Vim the ancient UNIX editor

Miguel Gonzalez – VFUK Digital

The plan for today is...

- Modes
- Open/Save
- Basic movements and commands
- Repetitions
- Advanced movements
- Searching
- Autocompletion
- Commands
- Visual mode
- UNIX commands

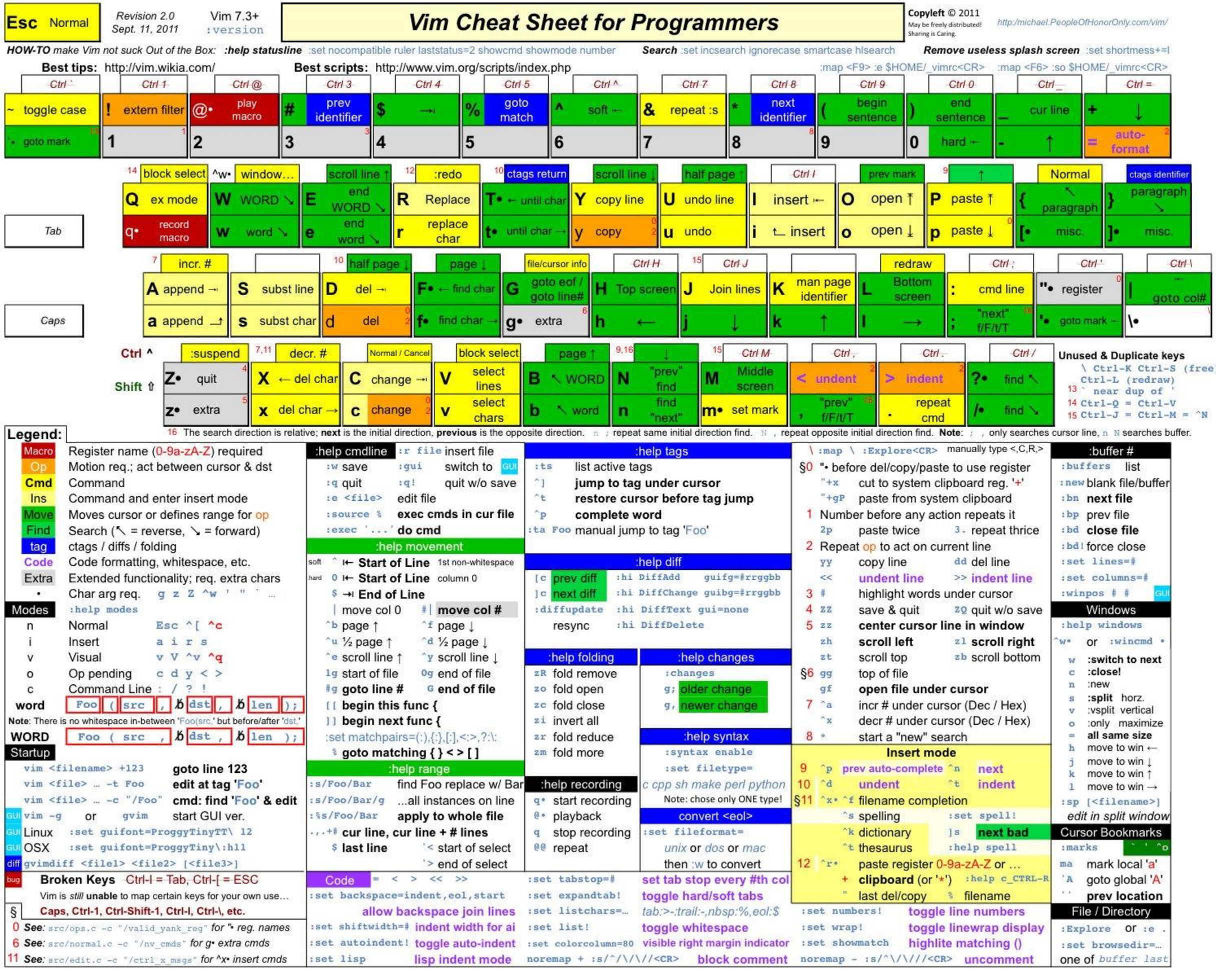
Setup and requisites

- Open a terminal in a UNIX environment
  - MacOS
  - Linux
  - Windows Subsystem for Linux
- Checkout this Git repo
  - https://github.com/ktzar/vim_workshop.git
- Type "vi" and press enter

