Visual Studio Code

Thursday, March 29, 2018 8:45 AM

Comparison to Sublime

- Pros
  - Emoji support is better
  - Git integration by default is nice
  - JavaScript auto-completion and general "awareness" of the language is nice
  - Customizing keyboard shortcuts is generally a nicer process.
- Cons
  - VSC's mouse control is not as customizable, e.g. you can't hold right-click and scroll to switch tabs (not that I used this).
  - When copy/pasting text from one source file to a blank file, the blank file in Sublime would be given the same syntax highlighting as the original file.
  - Joining lines
    - VSC's "join lines" functionality isn't aware of comments, so if you join a multi-line comment like "//a\n//b", you'll end up with "//a //b" instead of Sublime's "//a b".
    - VSC's join also works different with multiple cursor. In Sublime, if you have 10 lines with 1 cursor each and join them, you'd end up with a single line with all 10 original lines on it. In VSC, you end up with 5 lines and then can't even run the join operation again without getting an error. The workaround is to expand the selection to the entire line (expandLineSelection) and *then* join.
  - Fuzzy matching in the Quick Open bar doesn't always work how I'd expect. In Sublime, I could press ctrl+shift+P and type "jvs" to match "Set syntax: JavaScript". In VSC in the set-language menu (ctrl+K, M), "jvs" doesn't work, but "j*v*s" does.
    - VSC just has a different set of rules:
      - "jas" works because it's written as "JavaScript", and VSC sees the capitalized "S", so it must implicitly treat "jas" like "ja*s".
      - "asc" also works because the characters are contiguous
    - This is especially weird with something like "Open Settings (JSON)", because you can't search for "S J" even though both of those letters are capitalized; you have to search for "S (".
    - Note that in this particular case with language identifiers, you can apparently customize them.
- Differences (i.e. not pros/cons)
  - Selecting collapsed text in VSC is a little bit strange since you need to select some part of the line *after* the collapsed text for it to select everything that was collapsed. For example, write "Line 1" on the first line and "Line 2" on the second, then collapse "Line 2". If you don't have a line 3, you literally cannot copy the contents of line 2 without expanding it first.
  - Sublime projects are roughly the same as VSC workspaces
  - Almost all keyboard chords (e.g. ctrl+X followed by ctrl+Y) in Sublime had ctrl used for both combinations in the chord. In VSC, many times you'll find ctrl+X followed by just Y.
  - Some tabs open differently, e.g. ctrl+, in VSC opens the Settings, but opening another file after with ctrl+P will close settings. This is because the Settings tab was never focused with ctrl+K,enter (no ctrl on enter, just on "K").
  - By default, Sublime's ctrl+click for multiple cursors is alt+click in VSCode
  - In Sublime, while dragging the minimap, you can press escape to cancel the drag operation and snap back to where you started. In VSCode, at least on Windows, you can't do that, but if you drag far enough horizontally, it will snap back just like how all scrollbars work in Windows.

    - In Sublime, you can right-click URLs and open them from the context menu. In VSC, you ctrl+click them (which IMO is better, but not a huge pro to list separately above). Note that for VSC to accomplish this, it means that you can't have Sublime's ctrl+click functionality of placing multiple cursors (it's alt+click).
    - Sublime's duplicate behavior works differently from VSC's. In VSC, it's not actually called "Duplicate"; it's copyLinesDownAction), and it always duplicates the entire line. In Sublime, if you have a selection, it will just duplicate the selection. To get this behavior in VSC, install this plugin.
    - No incremental search in VSC by default. I'm going to just try not using it since it's pretty similar to ctrl+F.
    - In Sublime, deleting a line with a bookmark would retain the bookmark. In VSC, it deletes the bookmark. VSC's behavior is annoying for managing TODO lists because I delete text frequently, but I still want to know WHERE I deleted text.
    - In Sublime, if you have two editor groups open and close the last tab of one, it doesn't put you back into a single editor group, whereas in VSC it seems to (although it's customizable via workbench.editor.closeEmptyGroups)
    - "Copy file path" and "Reveal in Explorer" are in the *buffer's* right-click menu in Sublime, but in VSC, they're in the *tab's* menu.
    - Sublime lets you type quotation marks, brackets, parens, etc. when you have a selection to wrap the selection in that set of characters. VSC has this controllable via editor.autoSurround, but you have to add custom keyboard shortcuts to get Sublime's functionality (which would be "always" in the dropdown).

