



All About Vim!

Lecture 3 - Jules Yang, Kavya
Tummalapalli

Pet Tax

Richard "Richie" Parker



Charcoal



Noot Noot

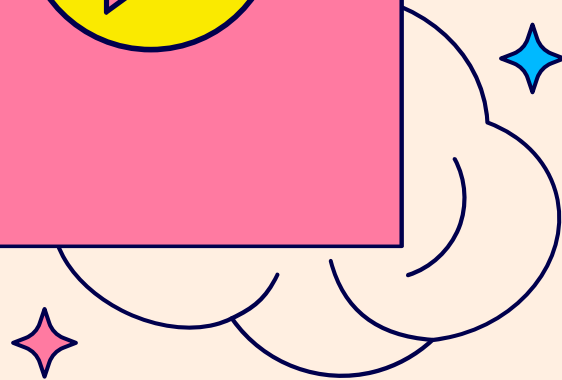
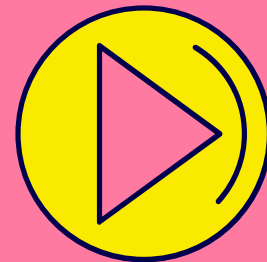


Pet Tax

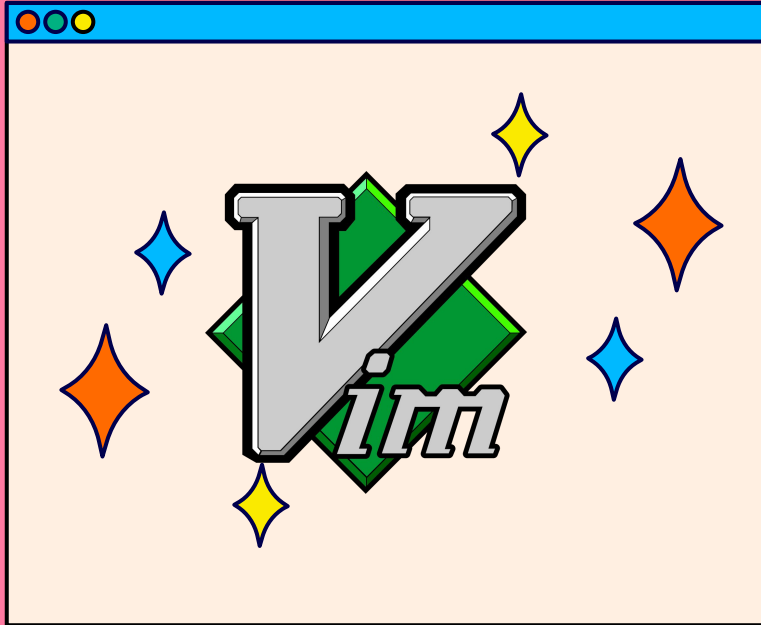




01
**What, Why,
How?**



What is Vim?



- A **screen-oriented** text editor
- Released in **1991**
 - Improved version of "vi" released in 1976
- A "**programmer's editor**"
- **No mouse or 'GUI'**



Why Vim?



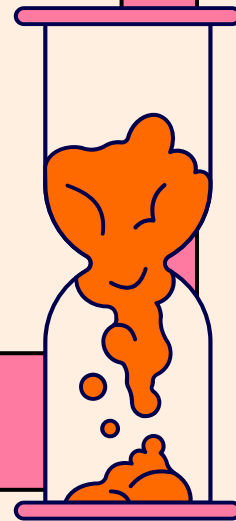
Efficiency

By only giving you the choice of using a keyboard, you can accomplish tasks much quickly. Customizable to your typing needs!



Ergonomics

Vim is designed to have your fingers always stay near the home row. This reduces hands fatigue and improves performance.





Why Vim?

It's Everywhere!

Vim exists on almost all machines. It's heavily used for system administration, programming, working with markup languages, and more!