Task

Enter Vim

Cheatsheet



Esc Normal

Revision 2.0
Sept. 11, 2011

Vim 7.3+
:version

Vim Cheat Sheet for Programmers

Copyleft © 2011
May be freely distributed!
Sharing is Caring.

http://michael.PeopleOfHonorOnly.com/vim/

HOW-TO make Vim not suck Out of the Box: ***:help statusline*** :set nocompatible ruler laststatus=2 showcmd showmode number ***Search*** :set incsearch ignorecase smartcase hlsearch ***Remove useless splash screen*** :set shortmess+=I

Best tips: http://vim.wikia.com/ **Best scripts:** http://www.vim.org/scripts/index.php :map <F9> :e $HOME/_vimrc<CR> :map <F6> :so $HOME/_vimrc<CR>

~~Ctrl `~~ ~ toggle case · `• goto mark [13]
~~Ctrl 1~~ ! extern filter · 1 [1]
~~Ctrl @~~ @• play macro · 2
~~Ctrl 3~~ # prev identifier · 3 [3]
~~Ctrl 4~~ $ →| · 4
~~Ctrl 5~~ % goto match · 5
~~Ctrl ^~~ ^ soft |← · 6
~~Ctrl 7~~ & repeat :s · 7
~~Ctrl 8~~ * next identifier · 8 [8]
~~Ctrl 9~~ (begin sentence · 9
~~Ctrl 0~~) end sentence · 0 hard |←
~~Ctrl _~~ _ cur line · - ↑
~~Ctrl =~~ + ↓ · = auto-format [2]

Tab
[14] block select: Q ex mode · q• record macro
^w• window…: W WORD ↘ · w word ↘
scroll line ↑: E end WORD ↘ · e end word ↘
[12] :redo: R Replace · r replace char
[10] ctags return: T• ← until char · t• until char →
scroll line ↓: Y copy line · y copy [0] [2]
half page ↑: U undo line · u undo
~~Ctrl I~~ I insert |← · i ⌙ insert
prev mark: O open ⤒ · o open ⤓
[9] ↑: P paste ⤒ · p paste ⤓ [0]
Normal: { ↖ paragraph · [• misc.
ctags identifier: } paragraph ↘ ·]• misc.

Caps
[7] incr. #: A append → · a append ⮥
S subst line · s subst char
[10] half page ↓: D del →| · d del [0] [2]
page ↓: F• ← find char · f• find char →
file/cursor info: G goto eof / goto line# · g• extra [6]
~~Ctrl H~~ H Top screen · h ←
[15] ~~Ctrl J~~ J Join lines · j ↓
K man page identifier · k ↑
redraw: L Bottom screen · l →
~~Ctrl ;~~ : cmd line · ; "next" f/F/t/T [16]
~~Ctrl '~~ "• register [0] · '• goto mark |←
~~Ctrl \~~ | |← goto col# · \•

Ctrl ^
Shift ⇧
:suspend: Z• quit [4] · z• extra [5]
[7,11] decr. #: X ← del char · x del char →
Normal / Cancel: C change →| · c change [0] [2]
block select: V select lines · v select chars
page ↑: B ↖ WORD · b ↖ word
[9,16] ↓: N "prev" find · n find "next"
[15] ~~Ctrl M~~ M Middle screen · m• set mark
~~Ctrl ,~~ < undent [2] · , "prev" f/F/t/T [16]
~~Ctrl .~~ > indent [2] · . repeat cmd
~~Ctrl /~~ ?• find ↖ · /• find ↘

Unused & Duplicate keys
\ Ctrl-K Ctrl-S (free)
Ctrl-L (redraw)
13 ` near dup of '
14 Ctrl-Q = Ctrl-V
15 Ctrl-J = Ctrl-M = ^N

16 The search direction is relative; **next** is the initial direction, **previous** is the opposite direction. n ; repeat same initial direction find. N , repeat opposite initial direction find. **Note**: ; , only searches cursor line, n N searches buffer.

