
Vim Workshop
Callum Howard
^H

What is a Text Editor?

What is a Text Editor

On lab computers (available in exams)


Nano


Gedit


Kwrite


Vim


Emacs

Other popular editors


Sublime Text 3


Atom.io


VS Code


Notepad++


(Notepad)


(TextEdit)

} Windows

Integrated Development Environment (IDE)


Xcode

} OSX


Visual Studio

} Windows


Eclipse


Netbeans


Geany

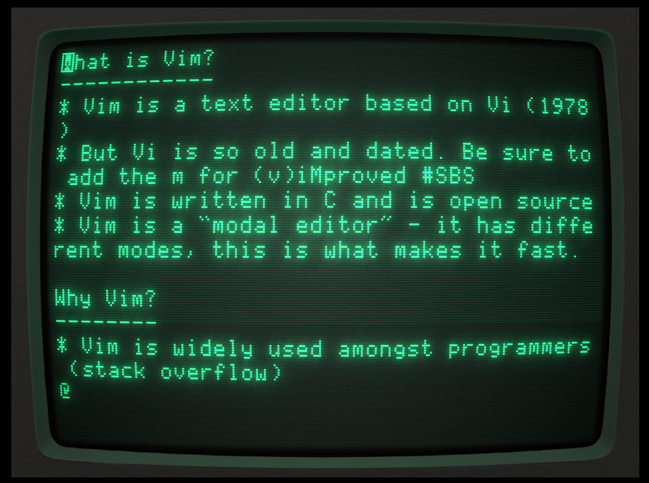
What is a Text Editor

Not Text Editors:

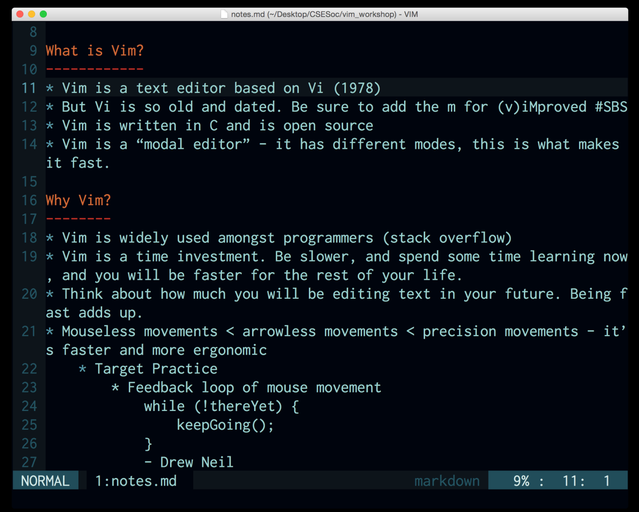

Vim

Vi vs. Vim vs. NeoVim

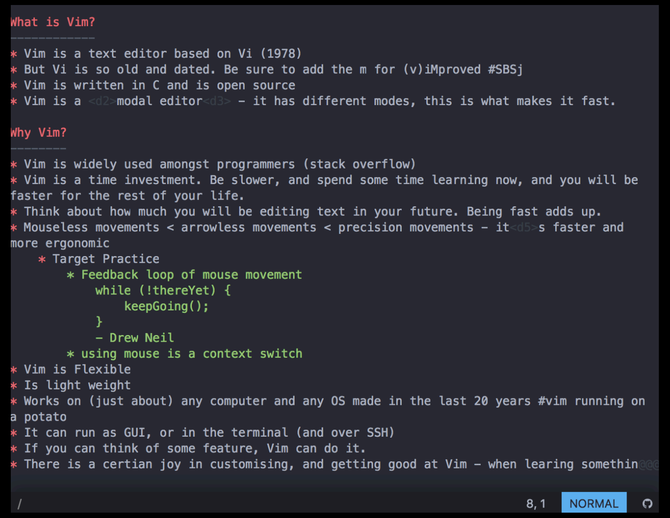
Vi (1978)



Vim



NeoVim



Vim is a ‘modal’ editor

Keys are used to move around and modify text

Type like you are used to, just like nano

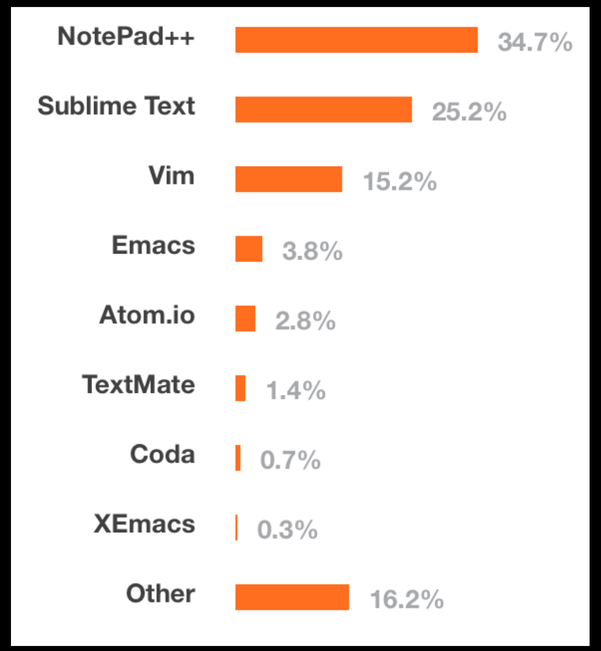
For selecting text

Mini command line for vim commands

But, Why Vim?

