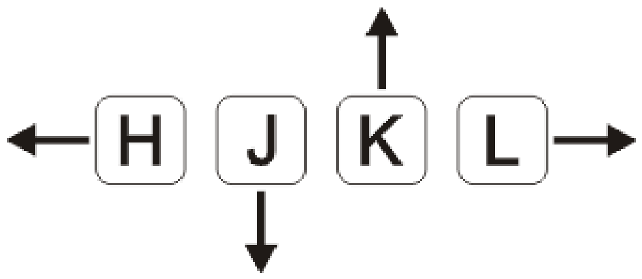
A Great Vim Cheat Sheet

Note: If you're decent at vim and want your mind blown, check out Advanced Vim.

I've compiled a list of *essential* vim commands that I use every day. I then give a few instructions on how to making vim as great as it should be, because it's painful without configuration.

Cursor movement (Inside command/normal mode)



- `w` - jump by start of words (punctuation considered words)
- `W` - jump by words (spaces separate words)
- `e` - jump to end of words (punctuation considered words)
- `E` - jump to end of words (no punctuation)
- `b` - jump backward by words (punctuation considered words)
- `B` - jump backward by words (no punctuation)
- `0` - (zero) start of line
- `^` - first non-blank character of line (same as 0w)
- `$` - end of line
- Advanced (in order of what I find useful)
  - `Ctrl+d` - move down half a page
  - `Ctrl+u` - move up half a page
  - `}` - go forward by paragraph (the next blank line)
  - `{` - go backward by paragraph (the next blank line)
  - `gg` - go to the top of the page
  - `G` - go the bottom of the page
  - `: [num] [enter]` - Go To that line in the document
  - Searching
    - `f [char]` - Move to the next char on the current line after the cursor
    - `F [char]` - Move to the next char on the current line before the cursor
    - `t [char]` - Move to before the next char on the current line after the cursor
    - `T [char]` - Move to before the next char on the current line before the cursor
    - All these commands can be followed by `;` (semicolon) to go to the next searched item, and `,` (comma) to go the the previous searched item

Insert/Appending/Editing Text

- Results in insert mode
  - `i` - start insert mode at cursor
  - `I` - insert at the beginning of the line
  - `a` - append after the cursor
  - `A` - append at the end of the line
  - `o` - open (append) blank line below current line (no need to press return)
  - `O` - open blank line above current line
  - `cc` - change (replace) an entire line
  - `c [movement command]` - change (replace) from the cursor to the move-to point.

    - ex. `ce` changes from the cursor to the end of the cursor word
- Esc - exit insert mode
- `r [char]` - replace a single character with the specified char (does not use insert mode)
- `d` - delete
    - `d` - [movement command] deletes from the cursor to the move-to point.
    - ex. `de` deletes from the cursor to the end of the current word
- `dd` - delete the current line
- Advanced
    - `J` - join line below to the current one

Marking text (visual mode)

- `v` - starts visual mode
    - From here you can move around as in normal mode (hjkl etc.) and can then do a command (such as `y`, `d`, or `c`)
- `V` - starts linewise visual mode
- `Ctrl+v` - start visual block mode
- `Esc` - exit visual mode
- Advanced
    - `O` - move to Other corner of block
    - `o` - move to other end of marked area

Visual commands

Type any of these while some text is selected to apply the action

- `y` - yank (copy) marked text
- `d` - delete marked text
- `c` - delete the marked text and go into insert mode (like c does above)

Cut and Paste

- `yy` - yank (copy) a line
- `p` - put (paste) the clipboard after cursor
- `P` - put (paste) before cursor
- `dd` - delete (cut) a line
- `x` - delete (cut) current character
- `X` - delete previous character (like backspace)

Exiting

- `:w` - write (save) the file, but don't exit
- `:wq` - write (save) and quit
- `:q` - quit (fails if anything has changed)
- `:q!` - quit and throw away changes

Search/Replace

- `/pattern` - search for pattern
- `?pattern` - search backward for pattern
- `n` - repeat search in same direction
- `N` - repeat search in opposite direction
- `:%s/old/new/g` - replace all old with new throughout file (gn is better though)
- `:%s/old/new/gc` - replace all old with new throughout file with confirmations

Working with multiple files

- `:e filename` - Edit a file
- `:tabe` - make a new tab
- `gt` - go to the next tab
- `gT` - go to the previous tab
- Advanced
  - `:vsp` - vertically split windows
  - `ctrl+ws` - Split windows horizontally
  - `ctrl+wv` - Split windows vertically
  - `ctrl+ww` - switch between windows
  - `ctrl+wq` - Quit a window

Marks

Marks allow you to jump to designated points in your code.

