common misconceptions about thinking and “smarter” responses

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Reading responses by block type

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 10: Structured Outputs and JSON Reliability

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Why structured output matters for production workflows

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Prompt-level techniques: schema specification, examples, and constraints

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

API-level structured outputs: the dedicated feature and how it differs from prompting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

JSON reliability: factors that improve or degrade format adherence

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Handling schema violations: tolerance, error recovery, and retries

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Complex schemas: nested objects, arrays, unions, and optional fields

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When to use structured outputs versus parsing free-form responses

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 11: Tool Use, Research, and Current Information

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

How tool use integrates with prompting: definitions, instructions, and constraints

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Tool use for research: prompting Claude to gather and synthesize current information

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Prompting for appropriate tool selection: avoiding overuse and underuse

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Tolerant tool-call handling: parsing errors, retries, and fallback behavior

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Long-running agentic workflows: structuring multi-step tool interactions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Combining tool results with reasoning: synthesis and verification patterns

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 12: Visual Inputs: Charts, Diagrams, and Technical Drawings

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

What kinds of visual inputs Sonnet 5.5 handles well

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Prompting for chart and graph analysis: reading values, trends, and patterns

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Technical drawings and schematics: structured extraction from diagrams

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Combining visual and textual prompts: coordinated instructions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Limitations and failure modes with complex visuals

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Crop tool: improving accuracy on dense charts and drawings

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 13: Long-Context Tasks and Massive Inputs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Context window capabilities and practical limits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Organizing large inputs: summaries, indexes, and structured access

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Prompting strategies that degrade with context length and alternatives

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Multi-pass processing: chunking, summarizing, and integrating

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Maintaining coherence across long conversations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When long context helps and when smaller focused contexts work better

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 14: Research, Writing, and Content Workflows

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Research prompts: scoping, sourcing, synthesizing, and citing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Writing prompts: drafting, voice, structure, and completeness

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Editing prompts: critique, revision, tone adjustment, and style enforcement

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summarization: faithful compression without loss of meaning

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Analysis prompts: comparing sources, extracting insights, and drawing conclusions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chain-of-prompts for complex content projects

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 15: Coding, Debugging, and Software Engineering

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Coding task scoping: when to be specific and when to trust the model

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Debugging prompts: providing context, error messages, and expected behavior

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Software engineering: architecture, refactoring, and design review

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Verification requirements: testing, edge cases, and self-check patterns

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Incremental development: building codebase changes step by step

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When additional tools or human review are non-negotiable

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 16: Data Extraction and Document Processing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Field extraction: schemas, mappings, and handling ambiguity

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Document classification and categorization

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Entity recognition and relationship extraction

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Dealing with poor-quality, noisy, or inconsistent source documents

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Batch processing patterns and maintaining consistency

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Validation and spot-checking strategies

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 17: Planning, Agentic Workflows, and Multi-Step Tasks

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Task planning: prompting Claude to generate executable plans

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Step-by-step execution: maintaining state and progress

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Agentic loops: when Claude should decide what to do next

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Mid-turn user messages: handling interruptions and new information

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Error recovery in long workflows: backtracking and correction

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Designing workflows that are robust to individual step failures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 18: Prompt Injection, Safety, and Refusals

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Prompt injection fundamentals: what it is and how it works

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

System prompt versus user message: trust boundaries

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Mid-turn prompt-injection considerations with user messages during long tasks

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Tool-result handling: why untrusted data needs sanitization

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Refusal behavior: categories, triggers, and workarounds (where legitimate)

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Reasoning-extraction requests: what is possible and what is restricted

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When application-level safeguards are required beyond prompts

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 19: Reliability Engineering: Making Prompts Production-Ready

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

What “reliable” means for LLM prompts: consistency, correctness, and recoverability

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Versioning prompts: tracking changes and impact

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

A/B testing prompts: comparing quality and efficiency

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Monitoring output quality: metrics and alerting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Handling model updates: when behavior changes and how to adapt

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Building fallback strategies for when prompts fail

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 20: Prompt Optimization: Improving Without Bloating

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Diagnose-first optimization: identifying the real bottleneck

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Reducing instruction overhead: compression techniques

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Improving accuracy without adding tokens

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Latency optimization: prompt structure, effort, and thinking choices

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Cost optimization: token efficiency across the request-response cycle

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

When to stop optimizing: diminishing returns and engineering judgment

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 21: Common Failures and Anti-Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Over-instructing: the paradox of too many rules

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Under-scoping: open-ended prompts that run too far

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Ambiguous success criteria: when “good” means nothing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Assuming capabilities: expecting what the model cannot reliably do

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Ignoring effort and thinking configuration

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Treating prompts as static: neglecting iteration and testing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Confusing prompt-level fixes with system-level requirements

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 22: Comparative Strategies: Matching Approach to Task Type

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Simple requests: minimal instruction, low effort, no thinking

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Reasoning-intensive tasks: structured thinking, higher effort, verification

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Coding tasks: context-rich, verification-aware, incremental

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Research tasks: tool-heavy, iterative, synthesis-focused

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Agentic workflows: stateful, adaptive, robust to failure

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Structured JSON generation: schema-driven, validation-heavy

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Multimodal tasks: coordinated visual and textual instructions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Decision guide: matching approach to task

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Chapter 23: Principles, Patterns, and What Comes Next

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Core principles of effective Sonnet 5.5 prompting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

The most impactful patterns reviewed

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

What prompting practice will look like as models evolve

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Staying current: monitoring Anthropic documentation and platform changes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Final thoughts on treating LLMs as collaborative tools, not oracles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

Conclusion: The Craft of Prompting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

From novice to practitioner: the journey mapped

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

What makes a great prompter: skills and mindset

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

The enduring value of understanding how the model works

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.

References

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/thesonnet55promptingguide>.