

keyboard driven

```
"key": "alt+b",  
"command": "workbench.action.toggleActivityBarVisibility",  
"when": "editorFocus"
```

VSCODE

Keyboard Driven VSCode

#vscode



Waylon Walker

19 Nov 2019 Originally published at waylonwalker.com • Updated on 28 Oct 2020 • 4 min read

Throw that mouse Away its time to setup some keyboard shortcuts.

Quick side note. This is a cross post from my personal blog at <https://waylonwalker.com/blog/keyboard-driven-vscode/>.

These sortcuts were the baseline for switching from tmux/vim to vscode. Most folks posts I was able to find gave great tips on replacing vim, but very few have focused on the hackability of tmux. tmux allows me to rapidly fire up a workspace, create new windows and splits. Then When I switch tasks I can leave that workspace open and and jump right back in later exactly where I left off. There is nothing quite like it. The shortcuts listed here make the transition a bit better. The worst thing I found when using vscode at first was no way to switch between the terminal and editor without the mouse. This first set of keybindings solve that issue.

The worst thing I found when using vscode at first was no way to switch between the terminal and editor without the mouse.



48



18



67



Navigation:

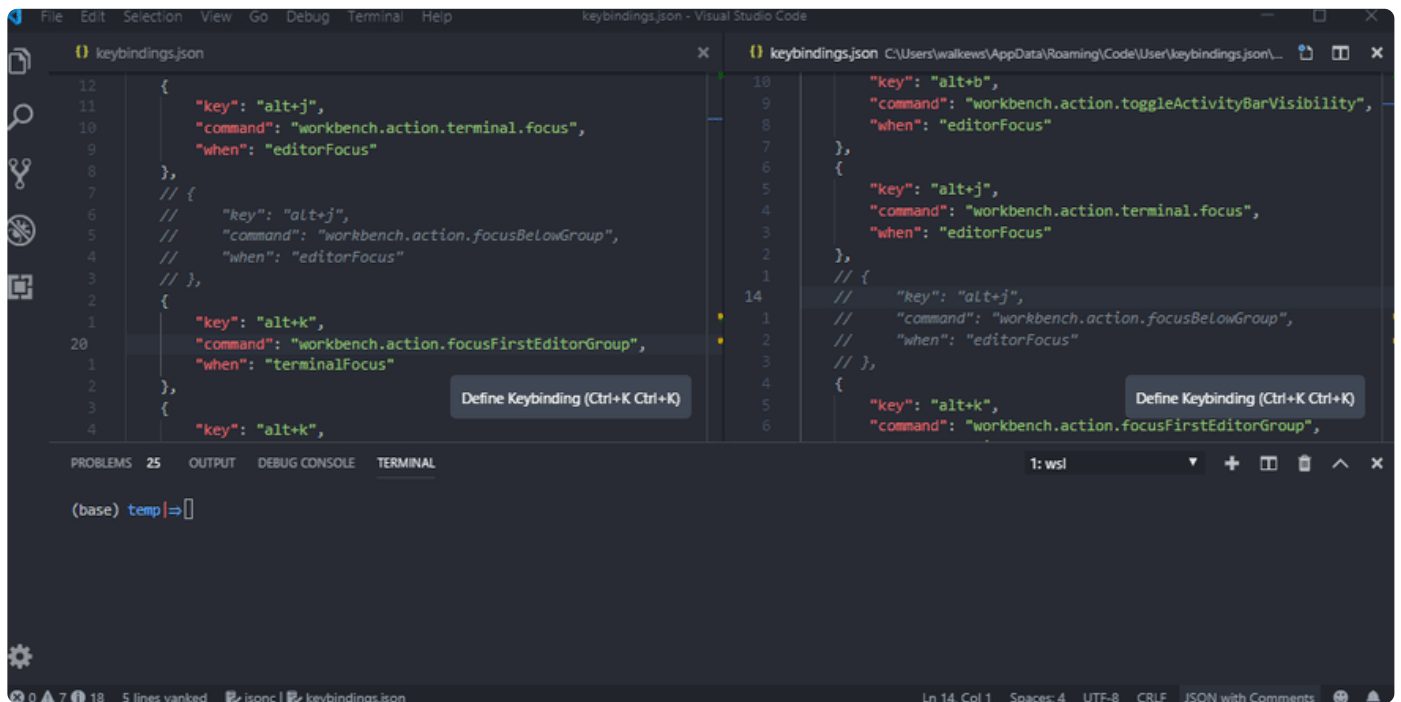
← jump to **left** split `Alt + h`

↓ jump to **terminal** from editor `Alt + j`

↑ jump to **editor** from terminal `Alt + k`

→ jump to **right** split `Alt + l`