Legend:

Macro	Register name (0-9a-zA-Z) required
Op	Motion req.; act between cursor & dst
Cmd	Command
Ins	Command and enter insert mode
Move	Moves cursor or defines range for op
Find	Search (↖ = reverse, ↘ = forward)
tag	ctags / diffs / folding
Code	Code formatting, whitespace, etc.
Extra	Extended functionality; req. extra chars
•	Char arg req. g z Z ^w ' " ` …

Modes :help modes

n	Normal	Esc ^[^c
i	Insert	a i r s
v	Visual	v V ^v ^q
o	Op pending	c d y < >
c	Command Line	: / ? !

word Foo (src , ␢ dst , ␢ len);
Note: There is no whitespace in-between 'Foo(src,' but before/after 'dst,'
WORD Foo (src , ␢ dst , ␢ len);

Startup

vim <filename> +123	goto line 123
vim <file> … -t Foo	edit at tag 'Foo'
vim <file> … -c "/Foo"	cmd: find 'Foo' & edit
GUI vim -g or gvim	start GUI ver.
GUI Linux	:set guifont=ProggyTinyTT\ 12
GUI OSX	:set guifont=ProggyTiny\:h11
diff gvimdiff <file1> <file2> [<file3>]	

bug **Broken Keys** ~~Ctrl-I = Tab, Ctrl-[= ESC~~
Vim is *still* **unable** to map certain keys for your own use…
Caps, Ctrl-1, Ctrl-Shift-1, Ctrl-I, Ctrl-\, etc.

§
0 *See:* src/ops.c -c "/valid_yank_reg" *for "• reg. names*
6 *See:* src/normal.c -c "/nv_cmds" *for g• extra cmds*
11 *See:* src/edit.c -c "/ctrl_x_msgs" *for ^x• insert cmds*

:help cmdline

:r file	insert file		
:w	save	:gui	switch to GUI
:q	quit	:q!	quit w/o save
:e <file>	edit file		
:source %	**exec cmds in cur file**		
:exec '…'	**do cmd**		

:help movement

soft	^	⇤ **Start of Line**	1st non-whitespace	
hard	0	⇤ **Start of Line**	column 0	
	$	⇥ **End of Line**		
	|	move col 0	#|	**move col #**
	^b	page ↑	^f	page ↓
	^u	½ page ↑	^d	½ page ↓
	^e	scroll line ↑	^y	scroll line ↓
	1g	start of file	0g	end of file
	#g	**goto line #**	G	**end of file**
	[[	**begin this func {**		
	]]	**begin next func {**		

:set matchpairs=(:),{:},[:],<:>,?:\:
% **goto matching { } < > [ ]**

:help range

:s/Foo/Bar	find Foo replace w/ Bar
:s/Foo/Bar/g	…all instances on line
:%s/Foo/Bar	**apply to whole file**
. , .+#	**cur line, cur line + # lines**
$	**last line**
'<	start of select
'>	end of select

Code = < > << >>

:set backspace=indent,eol,start	allow backspace join lines
:set shiftwidth=#	indent width for ai
:set autoindent!	toggle auto-indent
:set lisp	lisp indent mode

:help tags

:ts	list active tags
^]	**jump to tag under cursor**
^t	**restore cursor before tag jump**
^p	**complete word**
:ta Foo	manual jump to tag 'Foo'

:help diff

[c prev diff	:hi DiffAdd	guifg=#rrggbb
]c next diff	:hi DiffChange	guibg=#rrggbb
:diffupdate	:hi DiffText	gui=none
resync	:hi DiffDelete	

:help folding

zR	fold remove
zo	fold open
zc	fold close
zi	invert all
zr	fold reduce
zm	fold more

:help changes
:changes
g; older change
g, newer change

:help syntax
:syntax enable
:set filetype=
c cpp sh make perl python
Note: chose only ONE type!