(what's so good about it anyway?)

It’s pretty popular



PREFERRED TEXT EDITOR BY OCCUPATION

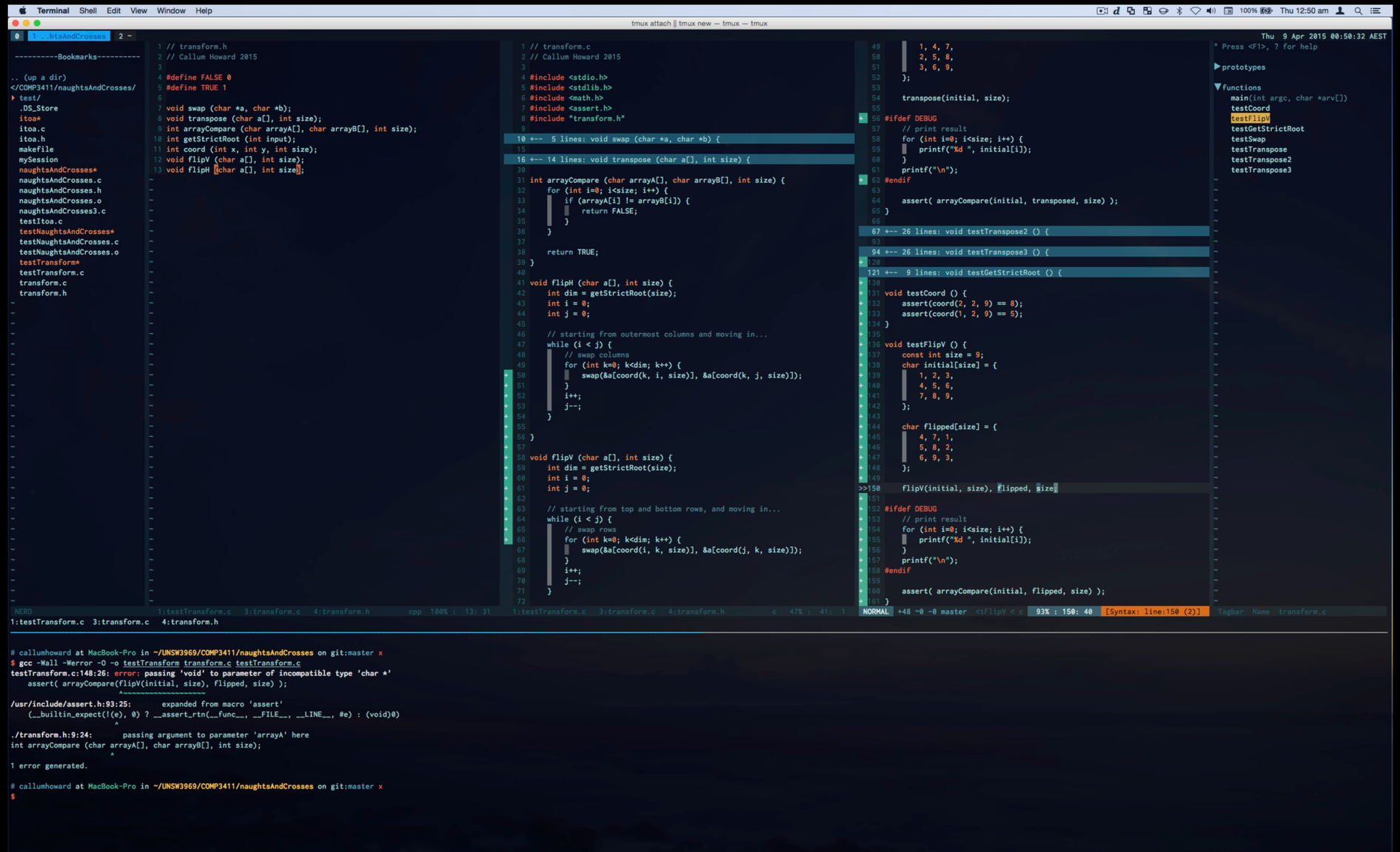
Occupation	Preferred text editor
Desktop Developers	NotePad++
Front-end web developers	Sublime Text
DevOps & Sys admins	Vim
Full-stack developers	Notepad++ and Sublime Text

http://stackoverflow.com/research/developer-survey-2015

Flexible


notes.md (~/Desktop...c/vim_workshop) - VIM
15
16 Why Vim?
17 --------
18 * Vim is widely used amongst programmers (stac#
19 * Vim is a time investment. Be slower, and spe#
20 * Think about how much you will be editing tex#
21 * Mouseless movements < arrowless movements < #
22 * Target Practice
23 * Feedback loop of mouse movement
24 while (!thereYet) {
25 keepGoing();
26 }
27 - Drew Neil
28 * using mouse is a context switch
29 * Vim is Flexible
30 * Is light weight
31 * Works on (just about) any computer and any O#
32 * It can run as GUI, or in the terminal (and o#
33 * If you can think of some feature, Vim can do#
34 * There is a certian joy in customising, and g#
35
36 Learning Vim
37 ------------
38 * Vimtutor
NORMAL 1:notes.md 25% : 29: 3
1:notes.md