This is by far the most useful set of keybindings that I use in vscode and is directly replicated from my tmux configuration. It allows me to quickly jump up, down, left, right. Do note that if you use vertical splits it does not work as well as tmux 😞.



```
[
  {
    "key": "alt+j",
    "command": "workbench.action.terminal.focus",
    "when": "editorFocus"
  },
  {
    "key": "alt+k",
    "command": "workbench.action.focusFirstEditorGroup",
    "when": "terminalFocus"
  },
  {
    "key": "alt+k",
    "command": "workbench.action.focusAboveGroup",
    "when": "editorFocus"
  }
]
```



48



18



67



```

    },
    {
      "command": "-toggleFindInSelection",
      "when": "editorFocus"
    },
    {
      "key": "alt+l",
      "command": "workbench.action.focusNextGroup",
      "when": "editorFocus"
    },
    {
      "key": "alt+h",
      "command": "workbench.action.focusPreviousGroup",
      "when": "editorFocus"
    },
    {
      "key": "alt+l",
      "command": "workbench.action.terminal.focusNextPane",
      "when": "terminalFocus"
    },
    {
      "key": "alt+h",
      "command": "workbench.action.terminal.focusPreviousPane",
      "when": "terminalFocus"
    }
  ],
]

```

toggle bloat

Since closing the sidebar is assigned to `ctrl+b` I thought that it made most sense to simulate the activity bar with `<kbd>Alt</kbd>+<kbd>b</kbd>`. There are many times when I just want to get as much out of the way as possible and this little bit does help.