:help recording

q•	start recording
@•	playback
q	stop recording
@@	repeat

convert <eol>
:set fileformat=
unix or *dos* or *mac*
then :w to convert

:set tabstop=#	set tab stop every #th col
:set expandtab!	toggle hard/soft tabs
:set listchars=…	*tab:>-,trail:-,nbsp:%,eol:$*
:set list!	toggle whitespace
:set colorcolumn=80	visible right margin indicator
noremap + :s/^/\/\//<CR>	block comment

\ :map \ :Explore<CR> manually type <,C,R,>

§0	"• before del/copy/paste to use register			
	"+x	cut to system clipboard reg. '+'		
	"+gP	paste from system clipboard		
1	Number before any action repeats it			
	2p	paste twice	3.	repeat thrice
2	Repeat op to act on current line			
	yy	copy line	dd	del line
	<<	undent line	>>	indent line
3	#	highlight words under cursor		
4	ZZ	save & quit	ZQ	quit w/o save
5	zz	**center cursor line in window**		
	zh	**scroll left**	zl	**scroll right**
	zt	scroll top	zb	scroll bottom
§6	gg	top of file		
	gf	**open file under cursor**		
7	^a	incr # under cursor (Dec / Hex)		
	^x	decr # under cursor (Dec / Hex)		
8	*	start a "new" search		

Insert mode

9	^p	prev auto-complete	^n	next
10	^d	undent	^t	indent
§11	^x• ^f	filename completion		
	^s	spelling	:set spell!	
	^k	dictionary	]s	next bad
	^t	thesaurus	:help spell	
12	^r•	paste register 0-9a-zA-Z or …		
	+	**clipboard** (or '*')	:help c_CTRL-R	
	"	last del/copy	%	filename

:set numbers!	toggle line numbers
:set wrap!	toggle linewrap display
:set showmatch	highlite matching ()
noremap - :s/^\/\/\//<CR>	uncomment

:buffer #

:buffers	list
:new	blank file/buffer
:bn	**next file**
:bp	prev file
:bd	**close file**
:bd!	force close
:set lines=#	
:set columns=#	
:winpos # #	GUI

Windows
:help windows
^w• or :wincmd •

w	**:switch to next**
c	**:close!**
n	:new
s	**:split** horz.
v	:vsplit vertical
o	:only maximize
=	**all same size**
h	move to win ←
j	move to win ↓
k	move to win ↑
l	move to win →

:sp [<filename>]
edit in split window

Cursor Bookmarks
:marks ` ' ^o

ma	mark local 'a'
'A	goto global 'A'
''	**prev location**

File / Directory
:Explore or :e .
:set browsedir=…
one of *buffer last*

Modes

01

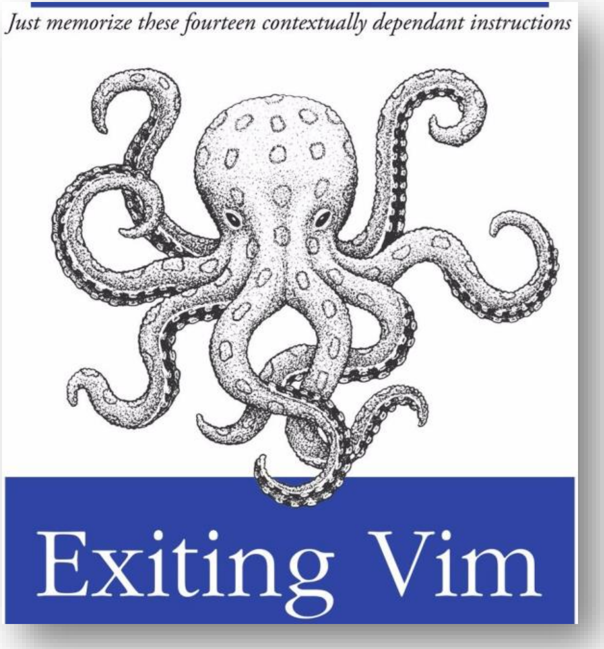
Vim is a modal editor

- Insert mode
- Command mode
- Visual mode

- I'll use Key for new keys
  - Esc Move from insert mode into command mode
  - i Go to insert mode from command mode

Task

Leave Vim, open it with shakespeare.txt file opened. Get into insert mode and add your name to the top.


