Announcements

- Office Hours are posted on **Piazza**!
- trainerlab due date has been extended to **next Thursday 09/17**
  - This week's lab, sportslab, is still due **next Thursday 09/17**
- **Extratations** are starting this week
  - Details will be announced on Piazza soon

All About Vim!

Lecture 2 - Jules Yang

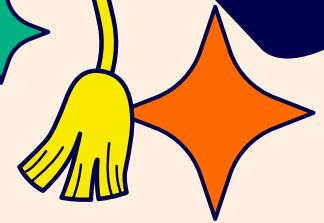
Pet Tax

Richard "Richie" Parker

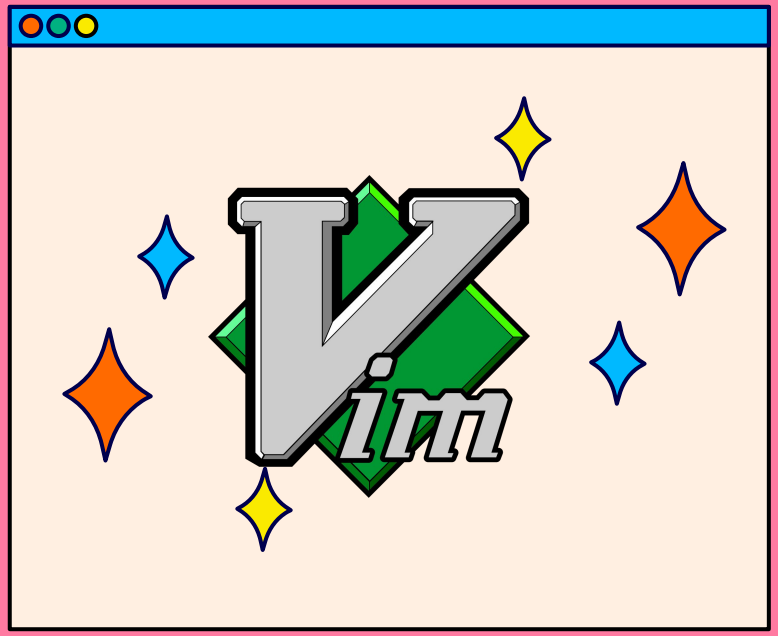
Charcoal

Noot Noot

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’**

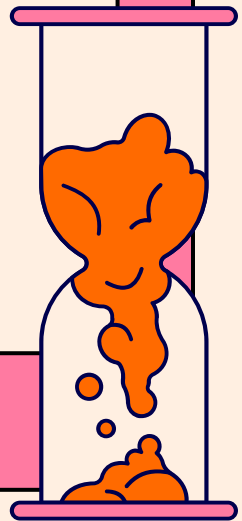
Why Vim?

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.



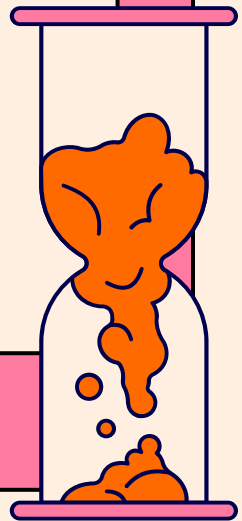
Why Vim?

It's Everywhere!

