 This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use.

[Learn more \(https://go.microsoft.com/fwlink/?linkid=845480\)](https://go.microsoft.com/fwlink/?linkid=845480)

Version 1.17 (/updates) is now available! Read about the new features and fixes from September.



TOPICS Key Bindings

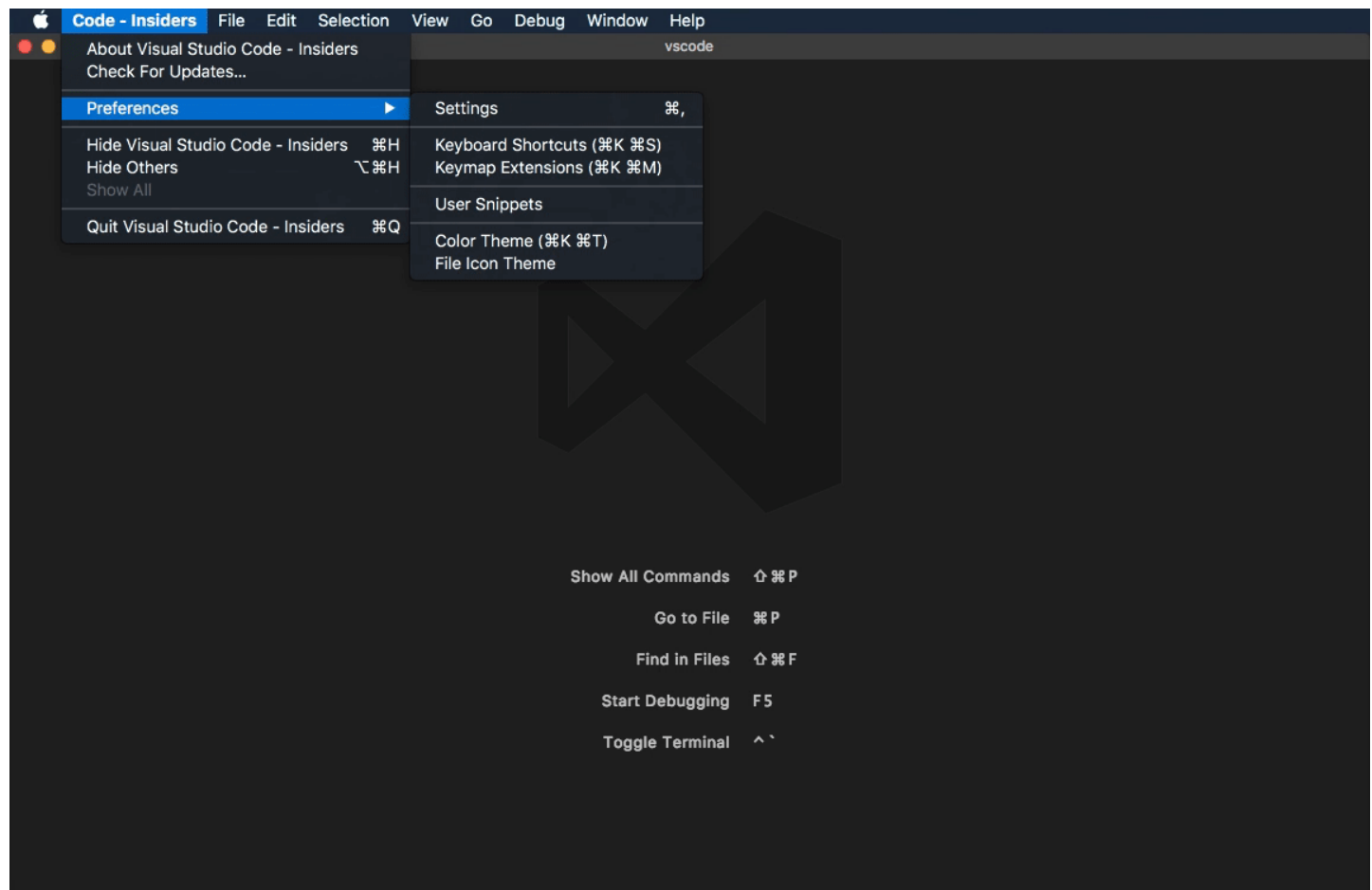
Key Bindings for Visual Studio Code <https://github.com/Microsoft/vscode-docs/blob/master/docs/getstarted/keybindings.md>

Visual Studio Code lets you perform most tasks directly from the keyboard. This page lists out the default bindings (keyboard shortcuts) and describes how you can update them.

Note: If you visit this page on a Mac, you will see the key bindings for the Mac. If you visit using Windows or Linux, you will see the keys for that platform. If you need the key binding for another platform, hover your mouse over the key you are interested in.

Keyboard Shortcuts Editor

Visual Studio Code provides a rich and easy keyboard shortcuts editing experience using **Keyboard Shortcuts** editor. It lists all available commands with and without keybindings and you can easily change / remove / reset their keybindings using the available actions. It also has a search box on the top that helps you in finding commands or keybindings. You can open this editor by going to the menu under **File > Preferences > Keyboard Shortcuts**. (**Code > Preferences > Keyboard Shortcuts** on Mac)




Most importantly, you can see keybindings according to your keyboard layout. For example, key binding `cmd+\\` in US keyboard layout will be shown as `ctrl+shift+alt+cmd+7` when layout is changed to German. The dialog to enter key binding will assign the correct and desired key binding as per your keyboard layout.


For doing more advanced keyboard shortcut customization, read [Advanced Customization \(/docs/getstarted/keybindings#_advanced-customization\)](/docs/getstarted/keybindings#_advanced-customization).