Flexible


Terminal Shell Edit View Window Help
tmux attach || tmux new — tmux — tmux
Thu 9 Apr 2015 00:50:32 AEST
Bookmarks
.. (up a dir)
</COMP3411/naughtsAndCrosses/
test/
.DS_Store
itoa*
itoa.c
itoa.h
makefile
mySession
naughtsAndCrosses*
naughtsAndCrosses.c
naughtsAndCrosses.h
naughtsAndCrosses.o
naughtsAndCrosses3.c
testItoa.c
testNaughtsAndCrosses*
testNaughtsAndCrosses.c
testNaughtsAndCrosses.o
testTransform*
testTransform.c
transform.c
transform.h
// transform.h
// Callum Howard 2015
#define FALSE 0
#define TRUE 1
void swap (char *a, char *b);
void transpose (char a[], int size);
int arrayCompare (char arrayA[], char arrayB[], int size);
int getStrictRoot (int input);
int coord (int x, int y, int size);
void flipV (char a[], int size);
void flipH (char a[], int size);
// transform.c
// Callum Howard 2015
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <assert.h>
#include "transform.h"
+-- 5 lines: void swap (char *a, char *b) {
+-- 14 lines: void transpose (char a[], int size) {
int arrayCompare (char arrayA[], char arrayB[], int size) {
for (int i=0; i<size; i++) {
if (arrayA[i] != arrayB[i]) {
return FALSE;
}
}
return TRUE;
}
void flipH (char a[], int size) {
int dim = getStrictRoot(size);
int i = 0;
int j = 0;
// starting from outermost columns and moving in...
while (i < j) {
// swap columns
for (int k=0; k<dim; k++) {
swap(&a[coord(k, i, size)], &a[coord(k, j, size)]);
}
i++;
j--;
}
}
void flipV (char a[], int size) {
int dim = getStrictRoot(size);
int i = 0;
int j = 0;
// starting from top and bottom rows, and moving in...
while (i < j) {
// swap rows
for (int k=0; k<dim; k++) {
swap(&a[coord(i, k, size)], &a[coord(j, k, size)]);
}
i++;
j--;
}
}
1, 4, 7,
2, 5, 8,
3, 6, 9,
};
transpose(initial, size);
#ifdef DEBUG
// print result
for (int i=0; i<size; i++) {
printf("%d ", initial[i]);
}
printf("\n");
#endif
assert(arrayCompare(initial, transposed, size));
}
+-- 26 lines: void testTranspose2 () {
+-- 26 lines: void testTranspose3 () {
+-- 9 lines: void testGetStrictRoot () {
void testCoord () {
assert(coord(2, 2, 9) == 8);
assert(coord(1, 2, 9) == 5);
}
void testFlipV () {
const int size = 9;
char initial[size] = {
1, 2, 3,
4, 5, 6,
7, 8, 9,
};
char flipped[size] = {
4, 7, 1,
5, 8, 2,
6, 9, 3,
};
flipV(initial, size), flipped, size);
#ifdef DEBUG
// print result
for (int i=0; i<size; i++) {
printf("%d ", initial[i]);
}
printf("\n");
#endif
assert(arrayCompare(initial, flipped, size));
}
" Press <F1>, ? for help
prototypes
functions
main(int argc, char *arv[])
testCoord
testFlipV
testGetStrictRoot
testSwap
testTranspose
testTranspose2
testTranspose3
NERD 1:testTransform.c 3:transform.c 4:transform.h cpp 100% : 13: 31
1:testTransform.c 3:transform.c 4:transform.h c 47% : 41: 1
NORMAL +48 ~0 -0 master <tFlipV < c 93% : 150: 40 [Syntax: line:150 (2)] Tagbar Name transform.c
1:testTransform.c 3:transform.c 4:transform.h
# callumhoward at MacBook-Pro in ~/UNSW3969/COMP3411/naughtsAndCrosses on git:master x
$ gcc -Wall -Werror -O -o testTransform transform.c testTransform.c
testTransform.c:148:26: error: passing 'void' to parameter of incompatible type 'char *'
assert(arrayCompare(flipV(initial, size), flipped, size));
^~~~~~~~~~~~~~~~~
/usr/include/assert.h:93:25: expanded from macro 'assert'
(__builtin_expect(!(e), 0) ? __assert_rtn(__func__, __FILE__, __LINE__, #e) : (void)0)
^
./transform.h:9:24: passing argument to parameter 'arrayA' here
int arrayCompare (char arrayA[], char arrayB[], int size);
^
1 error generated.
callumhoward at MacBook-Pro in ~/UNSW3969/COMP3411/naughtsAndCrosses on git:master x
$

