

Sidebar

Programming · VS Code Chat and AI Cheat Sheet

VS Code's built-in chat, agent mode and inline suggestions, plus the Claude Code extension: the settings to search for, what they do, and their defaults. Sheet 13 in the VS Code series. These settings change quickly between versions; this sheet was checked against VS Code 1.139.1 and Claude Code extension 2.1.285.

CHAT: GETTING AROUND

- `Ctrl+Alt+I` Open the Chat view (side bar)
- `Ctrl+I` Inline chat: ask for an edit right where the cursor is
- `@name` Address a chat participant, e.g. `@workspace` or `@terminal`
- `/command` Slash commands such as `/explain`, `/fix`, `/tests`, `/clear`
- `#file` `#selection` `#codebase` Attach context to a question: a file, your selection, or a search of the project
- `Ask / Edit / Agent` Chat modes: answers only / edits you review / autonomous multi-step work with tools
- `chat.newSession.defaultMode` Mode a new chat starts in (default empty, meaning the product default)

TOOL APPROVAL & SAFETY

- `chat.tools.global.autoApprove` `true` approves every tool call with no prompt (default `false`): use with great care
- `chat.tools.edits.autoApprove` File patterns the agent may edit without asking (object of pattern: true/false)
- `chat.tools.terminal.enableAutoApprove` Allow terminal commands to be auto-approved by rule (default `true`)
- `chat.tools.terminal.autoApprove` Commands approved (`true`) or always blocked (`false`), e.g. `{ "git status": true }`
- `chat.tools.urls.autoApprove` Web addresses the agent may fetch without asking (a short documentation list by default)
- `chat.tools.riskAssessment.enabled` Check tool calls for risk before running them (default `true`)
- `chat.agent.sandbox.enabled` `off` (default) or `on`: run agent commands in a sandbox
- `chat.agent.networkFilter` Filter the agent's network access (default `false`)
- `chat.agent.allowedNetworkDomains` / `deniedNetworkDomains` Domain lists for that filter (default `[]`)

AGENT & REQUESTS

- `chat.agent.enabled` Allow agent mode (default `true`)
- `chat.agent.maxRequests` How many steps an agent may take before asking to continue (default 50)
- `chat.permissions.default` `default` (default), `autoApprove`, `autopilot`: how freely tools may run
- `chat.checkpoints.enabled` Snapshot your files at each request so you can restore (default `true`)
- `chat.checkpoints.showFileChanges` Show which files changed at each checkpoint (default `false`)
- `chat.requestQueuing.defaultAction` A message sent mid-response: `steer` (default) the current one, or `queue` it
- `chat.saveBeforeSend` Save open files before sending a request (default `true`)
- `chat.restoreLastPanelSession` Reopen the last conversation in the panel (default `false`)
- `chat.defaultModel` Model chosen for new chats (default empty: picked for you)

REVIEWING EDITS

- `chat.editing.autoAcceptDelay` Seconds before edits are accepted automatically (default) means never
- `chat.editing.openChangedFileInDifffEditor` Open changed files as diffs (default `true`)
- `chat.editing.confirmEditRequestRemoval` Ask before removing an edit request (default `true`)
- `chat.editing.confirmEditRequestRetry` Ask before retrying an edit request (default `true`)
- `chat.editRequests` How you edit an earlier message: `inline` (default), `hover`, `input`, `none`
- `chat.undoRequests.restoreInput` Put the text back in the box when you undo a request (default `true`)
- `chat.implicitContext.enabled` Attach the active editor automatically: `never`, `first`, `always` (panel default `always`)
- `chat.pasteAsAttachmentThreshold` Pasted text longer than this many characters becomes an attachment (default 10000)