Keymap Extensions


Keyboard shortcuts are vital to productivity and changing keyboarding habits can be tough. To help with this, **File > Preferences > Keymap Extensions** shows you a list of popular keymap extensions. These extensions modify the VS Code shortcuts to match those of other editors so you don't need to learn new keyboard shortcuts. There is also a Keymaps category (<https://marketplace.visualstudio.com/search?target=VSCode&category=Keymaps&sortBy=Downloads>) of extensions in the Marketplace.




Vim
vscodevim 937.5K
Vim emulation for Visual Studio Code



Sublime Text Keymap
ms-vscode 239.8K
Popular Sublime Text bindings for VS Code.



Atom Keymap
ms-vscode 126.1K
Popular Atom bindings for Visual Studio Code



Visual Studio Keymap
ms-vscode 71.6K
Popular Visual Studio bindings for VS Code.

<https://marketplace.visualstudio.com/?itemName=vscodevim.vim>

<https://marketplace.visualstudio.com/?itemName=ms-vscode.sublime-keybindings>

<https://marketplace.visualstudio.com/?itemName=ms-vscode.atom-keybindings>

<https://marketplace.visualstudio.com/?itemName=ms-vscode.vs-keybindings>

Tip: Click on an extension tile above to read the description and reviews to decide which extension is best for you. See more in the Marketplace (<https://marketplace.visualstudio.com/vscode>).

Keyboard Shortcuts Reference

We also have a printable version of these keyboard shortcuts. **Help > Keyboard Shortcut Reference** displays a condensed PDF version suitable for printing as an easy reference.

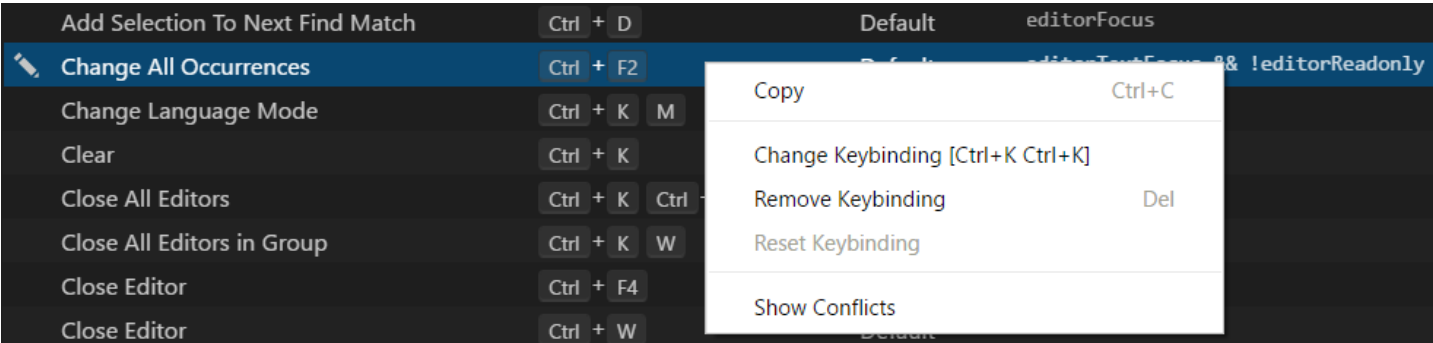
Below are links to the three platform-specific versions:

- Windows (<https://go.microsoft.com/fwlink/?linkid=832145>)
- macOS (<https://go.microsoft.com/fwlink/?linkid=832143>)
- Linux (<https://go.microsoft.com/fwlink/?linkid=832144>)

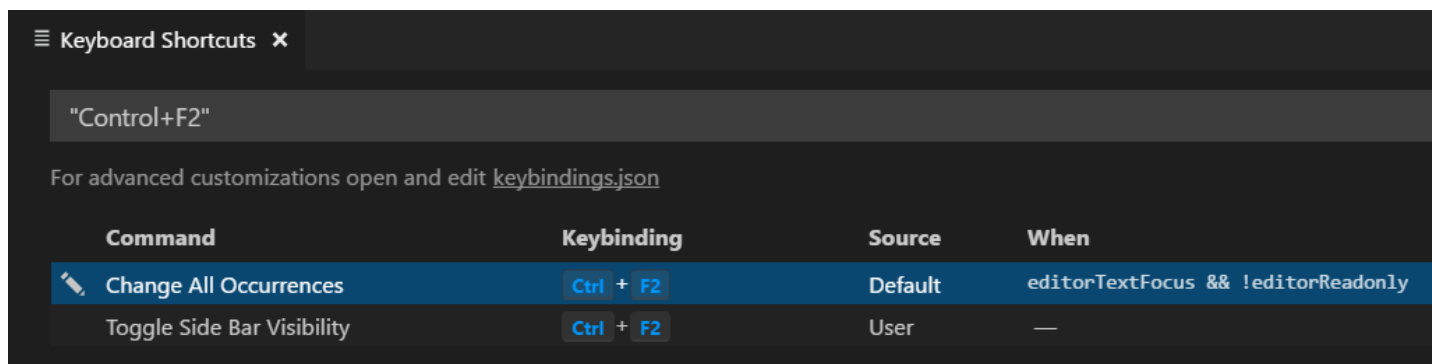
Detecting keybinding conflicts

If you have many extensions installed or you have customized (/docs/getstarted/keybindings#_advanced-customization) your keyboard shortcuts, you can sometimes have keybinding conflicts where the same keyboard shortcut is mapped to several commands. This can result in confusing behavior, especially if different keybindings are going in and out of scope as you move around the editor.

