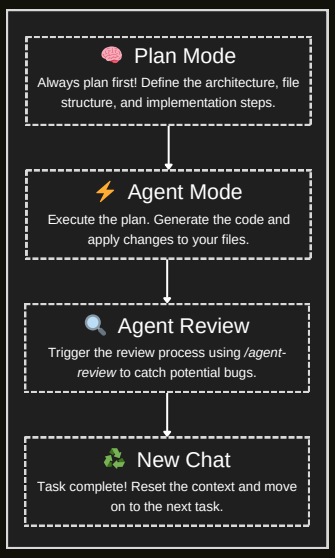
CURSOR CHEAT SHEET

Pricing per 1M Tokens

Model	Default Context	Max Mode	Input	Cache Write	Cache Read	Output
Claude 4.6 Opus	200k	1M	$5	$6.25	$0.5	$25
Claude 4.6 Sonnet	200k	1M	$3	$3.75	$0.3	$15
Composer 1.5	200k	-	$3.5	-	$0.35	$17.5
Gemini 3.1 Pro	200k	1M	$2	-	$0.2	$12
GPT-5.3 Codex	272k	-	$1.75	-	$0.175	$14

Recommended Workflow



Mode	Best For	Capabilities
Agent	Complex developments, refactoring	Autonomous exploration, multi-file editing
Ask	Learning, planning, questions	Read-only inspection, makes no automatic changes
Plan	Complex developments requiring planning	Generates detailed plans before execution, asks clarifying questions
Debug	Challenging bugs	Hypothesis generation, log instrumentation, runtime analysis

Privacy Settings	Description
Privacy Mode	Your code is never used as training data by Cursor or third-party model providers. Full details at: trust.cursor.com
Share Data	Your code data is shared to help improve Cursor for everyone.

Auto Mode

Allows Cursor to select the model best fit for the immediate task and with the highest reliability based on current demand. It can detect degraded output performance and automatically switch models to resolve it.

Pricing:

- Input + Cache Write: $1.25 per 1M tokens
- Output: $6.00 per 1M tokens
- Cache Read: $0.25 per 1M tokens

Max Mode

- Max Context: Expands the context window to the maximum supported by the model.
- Cost: Adds a +20% premium to the base API price.
- Best For: Users who want the best experience regardless of cost.

Agent Skills `/create-skill`

Agent Skills are portable, version-controlled packages that teach the AI Agent specific tasks and workflows. They keep context usage highly efficient by loading resources strictly on-demand. Cursor automatically discovers these skills on startup, and the Agent dynamically decides when they are relevant based on your current context.

- Type */migrate-to-skills* in the Agent chat to automatically convert your existing rules and commands into the "Skill" format.

Rules `/create-rule`

✅ Best Practices

- < 500 Lines: Keep rules short, split large ones into composable pieces.
- Clear & Focused: Avoid vague guidance, write like clear internal docs.
- Reference Files: Don't copy code, point to the referenced file instead.
- Automate: Convert your frequently repeated prompts into rules.

❌ What to Avoid

- Basic Tools: Don't teach common tools like npm, git, or pytest (Agent already knows them).
- Rare Edge Cases: Skip the edge cases, keep rules focused on frequently used patterns.

.cursorignore

Files added here are completely hidden from AI features (Agent, Tab, Inline Edit) and codebase search. Use this to protect passwords and API keys.

Subagents `/create-subagent`

Specialized AI assistants with their own isolated context windows that the main Agent can delegate complex tasks to. They can perform background research or execute multiple tasks in parallel without bloating the main chat's context. The Agent can automatically spawn a Subagent with the necessary context when faced with a complex task, or you can directly invoke one in the chat.

- Each subagent should have a single, clear responsibility. Avoid creating general-purpose subagents.

Emre ÖZCAN | Cursor Ambassador