Just memorize these fourteen contextually dependant instructions
Exiting Vim

Opening and saving

- These commands only work in "command mode"
- They start with colon (:)

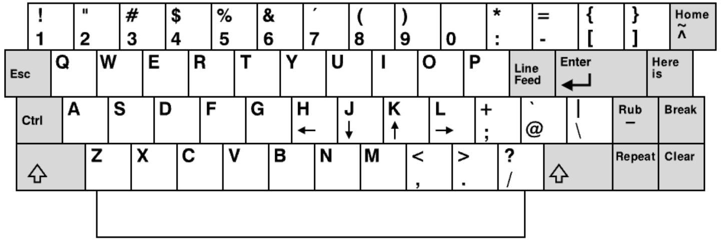
- New commands
  - :w
  - :o
  - :w+
  - :o {filename}
  - zz
- Now you're proficient with Vim as you probably are with Nano

Task

Save the file you've just changed. Exit Vim, start it and open the file with ":o". Open a new file within Vim.

Basic movements

- In Vim, once you're in command mode, you can move around the current opened file with movement commands.
- You can move using the arrows, but that's not the most efficient:
- Movements
  - j
  - k
  - l
  - h



- Jumps
  - w beginning of next Word
  - e end of next word
  - b back

- Same-line searches. Multi-character commands
  - f find
  - t 'til

Task

Move down 10 lines

Move to the 3rd word

Go back a few words

Go to the next comma

Advanced ways to get into insert mode

- There's some commands to move and get into insert mode at the same time

- Moving faster
  - a **a**dd
  - s **s**ubstitute the current character
  - o **o**pen a line

Task

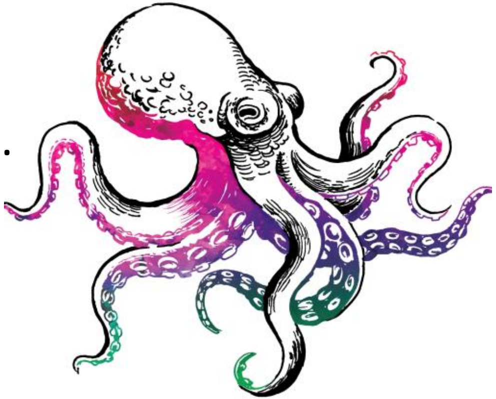
Open react.js and add a comment above line 6 with "o"

Composing commands

02

Composed commands

- In Vim, there's commands that expect, after them, a movement. This composes a "language" where sentences are:
  - Command + Movement



Task

Swap two contiguous words.

Delete from a word until the end of the sentence.

- Commands
  - `d` delete
  - `c` cut
  - `y` yank (copy)

What do these commands do?

`dw` `dj` `dl` `dh` `cw` `ct"` `ct.` `cek` `ctk`

- Pasting
  - `p` paste
    - **IMPORTANT**. Every time you delete something, it goes to the clipboard

Applying commands to a line

- Most of the movement commands we've seen can be applied to a full line by duplicating the command:

- Commands
  - dd delete the current line
  - yy yank the current line (what does yyp do?)
  - y yank (copy)

Task

Open react.js and add a comment above line 6 with "o"

Insert // at the beginning of a line, starting from somewhere in the middle of the line

- Also, you can use UPPERcase to take the current line as if it was a character
  - D delete line
  - S substitute
  - O open before line
  - J join

What do these commands do?

yyp cc

What's the difference between D and S?

Repeating a command

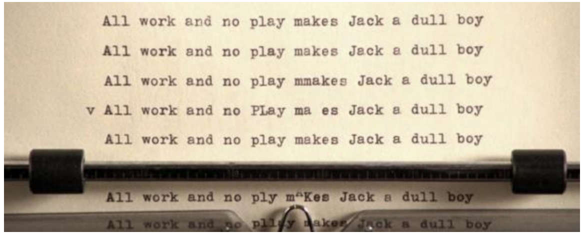
- Vim's "language" also includes a modifier to the "command + movement" we've seen:
    - Multiplier + command + movement

What do these commands do?
4dd 5cw 10j 3yyp

Task

Remove 5 lines
Go 3 words to the right
Go 10 lines up
Insert // at the beginning of a line, go a few lines later, and repeat that command