The **Keyboard Shortcuts** editor has a context menu command **Show Conflicts**, which will filter the keybindings based on a keyboard shortcut to display conflicts.



Pick a command with the keybinding you think is overloaded and you can see if multiple commands are defined, the source of the keybindings and when they are active.



Default Keyboard Shortcuts

Note: The following keys are rendered assuming a standard US keyboard layout. If you use a different keyboard layout, please read below (/docs/getstarted/keybindings#_keyboard-layouts). You can view the currently active keyboard shortcuts in VS Code in the **Command Palette** (View -> **Command Palette**) or in the **Keyboard Shortcuts** editor (File > Preferences > Keyboard Shortcuts).

Basic Editing

Key	Command	Command id
Ctrl+X	Cut line (empty selection)	editor.action.clipboardCutAction
Ctrl+C	Copy line (empty selection)	editor.action.clipboardCopyAction
Ctrl+Shift+K	Delete Line	editor.action.deleteLines
Ctrl+Enter	Insert Line Below	editor.action.insertLineAfter
Ctrl+Shift+Enter	Insert Line Above	editor.action.insertLineBefore
Alt+Down	Move Line Down	editor.action.moveLinesDownAction
Alt+Up	Move Line Up	editor.action.moveLinesUpAction
Shift+Alt+Down	Copy Line Down	editor.action.copyLinesDownAction
Shift+Alt+Up	Copy Line Up	editor.action.copyLinesUpAction
Ctrl+D	Add Selection To Next Find Match	editor.action.addSelectionToNextFindMatch
Ctrl+K Ctrl+D	Move Last Selection To Next Find Match	editor.action.moveSelectionToNextFindMatch
Ctrl+U	Undo last cursor operation	cursorUndo
Shift+Alt+I	Insert cursor at end of each line selected	editor.action.insertCursorAtEndOfEachLineSelected
Ctrl+Shift+L	Select all occurrences of current selection	editor.action.selectHighlights
Ctrl+F2	Select all occurrences of current word	editor.action.changeAll
Ctrl+I	Select current line	expandLineSelection
Ctrl+Alt+Down	Insert Cursor Below	editor.action.insertCursorBelow
Ctrl+Alt+Up	Insert Cursor Above	editor.action.insertCursorAbove
Ctrl+Shift+\	Jump to matching bracket	editor.action.jumpToBracket

Ctrl+]	Indent Line	editor.action.indentLines
Key	Command	Command id
Ctrl+[Outdent Line	editor.action.outdentLines
Home	Go to Beginning of Line	cursorHome
End	Go to End of Line	cursorEnd
Ctrl+End	Go to End of File	cursorBottom
Ctrl+Home	Go to Beginning of File	cursorTop
Ctrl+Down	Scroll Line Down	scrollLineDown
Ctrl+Up	Scroll Line Up	scrollLineUp
Alt+PageDown	Scroll Page Down	scrollPageDown
Alt+PageUp	Scroll Page Up	scrollPageUp
Ctrl+Shift+[Fold (collapse) region	editor.fold
Ctrl+Shift+]	Unfold (uncollapse) region	editor.unfold
Ctrl+K Ctrl+[Fold (collapse) all subregions	editor.foldRecursively
Ctrl+K Ctrl+]	Unfold (uncollapse) all subregions	editor.unfoldRecursively
Ctrl+K Ctrl+0	Fold (collapse) all regions	editor.foldAll
Ctrl+K Ctrl+J	Unfold (uncollapse) all regions	editor.unfoldAll
Ctrl+K Ctrl+C	Add Line Comment	editor.action.addCommentLine
Ctrl+K Ctrl+U	Remove Line Comment	editor.action.removeCommentLine
Ctrl+ /	Toggle Line Comment	editor.action.commentLine
Shift+Alt+A	Toggle Block Comment	editor.action.blockComment
Ctrl+F	Find	actions.find
Ctrl+H	Replace	editor.action.startFindReplaceAction
F3	Find Next	editor.action.nextMatchFindAction
Shift+F3	Find Previous	editor.action.previousMatchFindAction
Alt+Enter	Select All Occurrences of Find Match	editor.action.selectAllMatches
Alt+C	Toggle Find Case Sensitive	toggleFindCaseSensitive
Alt+R	Toggle Find Regex	toggleFindRegex
Alt+W	Toggle Find Whole Word	toggleFindWholeWord
Ctrl+M	Toggle Use of Tab Key for Setting Focus	editor.action.toggleTabFocusMode
unassigned	Toggle Render Whitespace	toggleRenderWhitespace
Alt+Z	Toggle Word Wrap	editor.action.toggleWordWrap

Key	Command	Command id
Ctrl+Space	Trigger Suggest	editor.action.triggerSuggest
Key	Command	Command id
Ctrl+Shift+Space	Trigger Parameter Hints	editor.action.triggerParameterHints
Shift+Alt+F	Format Document	editor.action.formatDocument
Ctrl+K Ctrl+F	Format Selection	editor.action.formatSelection
F12	Go to Definition	editor.action.goToDeclaration
Ctrl+K Ctrl+I	Show Hover	editor.action.showHover
Alt+F12	Peek Definition	editor.action.previewDeclaration
Ctrl+K F12	Open Definition to the Side	editor.action.openDeclarationToTheSide
Ctrl+.	Quick Fix	editor.action.quickFix
Shift+F12	Show References	editor.action.referenceSearch.trigger
F2	Rename Symbol	editor.action.rename
Ctrl+Shift+.	Replace with Next Value	editor.action.inPlaceReplace.down
Ctrl+Shift+,	Replace with Previous Value	editor.action.inPlaceReplace.up
Shift+Alt+Right	Expand AST Select	editor.action.smartSelect.grow
Shift+Alt+Left	Shrink AST Select	editor.action.smartSelect.shrink
Ctrl+K Ctrl+X	Trim Trailing Whitespace	editor.action.trimTrailingWhitespace
Ctrl+K M	Change Language Mode	workbench.action.editor.changeLanguageMode