Light weight

21.3 MB on my computer
(before plugins)

Will run on anything

GUI

MacVim

gVim (Linux, Windows)

Vim key bindings in other editors

SSH


tmux attach || tmux new — tmux — tmux
0 1 notes.md 2 frequency.c
Mon 13 Apr 2015 01:19:49 AEST
# callumhoward at MacBook-Pro in ~
$ ssh cse
# callumh at wagner in ~
# callumh at wagner in ~
$ cd AWESOMEFOLDER/
# callumh at wagner in ~/AWESOMEFOLDER
# callumh at wagner in ~/AWESOMEFOLDER
$
19 int main (int argc, char **argv) {
20 // declare an array to store the frequency
21 int frequency[MAX_ASCII];
22
23 // make sure the array is empty by initialising to 0
24 int i = 0;
25 while (i < MAX_ASCII) {
26 frequency[i] = 0;
27 i++;
28 }
29
30 // read characters in from STDIN (standard input) until Ctrl-d
NORMAL 1:frequency.c main() < c utf-8[unix] 36% : 22: 0

```
while (!thereYet) {
    keepGoing();
}
```

- Drew Neil, “Precision Editing at the Speed of Thought”

But can Vim do *XYZ*?

…yes, yes it can.

Vim is a time investment

Editing in Vim is FAST, but it takes time to learn

How to get good at Vim

How to get good at Vim

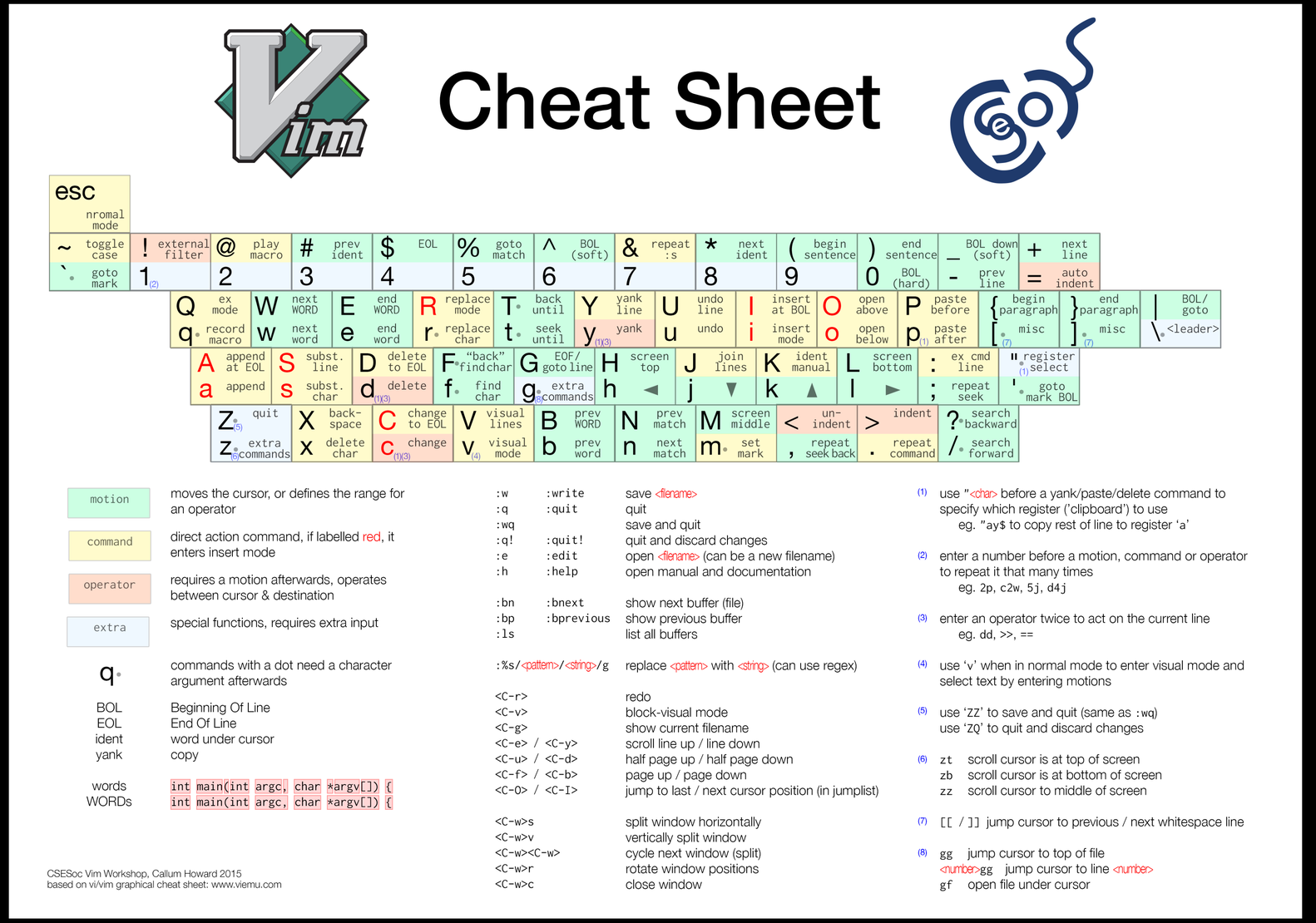
- Vimtutor
  - installed with vim
  - just type `vimtutor` at the terminal
- VimCasts.org
- Vim :help - built in help/documentation
- Google / StackOverflow

