INNOVATION AI LAB

INO AI LAB

[AI EDUCATION • COURSE MATERIAL]

Claude Deep Dive: Coding & Analysis

Sonnet 4.5, Opus 4.5, Projects, Artifacts

LEVEL	DURATION	LESSONS
Intermediate	**3.5 hours**	**8**

Innovation AI Lab — Tools, Prompts & Performance

// 2026 EDITION

[00]

Table of Contents

[LESSON 01]

Why Claude Wins on Code and Long Documents

Anthropic's Claude line has carved out two clear strengths: code editing and long-context reasoning. Sonnet 4.5 is the workhorse — fast, cheap enough for daily coding, and trained specifically to follow large refactors across many files. Opus 4.5 is the heavy hitter for architecture reviews, complex debugging, and 200K+ token documents. Most teams default to Sonnet and escalate to Opus only when Sonnet stalls.

Claude's tone leans careful and verbose — it will ask clarifying questions where ChatGPT charges ahead. This is a feature for high-stakes work and a friction for quick chat. Adjust with a system prompt: 'Be terse. Skip preamble. Don't ask permission for obvious next steps.' One line reshapes the whole experience and saves hundreds of tokens per turn.

// KEY TAKEAWAYS

- Sonnet by default, Opus when Sonnet stalls.
- Claude excels at code and 200K+ token docs.
- One system line tames the verbosity.

[LESSON 02]

Projects, Artifacts, and the Computer Use Tool

Claude Projects pin a knowledge base — paste your style guide, code conventions, and reference docs once, and every chat inherits them. Artifacts open a side panel for code, docs, diagrams, and small apps; they're the right surface for anything you'll iterate on more than twice. Click into an artifact, edit, and ask Claude to update — version history is built in.

Computer Use lets Claude control a desktop in a sandbox: open apps, click buttons, fill forms. It's still rough in 2026 — slow, occasionally confused — but useful for repetitive UI tasks no API covers. Combine with screenshots for verification. As with any agent, keep scopes narrow, log every action, and never give it credentials to systems you can't afford to lose.

// KEY TAKEAWAYS

- Projects pin knowledge across all chats.
- Artifacts beat chat for anything iterated 2+ times.
- Computer Use: powerful, slow, supervise heavily.

[LESSON 03]

Coding Workflows With Claude Code

Claude Code is the terminal-native coding agent that runs locally, reads your repo, edits files, runs tests, and commits. The workflow that works best: write a clear plan in a markdown file, ask Claude Code to implement against the plan, review the diff, and only then run tests. Skipping the plan step turns Claude Code into an expensive guess engine; with the plan, it ships PRs that pass review.

Set guardrails in a CLAUDE.md at the repo root: which directories to avoid, which patterns to follow, what 'done' means. Add a pre-commit hook for tests and linting so Claude can self-check before claiming completion. Track session cost; pair coding with Claude Code is genuinely productive but easy to over-spend if you let it loop on a hard bug for an hour.

// KEY TAKEAWAYS

- Plan in markdown before coding.
- CLAUDE.md sets repo-wide guardrails.
- Pre-commit hooks let the agent self-verify.

[LESSON 04]

Long-Context Patterns: 200K Tokens Done Right

With a 200K context, you can paste an entire small codebase, a full book, or a quarter of meeting transcripts. But context isn't free — quality drops on details in the middle ('lost in the middle' effect). Counter this by putting the most important context at the top and bottom, restating the task after long inputs, and using XML tags to delimit sections.

For very long inputs, prefer retrieval over context-stuffing. Build a simple RAG layer that pulls only relevant chunks per query. Reserve the full 200K for tasks that genuinely need cross-document synthesis — competitor analysis, due diligence, multi-quarter financial review. Measure: if a 20K-token version produces the same answer, the extra 180K is just burning money and latency.

// KEY TAKEAWAYS

- Put critical context at top and bottom.
- Restate the task after long inputs.
- Use RAG when full context isn't proven necessary.

[NEXT]

Keep Going

You've completed this course material. The real learning starts when you apply what you've read. Pick one idea from this PDF and run a small experiment this week. Document what worked and what didn't. Share your findings with the community.

Explore more free courses, daily AI tips, and curated tools at:

innovationailab.com

Have feedback or want to suggest a topic? We read every message.

hello@innovationailab.com

— Innovation AI Lab Team —

// part of LumiLife Tech