Navigation

Key	Command	Command id
Ctrl+T	Show All Symbols	workbench.action.showAllSymbols
Ctrl+G	Go to Line...	workbench.action.gotoLine
Ctrl+P	Go to File..., Quick Open	workbench.action.quickOpen
Ctrl+Shift+O	Go to Symbol...	workbench.action.gotoSymbol
Ctrl+Shift+M	Show Problems	workbench.actions.view.problems
F8	Go to Next Error or Warning	editor.action.marker.next
Shift+F8	Go to Previous Error or Warning	editor.action.marker.prev
Ctrl+Shift+P	Show All Commands	workbench.action.showCommands
Ctrl+Shift+Tab	Navigate Editor Group History	workbench.action.openPreviousRecentlyUsedEditorInGroup
Alt+Left	Go Back	workbench.action.navigateBack
Alt+Right	Go Forward	workbench.action.navigateForward

Editor/Window Management

Key	Command	Command id
-----	---------	------------

Ctrl+Shift+N	New Window	workbench.action.newWindow
Key	Command	Command id
Ctrl+W	Close Window	workbench.action.closeWindow
Ctrl+F4	Close Editor	workbench.action.closeActiveEditor
Ctrl+K F	Close Folder	workbench.action.closeFolder
unassigned	Cycle Between Editor Groups	workbench.action.navigateEditorGroups
Ctrl+\	Split Editor	workbench.action.splitEditor
Ctrl+1	Focus into First Editor Group	workbench.action.focusFirstEditorGroup
Ctrl+2	Focus into Second Editor Group	workbench.action.focusSecondEditorGroup
Ctrl+3	Focus into Third Editor Group	workbench.action.focusThirdEditorGroup
Ctrl+K Ctrl+Left	Focus into Editor Group on the Left	workbench.action.focusPreviousGroup
Ctrl+K Ctrl+Right	Focus into Editor Group on the Right	workbench.action.focusNextGroup
Ctrl+Shift+PageUp	Move Editor Left	workbench.action.moveEditorLeftInGroup
Ctrl+Shift+PageDown	Move Editor Right	workbench.action.moveEditorRightInGroup
Ctrl+K Left	Move Active Editor Group Left	workbench.action.moveActiveEditorGroupLeft
Ctrl+K Right	Move Active Editor Group Right	workbench.action.moveActiveEditorGroupRight
Ctrl+Alt+Right	Move Editor into Next Group	workbench.action.moveEditorToNextGroup
Ctrl+Alt+Left	Move Editor into Previous Group	workbench.action.moveEditorToPreviousGroup

File Management

Key	Command	Command id
Ctrl+N	New File	workbench.action.files.newUntitledFile
Ctrl+O	Open File...	workbench.action.files.openFile
Ctrl+S	Save	workbench.action.files.save
Ctrl+K S	Save All	workbench.action.files.saveAll
Ctrl+Shift+S	Save As...	workbench.action.files.saveAs
Ctrl+F4	Close	workbench.action.closeActiveEditor
unassigned	Close Others	workbench.action.closeOtherEditors
Ctrl+K W	Close Group	workbench.action.closeEditorsInGroup
unassigned	Close Other Groups	workbench.action.closeEditorsInOtherGroups
unassigned	Close Group to Left	workbench.action.closeEditorsToTheLeft
unassigned	Close Group to Right	workbench.action.closeEditorsToTheRight
Ctrl+K Ctrl+W	Close All	workbench.action.closeAllEditors
Ctrl+Shift+T	Reopen Closed Editor	workbench.action.reopenClosedEditor

Ctrl+K Enter	Keep Open	workbench.action.keepEditor
Ctrl+Tab	Open Next	workbench.action.openNextRecentlyUsedEditorInGroup
Key	Command	Command id
Ctrl+Shift+Tab	Open Previous	workbench.action.openPreviousRecentlyUsedEditorInGroup
Ctrl+K P	Copy Path of Active File	workbench.action.files.copyPathOfActiveFile
Ctrl+K R	Reveal Active File in Windows	workbench.action.files.revealActiveFileInWindows
Ctrl+K O	Show Opened File in New Window	workbench.action.files.showOpenedFileInNewWindow
unassigned	Compare Opened File With	workbench.files.action.compareFileWith

Display

Key	Command	Command id
F11	Toggle Full Screen	workbench.action.toggleFullScreen
Ctrl+K Z	Toggle Zen Mode	workbench.action.toggleZenMode
Escape Escape	Leave Zen Mode	workbench.action.exitZenMode
Ctrl+=	Zoom in	workbench.action.zoomIn
Ctrl+-	Zoom out	workbench.action.zoomOut
Ctrl+Numpad0	Reset Zoom	workbench.action.zoomReset
Ctrl+B	Toggle Sidebar Visibility	workbench.action.toggleSidebarVisibility
Ctrl+Shift+E	Show Explorer / Toggle Focus	workbench.view.explorer
Ctrl+Shift+D	Show Debug	workbench.view.debug
Ctrl+Shift+G	Show Source Control	workbench.view.scm
Ctrl+Shift+X	Show Extensions	workbench.view.extensions
Ctrl+Shift+U	Show Output	workbench.action.output.toggleOutput
Ctrl+Q	Quick Open View	workbench.action.quickOpenView
Ctrl+Shift+F	Show Search	workbench.view.search
Ctrl+Shift+H	Replace in Files	workbench.action.replaceInFiles
Ctrl+Shift+J	Toggle Search Details	workbench.action.search.toggleQueryDetails
Ctrl+Shift+C	Open New Command Prompt	workbench.action.terminal.openNativeConsole
Ctrl+Shift+V	Toggle Markdown Preview	markdown.showPreview
Ctrl+K V	Open Preview to the Side	markdown.showPreviewToSide
Ctrl+`	Toggle Integrated Terminal	workbench.action.terminal.toggleTerminal