APPEARANCE & NOTIFICATIONS

<code>chat.fontSize</code> / <code>chat.fontFamily</code>	Text size and font in the chat (defaults 13 and the system default)
<code>chat.editor.fontSize</code>	Code block size in chat (default 14; 12 on macOS)
<code>chat.editor.wordWrap</code>	Wrap lines in code blocks: <code>off</code> (default) or <code>on</code>
<code>chat.math.enabled</code>	Render maths formulas in answers (default <code>true</code>)
<code>chat.tips.enabled</code>	Show tips in the Chat view (default <code>true</code>)
<code>chat.contextUsage.enabled</code>	Show how much of the model's context window is used (default <code>true</code>)
<code>chat.tools.todos.showWidget</code>	Show the agent's to-do list while it works (default <code>true</code>)
<code>chat.notifyWindowOnResponseReceived</code>	<code>windowNotFocused</code> (default), <code>off</code> , <code>always</code> : alert when an answer arrives
<code>chat.notifyWindowOnConfirmation</code>	Same options, for when the agent needs your approval
<code>chat.viewSessions.orientation</code>	Session list layout: <code>sideBySide</code> (default) or <code>stacked</code>
<code>chat.agentSessions.showExternal</code>	<code>recent</code> (default), <code>none</code> , <code>last24Hours</code> , <code>last7Days</code> , <code>last30Days</code> : sessions from other tools

INLINE CHAT & INLINE SUGGESTIONS

<code>Tab</code> / <code>Esc</code>	Accept / dismiss a grey inline suggestion
<code>Ctrl+Right</code>	Accept just the next word of a suggestion
<code>Alt+]</code> / <code>Alt+[</code>	Show the next / previous suggestion
<code>editor.inlineSuggest.enabled</code>	Show inline suggestions at all (default <code>true</code>)
<code>editor.inlineSuggest.showToolbar</code>	<code>always</code> , <code>onHover</code> , <code>never</code> : the small toolbar under a suggestion
<code>editor.inlineSuggest.minShowDelay</code>	Wait in ms before showing a suggestion (default 0)
<code>editor.inlineSuggest.suppressSuggestions</code>	Hide the normal suggestion list while an inline one shows
<code>editor.inlineSuggest.syntaxHighlightingEnabled</code>	Colour the suggested code like real code
<code>github.copilot.nextEditSuggestions.enabled</code>	Suggest the <i>next</i> edit as well as the current line (needs the Copilot features enabled)
<code>inlineChat.fixDiagnostics</code>	Offer Fix with chat on errors and warnings (default <code>true</code>)
<code>inlineChat.askInChat</code>	Offer Ask in Chat from the inline chat box (default <code>true</code>)
<code>inlineChat.affordance</code>	<code>off</code> (default) or <code>editor</code> : a hint button next to the cursor
<code>inlineChat.defaultModel</code>	Model for inline chat (default empty)

CLAUDE CODE EXTENSION: SETTINGS

<code>claudeCode.preferredLocation</code>	<code>panel</code> (default) or <code>sidebar</code> : where Claude opens
<code>claudeCode.useTerminal</code>	Use the terminal instead of the built-in chat panel (default <code>false</code>)
<code>claudeCode.initialPermissionMode</code>	<code>default</code> , <code>manual</code> , <code>acceptEdits</code> , <code>plan</code> , <code>bypassPermissions</code> for new conversations (default unset)
<code>claudeCode.allowDangerouslySkipPermissions</code>	Permit bypass-permissions mode; only for sandboxes with no internet
<code>claudeCode.autosave</code>	Save files before Claude reads or writes them (default <code>true</code>)
<code>claudeCode.attachOpenFile</code>	Add the open file to your messages (default <code>true</code>)
<code>claudeCode.useCtrlEnterToSend</code>	Ctrl+Enter sends and Enter makes a new line (default <code>false</code>)
<code>claudeCode.focusView</code>	Hide tool calls; show just your prompts and the replies (default <code>false</code>)
<code>claudeCode.respectGitIgnore</code>	Honour <code>.gitignore</code> in file searches (default <code>true</code>)
<code>claudeCode.lockEditorGroups</code>	Keep Claude's tabs in their own editor group (default <code>true</code>)
<code>claudeCode.continueAfterReload</code>	Resume an interrupted step after a window reload (default <code>true</code>)
<code>claudeCode.scrollToBottomOnSend</code>	Jump to the newest message when you send (default <code>true</code>)
<code>claudeCode.showMessageTimestamps</code>	Show when each message was sent (default <code>false</code>)
<code>claudeCode.archiveInactiveSessions</code>	Archive sessions after 0, 1, 2, 7 or 14 days idle (default 14)
<code>claudeCode.usePythonEnvironment</code>	Activate the workspace's Python environment when Claude runs (default <code>true</code>)
<code>claudeCode.environmentVariables</code>	Extra environment variables for the Claude process (default <code>[]</code>)
<code>claudeCode.hideOnboarding</code> / <code>disableLoginPrompt</code>	Hide the getting-started checklist / never prompt to log in (both default <code>false</code>)

