AI Coding Cheatsheet 2026

EG3 · EG3.COM

Reference Card · Updated April 2026

Tool Comparison

TOOL	TYPE	PRICING	MODELS	BEST FOR	KEY LIMITATION
Cursor	IDE (VS Code fork)	Hobby (free, limited), Pro ($20/mo), Pro+ ($60/mo), Ultra ($200/mo)	Claude, GPT, Gemini (usage credits scale with tier)	Inline editing, Composer mode, visual agent workflows	Hits usage limits fast on complex agent tasks in lower tiers
VS Code + GitHub Copilot	IDE	Pro ($10/mo), Pro+ ($39/mo)	Multiple including Claude Opus 4.6 on higher tiers	Familiar VS Code users wanting lightweight autocomplete and chat	Less autonomous than dedicated agents. Context window and agent depth lag behind frontier tools
Windsurf	IDE (AI-native editor)	Free for individuals. Paid for teams (~$15+/user/mo)	Proprietary + frontier models via integrations	Flow-state editing. Developers who want an agent-powered editor without switching from VS Code habits	Newer ecosystem. Less mature third-party extensions than VS Code
Claude Code CLI	CLI	Access via Anthropic Pro ($20/mo) or higher tiers	Sonnet 4.6 (default), Opus 4.6, Haiku 4.5	Complex refactors, automation, multi-file changes, agent teams	Steeper learning curve. No built-in visual editor. Requires careful permission setup

Cursor Strengths

- Deep IDE integration with tab completions that feel native.
- Composer mode handles multi-file edits cleanly.
- Supports MCP, skills, hooks, and cloud agents on paid plans.
- Excellent for iterative vibe coding inside the editor.

Cursor Gotchas

- Credit system burns through fast when you chain agents or use expensive models.
- Can feel like a heavier VS Code fork. Some extensions break.
- Agent reliability drops on very large codebases without careful context management.

Claude Code CLI Strengths

- Raw model access with massive context. It reads your entire codebase effectively.
- Built for delegation. It runs commands, edits files, runs tests, and iterates in a loop.
- MCP servers, hooks, skills, and experimental agent teams let you build real automation.
- Headless mode drops straight into CI/CD.

Claude Code CLI Gotchas

- Terminal only. You lose the visual diff comfort of an IDE.
- Permission model is strict. You'll fight .claudeignore and approval prompts early on.
- Agent teams are still experimental. Coordination sometimes falls apart on ambiguous tasks.

VS Code + GitHub Copilot Strengths

- Cheap and everywhere. $10/mo gets you solid daily assistance.
- Works with your existing setup and extensions.
- Cloud agent and review features on Pro+.

VS Code + GitHub Copilot Gotchas

- Agentic capabilities are more conservative than Cursor or Claude Code.
- Context management feels less intelligent on large projects.

Windsurf Strengths

- Designed from the ground up for AI flow state.
- Free individual tier is genuinely useful.
- Cascade agent feature handles autonomous tasks well.

Windsurf Gotchas

- Smaller extension ecosystem.
- Less documentation and community knowledge than the other three.

IDE vs CLI Use IDE tools when you want to stay in flow. Cursor or Windsurf keep you inside the editor with inline suggestions, visual diffs, and quick iterations. They shine for feature implementation, debugging, and refactoring where you want to see the code change immediately.

Use CLI when you need delegation and automation. Claude Code excels at long-running tasks like "refactor this authentication layer across 17 files, update tests,. And make sure the build still passes." The terminal gives cleaner context control, easier scripting, and better integration with git hooks or CI. Nobody builds reliable agent pipelines inside a GUI as easily as they do in the shell.

CLI matters for automation because it's scriptable, deterministic, and plays nicely with existing devops tools. IDE agents are better pair programmers. CLI agents are better junior engineers you can assign tickets to.

Best Practices for AI-Assisted Coding Write specific prompts. Give the agent the exact file, the surrounding context, the acceptance criteria, and examples of the style you want. Vague prompts produce vague code.

Use agent mode for multi-step tasks. Inline chat or tab autocomplete for small fixes. Switch to full agent when the change spans files or requires running tests.

Review every line of AI output. AI doesn't understand your business logic or the subtle performance traps in your stack. Treat it like a very fast but overconfident intern. Run the tests, check for security issues, and verify edge cases yourself.

Common pitfalls include accepting hallucinated APIs, letting agents touch configuration files without review, and over-relying on one model. Switch models when one struggles, and Claude is strong at reasoning. GPT variants sometimes produce more practical code.

Practical Workflow Example (IDE + CLI) Start in Cursor for initial exploration. Use tab completions and Composer to sketch a new feature in the files you're actively editing.

When the scope grows, export the task description to a CLAUDE.md file in the project root.

Switch to terminal and run Claude Code CLI with a precise prompt. "Implement the payment retry logic described in CLAUDE.md. Update the service layer, add tests, and run the full test suite." Use hooks to enforce linting and security checks on every edit.

Review the diffs in your IDE. Merge what looks good. Send the remaining issues back to the CLI agent with a follow-up command.

Iterate between the two tools. Cursor for speed and visibility. Claude Code for depth and autonomy. This hybrid approach avoids the weaknesses of either environment alone.

If your project grows beyond what one agent can handle, enable agent teams in Claude Code. The lead agent delegates subtasks while you watch progress from the IDE. That combination is where real productivity shows up in 2026.