An Interactive Virtual Keyboard to Visualize any Collection of Shortcuts

Table of Contents

Recent Posts

- Why Keyboard Shortcuts don't work on non-US Layouts and how Devs could fix it
- An Interactive Virtual Keyboard to Visualize any Collection of Shortcuts
- An app to show the shortcuts of the current application for Windows, Linux, and macOS
- Automatically add <kbd>-tags with a Single Regex

Popular Posts

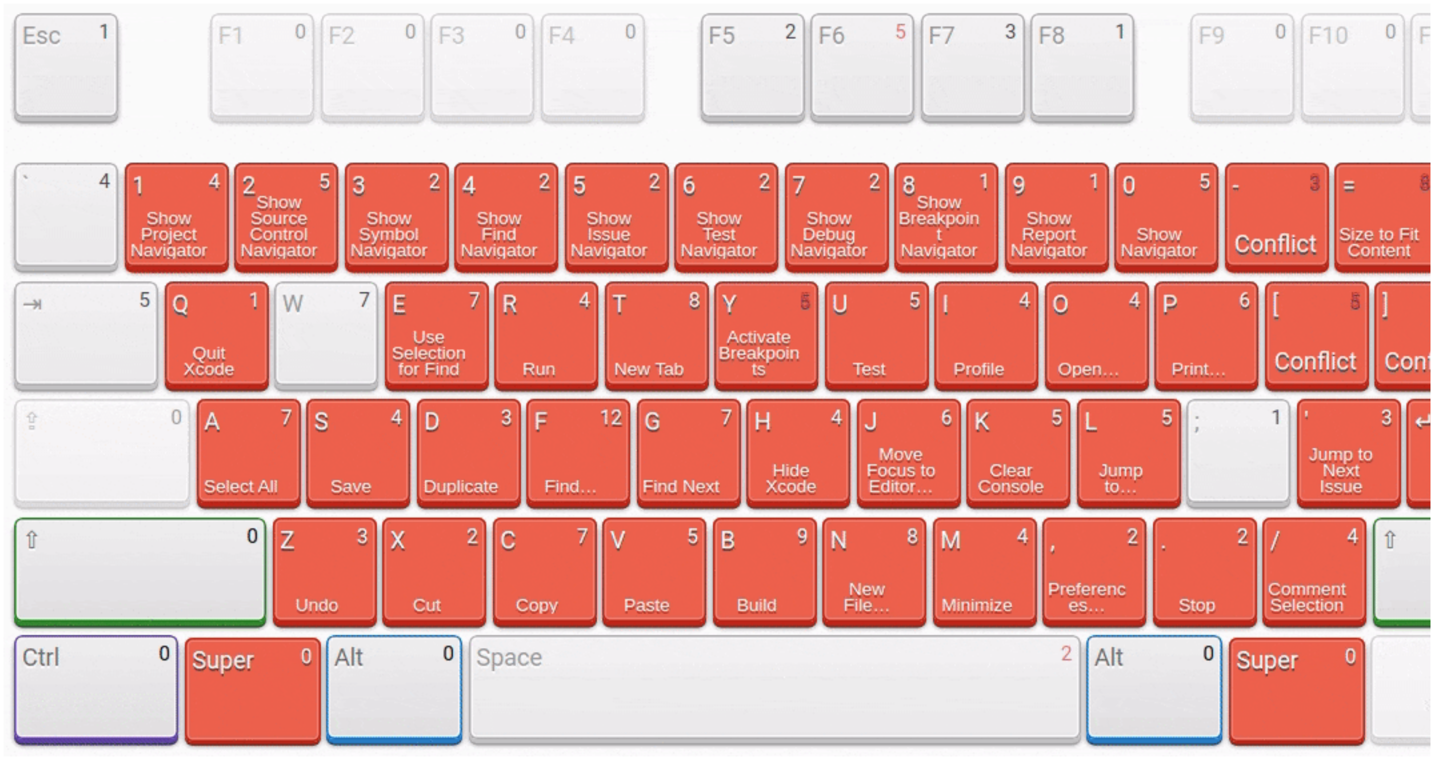
- Learning all VSCode shortcuts evolved my developing habits
- Managing my personal knowledge base
- Setting up a Linux Workstation for Software Development
- Using Hugo, GitLab Pages, and Cloudflare to create and run this Website

