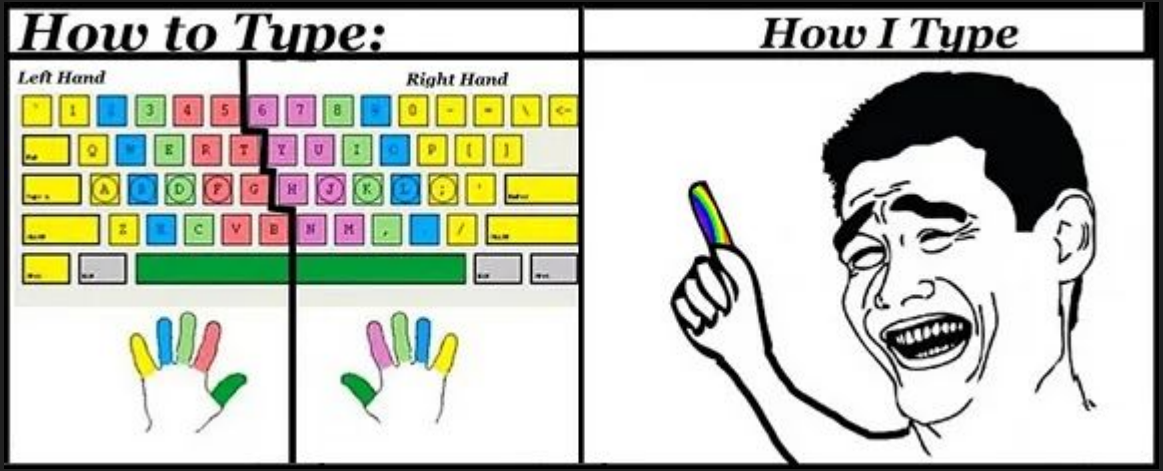
Neovim Tech Talk

JUSTIN EDGAR

Prerequisites

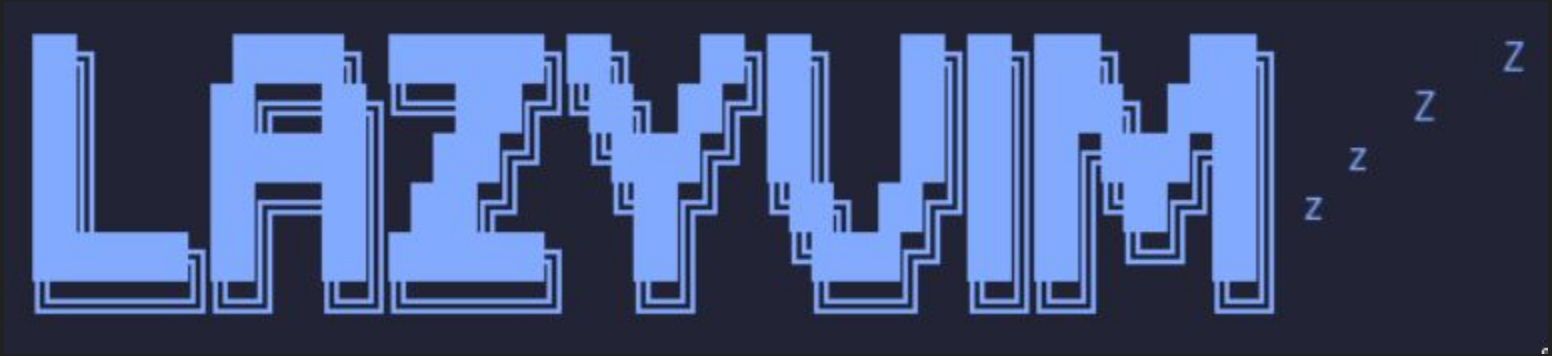
- Have Git installed
- Have VsCode/Neovim installed
- Know how to Touch-Type (LEARN THIS FIRST)



