BOOST YOUR CODING FU

with VSCode and Vim

CHEATSHEET

BABY STEPS

- `h` move cursor to the left
- `j` move down
- `k` move up
- `l` move right
- `i` Go into Insert mode
- `<ESC>` `<C-C>` `<C-[>` Go back to Normal mode

MOVE FAST WORD BY WORD

- `w` move to the beginning of next word
- `b` move to the beginning of the previous word
- `e` move to the end of the next word
- `ge` move to the end of the previous word

FIND CHARACTER

- `f{character}` Find next occurrence of character
- `F{character}` Find previous occurrence of character
- `t{character}` Find next occurrence of character and place cursor just before it
- `T{character}` Find previous occurrence of character and place cursor just before it

- `W` move to the beginning of next WORD
- `B` move to the beginning of the previous WORD
- `E` move to the end of the next WORD
- `gE` move to the end of the previous WORD

- `;` Go to next occurrence of {character}
- `,` Go to previous occurrence of {character}

MOVE EXTREMELY HORIZONTALLY

- `0` Moves to the first character of a line
- `^` Moves to the first non-blank character of a line
- `$` Moves to the end of a line
- `g_` Moves to the non-blank character at the end of a line

MOVE FASTER VERTICALLY

- `}` Jumps entire paragraphs downwards
- `{` similarly but upwards
- `CTRL-D` lets you move down half a page by scrolling the page
- `CTRL-U` lets you move up half a page also by scrolling

HIGH PRECISION VERTICAL MOTIONS WITH SEARCH

- `/{pattern}` Search for {pattern}. {pattern} is a regex.
- `?{pattern}` Search for {pattern} backwards.

- `/` Repeat last search forwards
- `?` Repeat last search backwards

- `n` Go to next match
- `N` Go to previous match

Move Faster With Counts

- `{count}{motion}` Repeat {motion} {count} times

- `2w` Jump to second word
- `4f"` Jump to fourth occurrence of the " character
- `3/cucumber` Jump to third match of "cucumber"

Move Semantically

- `gd` Go to definition (of the word under the cursor)
- `gf` Go to file (for file under the cursor)

More Nifty Core Motions

- `gg` Go to the top of the file
- `{line}gg` Go to {line}
- `G` Go to the end of the file
- `%` jump to matching ({[]})

Edit Like Magic With Vim Operators

- `{operator}{count}{motion}` Apply operator on bit of text covered by motion

- `d` delete
- `c` change
- `y` yank (copy)
- `p` p (paste)
- `g~` switch case

Linewise Operators

- `dd` delete a line
- `cc` change a line
- `yy` yank (copy) a line
- `g~~` switch case of a line
- `>>` shift line right
- `<<` shift lineleft
- `==` format line

Capital Case (Stronger Version) Operators

- `D` delete from cursor to the end of the line
- `C` change from cursor to the end of the line
- `Y` yank (copy) a line. Like yy
- `P` put (paste) before the cursor

- `>` shift right
- `<` shift left
- `=` format

TEXT OBJECTS

- `{operator}a{text-object}` Apply operator to all text-object including trailing whitespace
- `{operator}i{text-object}` Apply operator inside text-object

- `diw` delete inner word
- `daw` delete a word
- `dis` delete inner sentence
- `das` delete a sentence
- `dip` delete inner paragraph
- `dap` delete a paragraph
- `di(` `dib` delete inside parentheses
- `da(` `dab` delete text inside parentheses (including parentheses)
- `di{` `diB` delete inside braces
- `da{` `daB` delete text inside braces (including braces)
- `di[` delete inside brackets
- `da[` delete text inside brackets (including brackets)
- `di"` delete inside quotes

REPEAT LAST CHANGE

- `.` Repeat the last change

CHARACTER EDITING COMMANDS

- `x` delete a character. Like dl
- `X` delete character before the cursor. Like dh
- `s` change a character. Like cl
- `~` switch case of a character

- `da"` delete a quoted text (including quotes)
- `dit` delete inside tag
- `dat` delete a tag (including tag)

- `ciw` same goes for other operators...

UNDO AND REDO

- `u` undo last change
- `C-R` redo last undo
- `{count}u` undo last {count} changes

INSERTING TEXT

- `i` go into insert mode before the cursor
- `a` go into insert mode after the cursor

- `I` go into insert mode at the beginning of a line
- `A` go into insert mode at the end of a line

- `o` insert new line below current line and go into insert mode
- `O` insert new line above current line and go into insert mode

---

- `gi` go to the last place you left insert mode

- `C-H` delete last character
- `C-W` delete last word
- `C-U` delete last line

VISUAL MODE

- `v` go into character-wise visual mode
- `V` go into line-wise visual mode
- `C-V` go into block-wise visual mode (to select rectangular blocks of text)

- `{trigger visual mode}{motion}` `{operator}` Visual mode operates in kind of the opposite way to normal mode. First you specify the motion to select text, and then you apply the operator

Operate on next search match

- `{operator}gn` Apply operator on next match

- `.` After using {op}gn, the dot commant repeats the last change on the next match. Woooot!