Vim Keybindings

There are many music players, organizers, web browsers, file viewers, terminals, IDEs, and more that take advantage of Vim's keybindings.

```
1 #include <stdio>
2
3 using namespace std;
4
5 int main() {
6     int* primes = new int[ 1000 ];
7     int candidate;
8
9     for ( candidate = 3; candidate < count; candidate ) {
10    }
11    return 0;
12 }
```

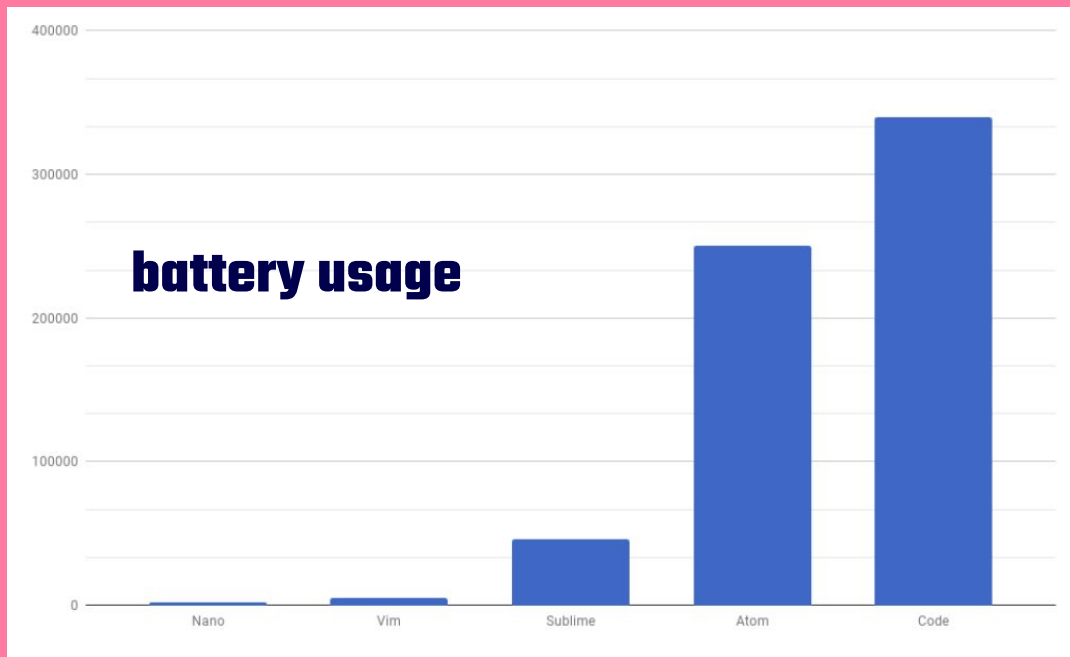
Old keyboard

SHIFT+4bbbxx

9,45

ALL

Why Vim?



Vim is Cool 🥰 👨💻 👩💻



How: The Vim Philosophy

Efficiency



Modal Editing

Think

How can I do what I want faster?
Don't try to memorize all these
commands, understand what you
need to do next.

Practice, Practice, Practice

The best and only way to learn
and gain familiarity! Mastery
takes time.

Teach Yourself!

Vim Wiki

Great online resource for
answering questions

Vimtutor

As easy as typing vimtutor
into the terminal

`:help <cmmd>`

Internal vim help pages for
a specific command

All Over the Internet

Many years of documentation
and resources



02

**The Modes of
Vim**





01 Normal mode (Esc)

Editor commands are used in this mode. The default mode.

02 Insert mode (i)

Editing a text buffer. Similar to modern text editors.

03 Visual mode (v)

Used for highlighting areas for text.

04 Command-line mode (:)

Single line input, used mostly for file navigation. Exits mode after command

05 Select mode (gh)

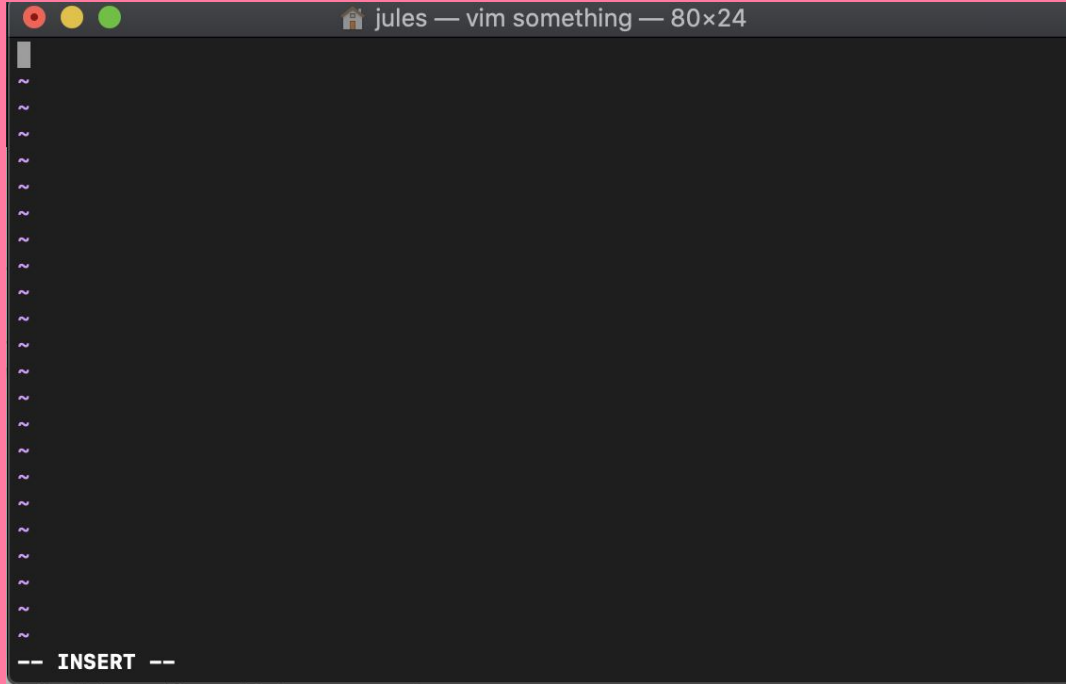
Printable characters replace text and Vim enters insert mode.

