VIM AND VIGOR

WHAT IS VIM?

Vim is a text editor, like notepad or Visual Studio Code. Vim stands for Vi Improved because it is an improved version of the Vi editor.

WHY LEARN VIM?

- You might get trapped in it someday.
- Available by default on Mac and most Linux distros.
- Incredibly lightweight; will run on your toaster.
- Powerful keybindings that will be useful for a lifetime.

COMMON PLACES VIM SHOWS UP

- `git commit`
- Using Linux, especially without a GUI
  - SSH to a Linux server
- Nearly every major editor via a Vim mode or plugin.
- Random sites include things like `hjkl`.
  - YouTube
  - Gmail
  - DuckDuckGo
  - GitHub

WHAT PEOPLE THINK OF VIM

Image credit: https://stackoverflow.blog/2017/05/23/stack-overflow-helping-one-million-developers-exit-vim/

REALITY

REALITY

You can exit Vim in 3 easy steps.

REALITY

You can exit Vim in 3 easy steps.

1. Don't panic.
2. Press Esc (if necessary)
3. Type `:q` or `:q!` plus enter
4. (Optional) Consider reconfiguring your default editor so this doesn't happen in the future.

VIM IN 4 STEPS

Vim is a **modal** text editor.

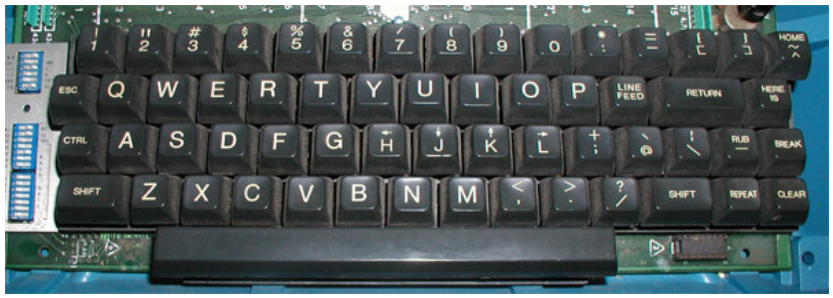
1. Press `i` to enter Insert Mode.
2. Make all the edits you want.
3. Type `Esc` to exit insert mode.
4. Type `:wq` to **write** changes and **quit** the program.

GETTING VIM

Vim is open source and widely available.

- Bundled with GitBash for Windows.
- Included with MacOS.
- Probably included with your favorite Linux distribution, definitely in repositories.
- You can get a Vim extension for your editor, but the quality of these vary.

REMAPPING ESCAPE



Vim uses escape often as a way to switch modes. It is recommended to remap your caps lock to escape for reachability.

Image credit: https://superuser.com/questions/599150/why-arrow-keys-are-not-recommended-in-vim

GROKKING VIM

***grok**: to understand profoundly and intuitively*

There are two ideas you must accept to grok Vim.

1. Vim is modal.
2. Vim is a language.

Definition from Mirriam-Webster

VIM IS MODAL.

A keystroke can mean different things depending on the mode.

VIM IN 4 STEPS

(Begin in Normal Mode)

1. Press `i` to enter Insert Mode. (Normal => Insert)
2. Make all the edits you want. (Insert Mode)
3. Type `Esc` to exit insert mode. (Insert => Normal)
4. Type `:wq` to **write** changes and **quit** the program. (Normal => Command => Exit)

MAIN MODES

- Normal Mode: default mode with helpful commands to navigate the file
- Insert Mode: typing mode; most like a regular editor
- Command Mode: full of useful utility commands like search and replace
- Visual Mode: for selecting text with a visual highlight

VIM IS A LANGUAGE

There are nouns, verbs, and adjectives.

COMMON MOTIONS (NOUNS)

- `h`/`j`/`k`/`l`: left/down/up/right
- `w`/`W`: forward one word/WORD
- `e`/`E`: forward to the end of a word/WORD
- `b`/`B`: back one word/WORD
- `0`: beginning of line
- `$`: end of line
- `%`: to matching parenthesis, bracket, brace, or tag
- `(`/`)`|`{`/`}`: back/forward one sentence|paragraph
- `f`/`F`|`t`/`T`: forward/backward including|excluding next character

COMMON COMMANDS (VERBS)

- `(none)`: move
- `d`: delete (cut)
- `c`: change (delete and to Insert Mode)
- `y`: yank (copy)
- `p`/`P`: paste before/after cursor
- `r`: replace

OTHER VERBS AND SHORTHAND

- `u`: undo (unlimited if configured)
- `i`/`I`: insert - Insert Mode at cursor/at beginning of line
- `a`/`A`: append - Insert Mode after cursor/at end of line
- `o`/`O`: open - Insert Mode on a new line below/above

MORE COMMON COMMANDS

- `dd/yy/cc/Y`: delete/copy/change current line
- `D/C`: delete/change until end of current line
- `gg`: beginning of file
- `G`: end of file
- `~`: switch case of letter under cursor
- `r<X>`: replace letter under cursor with X
- `x`: cut letter under cursor

COUNTS (ADJECTIVES)

A number or count can be used to modify a verb or noun.

- `c2w` == "change 2 words"
- `2dd` == "delete 2 lines"
- `5p` == "paste 5 times"
- `2fa` == "find 2nd a"

THE .

. repeats the last change.

SEARCHING

- `/`: search forwards
- `?`: search backwards
- `n`/`N`: next/previous search

INDENTATION

- =: correct indentation command
- >: more indentation command
- <: less indentation command

REGISTERS

Vim has more than 3 dozen registers or clipboards. Every single letter, number, and most punctuation characters are a clipboard.

REGISTERS (CONT)

- `""`: default register
- `"ayW`: yank a WORD into the a register
- `"+p`: paste from the system clipboard
- `"*yl`: yank a character into the X clipboard (Linux)
- `"0`: last yanked register
- Numerical registers (`"1`, `"2`, ...) are a history of things deleted.
- `"%` always has the file name

MACROS

Macros are an extremely powerful tool to repeat actions several times.

- `qq`: record macro into register q
- `q` (while recording): quit recording
- `@q`: run macro in register q
- `@@`: run last run macro

TERMINAL INTERACTION

- `!<shell command>`: run command in terminal
- `%` in command mode refers to the current filename
- Eg: `!python %`: run current file through Python
- `:r<command>`: read command output into buffer
- Eg: `:r!date`: read date from `date` command into buffer

MARKS

Marks are a way to "mark" a spot in a file for later use as a motion.

- `m<lowercase letter>`: set per file mark
- `m<UPPERCASE letter>`: set global mark
- `'<mark>`: jump to line of mark
- `` `<mark> ``: jump to exact mark

CONTROL COMMANDS

The `c-` prefix indicates you should use type control and the indicated key.

- `c-e`/`c-y`: move up/down without moving cursor
- `c-n`/`c-p`: next/previous keyword completion

GETTING HELP

Forgot what a key does in Vim? Don't worry, Vim has you covered!

The `:h <topic/key/command>` command will open Vim's built in manual.

`vimtutor`

The `vimtutor` command (run in terminal) will allow you to access Vim's built in tutorial.

AND MORE!

Vim is highly, highly configurable, and the plugin community is active. And of course, there are even more commands (literally every key is mapped to something in Normal Mode).