January 22, 2021 8 minutes read
Last modified on February 17, 2021
productivity • keycombiner • vuejs

Introduction

An important part of KeyCombiner is displaying collections of keyboard shortcuts. Therefore, I have invested a lot of time to design searching and filtering features that help to browse even large collections.

Unfortunately, these features are not sufficient when you want to understand a collection of hundreds of shortcuts at a glance. I have been thinking about this problem since I started working on KeyCombiner almost precisely one year ago. Today, I am happy to announce that KeyCombiner offers a solution:
The Shortcut Collection Visualizer



Collection Visualizer for XCode, one of very few applications that use all 4 modifier keys at the same time.

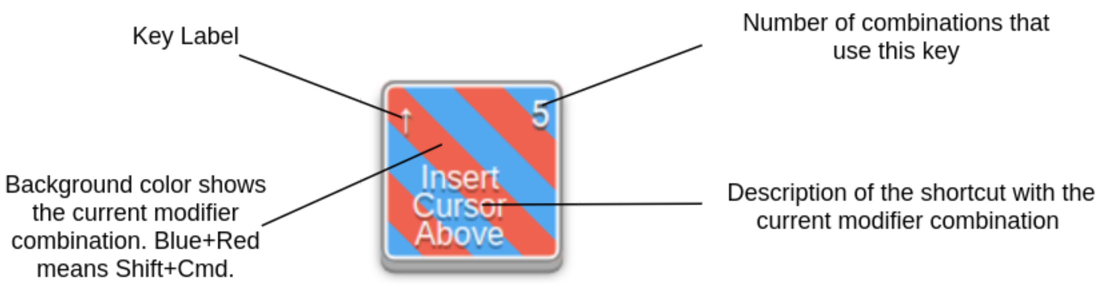
It is heavily inspired by Waldo Bronchart's open-source Application Shortcut Mapper. However, it is a new VueJS-based implementation, adding several additional features that work together with the rest of KeyCombiner. Most importantly, it can efficiently process KeyCombiner's collection tables and hence works for any shortcut collection on KeyCombiner, even search results.

If you want to play around with it right away, go to any public KeyCombiner collection, e.g. for VSCode, IntelliJ IDEA, XCode, Chrome or one of the other 60+ public collections. If you want to fully understand its potential, please read on.

Features

Overview

The virtual keyboard packs a lot of data into a relatively small space. Each button of the keyboard consists of the following elements:


Key Label
Number of combinations that use this key
↑
5
Insert Cursor Above
Background color shows the current modifier combination. Blue+Red means Shift+Cmd.
Description of the shortcut with the current modifier combination

Elements on each key of the virtual keyboard.

Grouping by Modifier Combination

A proper keyboard shortcut consists of 0 or more modifier keys and exactly one non-modifier key. There are 4 modifier keys:

1. `Ctrl`
2. `Shift`
3. `Alt`
4. `Cmd` (macOS) / `Super` (Windows and Linux)

This order of modifiers is not random. KeyCombiner *always* shows keyboard shortcuts with precisely this order. There are good reasons for this.

This means that we have four boolean variables, resulting in 2^4 possible modifier combinations. For each of these 16 states, the Collection Visualizer uses a different background color, or background gradient if there are multiple active modifiers.