- `m{a-z}` - Set mark {a-z} at cursor position
- A capital mark {A-Z} sets a global mark and will work between files
- `'{a-z}` - move the cursor to the start of the line where the mark was set
- `''` - go back to the previous jump location

General

- `u` - undo
- `Ctrl+r` - redo
- `.` - repeat last command

Making Vim actually useful

Vim is quite unpleasant out of the box. For example, typeing `:w` for every file save is awkward and copying and pasting to the system clipboard does not work. But a few changes will get you much closer to the editor of your dreams.

.vimrc

- My .vimrc file has some pretty great ideas I haven't seen elsewhere.
- This is a minimal vimrc that focuses on three priorities:
  - adding options that are strictly better (like more information showing in autocomplete)
  - more convenient keystrokes (like `[space]w` for write, instead of `:w [enter]`)
  - a similar workflow to normal text editors (like enabling the mouse)

Installation

- Copy this to your home directory and restart vim. Read through it to see what you can now do (like `[space]w` to save a file)
  - mac users - making a hidden normal file is suprisingly tricky. Here’s one way:
    - in the command line, go to the home directory
    - type `nano .vimrc`
    - paste in the contents of the .vimrc file
    - `ctrl+x`, `y`, `[enter]` to save
- You should now be able to press `[space]w` in normal mode to save a file.
- `[space]p` should paste from the system clipboard (outside of vim).
  - If you can’t paste, it’s probably because vim was not built with the system clipboard option. To check, run `vim --version` and see if `+clipboard` exists. If it says `-clipboard`, you will not be able to copy

from outside of vim.
  - For mac users, homebrew install vim with the clipboard option. Install homebrew and then run `brew install vim`.
    - then move the old vim binary: `$ mv /usr/bin/vim /usr/bin/vimold`
    - restart your terminal and you should see `vim --version` now with `+clipboard`

Plugins

- The easiest way to make vim more powerful is to use Vintageous in sublime (version 3). This gives you Vim mode inside sublime. I suggest this (or a similar setup with the Atom editor) if you aren't a vim master. Check out Advanced Vim if you are.

- Vintageous is great, but I suggest you change a few settings to make it better.

  - Clone this repository to `~/.config/sublime-text-3/Packages/Vintageous`, or similar. Then check out the "custom" branch.
    - Alternatively, you can get a more updated Vintageous version by cloning the official repo and then copying over this patch.
  - Change the user settings (`User/Preferences.sublime-settings`) to include:
    - `"caret_style": "solid"`
    - This will make the cursor not blink, like in vim.
    - sublime might freeze when you do this. It's a bug; just restart sublime after changing the file.
  - `ctrl+r` in vim means "redo". But there is a handy ctrl+r shortcut in sublime that gives an "outline" of a file. I remapped it to alt+r by putting this in the User keymap
    - `{ "keys": ["alt+r"], "command": "show_overlay", "args": {"overlay": "goto", "text": "@"} },`
  - Add the ability to toggle vintageous on and off
  - Mac users: you will not have the ability to hold down a navigation key (like holding j to go down). To fix this, run the commands specified here: https://gist.github.com/kconragan/2510186

- Now you should be able to restart sublime and have a great vim environment! Sweet Dude.

Switch Caps Lock and Escape

- I highly recommend you switch the mapping of your caps lock and escape keys. You'll love it, promise! Switching the two keys is platform dependent; google should get you the answer

Other

I don't personally use these yet, but I've heard other people do!

- `:wqa` - Write and quit all open tabs (thanks Brian Zick)