Surrounding the selection in ' and " in plaintext files

If you select a word in a plaintext file and type an apostrophe or quotation marks, it will *replace* the selection with what you typed, not *surround* it. To get it to surround, add this to your keyboard shortcuts JSON:

```
{
  "key": "'", // Single Quote ('')
  "command": "editor.action.insertSnippet",
  "when": "editorLangId == plaintext && editorTextFocus && editorHasSelection",
  "args": {
    "snippet": "'${1:${TM_SELECTED_TEXT}}'"
  }
},
{
  "key": "shift+'", // Double Quote ("")
  "command": "editor.action.insertSnippet",
  "when": "editorLangId == plaintext && editorTextFocus && editorHasSelection",
  "args": {
    "snippet": "\"${1:${TM_SELECTED_TEXT}}\""
  }
}
```

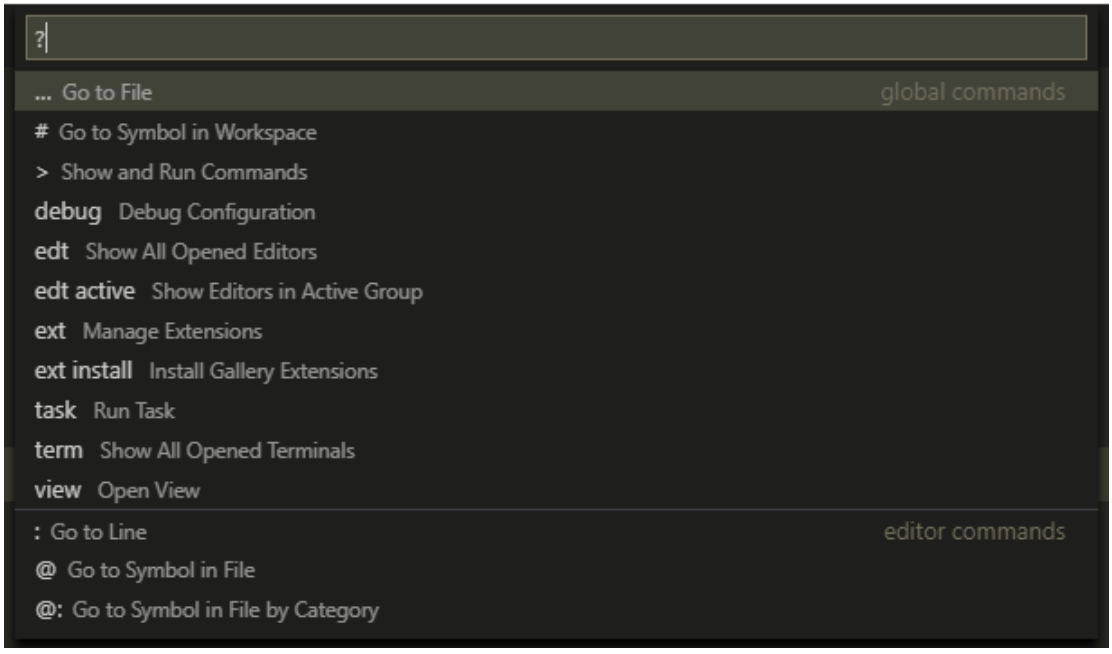
This is needed because editor.autoSurround has no "always" option.

[15:21] HiDeoo: Adam13531 Btw, found the reason why it doesn't work for quote in plaintext & will prolly never work, this a language definition feature, but plaintext is not a language, it's the editor without any language https://github.com/Microsoft/vscode/tree/master/extensions

Random tips

- All of the Git stuff takes place in SCM windows (Source Control Management)
- When using ctrl+P to quick-open files, you can press the right arrow to open the file outside of the temporary buffer.
- In the settings screen, if you click a pencil icon, it will automatically copy it over to your user settings and give it a particular value.
- VSC has "GitGutter" from Sublime baked in by default; you can see changes you made to a particular file on the left of the line, and clicking that shows a diff.
    - Alternatively, you can click the magnifying glass at the upper right of the editor to open it in

a new buffer entirely

- You can search for extensions directly through the editor by doing ctrl+shift+P --> Extensions --> Install Extensions (the default shortcut is ctrl+shift+X)
- You can hide or show the Activity Bar (the vertical icons on the left side by default) in the View --> "[Show|Hide] Activity Bar"
- Typing a "?" in Quick Open will give you this help dialog:
  - 

Extensions

Searching the Internet for extensions

ChiefZ suggested searching for "VSCode" rather than "Visual Studio Code", that way you don't get a million results for Visual Studio (the full IDE).