If you already have Neovim Installed

Kickstart: https://github.com/nvim-lua/kickstart.nvim

Lazyvim: https://www.lazyvim.org/



If you don’t

VSCodeVim

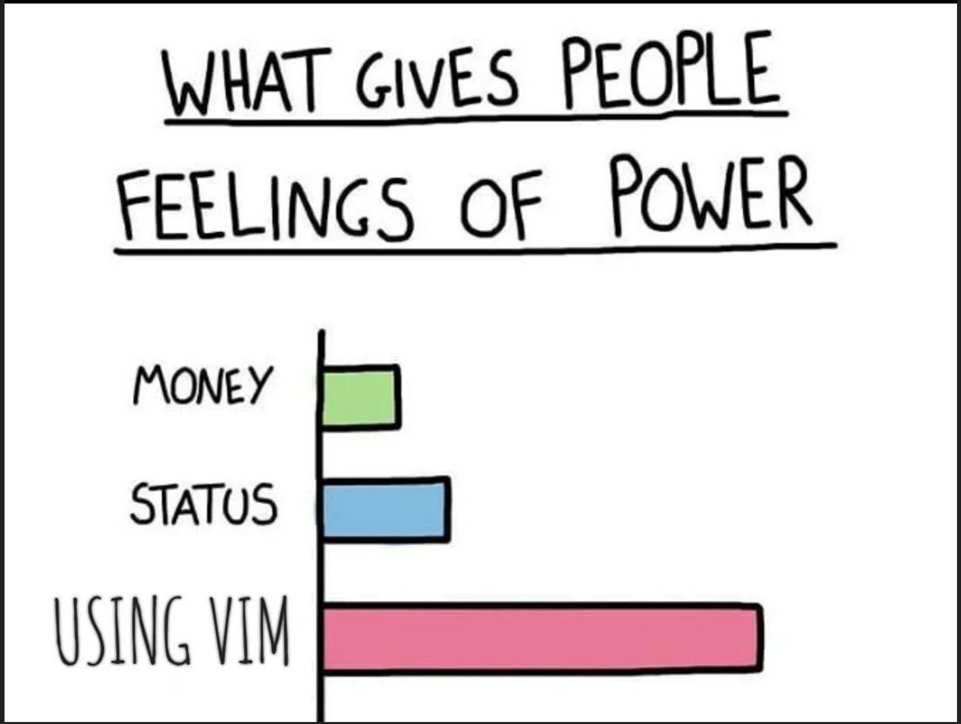
Vim emulation for Visual Studio Code

Slides + Demo

```
$ git clone https://github.com/Heaveil/Vim-Talk.git
```

Outline

- Introduction
- Basics
- Plugins
- Q&A



What

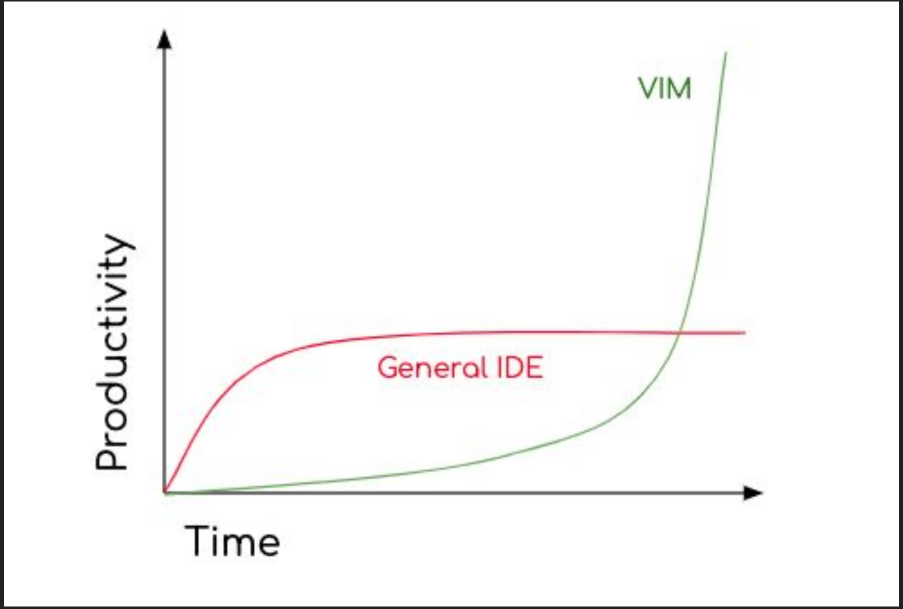
- Vim is a text editor (it edits text)
- Neovim is a community fork of Vim
- Neovim ≠ Terminal
- Neovim is a program that lives inside the terminal
- The philosophy is to never leave your terminal