48

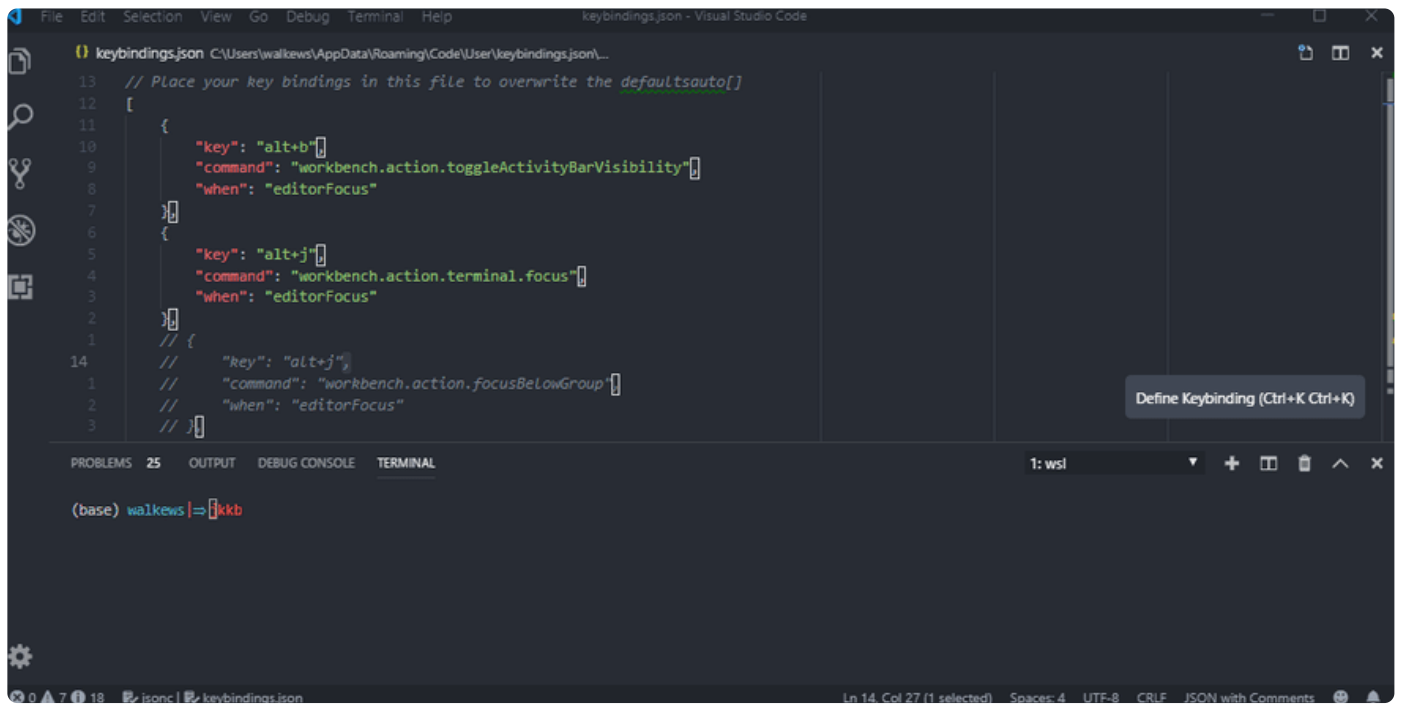


18



67





```
keybindings.json C:\Users\walkews\AppData\Roaming\Code\User\keybindings.json...
13 // Place your key bindings in this file to overwrite the defaultsauto[]
12 [
11   {
10     "key": "alt+b",
9     "command": "workbench.action.toggleActivityBarVisibility",
8     "when": "editorFocus"
7   },
6   {
5     "key": "alt+j",
4     "command": "workbench.action.terminal.focus",
3     "when": "editorFocus"
2   },
1   // {
14 //   "key": "alt+j",
1 //   "command": "workbench.action.focusBelowGroup",
2 //   "when": "editorFocus"
3 // }
]

PROBLEMS 25 OUTPUT DEBUG CONSOLE TERMINAL 1: wsl
(base) walkews | => kkb
```

```
[
  {
    "key": "alt+b",
    "command": "workbench.action.toggleActivityBarVisibility",
    "when": "editorFocus"
  },
]
```

Alt+[svx]

Split it up

👉 Split horizontally **Alt+s**

👉 Vertically **Alt+v**

💥 Close **Alt+x**

This is another one replicated from tmux for quickly creating horizontal (s) and vertical (v) splits. Once I am done with them I can close them with **Alt+x**.



48



18



67



```
keybindings.json C:\Users\walkew\AppData\Roaming\Code\User\keybindings.json alt+l
8   "key": "alt+l",
7   "command": "-toggleFindInSelection",
6   "when": "editorFocus"
5 },
4 },
3   "key": "alt+l",
2   "command": "workbench.action.focusNextGroup",
1   "when": "editorFocus"
42 },
1 {
2   "key": "alt+h",
3   "command": "workbench.action.focusPreviousGroup",
4   "when": "editorFocus"
5 },
6 {
7   "key": "alt+n",
8   "command": "workbench.action.terminal.focusNext",
9   "when": "terminalFocus"

Define Keybinding (Ctrl+K Ctrl+K)

PROBLEMS 25 OUTPUT DEBUG CONSOLE TERMINAL 1: wsl
(base) temp|=|
```

```
[
  {
    "key": "alt+s",
    "command": "workbench.action.terminal.split",
    "when": "terminalFocus"
  },
  {
    "key": "alt+s",
    "command": "workbench.action.splitEditor",
    "when": "editorFocus"
  },
  {
    "key": "alt+v",
    "command": "workbench.action.splitEditorOrthogonal",
    "when": "editorFocus"
  },
  {
    "key": "alt+x",
    "command": "workbench.action.terminal.kill",
    "when": "terminalFocus"
  },
  {
    "key": "alt+x",
    "command": "workbench.action.closeActiveEditor",
    "when": "editorFocus"
  }
]
```