CLAUDE CODE EXTENSION: SHORTCUTS & COMMANDS

<code>Ctrl+Escape</code>	Focus the Claude input; press again to return to the editor
<code>Ctrl+Shift+Escape</code>	Open Claude in a new editor tab
<code>Alt+K</code>	Insert an @-mention of the current file and selection in the editor
<code>Ctrl+Alt+F</code>	Toggle Focus view in a Claude panel
<code>Ctrl+Shift+T</code>	Reopen the Claude session tab you just closed (when enabled)
<code>Ctrl+N</code>	New conversation, when <code>claudeCode.enableNewConversationShortcut</code> is on
<code>Claude Code: Open in Side Bar / New Tab / New Window / Terminal</code>	Choose where to open the Claude Code interface (when enabled, Claude Code lives for the duration of the session)
<code>Claude Code: New Conversation</code>	Start a fresh conversation
<code>Claude Code: Create Worktree</code>	Make a Git worktree for a separate line of work
<code>Claude Code: Accept / Reject Proposed Changes</code>	Apply or discard the diff Claude proposes (also per change at the cursor)
<code>Claude Code: Insert @-Mention Reference</code>	Add the current file or selection as a reference
<code>Claude Code: Rename Session Tab / Mark Session as Unread</code>	Organise your sessions
<code>Claude Code: Install Plugin / Show Logs</code>	Add a plugin / open the extension's log

A REASONABLE STARTER SET (MY SUGGESTIONS, NOT DEFAULTS)

"chat.tools.global.autoApprove": false	Keep it off (the default); approve tools as they come
"chat.editing.autoAcceptDelay": 0	Review every edit yourself (the default)
"chat.notifyWindowOnResponseReceived": "windowNotFocused"	Hear about long-running answers while you work elsewhere (the default)
"editor.inlineSuggest.showToolbar": "onHover"	Toolbar only when you want it
"claudeCode.initialPermissionMode": "plan"	Start Claude conversations by planning rather than editing
"claudeCode.useCtrlEnterToSend": true	Enter for new lines when writing long prompts

AUTO-APPROVAL: CONVENIENT, AND THE RISKIEST SETTING HERE

With `chat.tools.global.autoApprove` on, the agent can run commands and edit files without asking. Prefer narrow rules instead: allow specific safe commands in `chat.tools.terminal.autoApprove` (for example read-only ones such as `git status`), keep file-edit approvals to particular patterns, and rely on checkpoints to undo. The same thinking applies to Claude Code's `bypassPermissions` mode, which is why the extension asks you to opt in separately.

CHAT SETTINGS YOU CANNOT FIND? THEY MOVE

Many chat options are marked experimental, are managed by an organisation policy, or are renamed between releases. Search the Settings editor for `chat.` or `@tag:experimental`, and use *Open Default Settings (JSON)* to see what your own version actually has.