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

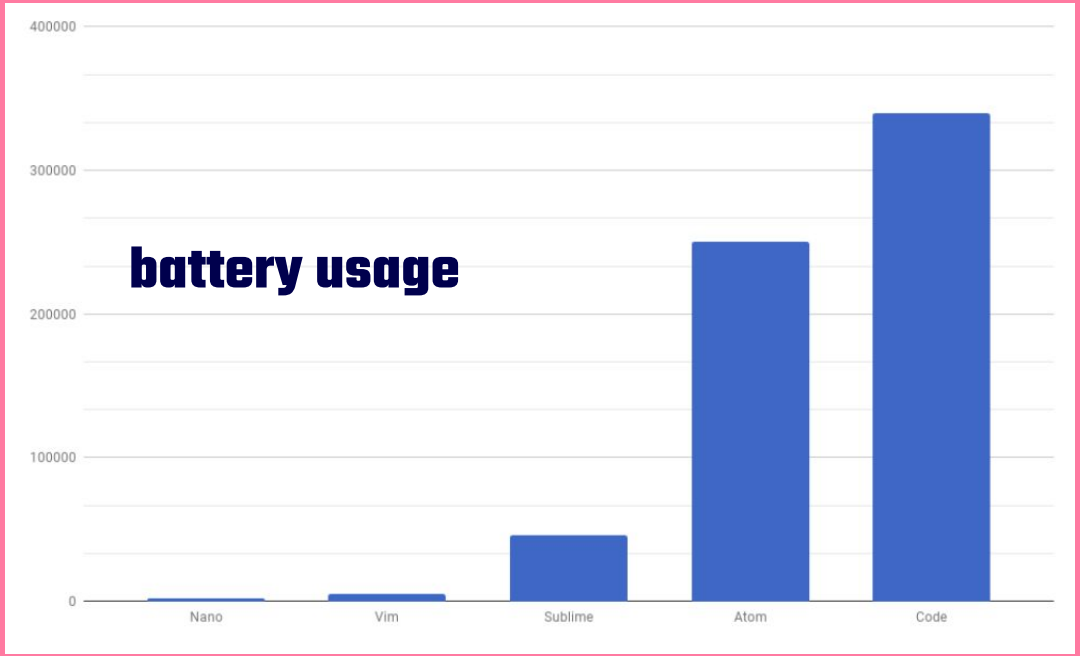
Vim Keybindings

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




1 #include <cstdio>
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; andidate) {
10 }
11 return 0;