- Commands
    - . repeat what you just did



All work and no play makes Jack a dull boy
All work and no play makes Jack a dull boy
All work and no play mmakes Jack a dull boy
v All work and no PLay ma es Jack a dull boy
All work and no play makes Jack a dull boy
All work and no ply m^Kes Jack a dull boy
All work and no pllay makes Jack a dull boy

Advanced movements

03

Inner movements

- When you are inside a text object, you can apply moveme that work in both directions, until they reach the "edges o the object". These are **w**ords, **p**aragraphs, [ ], ( ), t (for tag
- The modifier key you apply here is "i" for "inner", since yo inside the object. Then, your "vim sentence" becomes:
  - Command + i + object

Task

Open costume.go and remove all the contents of "Costume" struct

Go to line 37 and replace the params with "param1, param2"

Open react.js and remove the contents of the th tag in line 15

What do these commands do?

ci{ di[ci[diw dit

Moving quickly

- Moving in a file is what we do the most often. 90% of the time we're navigating a source file and 10% we're writing code.

- Commands
  - `gg` go to the top of the file
  - `G` go to the bottom of the file
  - `H` go to the top of the screen (high, or "vim left")
  - `M` go to the middle of the screen
  - `L` go to the bottom of the screen (low, or "vim right")
  - `:` go to a line number
  - `C-u` scroll Up
  - `C-d` scroll Down

Task

Open shakespeare.txt and scroll down a couple of screens.

Go to line 15

Copy and paste the whole contents of the file, from top to bottom

Searching

- Searching is usually the fastest way to move in a file

- Search commands
  - `/` go to the next occurrence of a text
  - `?` go to the previous occurrence of a text
  - `n` go to the next occurrence of the current search
  - `N` go to the previous occurrence of the current search
  - `*` search the word under the cursor and go to the next

Task

Open shakespeare.txt and search for "shall". Jump to the next "shall"

Open costume.go, move to the Costume definition, and search using * where it's used. Cycle through the results.

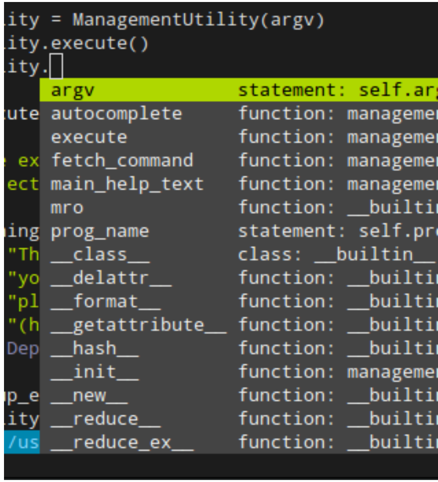
More commands

04

Undo and redo, autocomplete

- You undo commands, not individual character changes. This is super powerful
  - New commands
    - `u` undo
    - `C-R` (Control + R) Redo

- Also, you can autocomplete what you're typing (using words from the same file)
  - `C-P` looks for words previously written
  - `C-N` looks for words from the current position to the end



```
ity = ManagementUtility(argv)
ity.execute()
ity.
    argv               statement: self.ar
ute autocomplete       function: manageme
    execute            function: manageme
 ex fetch_command      function: manageme
ect main_help_text     function: manageme
    mro                function: __builti
ing prog_name          statement: self.pr
"Th __class__          class: __builtin__
"yo __delattr__        function: __builti
"pl __format__         function: __builti
"(h __getattribute__   function: __builti
Dep __hash__           function: __builti
    __init__           function: manageme
p_e __new__            function: __builti
ity __reduce__         function: __builti
/us __reduce_ex__      function: __builti
```

Task

Remove a paragraph. Undo it.

In Costume.go, type anywhere "fmt.P" and then try C-P and C-N

Indentation

- Indenting
  - `>` indent to the right
  - `<` indent to the left
  - `=` autoindent (doesn't work if Vim doesn't know which file type you have loaded)

Task

Fully unindent a section of typescript.ts and then autoindent it.

Now ty the same with pygame.py. You'll have to indent it manually

Visual mode

- You can select a block of text to do actions on it. You get into visual mode with "V", and leave with "Esc"
  - V enter visual mode for full lines
  - v enter visual mode for characters

Once in, you use all the normal movements you've already learnt, and then perform a command. Your "vim language" changes a bit, for example V3jd would be equivalent to 3dd.