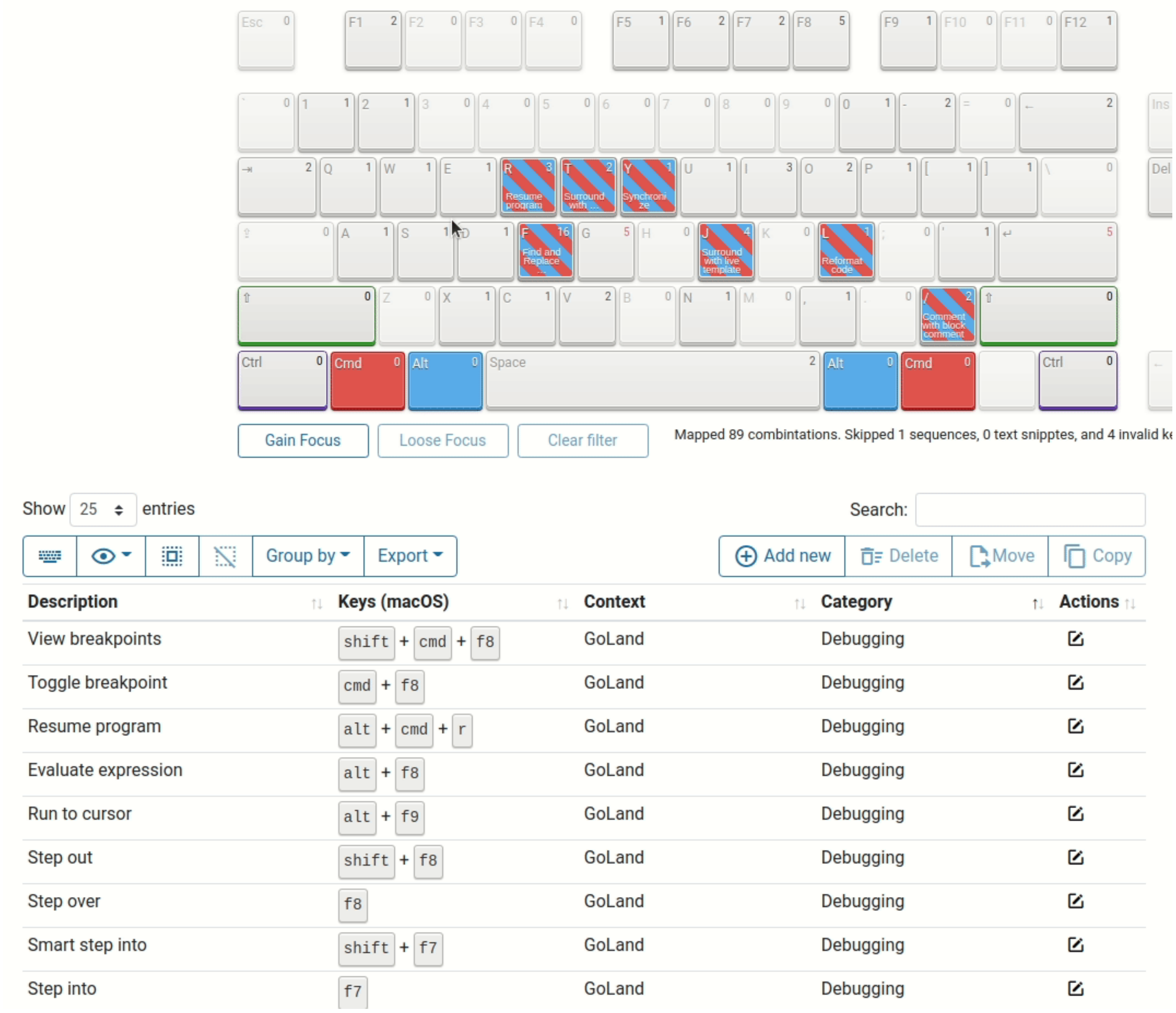
To toggle modifiers, click on the virtual buttons with your mouse, or press the respective modifier key on your physical keyboard. The entire virtual keyboard will then update according to the active combination of modifiers.