How to get good at Vim

- Practice!
- Don’t stop learning new things,
- Seek a faster way

Let’s Learn some Vim!

http://callumh.web.cse.unsw.edu.au/vimcheatsheet.pdf



motion — moves the cursor, or defines the range for an operator

command — direct action command, if labelled red, it enters insert mode

operator — requires a motion afterwards, operates between cursor & destination

extra — special functions, requires extra input

q• — commands with a dot need a character argument afterwards

BOL — Beginning Of Line
EOL — End Of Line
ident — word under cursor
yank — copy

words
WORDs

```
int main(int argc, char *argv[]) {
int main(int argc, char *argv[]) {
```

:w	:write	save <filename>
:q	:quit	quit
:wq		save and quit
:q!	:quit!	quit and discard changes
:e	:edit	open <filename> (can be a new filename)
:h	:help	open manual and documentation
:bn	:bnext	show next buffer (file)
:bp	:bprevious	show previous buffer
:ls		list all buffers
:%s/<pattern>/<string>/g		replace <pattern> with <string> (can use regex)
<C-r>		redo
<C-v>		block-visual mode
<C-g>		show current filename
<C-e> / <C-y>		scroll line up / line down
<C-u> / <C-d>		half page up / half page down
<C-f> / <C-b>		page up / page down
<C-O> / <C-I>		jump to last / next cursor position (in jumplist)
<C-w>s		split window horizontally
<C-w>v		vertically split window
<C-w><C-w>		cycle next window (split)
<C-w>r		rotate window positions
<C-w>c		close window

(1) use "<char> before a yank/paste/delete command to specify which register ('clipboard') to use
eg. "ay$ to copy rest of line to register 'a'

(2) enter a number before a motion, command or operator to repeat it that many times
eg. 2p, c2w, 5j, d4j

(3) enter an operator twice to act on the current line
eg. dd, >>, ==

(4) use 'v' when in normal mode to enter visual mode and select text by entering motions

(5) use 'ZZ' to save and quit (same as :wq)
use 'ZQ' to quit and discard changes

(6) zt scroll cursor is at top of screen
zb scroll cursor is at bottom of screen
zz scroll cursor to middle of screen

(7) [[/]] jump cursor to previous / next whitespace line

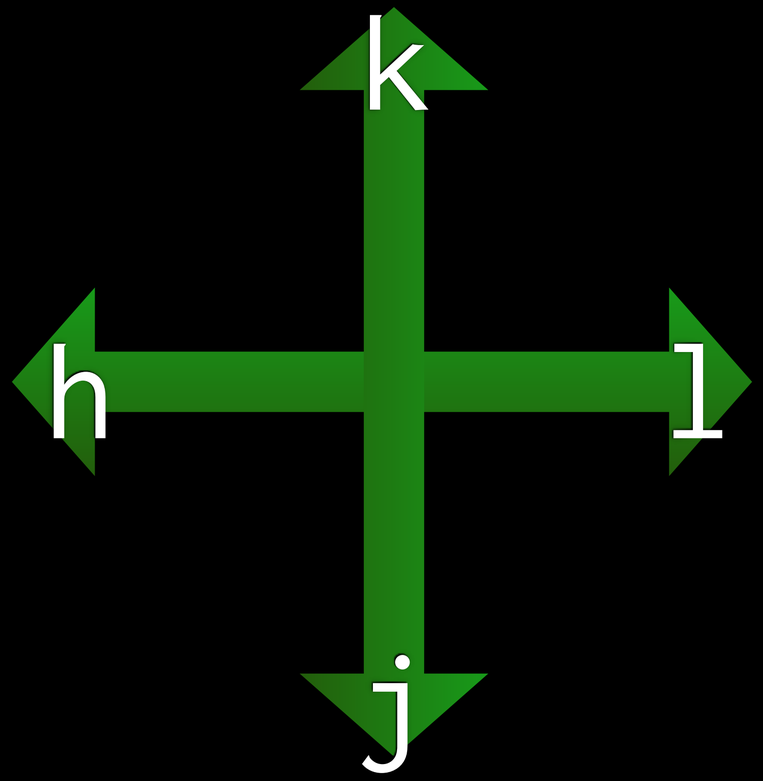
(8) gg jump cursor to top of file
<number>gg jump cursor to line <number>
gf open file under cursor

CSESoc Vim Workshop, Callum Howard 2015
based on vi/vim graphical cheat sheet: www.viemu.com