Copying and pasting

- `y{motion}` yank (copy) text covered by motion
- `p` put (paste) after cursor
- `P` paste before cursor

- `yy` copy line
- `Y` copy line

- `yyp` duplicate line
- `ddp` swap lines
- `xp` swap characters

- `"ay{motion}` copy to register a
- `"Ay{motion}` copy and append to register a
- `"ap` paste from register a

- `"` unnamed register
- `0` yank register
- `1-9` delete registers
- `[a-z]` named registers

- `C-R a` paste from register a when in Insert mode

Command-line mode

- `:edit {file}` `:e {file}` create or edit file
- `:write` `:w` save file
- `:quit` `:q` close file
- `:write!` `:w!` force save file
- `:quit!` `:q!` close file without saving
- `:wq` save and close file
- `:wall` `:wa` save all files
- `:qall` `:qa` close all files
- `:wqall` `:wqa` save and close all files
- `:qall!` `:qa!` close all files without saving

- `:[range]delete [register]` `:[r]d [r]` delete multiple lines into register

- `@:` repeat last ex command
- `@@` after repeating it once, you can continue repeating with this

COMMAND-LINE MODE RANGES

- `:{start},{end}` start and end lines of range e.g. :1,2d
- `:{start},{offset}` start and offset lines of range e.g. :1,+2d
- `.` current line e.g. :.,+2d
- `%` whole file e.g. :%d
- `0` beginning of file e.g. :0,10d
- `$` end of file e.g. :10,$d
- `:'<,'>` visual selection

COMMAND-LINE MODE SUBSTITUTE

- `:[range]/{pattern}/{substitute}/[flags]` substitute matched pattern for string literal in given range

- `g flag` substitute all matches in a line
- `i flag` case insensitive search
- `c flag` confirm substitution for each match

SPLIT WINDOWS

- `:sp {file}` Open file in a horizontal split
- `:vsp {file}` Open file in a vertical split

- `C-W S` Open same file in a horizontal split
- `C-W V` Open same file in a vertical split

- `C-W h` Move to split to the left
- `C-W j` Move to split below
- `C-W k` Move to split above
- `C-W l` Move to split to the right

TABS

- `:tabnew {file}` Open file in new tab
- `:tabnext` `:tabn` Jump to next tab
- `:tabprev` `:tabp` Jump to previous tab
- `:tabonly` `:tabo` Close all other tabs

VIM SURROUND

- `ds` delete surroundings e.g. ds"
- `cs` change surroundings e.g. cs*tem>
- `ys` add surroundings e.g. ysiw"

- `ds"` delete surrounding quotes
- `cs*tem>` change surrounding * for the tag

CUSTOM MAPPINGS (THEY NEED TO BE ADDED TO YOUR CONFIG)

- `ysiw"` surround word under the cursor with quotes

- `S` In visual mode you can select some text, then type S to add surroundings. e.g. Stp> to wrap the selection in a <p> tag

Insert mode

- `jk` Go back to normal mode

Normal mode

- `J` Go down faster
- `K` Go up faster
- `<leader>j` Join lines
- `<leader>/` Clear highlighted text
- `<leader>w` Save file

- `<leader>p` Open command palette
- `<leader>t` Go to symbol in file

- `<C-H>` Move to the window on the left
- `<C-J>` Move to the window below
- `<C-K>` Move to the window above
- `<C-L>` Move to the window on the right

- `<leader>tt` Create new tab
- `<leader>tn` Move to next tab
- `<leader>tp` Move to previous tab
- `<leader>to` Close all tabs but the current one

MOVING FASTER WITH VIM-

MOVING FASTER WITH

MULTIPLE CURSORS

SNEAK

- `s{char}{char}` Jump to the next ocurrence of {char}{char}
- `S{char}{char}` Jump to the previous ocurrence of {char}{char}
- `;` Go to next occurrence of {char}{char}
- `,` Go to previous occurrence of {char}{char}
- `{op}z{char}{char}` Apply operator on text traversed by vim sneak motion

EASYMOTION

- `<leader><leader>w` start of words
- `<leader><leader>b` start of words backwards
- `<leader><leader>bdw` start of words everywhere. The bd stands for bidirectional
- `<leader><leader>e` end of words
- `<leader><leader>ge` end of words backwards
- `<leader><leader>bdw` end of words everywhere
- `<leader><leader>j` beginning of lines
- `<leader><leader>k` beginning of lines backwards
- `<leader><leader>f{char}` find character
- `<leader><leader>F{char}` find character backwards
- `<leader><leader>t{char}` until character
- `<leader><leader>T{char}` until character backwards
- `<leader><leader>s{char}` search character everywhere

Based on Search

- `<CMD-D>` `<C-D>` `gb` Adds an additional cursor. This command puts you in visual mode. In effect, it selects all the words under each new cursor at once.
- `{operator}` Applies operator on words selected by the multiple cursors.
- `I` Insert before multiple words at the same time
- `A` Append after multiple words at the same time

In Consecutive Rows

- `<C-V>` Go into Visual-block mode and select text using motions
- `{operator}` Applies operator on visual selection.
- `I` Insert before visual selection
- `A` Append after visual selection

Reusable Editing With Macros

- `q{register}` Start recording a new macro in {register}
- `q` Stop recording macro
- `@{register}` Replay macro in {register}
- `@@` Replay the last macro that you executed

Integrating With Neovim

- `:[range]copy {address}` Copy lines after {address} line
- `:[range]move {address}` Move lines after {address} line
- `:[range]normal {commands}` Apply normal mode commands on a range of lines
- `:execute "{ex-command}"` Execute string as an Ex command. Useful in combination with normal when the normal mode commands include special characters.
- `:[range]global/{pattern}/{command}` Execute an Ex command on lines within a range that match a pattern. The default range is the whole file. Really useful in combination with the normal command.
- `:[range]global!/{pattern}/{command}` Execute an Ex command on lines

Some Handy VSCode Only Mappings

- `af` Select increasingly bigger blocks of text
- `gh` Equivalent to hovering the mouse over a bit of text.

within a range that do not match a pattern.

Bonus 01: Entire and arguments text objects

- `dae` Delete complete buffer
- `yae` Yank complete buffer
- `cia` Change function argument
- `daa` Delete function argument including separator