Listing all installed extensions from the command line

code.cmd --list-extensions

(code.exe doesn't work with this argument on Windows)

Checking settings

To see an extension's settings, just press ctrl+, and search for something like "powermode".

Clicking an extension's title will open the marketplace link in the browser so that you can share it easily.

Disabling all extensions

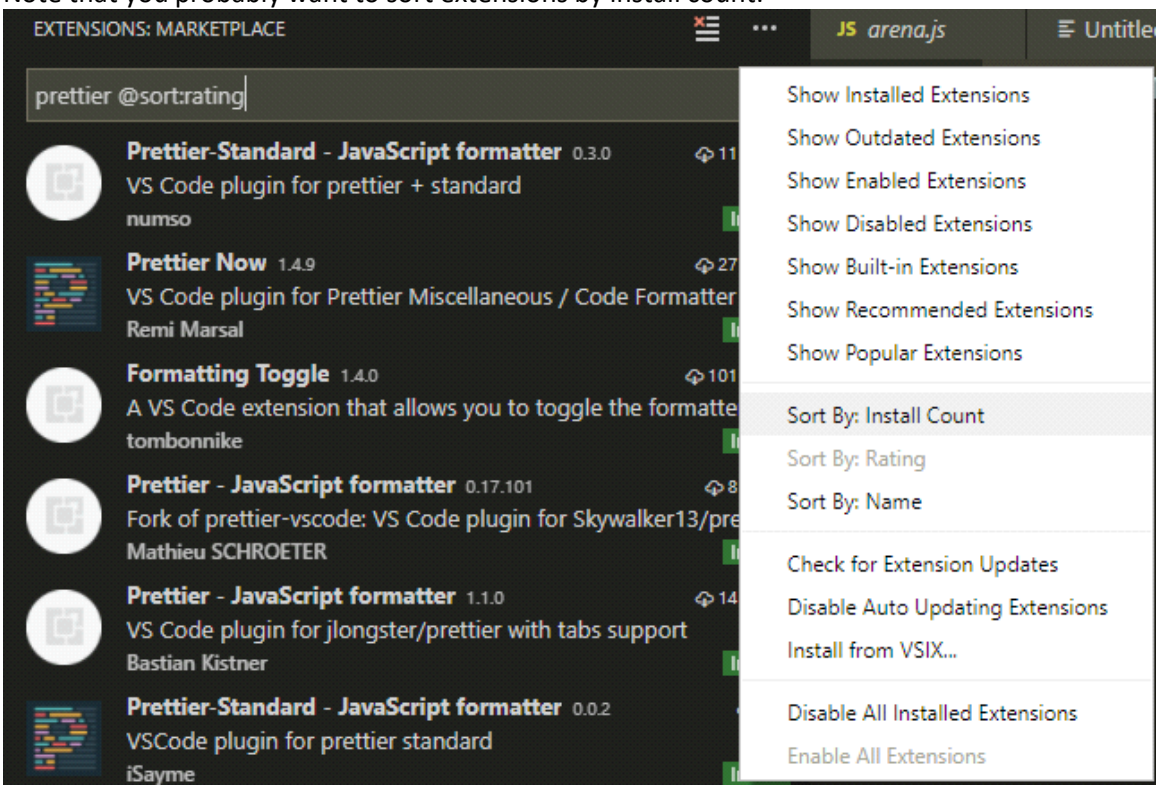
code.cmd --disable-extensions

This is helpful for debugging whether a particular problem is happening due to an extension.

Installing

To install an extension, just ctrl+shift+P → Install. Alternatively, assign a shortcut to "workbench.view.extensions" (I made it ctrl+shift+X).