Vim can be frustrating at first,
but persevere! Soon enough
it will be muscle memory

http://callumh.web.cse.unsw.edu.au/vimcheatsheet.pdf

Movement



http://callumh.web.cse.unsw.edu.au/vimcheatsheet.pdf

Exiting Vim

`:w`	`:write`	save <filename>
`:q`	`:quit`	quit
`:wq`		save and quit
`:q!`	`:quit!`	quit and discard changes
`:e`	`:edit`	open <filename> (can be a new filename)
`:h`	`:help`	open manual and documentation

use ‘`ZZ`’ to save and quit (same as `:wq`)
use ‘`ZQ`’ to quit and discard changes

x to Delete

it’s like crossing it out

Inserting (with `i`)

NORMAL

Keys are used to move around and modify text

INSERT

Type like you are used to, just like nano

VISUAL

For selecting text

Mini command line for vim commands

i to insert before cursor

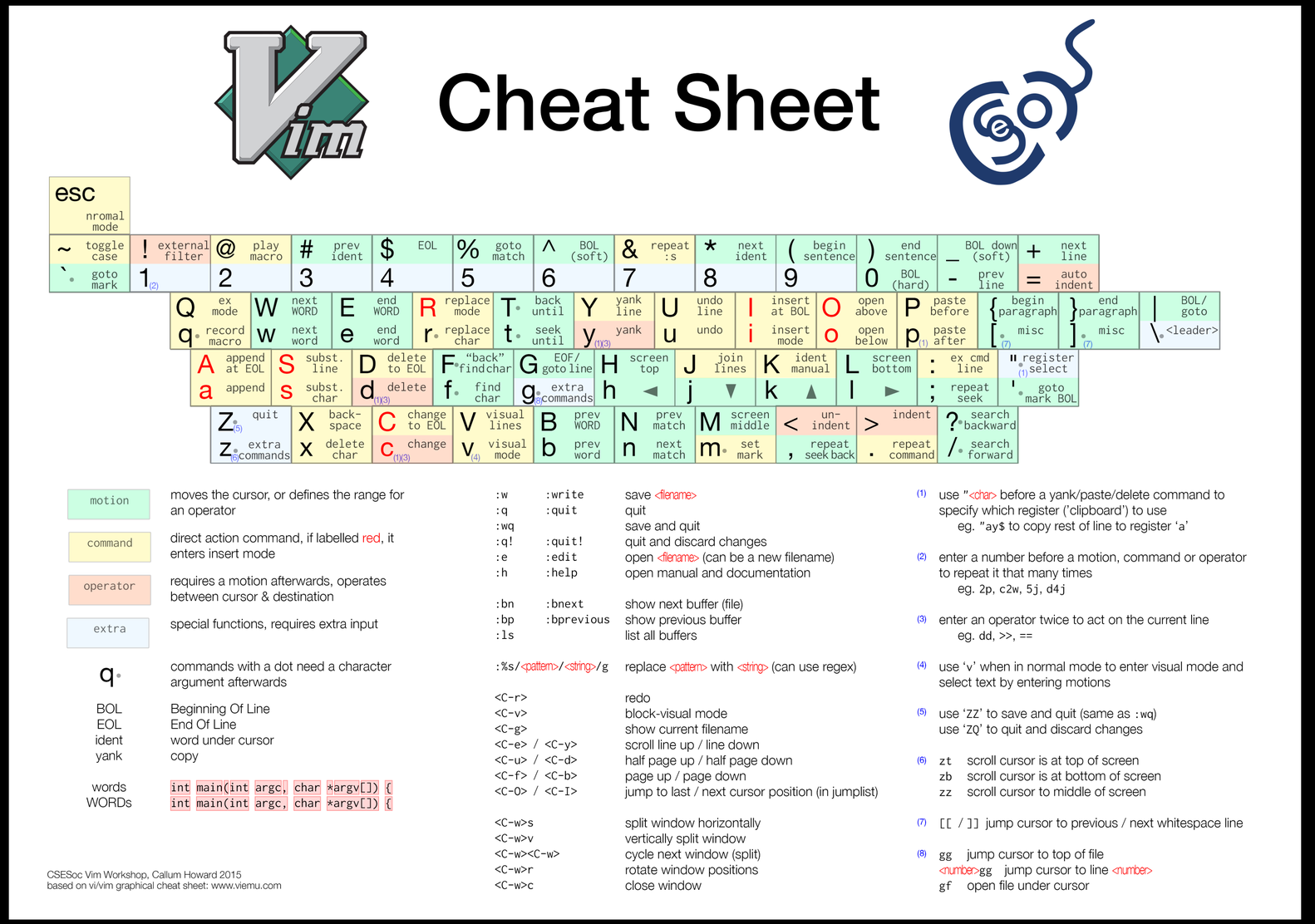
a to insert after cursor

I to insert before line

A to append at end of line

Emacs users hate him, you won’t believe these 6 AMAZING productivity tips… more

Memorise using the mnemonics



motion — moves the cursor, or defines the range for an operator

command — direct action command, if labelled red, it enters insert mode

operator — requires a motion afterwards, operates between cursor & destination

extra — special functions, requires extra input

q• — commands with a dot need a character argument afterwards

BOL — Beginning Of Line
EOL — End Of Line
ident — word under cursor
yank — copy

words
WORDs

```
int main(int argc, char *argv[]) {
int main(int argc, char *argv[]) {
```