Preferences

Key	Command	Command id
Ctrl+,	Open User Settings	workbench.action.openGlobalSettings
unassigned	Open Workspace Settings	workbench.action.openWorkspaceSettings

Ctrl+K Ctrl+S	Open Keyboard Shortcuts	workbench.action.openGlobalKeybindings
---------------	-------------------------	--

Key	Command	Command id
unassigned	Open User Snippets	workbench.action.openSnippets
Ctrl+K Ctrl+T	Select Color Theme	workbench.action.selectTheme
unassigned	Configure Display Language	workbench.action.configureLocale

Debug

Key	Command	Command id
F9	Toggle Breakpoint	editor.debug.action.toggleBreakpoint
F5	Start	workbench.action.debug.start
F5	Continue	workbench.action.debug.continue
Ctrl+F5	Start (without debugging)	workbench.action.debug.run
F6	Pause	workbench.action.debug.pause
F11	Step Into	workbench.action.debug.stepInto
Shift+F11	Step Out	workbench.action.debug.stepOut
F10	Step Over	workbench.action.debug.stepOver
Shift+F5	Stop	workbench.action.debug.stop
Ctrl+K Ctrl+I	Show Hover	editor.debug.action.showDebugHover

Tasks

Key	Command	Command id
Ctrl+Shift+B	Run Build Task	workbench.action.tasks.build
unassigned	Run Test Task	workbench.action.tasks.test

Extensions

Key	Command	Command id
unassigned	Install Extension	workbench.extensions.action.installExtension
unassigned	Show Installed Extensions	workbench.extensions.action.showInstalledExtensions
unassigned	Show Outdated Extensions	workbench.extensions.action.listOutdatedExtensions
unassigned	Show Recommended Extensions	workbench.extensions.action.showRecommendedExtensions
unassigned	Show Popular Extensions	workbench.extensions.action.showPopularExtensions
unassigned	Update All Extensions	workbench.extensions.action.updateAllExtensions

Advanced customization

All keyboard shortcuts in VS Code can be customized via the `keybindings.json` file.

- To configure keyboard shortcuts the way you want, open **Keyboard Shortcuts** editor and click on the link `keybindings.json`.
- This will open the **Default Keyboard Shortcuts** on the left and your `keybindings.json` file where you can overwrite the default bindings on the

right.

- The list above isn't exhaustive. More commands may be listed under "Here are other available commands" in **Default Keyboard Shortcuts**.

Keyboard Rules

The keyboard shortcuts dispatching is done by analyzing a list of rules that are expressed in JSON. Here are some examples:

```
// Keybindings that are active when the focus is in the editor
{ "key": "home",          "command": "cursorHome",          "when": "editorTextFocus" },
{ "key": "shift+home",    "command": "cursorHomeSelect",    "when": "editorTextFocus" },

// Keybindings that are complementary
{ "key": "f5",            "command": "workbench.action.debug.continue", "when": "inDebugMode" },
{ "key": "f5",            "command": "workbench.action.debug.start",    "when": "!inDebugMode" },

// Global keybindings
{ "key": "ctrl+f",        "command": "actions.find" },
{ "key": "alt+left",      "command": "workbench.action.navigateBack" },
{ "key": "alt+right",     "command": "workbench.action.navigateForward" },

// Global keybindings using chords (two separate keypress actions)
{ "key": "ctrl+k enter",  "command": "workbench.action.keepEditor" },
{ "key": "ctrl+k ctrl+w", "command": "workbench.action.closeAllEditors" },
```

Each rule consists of:

- a **key** that describes the pressed keys.
- a **command** containing the identifier of the command to execute.
- an **optional** **when** clause containing a boolean expression that will be evaluated depending on the current **context**.

Chords (two separate keypress actions) are described by separating the two keypresses with a space. E.g.: `ctrl+k ctrl+c`.

When a key is pressed:

- the rules are evaluated from **bottom to top**.
- the first rule that matches, both the **key** and in terms of **when**, is accepted.
- no more rules are processed.
- if a rule is found and has a **command** set, the **command** is executed.

The additional `keybindings.json` rules are appended at runtime to the bottom of the default rules, thus allowing them to overwrite the default rules. The `keybindings.json` file is watched by VS Code so editing it while VS Code is running will update the rules at runtime.

Accepted keys

The **key** is made up of modifiers and the key itself.

The following modifiers are accepted:

Platform	Modifiers
Mac	ctrl+, shift+, alt+, cmd+
Windows	ctrl+, shift+, alt+, win+
Linux	ctrl+, shift+, alt+, meta+

The following keys are accepted:

- f1-f19, a-z, 0-9
- ` , - , = , [,] , \ , ; , ' , , , . , /
- left, up, right, down, pageup, pagedown, end, home
- tab, enter, escape, space, backspace, delete
- pausebreak, capslock, insert
- numpad0-numpad9, numpad_multiply, numpad_add, nupad_separator
- numpad_subtract, numpad_decimal, numpad_divide

Command arguments

You can invoke a command with arguments. This is useful if you often perform the same operation on a specific file or folder. You can add a custom keyboard shortcut to do exactly what you want.