48



18



67



Alt+[cnp]

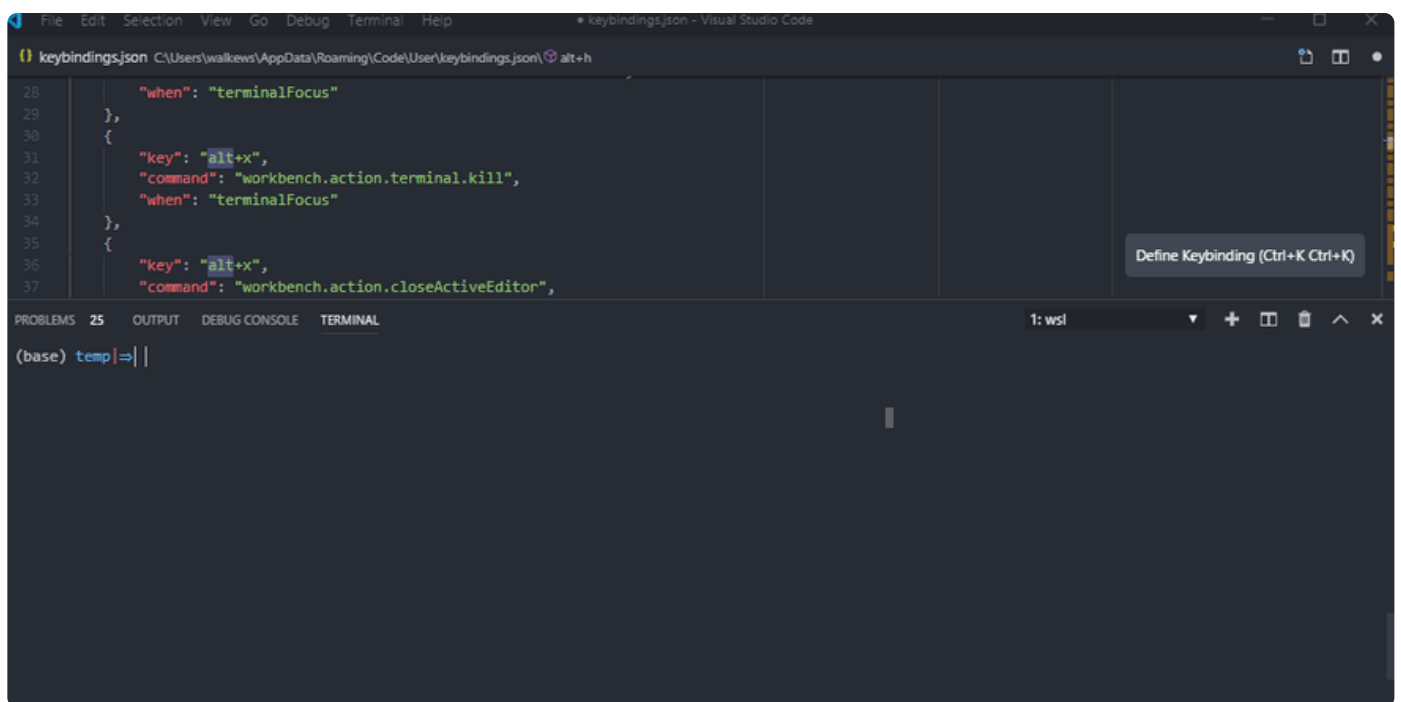
👉 **Create** new workspace **Alt + c**

➡ **jump to next** workspace **Alt + n**

⬅ **jump to previous** workspace **Alt + p**

Sometimes the terminal window gets a bit cramped inside of splits and you need to use different panes. I replicated the cnp pattern from tmux here as well. I can create (c) new panes, then go to the next (n), or previous (p) without leaving the comfort of my keyboard.