Note that you probably want to sort extensions by install count:



After that, you will likely have to press the "Reload" button where "Install" used to be (or you can access it via the command palette).

Find in files

This works pretty similarly to Sublime:

- Ctrl+shift+F pops up the find-in-files dialog
- [Shift+]F4 will go between results
- If the find-in-files dialog has the focus, then the up and down arrows will also go between results, but the focus won't be in the editor obviously.
- You can dismiss an entire file's worth of results with the Dismiss button on the right side or pressing "delete" when the find pan has the focus

Snippets (reference)

In VSC, all of your snippets for a particular language go into a single file rather than being defined in individual files like in Sublime.

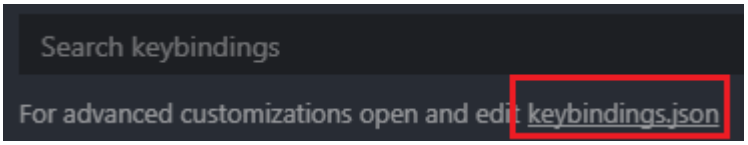
Summary:

- Ctrl+shift+P: "user snippets"
- Pick the language you want (or "global")
- Follow the instructions in the file that is created or the reference link above

Keyboard shortcuts

Basics

To modify shortcuts, press ctrl+K,ctrl+S (Preferences → Open Keyboard Shortcuts). To focus the search, press ctrl+F after that. To edit them as JSON, click the link just below the search field:



When searching for shortcuts, you don't need to type "+", e.g. "ctrl shift b".

If you see a "-" next to the command, it means that the shortcut is not used.

- Definition-related things
  - Shift+F12: see references
  - Alt+F12: peek definition (this seems to do nothing)
  - Ctrl+F12 (or alt+click): go to definition
- Ctrl+shift+G is the default Git Integration shortcut
- Ctrl+K, M (**no** ctrl on the "M" part): change language, e.g. to make a plain text file into a JS-highlighted file

"when" clause

Boolean OR

This is not supported (reference), but you can represent (A || B) with !(!A && !B), so:

BAD: "when": "(editorLangId == plaintext || editorLangId == todo) && editorTextFocus && editorHasSelection"

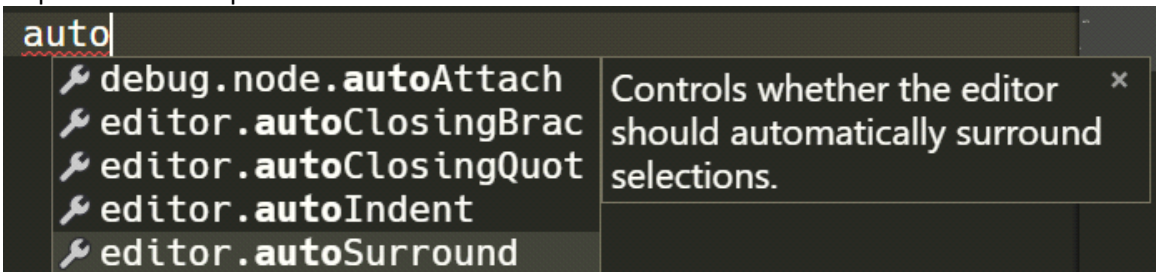
GOOD: "when": "!(editorLangId != plaintext && editorLangId != todo) && editorTextFocus && editorHasSelection"

Settings

Modifying arbitrary settings

Ctrl+, gets you to the main settings window, but then if you want to modify the JSON file directly, you have to click the three dots at the top right (although you can set a keyboard shortcut to open it directly or type "settings (" in Quick Open).

While modifying the JSON file directly, you can just start typing the name of a setting and VSC will autocomplete it and its options:



Git workflow

Fixing conflicts

First, to even see that you have conflicts, make sure that you've pulled from the right remote/branch (and if it's a pull request, make sure you've rebased to that remote/branch; I think it's better to do this from the command line).

Click the file in the SCM to open the diff. Fix the conflicts. Finally, stage the file with the "+" button.

git rm

The easiest way to do this is to just delete the file from your hard drive (e.g. with Windows Explorer). The VSC SCM will detect the change and then you can stage it.