The following is an example overriding the `Enter` key to print some text:

```
{ "key": "enter", "command": "type",
  "args": { "text": "Hello World" },
  "when": "editorTextFocus" }
```

The `type` command will receive `{ "text": "Hello World" }` as its first argument and add "Hello World" to the file instead of producing the default command.

'when' clause contexts

VS Code gives you fine control over when your key bindings are enabled through the optional `when` clause. If your key binding doesn't have a `when` clause, the key binding is globally available at all times.

Below are the some of the possible `when` clause contexts which evaluate to Boolean `true/false`:

Context name	True when
Editor contexts	
<code>editorFocus</code>	An editor has focus, either the text or a widget.
<code>editorTextFocus</code>	The text in an editor has focus (cursor is blinking).
<code>editorHasSelection</code>	Text is selected in the editor.
<code>editorHasMultipleSelections</code>	Multiple regions of text are selected (multiple cursors).
<code>editorReadOnly</code>	The editor is read only.
<code>editorLangId</code>	True when the editor's associated language <code>Id</code> (<code>/docs/languages/identifiers</code>) matches. Example: <code>"editorLangId == typescript"</code> .
<code>textCompareEditorVisible</code>	Diff (compare) view is visible.
Mode contexts	
<code>inDebugMode</code>	A debug session is running.
<code>inSnippetMode</code>	The editor is in snippet mode.
<code>inQuickOpen</code>	The Quick Open drop-down has focus.
Explorer contexts	
<code>explorerViewletVisible</code>	True if Explorer view is visible.
<code>explorerViewletFocus</code>	True if Explorer view has keyboard focus.
<code>filesExplorerFocus</code>	True if File Explorer section has keyboard focus.
<code>openEditorsFocus</code>	True if OPEN EDITORS section has keyboard focus.
<code>explorerResourceIsFolder</code>	True if a folder is selected in the Explorer.
Editor widget contexts	
<code>findWidgetVisible</code>	Editor Find widget is visible.
<code>suggestWidgetVisible</code>	Suggestion widget (IntelliSense) is visible.

suggestWidgetMultipleSuggestions	Multiple suggestions are displayed.
renameInputVisible	Rename input text box is visible.
Context name	True when
referenceSearchVisible	Find All References peek window is open.
inReferenceSearchEditor	The Find All References peek window editor has focus.
config.editor.stablePeek	Keep peek editors open (controlled by <code>editor.stablePeek</code> setting).
quickFixWidgetVisible	Quick Fix widget is visible.
parameterHintsVisible	Parameter hints are visible (controlled by <code>editor.parameterHints</code> setting).
parameterHintsMultipleSignatures	Multiple parameter hints are displayed.
Integrated terminal contexts	
terminalFocus	An integrated terminal has focus.
Global UI contexts	
resourceLangId	True when the Explorer or editor title language Id (<code>/docs/languages/identifiers</code>) matches. Example: <code>"resourceLangId == markdown"</code>
resourceFilename	True when the Explorer or editor filename matches. Example: <code>"resourceFilename == gulpfile.js"</code>
globalMessageVisible	Message box is visible at the top of VS Code.
searchViewletVisible	Search view is open.
replaceActive	Search view Replace text box is open.
Configuration settings contexts	
config.editor.minimap.enabled	True when the setting <code>editor.minimap.enabled</code> is <code>true</code> .

Note: You can use any user or workspace setting that evaluates to a boolean here with the prefix `"config."`.

The list above isn't exhaustive and you may see some `when` contexts for specific VS Code UI in the **Default Keyboard Shortcuts**.

Removing a specific key binding rule

You can write a key binding rule that targets the removal of a specific default key binding. With the `keybindings.json`, it was always possible to redefine all the key bindings of VS Code, but it can be very difficult to make a small tweak, especially around overloaded keys, such as `Tab` or `Escape`. To remove a specific key binding, add a `-` to the `command` and the rule will be a removal rule.

Here is an example:

```
// In Default Keyboard Shortcuts
...
{ "key": "tab", "command": "tab", "when": ... },
{ "key": "tab", "command": "jumpToNextSnippetPlaceholder", "when": ... },
{ "key": "tab", "command": "acceptSelectedSuggestion", "when": ... },
...

// To remove the second rule, for example, add in keybindings.json:
{ "key": "tab", "command": "-jumpToNextSnippetPlaceholder" }
```