06 Ex mode (Q)

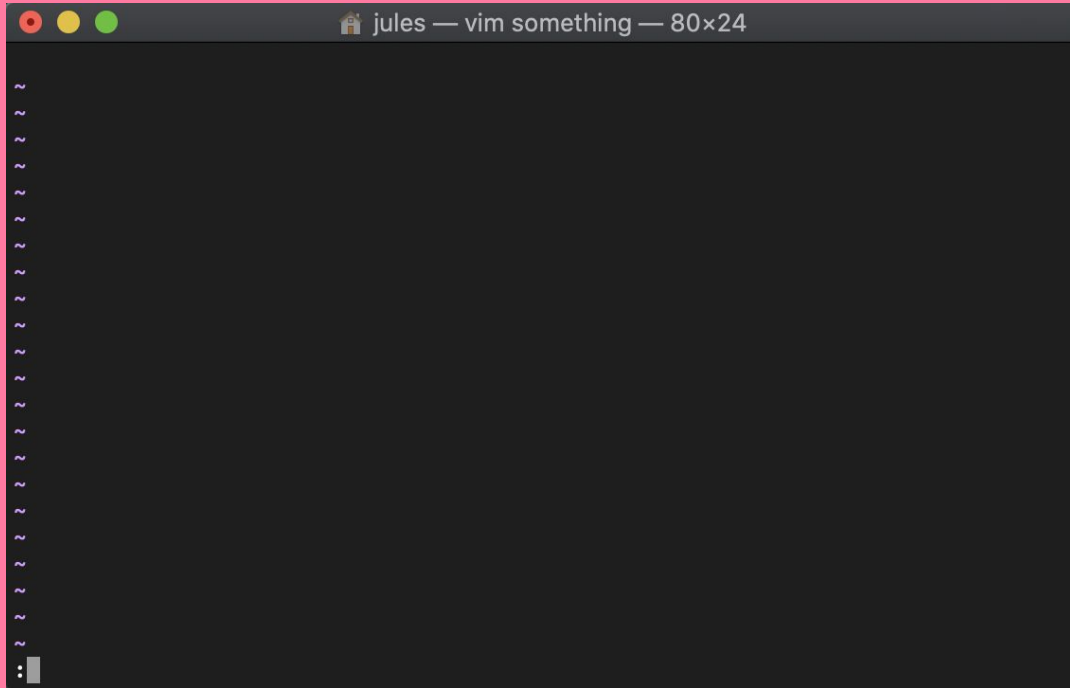
Similar to command-line mode, but does not exit mode after command.