Why

- Mouseless Development
- Life Long Skill
- Super Efficient
- Blazingly Fast 🚀
- Vast Plugin Ecosystem
- Makes Coding Super Fun
- Bragging Rights (I Use Neovim Btw)

Caveats

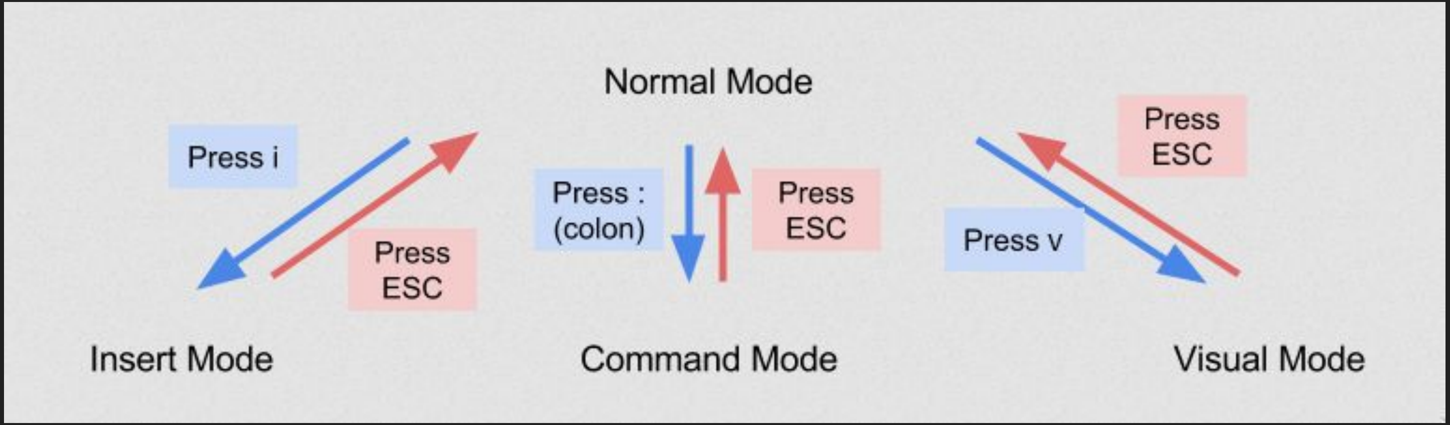
- Steep Learning Curve
- Configuration Hell