Keyboard layouts

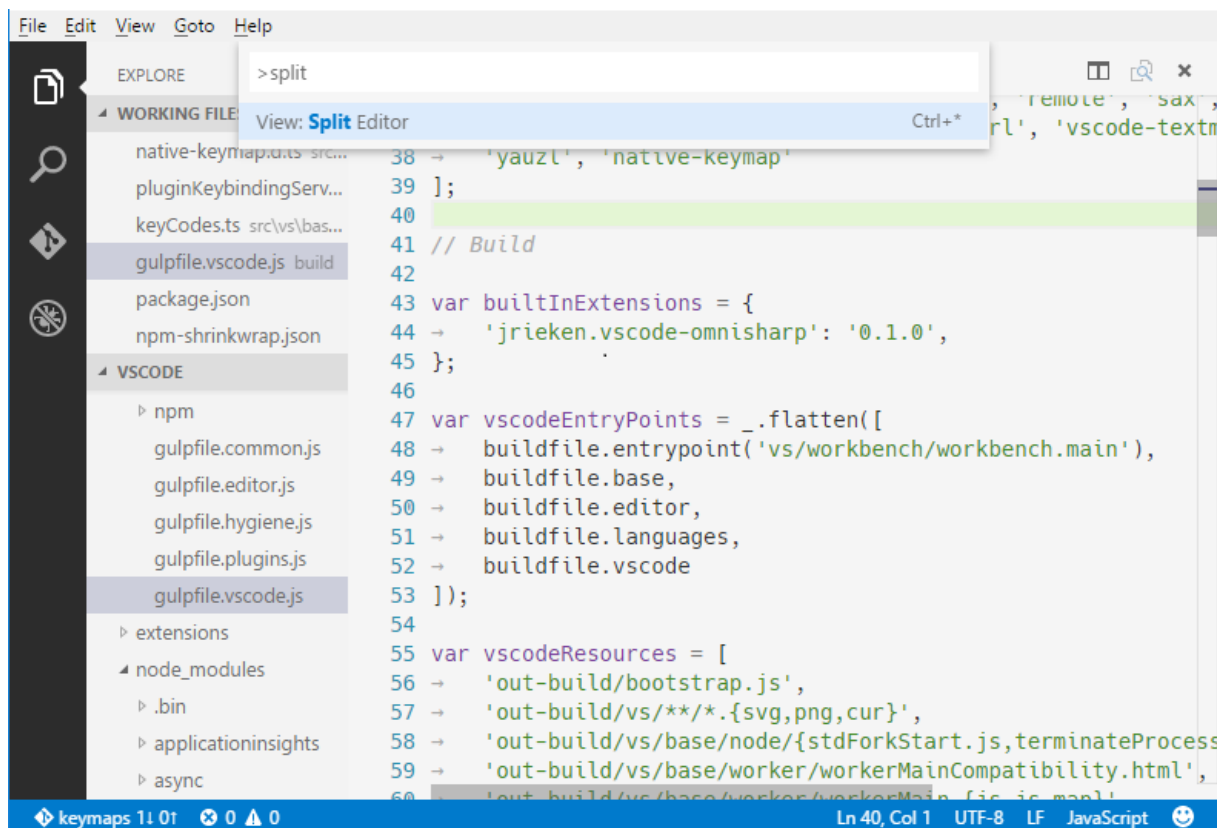
Note: This section relates only to key bindings, not to typing in the editor.

The keys above are string representations for virtual keys and do not necessarily relate to the produced character when they are pressed. More precisely:

- Reference: Virtual-Key Codes (Windows) (<https://msdn.microsoft.com/en-us/library/windows/desktop/dd375731>)
- tab for VK_TAB (0x09)
- ; for VK_OEM_1 (0xBA)
- = for VK_OEM_PLUS (0xBB)
- , for VK_OEM_COMMA (0xBC)
- - for VK_OEM_MINUS (0xBD)
- . for VK_OEM_PERIOD (0xBE)
- / for VK_OEM_2 (0xBF)
- ` for VK_OEM_3 (0xC0)
- [for VK_OEM_4 (0xDB)
- \ for VK_OEM_5 (0xDC)
-] for VK_OEM_6 (0xDD)
- ' for VK_OEM_7 (0xDE)
- etc.

Different keyboard layouts usually reposition the above virtual keys or change the characters produced when they are pressed. When using a different keyboard layout than the standard US, Visual Studio Code does the following:

All the key bindings are rendered in the UI using the current system's keyboard layout. For example, `Split Editor` when using a French (France) keyboard layout is now rendered as `Ctrl+*`:



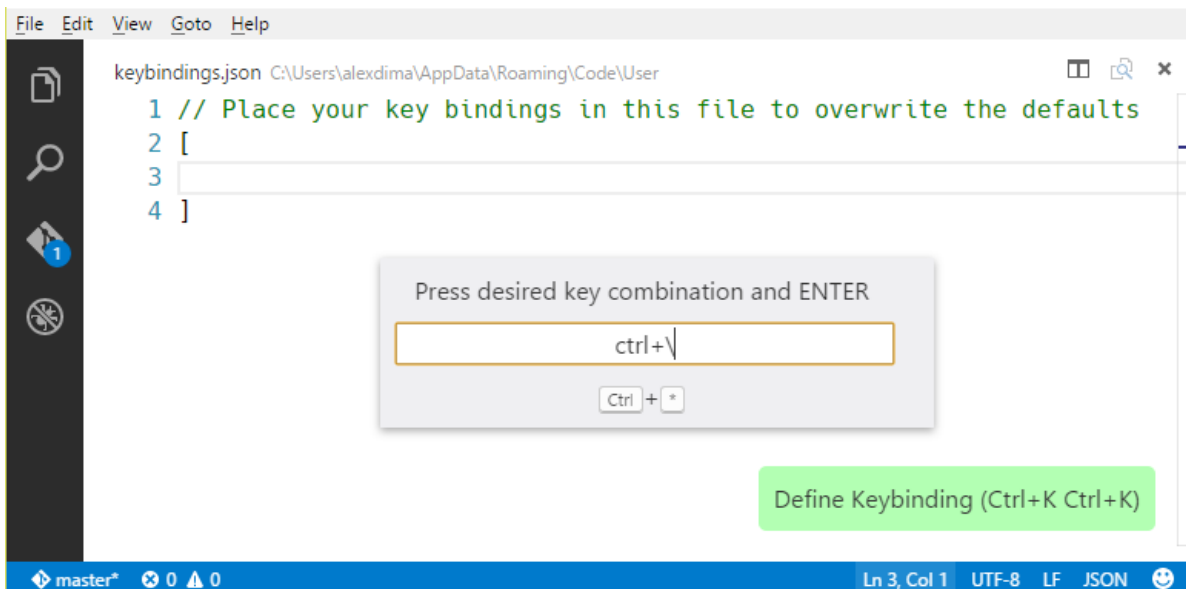
When editing `keybindings.json`, VS Code highlights misleading key bindings - those that are represented in the file with the character produced under the standard US keyboard layout, but which need pressing keys with different labels under the current system's keyboard layout. For example, here is how the **Default Keyboard Shortcuts** rules look like when using a French (France) keyboard layout:

```

300 { "key": "ctrl+shift+j",      "command": "workbench.action.search.toggleQueryDetails",
301 ..... "when": "searchViewletVisible" },
302 { "key": "ctrl+t",           "command": "workbench.action.showAllSymbols" },
303 { "key": "f1",               "command": "workbench.action.showCommands" },
304 { "key": "c",                 "command": "workbench.action.showCommands" },
305 { "key": "C",                 "command": "workbench.action.showErrorsWarnings" },
306 { "key": "ctrl+\\",           "command": "workbench.action.splitEditor" },
307 { "key": "ctrl+shift+b",      "command": "workbench.action.tasks.build" },
308 { "key": "ctrl+shift+t",      "command": "workbench.action.tasks.test" },
309 { "key": "ctrl+shift+c",      "command": "workbench.action.terminal.openNativeConsole"
310 { "key": "f11",              "command": "workbench.action.toggleFullScreen" },
311 { "key": "ctrl+b",           "command": "workbench.action.toggleSidebarVisibility" },
312 { "key": "ctrl+=",           "command": "workbench.action.zoomIn" },
313 { "key": "ctrl+-",           "command": "workbench.action.zoomOut" },
314 { "key": "ctrl+k enter",     "command": "workbench.files.action.addToWorkingFiles" },
315 { "key": "ctrl+k ctrl+w",    "command": "workbench.files.action.closeAllFiles" },