It is very rare that a key has a shortcut for every modifier combination. However, it can happen, especially when combining shortcuts of multiple applications in personal collections.
(Please don’t tell me I forgot one of the 16 modifier combinations - it took me way too long to create this animation.)

Filter Collection Table

One of my favorite things about KeyCombiner’s shortcut collections is that I can filter them by context, category, or modifier combination with a single click using the panes on the side.

The collection visualizer expands on this concept. If you click on any non-modifier key, the collection table will show all shortcuts that use this particular key. To show all shortcuts containing the key F click on the F button on the virtual keyboard.



Mapped 89 combintations. Skipped 1 sequences, 0 text snipptes, and 4 invalid ke

Description	Keys (macOS)	Context	Category	Actions
View breakpoints	shift + cmd + f8	GoLand	Debugging	
Toggle breakpoint	cmd + f8	GoLand	Debugging	
Resume program	alt + cmd + r	GoLand	Debugging	
Evaluate expression	alt + f8	GoLand	Debugging	
Run to cursor	alt + f9	GoLand	Debugging	
Step out	shift + f8	GoLand	Debugging	
Step over	f8	GoLand	Debugging	
Smart step into	shift + f7	GoLand	Debugging	
Step into	f7	GoLand	Debugging	

Filtering the collection table for all shortcuts that contain the F key.

Real-time updates on changes

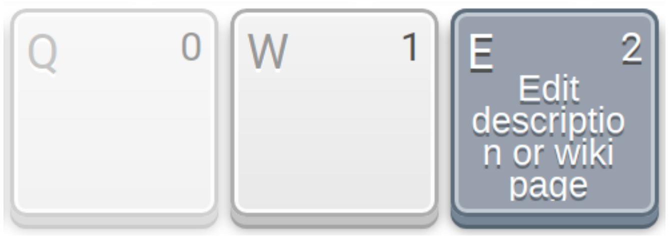
Building personal collections of keyboard shortcuts and text snippets is the foundational concept behind KeyCombiner. You can then practice these collections with its interactive trainer, relying on spaced repetition techniques and advanced statistics to guide your learning progress. You can also use KeyCombiner Desktop to instantly look up all combinations in your collections without leaving your current context.

Oh wait, I am getting side-tracked. I meant to say that the collection visualizer updates immediately whenever you make a change to one of your collections. A change could be adding new shortcuts, editing existing entries, or re(moving) entries. This works in the blink of an eye, even if you remove hundreds of combinations at once.

Additional Features

I am getting the sense that this post will be too long for the average person's interest in keyboard shortcuts. So, I will list some additional features in shorter form:

- There are three levels of opacity:
    1. Keys without any mapped combinations
    2. Keys with mapped shortcuts, but none that use the current modifiers
    3. Keys that have a combination with the currently activated modifiers



Different levels of opacity carry information.

- If, for any modifier combination, there are two or more shortcuts bound to a key, the number of combinations in the top right of the button is marked red.
- If there are two or more combinations on a key for the current modifier state, the shortcut description for this key says *Conflict*.
- There is a small text below the virtual keyboard saying how many shortcuts are mapped onto the virtual keyboard, and how many combinations had to be skipped. (See Current Limitations)
- The keyboard must be in focus if you want to activate modifiers by pressing the respective buttons on your physical keyboard. This is so that you can still use `Ctrl` and `Shift` for table selection operations without affecting the visualizer. Buttons below the virtual keyboard allow toggling the focus.

Use Cases

Quickly Grasp a Set of Shortcuts

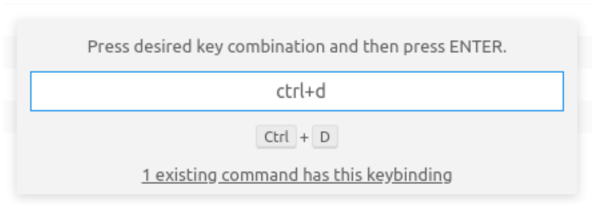
Perhaps the most obvious use case is exploring a collection of shortcuts. The visual keyboard helps immensely in this process. Within seconds, you can get a feeling of which modifiers are used by a specific application and whether it uses Vim-like home row navigation or something else entirely.