12 }
OSD Notify
SHIFT+4bbbxx
9,45 All

Why Vim?


battery usage
400000
300000
200000
100000
0
Nano
Vim
Sublime
Atom
Code

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

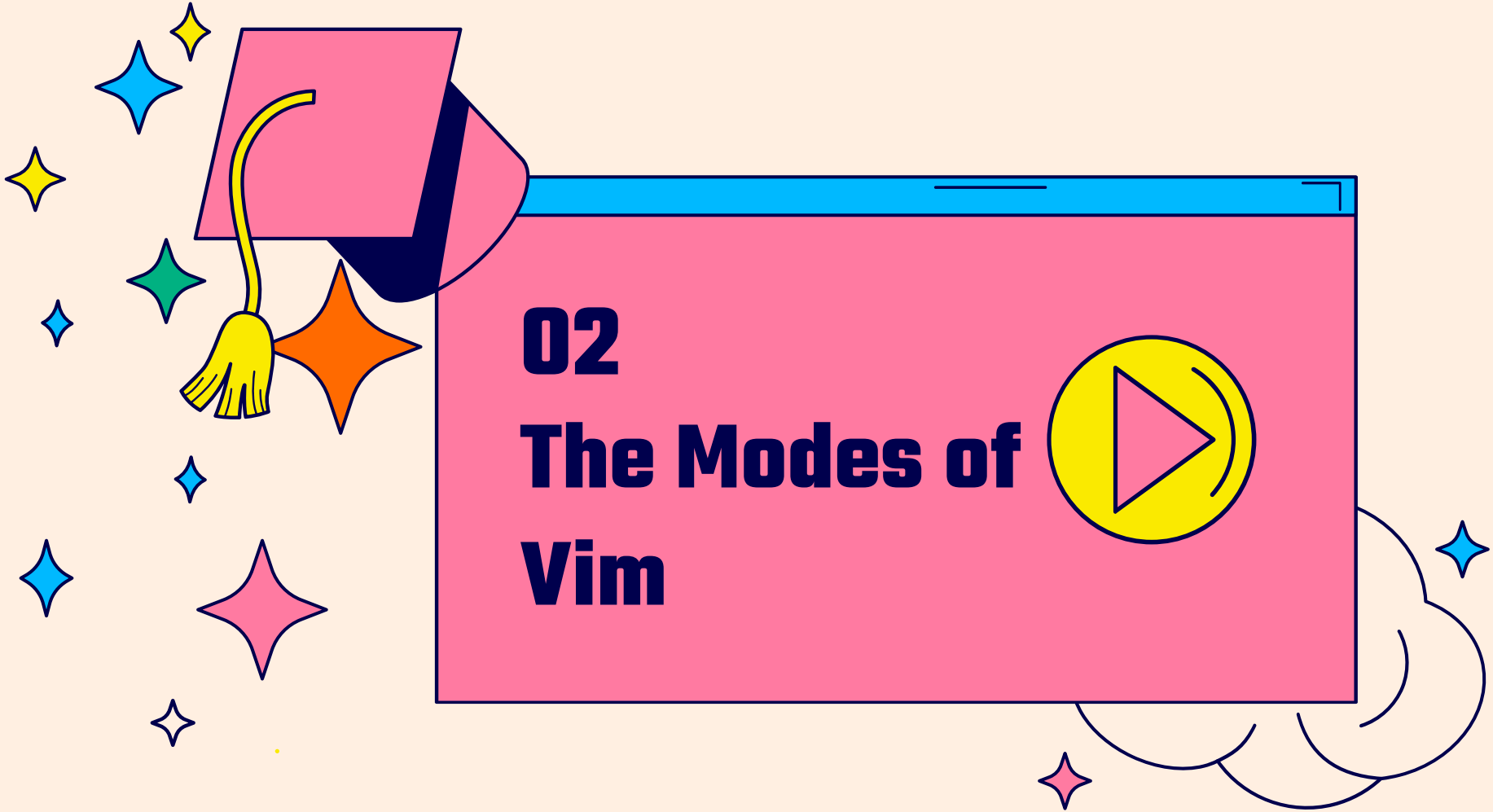
As easy as typing vimtutor into the terminal

