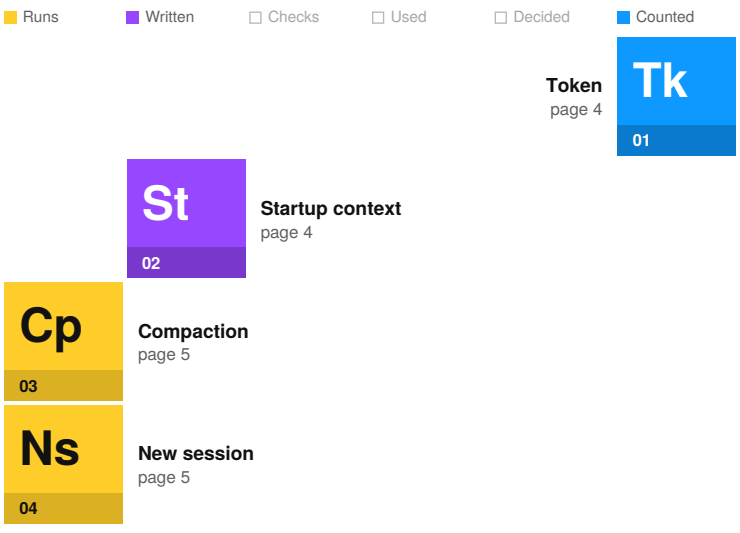
# Context Windows

A pocket-sized guide to what fills a coding agent's context window, as of October 2026.




*My friend has a lot of questions.*

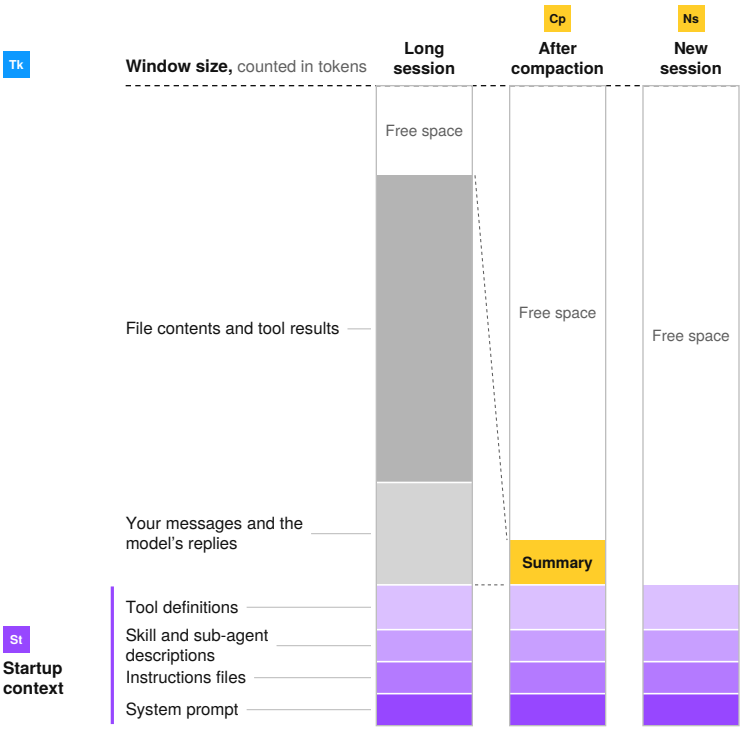
# What is the context window?

The context window is everything the model can see on one call. It holds the instructions and tool descriptions the harness loads, your conversation, and every file and command output the agent has read. The window has a fixed size, so a long session fills it.

An Accidental Guide to: Agents, Harnesses, Skills, and Tools covers the basics and lists what the window holds. These four ideas explain how the window fills up and how you keep room for the work.

| | |
|---|---|
| **Token** | A short piece of text used to count the window's size and usage. |
| **Startup context** | What the harness loads before your first message. |
| **Compaction** | Replacing older conversation with a summary when the window is nearly full. |
| **New session** | An empty conversation that starts with only the startup context. |

# What takes up room?




The startup context loads before you type your first message and stays in the window for the whole session. The conversation grows with every message you exchange and every file and command output the agent reads. Compaction replaces the conversation with a short summary. A new session has only the startup context. The sizes in the picture are examples.