`:w`	`:write`	save <filename>
`:q`	`:quit`	quit
`:wq`		save and quit
`:q!`	`:quit!`	quit and discard changes
`:e`	`:edit`	open <filename> (can be a new filename)
`:h`	`:help`	open manual and documentation
`:bn`	`:bnext`	show next buffer (file)
`:bp`	`:bprevious`	show previous buffer
`:ls`		list all buffers
`:%s/<pattern>/<string>/g`		replace <pattern> with <string> (can use regex)
`<C-r>`		redo
`<C-v>`		block-visual mode
`<C-g>`		show current filename
`<C-e>` / `<C-y>`		scroll line up / line down
`<C-u>` / `<C-d>`		half page up / half page down
`<C-f>` / `<C-b>`		page up / page down
`<C-O>` / `<C-I>`		jump to last / next cursor position (in jumplist)
`<C-w>s`		split window horizontally
`<C-w>v`		vertically split window
`<C-w><C-w>`		cycle next window (split)
`<C-w>r`		rotate window positions
`<C-w>c`		close window

(1) use "<char> before a yank/paste/delete command to specify which register ('clipboard') to use
eg. "ay$ to copy rest of line to register 'a'

(2) enter a number before a motion, command or operator to repeat it that many times
eg. 2p, c2w, 5j, d4j

(3) enter an operator twice to act on the current line
eg. dd, >>, ==

(4) use 'v' when in normal mode to enter visual mode and select text by entering motions

(5) use 'ZZ' to save and quit (same as :wq)
use 'ZQ' to quit and discard changes

(6) zt scroll cursor is at top of screen
zb scroll cursor is at bottom of screen
zz scroll cursor to middle of screen

(7) [[/]] jump cursor to previous / next whitespace line

(8) gg jump cursor to top of file
<number>gg jump cursor to line <number>
gf open file under cursor

CSESoc Vim Workshop, Callum Howard 2015
based on vi/vim graphical cheat sheet: www.viemu.com

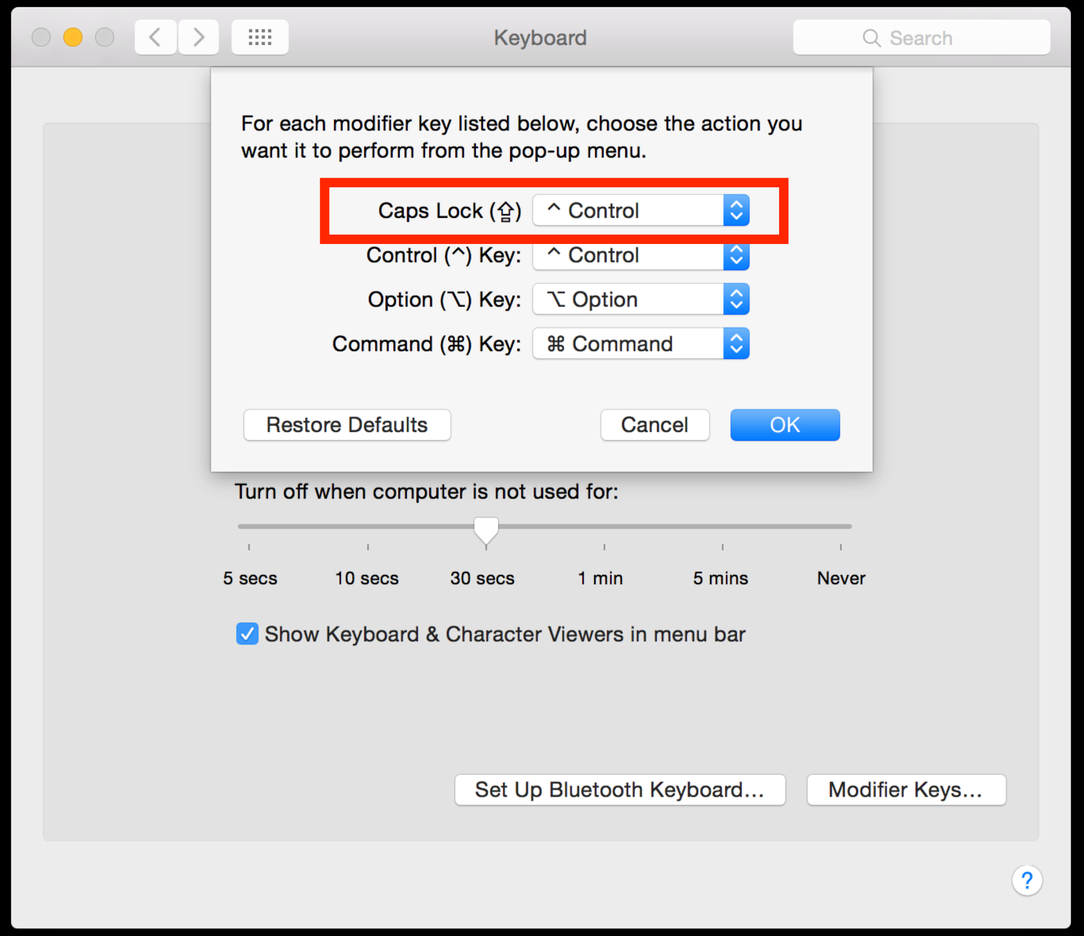
ProTip: Touch-Type

ProTip:

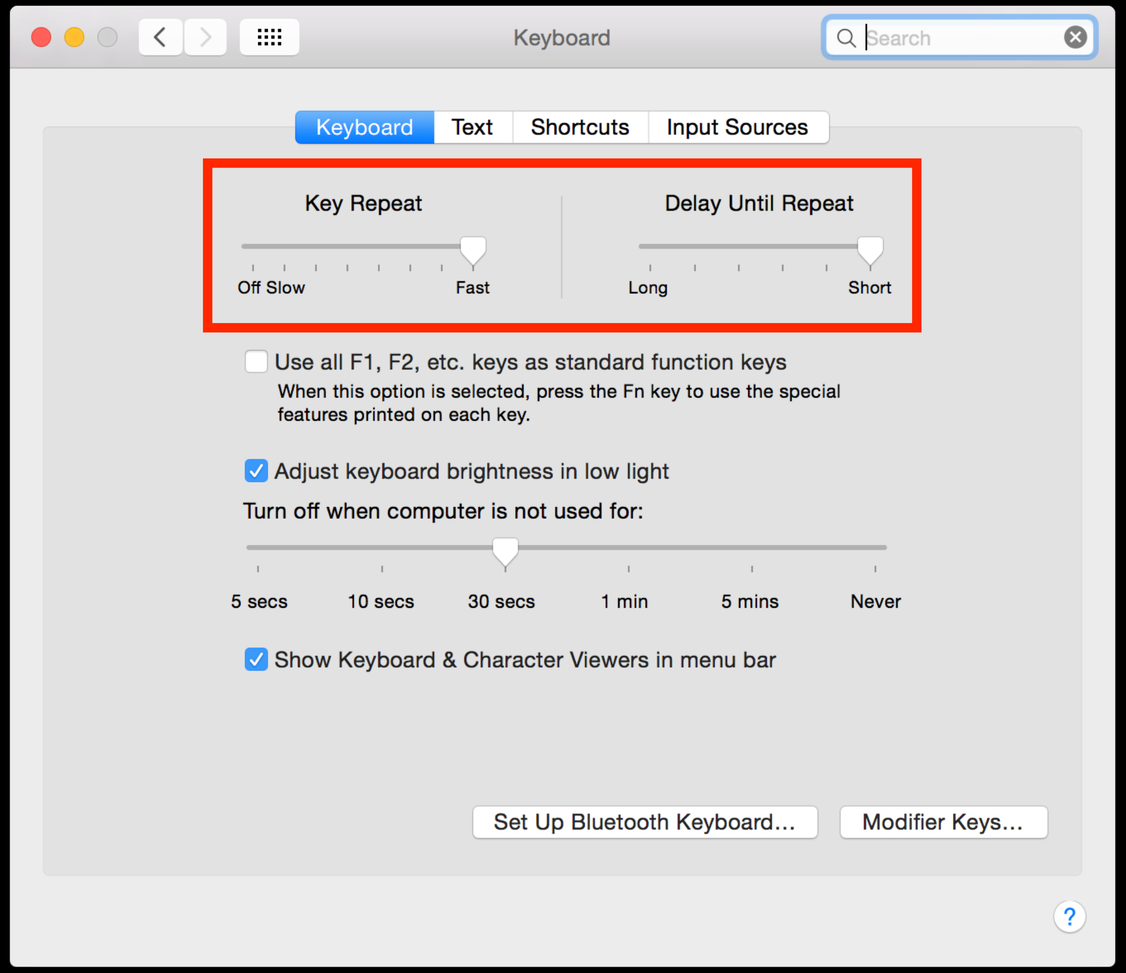
`:imap kj <ESC>`

Got to first `:set nocp`

Protip:
Remap caps-lock to CTRL



Protip:
speed up key-repeat



ProTip:
Disable arrow keys

```
map <Up> <NOP>
map <Down> <NOP>
map <Left> <NOP>
map <Right> <NOP>
```

put this in your .vimrc

gVim vs Terminal Vim

gVim

+ Smoother Scrolling
+ Dynamic cursor (| in insert mode, ▉ in normal mode)
+ Can display Millions of colours
- Has toolbar that you should turn off!

Terminal Vim

+ Access to the Command Line <C-z>
+ 256 colours
+ Tmux
+ SSH

Plugins or Vanilla Vim

Plugins

+ Super Powerful
+ Super Customisable
+ Super
+ Can slow down Vim

Vanilla Vim

+ Can use in an exam
+ Can use another person's account/computer without your customisations
+ Can use over SSH