Staging hunks

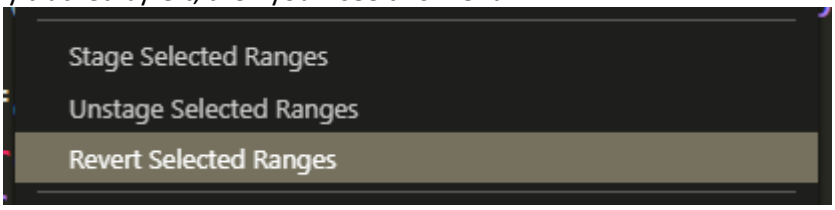
If you have a file with two commits' worth of changes, then you can stage hunks to separate the changes into two commits without having to back up the file and restore it later.

The process is very simple:

1. Go to any file with changes
2. Right-click the lines that you want to commit right now and choose "Stage Selected Lines/Ranges"
3. Commit

Reverting selected changes

When viewing diffs side by side, you have to right-click in the *right* panel as opposed to the changes already tracked by Git, then you'll see this menu:



After choosing that option, make sure you save since it's not automatic.

Git setup

This process took me several hours to figure out. It's relatively basic in the end. The goal was to get VSC to be able to run "git fetch" without asking for my password. Note that it *does* have to ask once per OS restart, but I was fine with that. The idea behind these steps is that VSC will inherit the environment from the caller, so as long as the caller is Bash with ssh-agent set up correctly, VSC should work too.

General steps:

- Get https://gitforwindows.org/
- Set up SSH keys here. For me, this involved:
    - Launch git-bash.exe (*not* just bash.exe)
    - Put the auto-launch script from here into ~/.bash_profile
    - Do "$ source ~/.bash_profile" to make sure that it asks me for my password and loads the key.
- Then, launch VSC from git-bash where everything is already configured so that VSC inherits those settings (reference):
    - Launch git-bash.exe
    - $ /c/Program\ Files/Microsoft\ VS\ Code/Code.exe &
    - $ disown
    - $ exit

Note: I do not think I had to run "git config --global credential.helper wincred" since I don't see that there when I do "git config --edit --global".

I put those last steps into a function in my bash_profile too:

```
function startVscAndExit() {
    # put three steps here
}
```

Then, from git-bash, simply type "startVscAndExit". Alternatively, from CMD, you can start git-bash and have it run that command:

```
start "" "%ProgramFiles%\Git\git-bash.exe" -c "startVscAndExit"
```

I'm not sure what's happening exactly, but I apparently don't need to launch from this CMD every time. Maybe it's just once per OS restart.

Also, apparently you can setup VSC to use PuTTY/Pageant:
https://www.cgranade.com/blog/2016/06/06/ssh-keys-in-vscode.html

[10:17] maggges: if you use putty: GIT_SSH=c:\Program Files\Putty\plink.exe
[10:17] maggges: if you use keepass, the plugin KeeAgent is suuuuper useful, it'll take the passphrase from the password entry

Snippets

Transformations (reference1, reference2)

Suppose you want to turn "Hello world" into "HELLO WORLD" (i.e. capitalize it); you can use transformations for that. The only weird thing is that you need to press tab for the transformation to work. You can't press enter or escape.

Here's an example snippet of how to capitalize the name that you type (remember: you have to press tab after typing the name):

```
"capitalizeName": {
  "prefix": "capName",
  "body": "${1:name} --capitalize--> ${1/(.*)/${1:/upcase}/}",
  "description": "",
  "scope": "javascript, javascriptreact"
},
```

Troubleshooting

Can't pull a repository that you've been able to pull before

It's possible that you're trying to do "Git pull..." before the SCM has finished loading.

Can't use VSC as the difftool (DiffDirectoryCommand Error: No diff tool found)

The problem is that I tried to run GitLens commands like "Show current branch history" → "Open directory compare with previous revision" and it gave me "DiffDirectoryCommand Error: No diff tool found".

I never actually found a solution for this. I started by looking here and I tried all manners of configuring my path to use code.cmd vs. code.exe, but no matter what I did, I kept getting "--wait [and] --diff are not valid options" (paraphrasing). However, running "code.exe --wait --diff file1 file2" on the command line DID work. It even worked from the terminal inside of VSC.

Performance issues

Look at this.