The different layers of opacity aid this use case. Without activating any modifiers, you can already understand where the most shortcuts are located. This is supported further by the combination count in each virtual keyboard button's top right. Filtering the collection table by clicking on a specific key lets you see all shortcuts for that key and understand how they are related.

See Conflicts and Free Combinations

At the moment, this is my favorite use case, as I have used it plenty of times already with great success.

I recently learned all VSCode shortcuts with KeyCombiner's interactive trainer. However, since then, I have started to experiment with Foam and picked up some other extensions. All of these come with their own set of commands. So, I frequently have to find an available key combination for a new command I want to use efficiently. VSCode itself is not much help with that. It tells you *after* setting a combination that it is already taken:


Press desired key combination and then press ENTER.

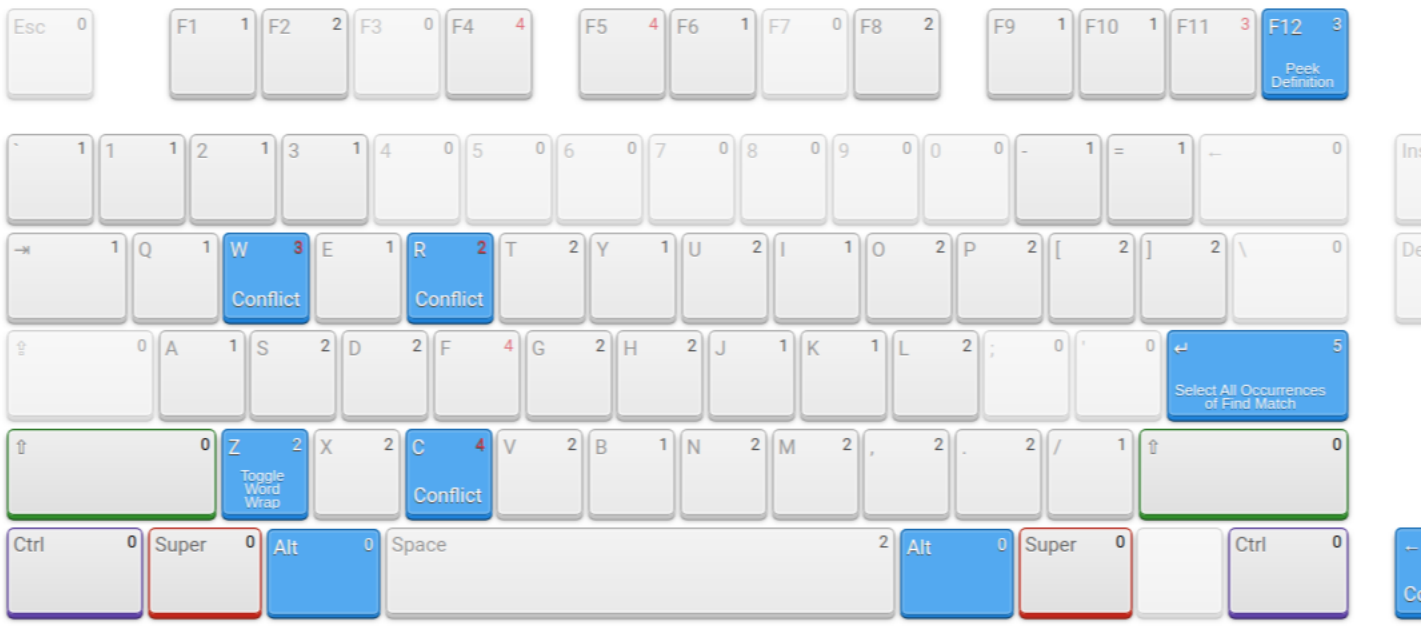
ctrl+d

Ctrl + D

1 existing command has this keybinding

Different levels of opacity carry information.