```

There is also a widget that helps input the key binding rule when editing `keybindings.json`. To launch the **Define Keybinding** widget, press `Ctrl+K` `Ctrl+K`. The widget listens for key presses and renders the serialized JSON representation in the text box and below it, the keys that VS Code has detected under your current keyboard layout. Once you've typed the key combination you want, you can press `Enter` and a rule snippet will be inserted.



Note: Visual Studio Code detects your current keyboard layout on start-up and then caches this information. For a good experience, we recommend restarting VS Code if you change your keyboard layout.

Next Steps

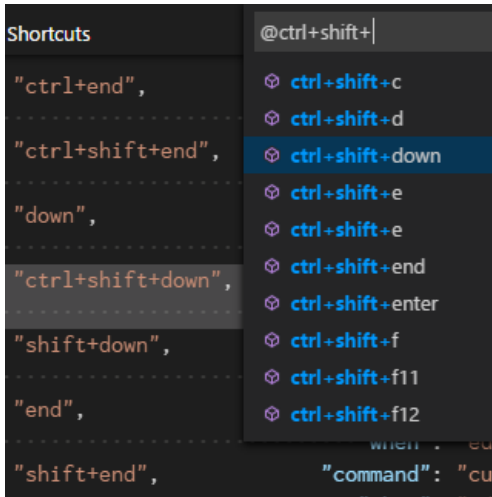
Now that you know about our Key binding support, what's next...

- Language Support (/docs/languages/overview) - Our Good, Better, Best language grid to see what you can expect
- Debugging (/docs/editor/debugging) - This is where VS Code really shines
- Node.js (/docs/nodejs/nodejs-tutorial) - End to end Node.js scenario with a sample app

Common Questions

Q: How to find out what command is bound to a specific key?

A: In the **Default Keyboard Shortcuts**, open **Quick Outline** by pressing `Ctrl+Shift+O`



Q: How to add a key binding to an action? For example add Ctrl+D to Delete Lines

A: Find a rule that triggers the action in the **Default Keyboard Shortcuts** and write a modified version of it in your `keybindings.json` file:

```
// Original, in Default Keyboard Shortcuts
{ "key": "ctrl+shift+k",      "command": "editor.action.deleteLines",
  "when": "editorTextFocus" },
// Modified, in User/keybindings.json, Ctrl+D now will also trigger this action
{ "key": "ctrl+d",           "command": "editor.action.deleteLines",
  "when": "editorTextFocus" },
```

Q: How can I add a key binding for only certain file types?

A: Use the `editorLangId` context key in your `when` clause:

```
{ "key": "shift+alt+a",      "command": "editor.action.blockComment",
  "when": "editorTextFocus && editorLangId == csharp" },
```

Q: I have modified my key bindings in `keybindings.json`, why don't they work?

A: The most common problem is a syntax error in the file. Otherwise, try removing the `when` clause or picking a different `key`. Unfortunately, at this point, it is a trial and error process.

Was this documentation helpful?

Yes No

Last updated on 10/5/2017

Hello from Seattle. Follow @code ()

Star 36,548

Support (https://support.microsoft.com/en-us/getsupport?wf=0&tenant=ClassicCommercial&oaspsworkflow=start_1.0.0.0&locale=en-us&supportregion=en-us&pesid=16064&ccsid=636196895839595242)

Privacy (<https://www.microsoft.com/privacystatement/en-us/core/default.aspx>)

Terms of Use (<https://www.microsoft.com/en-us/legal/intellectualproperty/copyright/default.aspx>) License (/License)

 Microsoft (<https://www.microsoft.com>)
© 2017 Microsoft