What does this commands do?
V3j2wd

Task

Enter Visual mode for characters and move a couple words. Yank and paste somewhere else.

Enter visual mode for lines and select a function, delete it.

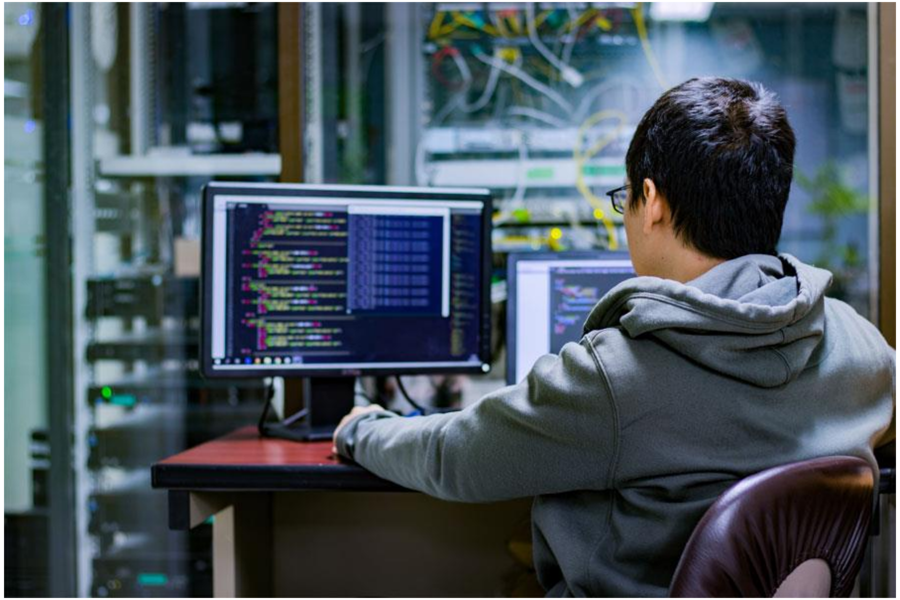
Enter visual mode and do a inner movement

Next steps in your Vim journey

04

More things

- Running UNIX commands
  - ! (using % as a variable)
    - Listing files
    - Filtering the current file through a UNIX command
- Customising VIM
  - You can have your own mappings
  - Store your config in .vimrc
- Macros
  - By far the most powerful feature of Vim, in my opinion
- Registers
  - You don't only have one clipboard, you have 52, one per letter
  - Just prepend "<letter> before yanking or pasting



More things (ii)

- Window and file organisation
  - Buffers
  - Splits
- Undo history
- Plugins
  - Managed with Plug, and you can store them in your `.vimrc`
  - Most popular:
    - NERDTree
    - C-T
    - SnipMate
    - NERDCommenter
    - Matchit
    - FUF
- You can use "Vim mode" inside your favourite edito

What to learn next?

Try to start using Vim every now and then

Get used to the basics

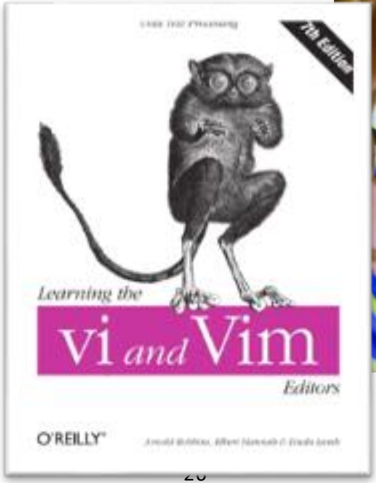
Every now and then question yourself:

- can I do X more efficiently?
- Research
- Make part of your routine

Use it in the command line

Materials for further learning

- Vim adventures
  - https://vim-adventures.com/
- Not a bad introduction:
  - http://mislav.uniqpath.com/2011/12/vim-revisited/
- Amazing official book. Only 576 pages:
  - https://www.iopb.res.in/vimbook-OPL.pdf
- Vim Tips Fandom
  - https://vim.fandom.com/wiki/Vim_Tips_Wiki



Thanks!!!

Questions?