I guess it's better than nothing, but trying multiple combinations and manually checking what other combination is already using that binding and whether you might be able to remove that other binding is not much fun. The collection visualizer made it trivial to see that there are actually plenty of free combinations in VSCode, only `Ctrl` and `Shift` are quite busy by default. Things start happening if you mix in `Alt`:



All modifier combinations with `Alt` are wide open for your own assignments in VSCode.

You can then go one step further and find free combinations that are easy to type. For me, these are combinations that I can type with just my left hand. If the non-modifier key is on the home row, that’s another big plus. In any case, a convenient shortcut should have a maximum of two modifiers.

Design a Coherent Set of Shortcuts

The collection visualizer helps design a coherent set of shortcuts, either for yourself or for an application you are developing.

Unfortunately, many application designers do not think very hard about keyboard shortcuts. Often, you end up with a set that is neither intuitive nor easy to type. Heck, even KeyCombiner’s own shortcuts are all over the place with sequences and different modifier combinations. Given that I work more or less alone on the project and try to be very efficient with my time, I didn’t think about these bindings enough. The collection visualizer makes this painfully obvious, and I will soon come up with new shortcuts. However, it will be very hard not to annoy users who have already memorized these shortcuts. So, I recommend that you be smarter than me and start to design a coherent set of shortcuts for your application right away. The collection visualizer is here to help you with that.

If you are not an application designer, you might still want to design a coherent set of key bindings for your personal use. Without any tools to assist you, this is a surprisingly hard tasks, especially when you try to find a coherent set for or *multiple* applications. You have to keep in mind which commands are available in these different apps, what the defaults are, and how to resolve these constraints into a set that works everywhere. The collection visualizer, along with KeyCombiner’s other collection management features, can help you get there.

Current Limitations

Above, I have written that a proper keyboard shortcut consists of 0 or more modifier keys and exactly one non-modifier key. However, KeyCombiner also allows sequences, such as the *Go To* shortcuts used by Gmail. I have been thinking a lot about how to visualize those on a virtual keyboard, but have not found a good solution yet.

Furthermore, KeyCombiner collections can also hold short text snippets, such as commands and programming language syntax. Many people use these snippets with the Desktop Apps' instant lookup. It turns your collections into an instant, context-aware, searchable cheatsheet. However, I struggle to find a way to visualize them on a keyboard.

Conclusion

In its first days, the collection visualizer has already helped me plenty of times. I improved my VSCode bindings, realized that KeyCombiner’s own default bindings are not intuitive, and found better ways to reuse my VSCode bindings in PyCharm and Eclipse. I’d be thrilled to hear about your experiences in the comments below or via mail.

I will write about the collection visualizer’s implementation in a future blog post. Spoiler: Vue and (S)CSS do the heavy lifting.

2 Comments tkainrad.dev Disqus' Privacy Policy 1 Login

Favorite 5 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

Epinephelus Itajara • 3 months ago
I did something very similar in AutoHotKey. Yours looks more professional

Oddly.. I dont see a link to download the thing here! :) If I cant find it in a few minutes, ill probably just email you

• Reply • Share ›

tkainrad Mod Epinephelus Itajara • 3 months ago
Hey, interesting! I don't think you mean the popular decades-old AutoHotKey for Windows. Do you have a link to your software?

To use what's described in this blog post, you don't need to download anything. Just go to https://keycombiner.com ;)

• Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Do Not Sell My Data

Blog Post Tags

architecture (1) • blogging (6) • cloud (1) • cloudflare (1) • database (1) • devops (4) • django (3) • gitlab (2) • hpc (2) • hugo (1) • java (1) • javascript (2) • keycombiner (7) • linux (3) • nosql (1) • notion (1) • productivity (9) • python (3) • saas (1) • twitter (1) • vscode (2) • vuejs (2) • wagtail (1) •

Thomas Kainrad | Vienna |