I am often using this one when I have a process running that I watch like gatsby, and I need to quickly pop into a new pane to run a git command and back in to gatsby before jumping up to my editor.



```
[
  {
    "key": "alt+c",
    "command": "workbench.action.terminal.new",
    "when": "terminalFocus"
  },
  {
    "key": "alt+n",
    "command": "workbench.action.terminal.focusNext",
    "when": "terminalFocus"
  }
]
```



48



18



67



```

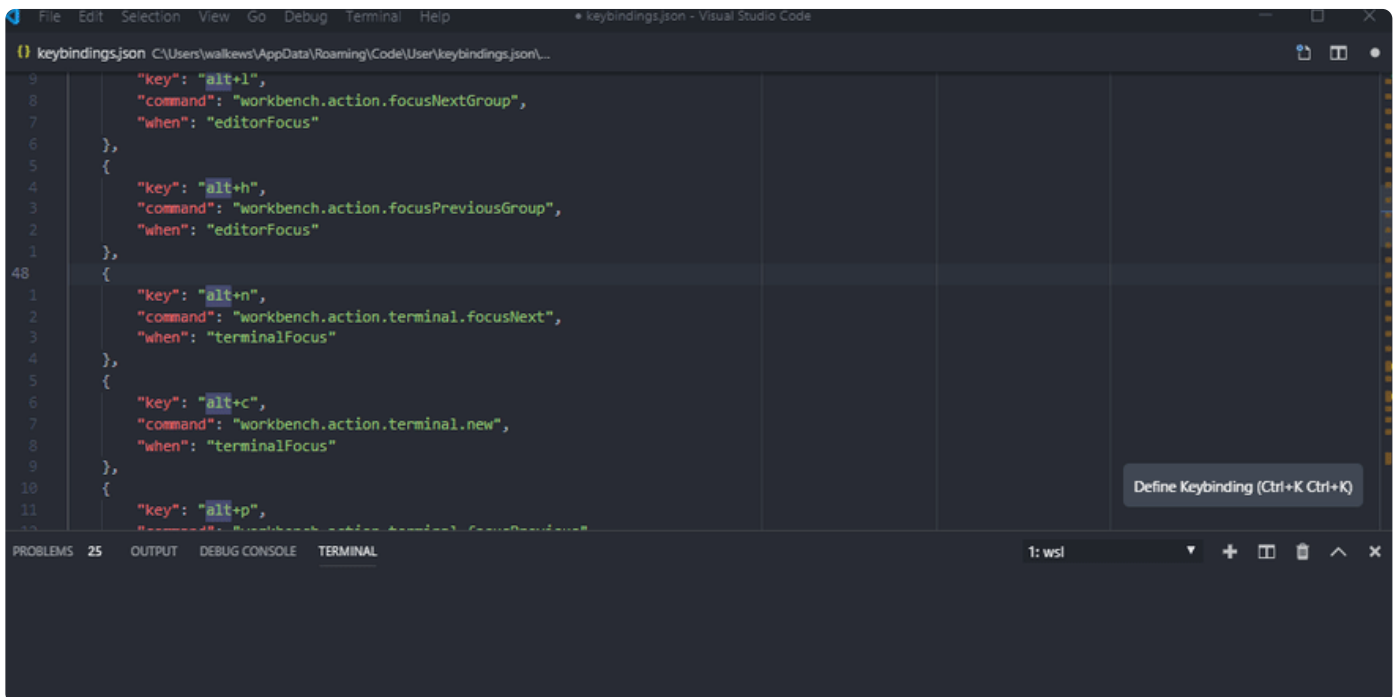
    "key": "alt+p",
    "command": "workbench.action.terminal.focusPrevious",
    "when": "terminalFocus"
  },
]