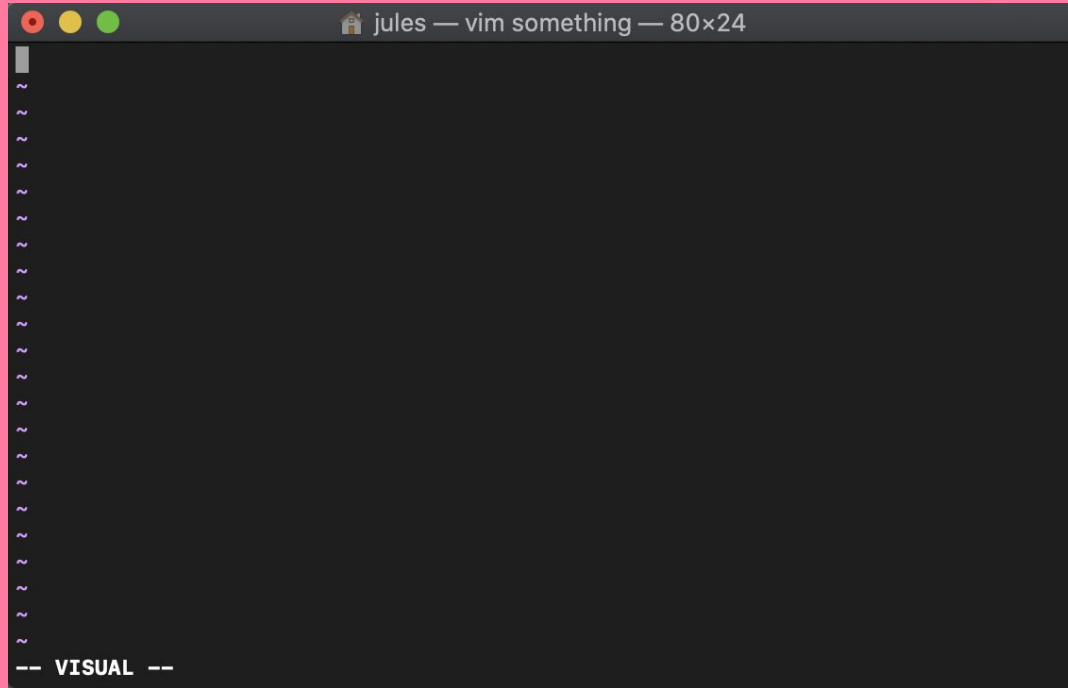
What Mode Am I In?



What Mode Am I In?



What Mode Am I In?



A terminal window titled "jules — vim something — 80x24" is shown. The terminal content consists of 18 lines of tilde characters (~) and the text "-- VISUAL --" at the bottom. The terminal window is set against a pink background.

```
jules — vim something — 80x24
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- VISUAL --
```



Opening/Creating a File in Vim

```
$ vim </path/filename>
```

↳ opens file in Normal Mode

:w

Save

:wq

Save and Quit

:q

Quit

:q!

Force Quit

Normal Mode: Navigating

So how do we move around without a mouse?

Ergonomically best to navigate letterwise:



Normal Mode: Navigating

Inline

- **f**ind <next occurrence of letter>
 - **F**ind <previous>
- **w** <next word>
 - **b** <back a word>
- **0** (zero) - beginning of line
- **\$** - end of line

Filewide

- **/<search text><enter>**
 - **?<back search text><enter>**
- **n**ext occurrence of search
 - **N** - previous
- **G**o to the end of file
 - **gg**o to beginning

The Parallel Shift

Normally, there is a pattern to Vim commands, but **THIS IS NOT ALWAYS TRUE**

t <char>	Look <i>forwards</i> for char exclusive	T <char>	Look <i>backwards</i> for char exclusive
f <char>	Look <i>forwards</i> for char inclusive	F <char>	Look <i>backwards</i> for char inclusive
o	Open new line <i>below</i>	O	Open new line <i>above</i>
i	Insert <i>before cursor</i>	I	Insert at <i>beginning of line</i>
a	Insert <i>after cursor</i>	A	Insert at <i>end of line</i>

back

O
line

^
non-blank

Fx
find x
,
previous x

Tx
after x

b
word
B
delimited
word

ge
end
gE
delimited
end

h
left

k
up

l
right

e
end

w
word

tx
before x

fx
find x

\$
line

forward

previous

gg

first line

?text

find text

#
find word
under cursor

N
previous text

C-b

page

C-u

1/2 page

H

screen

{

paragraph

(

sentence

k

up

j

down

E

delimited
end

w

delimited word

;

next x

)

sentence

}

paragraph

L

screen

C-d

1/2 page

C-f

page

/text

find text

n

next text

find word
under cursor

G

last line

next

absolute movements

' '

last location

'.

last edit

#G

line #

%

matching
bracket





Normal Mode: Editing



<COUNT> <VERB> <TEXT OBJECT>

2 **d**delete **w**word

This command would delete two words from the current cursor position.

Vim has its own intuitive **“language”**. It has verbs and objects, and the commands are similar to English counterparts. You can create **“clauses”** to manipulate this text.