**Tip.** A crowded window can give worse answers before it's full, because the model has more to read and the details you need are harder to find. To see how full yours is, run `/context` in Claude Code or GitHub Copilot CLI, or `/status` in Codex, as of October 2026.

# Token

Tk
01

**Identify by** A window size, usage count or price stated in tokens.

**What** **A token is a short piece of text.** Window sizes, usage and prices are all counted in tokens.

**So what** **The whole window is sent on every call.** A large file or log costs its tokens again on every later call.

**Field marks**

| Controlled by | Cannot |
| --- | --- |
| **The model you pick.** | **Be counted from words.** |

**Confused with** **Word**
A short, common word is often one token. A long or rare word can take several.

# Startup context

St
02

**Identify by** Content in the window before your first message.

**What** **Startup context is what the harness loads first.** That's the system prompt, instructions files, skill and sub-agent descriptions, and tool definitions.

**So what** **Startup context is sent on every call.** Long instructions files and many MCP servers take room before you start.

**Field marks**

| Controlled by | Cannot |
| --- | --- |
| **You and the harness.** | **Hold every procedure cheaply.** |

**Confused with** **Memory**
Harness memory is a file of notes loaded with the startup context. The model keeps nothing between calls.

# Compaction

Cp
03

**Identify by** A message saying the conversation was compacted.

**What** **Compaction swaps older conversation for a summary.** GitHub Copilot CLI does it at about 80 percent full, as of October 2026.

**So what** **Compaction can lose early instructions and exact output.** Rules that must last belong in an instructions file.

**Field marks**

| Controlled by | Cannot |
|---|---|
| **The harness and you.** | **Bring lost detail back.** |

**Confused with** **Trimming**
Trimming drops old content without a summary. Some harnesses trim before they compact.

# New session

Ns
04

**Identify by** An empty conversation after `/clear`, `/new` or opening a session.

**What** **A new session starts with only the startup context.** Only content saved in files carries over.

**So what** **A fresh window has room for a new task.** Have the agent save its plan and progress to a file first.

**Field marks**

| Controlled by | Cannot |
|---|---|
| **You.** | **Recall the old conversation.** |

**Confused with** Cp **Compaction**
Compaction keeps a summary in the same session. A new session keeps only files.

# What keeps the window useful?

A few habits keep the window small and on topic, so the model can find what matters.

| | |
|---|---|
| **Start fresh for unrelated work** | A new session gives the next task an empty conversation. |
| **Save the plan and progress** | A file carries what matters into the next session. |
| **Send long work to sub-agents** | Searches, test runs and log reading stay in separate windows. Only summaries return. |
| **Read only what you need** | Ask for one function or only the failing tests. |
| **Keep instructions files short** | Startup context is sent on every call. Move specific procedures into skills. |
| **Compact at a natural break** | When one part of the work is done, run `/compact` and say what to keep. |

# Sources

These public sources were checked on 22 October 2026, and harnesses change often.


**Anthropic, Claude Code: Explore the context window**
`code.claude.com/docs/en/context-window`

**Anthropic, Claude Code: How Claude Code works**
`code.claude.com/docs/en/how-claude-code-works`

**Anthropic, Claude Code: Manage costs effectively**
`code.claude.com/docs/en/costs`

**Anthropic, Context windows**
`platform.claude.com/docs/en/build-with-claude/context-windows`

**Anthropic, Effective context engineering for AI agents**
`anthropic.com/engineering/effective-context-engineering-for-ai-agents`

**GitHub, About context management**
`docs.github.com/en/copilot/concepts/prompting-and-context/context-management`

**OpenAI, Codex commands**
`learn.chatgpt.com/docs/developer-commands`

**Chroma, Context Rot**
`trychroma.com/research/context-rot`

**HumanLayer, Getting AI to Work in Complex Codebases**
`github.com/humanlayer/advanced-context-engineering-for-coding-agents`






An Accidental Guide to…

**Context Windows**

A pocket-sized guide to what fills a coding agent's context window, as of October 2026.
Written and reviewed by a person. AI does the grunt work. Set in TeX Gyre Heros and Newsreader.
Leave a star: `accidental.info/star`


*My friend has a lot of questions.*


oops@accidental.info · CC BY 4.0 · October 2026