```

Alt + z

 Zoom into terminal Alt + z

This last one doesn't work as well as I would like but it sure does get the job done. It Zooms (z) into the terminal from anywhere in vscode. I can be in my editor and zoom out of the terminal to make more room, or zoom in to see what happened from my last edit.



```

[
  {
    "key": "alt+z",
    "command": "workbench.action.toggleMaximizedPanel",
  },
]

```

Bonus

Alt + r



48



18



67



I recently found this one, and find it quite useful to quickly do things like revert selected range, or extract variable without leaving the keyboard.

```
[
  {
    "key": "alt+r",
    "command": "editor.action.showContextMenu",
  },
]
```

Discussion

[Subscribe](#)

Add to the discussion



Gourav • Nov 22 '19



Very nice keyboard hacks. Here's how I use it:

1) Single shortcut (Ctrl+j) to open/close terminal + also set the focus in terminal. It works out of the box. :D

2) Single shortcut (ctrl+l) to toggle focus between all opened editors, it also works even if focus is on terminal:

```
{
  "key": "ctrl+l",
  "command": "workbench.action.focusNextGroup",
  "when": "editorFocus"
},
{
  "key": "ctrl+l",
  "command": "workbench.action.focusActiveEditorGroup",
  "when": "!editorFocus"
},
```

3) open/close sidebar (ctrl+b).

Apart from these my best keyboard hack is modified keyboard win key as mouse left click using autohotkey software and I never touched the mouse again in my life.



48



18



67





Waylon Walker 🌟 • Nov 23 '19



That is awesome! I use the win key for too many shortcuts, I would need to find a different hotkey for that. How do you move the mouse, autohotkey as well? any api I have tried to use has been unsuccessful for me.

I have a bunch of stuff in an always running autohotkey, the only one that I use often is "_" mapped to Shift+space. I literally use this many times per minute.

apart from that I have a vortex pok3r keyboard, which I have done my best to map some basic vim keys to. Now I have hjkl a i o d y p everywhere! It's far from perfect, but eases the pain.

♡ 1 💬



Gourav • Nov 23 '19



my win key is adjacent to left ctrl key so after mapping it becomes super convenient to do ctrl+click (and drag). I still needed my win key so I've mapped it to right ctrl. I still use trackpad for mouse movements but with some gestures like 2 fingers tap for right click and 3 finger swipe to swap b/w apps like macbook.

mapping "_" to the shift space is a nice hack.

♡ 1 💬



Madyan 🌱 • Aug 7 '20



I try using the Command Palette as much as possible to avoid using my mouse. Configuring these shortcuts will defo help me reduce my mouse usage. Thanks for the great tips dude!

♡ 2 💬



Waylon Walker 🌟 • Aug 7 '20



You're welcome. Getting in and out of the terminal is the #1 greatest hotkey for me. It's also the one that took me the longest to figure out. If it wasn't for that hotkey I would be using tmux and vim, it would have been a hard stop on vscode.

Thanks for the follow ♥'s and all the comments today!

♡ 48

🔥 18

🔖 67





KiritchoukC • Nov 19 '19



Thanks for sharing !

♡ 2 💬



Waylon Walker 🌟 • Nov 19 '19



You're welcome, I really enjoyed putting it together

♡ 1 💬



Udi • Nov 19 '19



I love it, thank you!

♡ 2 💬



Waylon Walker 🌟 • Nov 19 '19



😊 Glad you enjoyed it!

♡ 1 💬

[Code of Conduct](#) • [Report abuse](#)



Waylon Walker

Data Engineering with python, kedro super user.

Follow

WORK

Sr. Data Scientist

LOCATION

Peoria, Illinois

EDUCATION

Iowa State University

♡ 48

👏 18

🔖 67



Trending on DEV Community 🔥



50+ Free Awesome Certificates to Earn in 2021

#career #computerscience #showdev #contributorswanted



How to run VS Code as a Container for Remote Development

#vscode #docker #productivity #devops



10 Fun APIs to Use For Your Next Project

#javascript #webdev #todayilearned #productivity



48



18



67