Normal Mode: Editing

- **y**ank (copy)
 - **Y**ank or **yy** a line
- **d**elete (cut)
 - **dd**elete (cut) line
- **p**aste below
 - **P**aste above
- **u**ndo
- **Ctrl-r**edo



c = copy

y = YEET

Switching Modes

Normal → Insert

- i** the typical way
- c** hange - deletes the following text object
- A** ppend - jumps to end of line
- o** pens a new line

Most Modes → Normal

Esc

Normal → Visual

- v** character mode
- V** line mode
- Ctrl + v** block mode

Command-line Mode Commands



How do you generate a random string?
Put a web designer in front of VIM
and tell him to save and exit.



I Am Developer
@iamdeveloper

Following

I've been using Vim for about 2 years now,
mostly because I can't figure out how to exit it.

RETWEETS
14,083

LIKES
8,154



3:26 PM - 17 Feb 2014



314



14K



8.2K



Command-line Mode Commands

/<pattern><Enter>

Jump to pattern instance

:<number>

Jump to line number

:+<number>

Jump ahead <number> of lines

:-<number>

Jump back <number> of lines

:w

Save

:wq

Save and Quit

:q

Quit

:<>!

Force

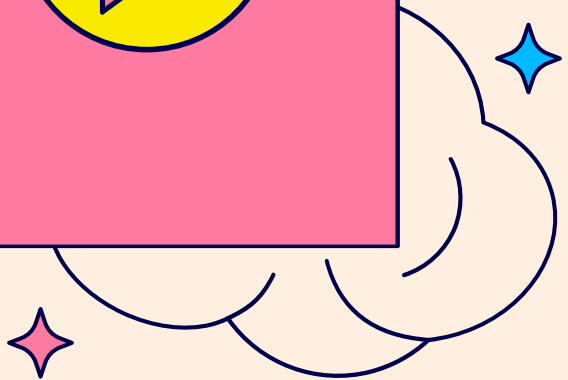
A Very Useful Command: Find and Replace (especially for the lab 🤖)

```
:<range>s:<find>:<replace>:<flags>
```

- **range**
 - **%** - through the whole file
 - **a,b** - lines a to b inclusive
- **flags**
 - **g** - flag replaces every instance in a single line
 - **c** - flag asks for confirmation before each change



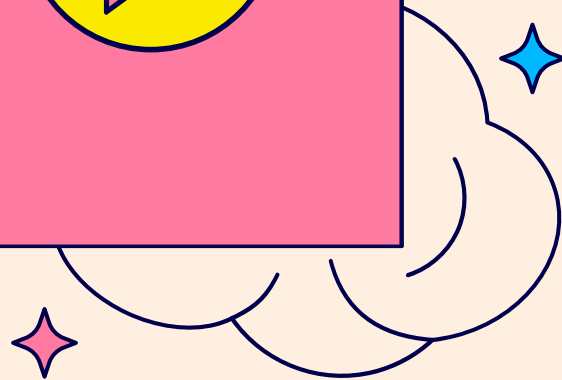
A large pink rectangular box with a blue header bar is the central focus. Inside the box, the text "03" is written in a bold, dark blue font at the top left. Below it, the text "Demo Time" is written in a larger, bold, dark blue font. To the right of the text is a yellow circular play button icon with a black triangle pointing to the right. The box has a thin blue border and a slight shadow effect.





04

Go Forth



Closing Vim Thoughts

Further Resources/Reading

- vimtutor
 - Walks you through a tutorial of vim
- *Practical Vim: Edit Text at the Speed of Thought* by Drew Neil

How to Help Yourself

- Practice, practice, practice!
- Google is your friend!
- [Vim Cheatsheet](#)
- Vim Wiki!
- :help <topic>

A Whole New World

- Registers
- Macros
- Interfacing with STDIN/STDOUT
- Customizing Vim to be an IDE
 - Vim plugins
- Code completion, folding, markers, etc.
- Using Vim in other editors



Announcements!

1. Extratation on VSCode **this Saturday 09/18 1-2** in **GHC 4211**
2. **Feedback Form:**

Lecture pacing too slow/fast? Are there broken links or typos? Let us know through the feedback form!

Link will also be available on the course website


Form:

tinyurl.com/f21-gpi-feedback



LPT: Lab Pro Tips



- 
- If you get an error message that says something like “Vundle” just press enter to continue
 - If you need line numbers to display, use `:set number`
 - If you really need to use your mouse, use `:set mouse=a`
 - If you get a “merge commit” screen on pull, then type `“:wq!”` and press enter to exit vim and complete the merge
 - Vim itself is used for other command tools such as Git
 - The last instruction on sliding should be (I always practice in grassy fields).
 - “Written by yours truly” (no quotes) should both be directly ABOVE and BELOW those lines
 - If you screw up on a file, ``git checkout <filename>`` to reset