: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.

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 occurence 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 occurence 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**

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

previous

gg first line

find word under cursor

N previous *text*

?*text* find *text*

C-b page

C-u 1/2 page

H screen

{ paragraph

(sentence

, previous *x*

B delimited word

gE delimited end

k up

back

0 line

^ non-blank

F*x* find *x*

T*x* after *x*

b word

ge end

h left

j down

l right

e end

w word

t*x* before *x*

f*x* find *x*

$ line

forward

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

vim - movement commands

Normal Mode: Editing

<COUNT> <VERB> <TEXT OBJECT>

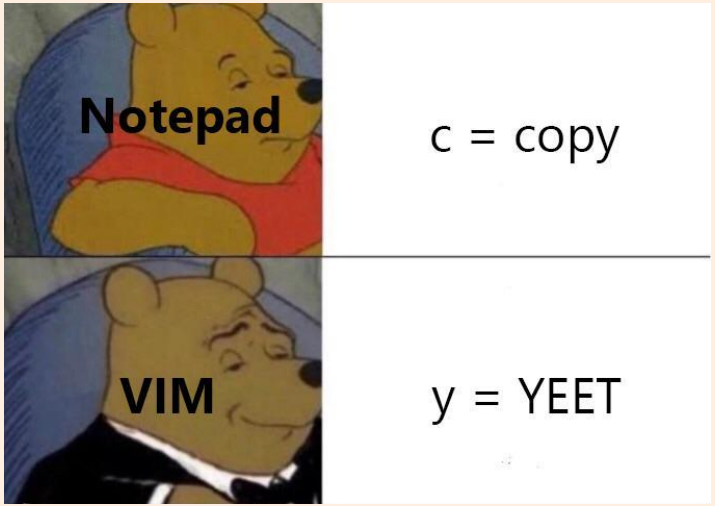
2 delete word

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



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 Devloper
@iamdevloper

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

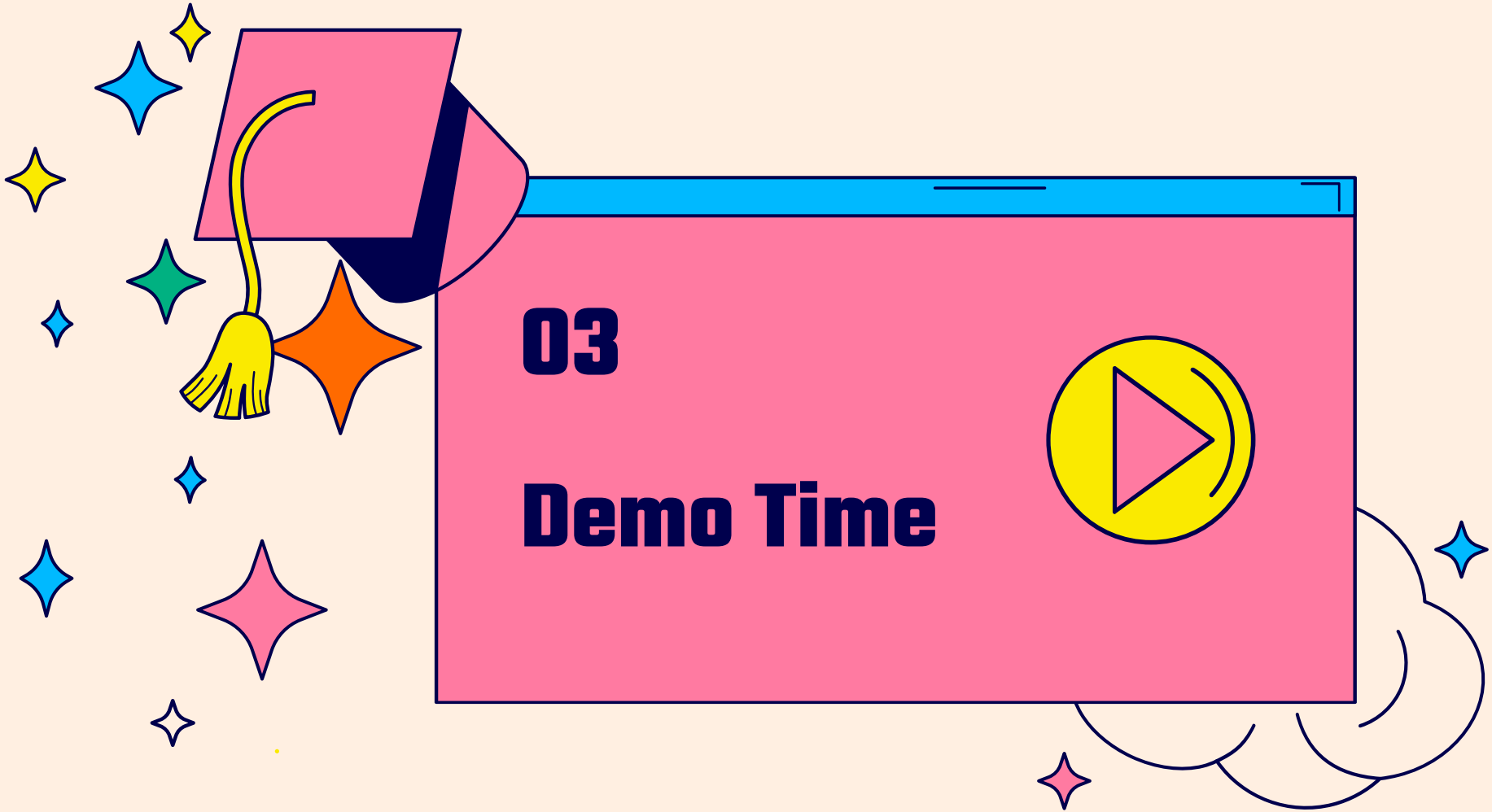
`:<cmd>!`	Force
`/<pattern><Enter>`	Jump to pattern instance
`:<number>`	Jump to line number
`:w`	Save
`:wq`	Save and Quit
`:q`	Quit

A Very Useful Command

(especially for the lab 😗)

```
:%s:<find>:<replace>:g
```

- **%** - through the whole file
- **s** - search command being run
- **g** - flag replaces every instance globally


03
Demo Time


03
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

LPT: Lab Pro Tips

- If you get an error message that says something like “Vundle” just press enter to continue
- 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
- Treat both words “More dust” as dust
- 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