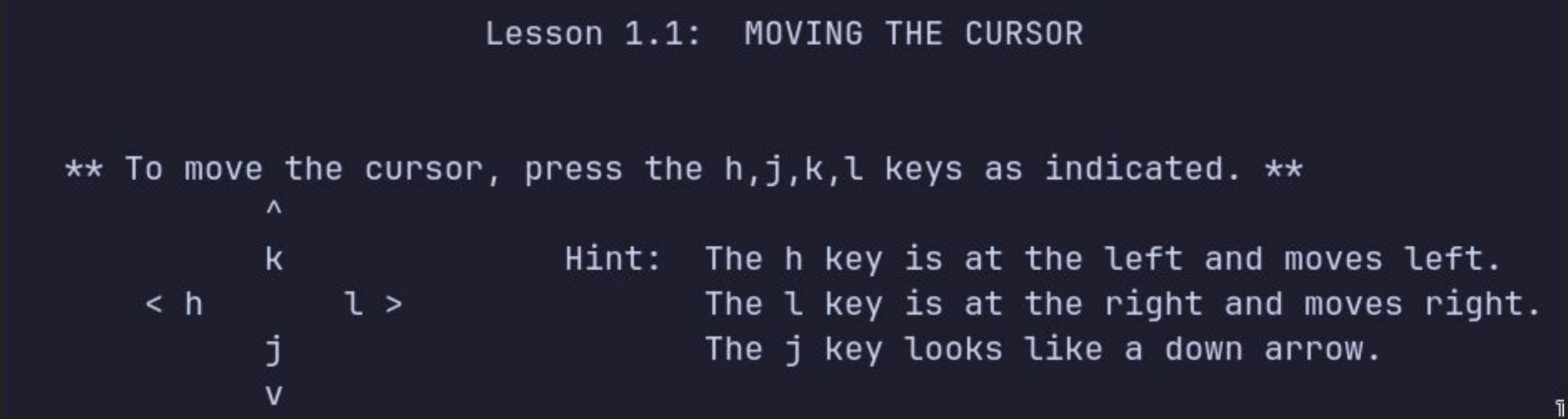
Let's Start!

Modal Editor

- Vim is a modal editor. It has 4 main modes
  - **Normal**: Navigate
  - **Insert**: Modify
  - **Visual**: Select
  - **Command**: Commands
- Press **esc** on any mode to go into normal mode, this should be your default



Basic Navigation



```
                      Lesson 1.1:  MOVING THE CURSOR


  ** To move the cursor, press the h,j,k,l keys as indicated. **
        ^
        k              Hint:  The h key is at the left and moves left.
    < h       l >             The l key is at the right and moves right.
        j                     The j key looks like a down arrow.
        v
```

Horizontal Navigation

- **w**ord
- **e**nd of word
- **b**ack
- **^** - first character
- **$** - last character
- **%** - matching pairs

- **f**ind / **F**ind <character>
- **t**ill / **T**ill <character>
- **;** - go to next match
- **,** - go to prev match

Vertical Navigation

- **ctrl-u**p - moves cursor up ½ page
- **ctrl-d**own - moves cursor down ½ page
- **gg** - moves cursor to start of file
- **G** - moves cursor to end of file

RELATIVE LINE NUMBERS

```
<number><j/k>
```

Insert Mode

- **i**nsert - before cursor
- **a**ppend - after cursor
- **o**pen - new line below

- **I**nsert - start of line
- **A**ppend - end of line
- **O**pen - new line above

Visual Mode

Visual Mode - **v**

Visual Line Mode - **V**

Visual Block Mode - **ctrl-v**

Opposite End - **o**

Uppercase - **U**

Lowercase - **u**

Increment - **ctrl-a**

Decrement - **ctrl-x**

Global Increment - **g ctrl-a**

Global Increment - **g ctrl-x**

More Actions

- **d**elete
- **c**hange
- **y**ank
- **p**aste
- **u**ndo
- **ctrl-r**edo

- **r**eplace - replace a character
- **R**eplace - go to replace mode

- / - go to search mode
- **n**ext - find next occurrence
- **N**ext - find previous occurrence

Windows

Open Vertical - **ctrl-w v**

Open Horizontal - **ctrl-w s**

Window Left - **ctrl-w h**

Window Down - **ctrl-w j**

Window Up - **ctrl-w k**

Window Right - **ctrl-w l**

Command Mode

- Press colon “**:**” while in normal mode to go to command mode
- In command mode
    - **w** - write/save
    - **q** - quit
    - **wq** - write/save and quit
    - **q!** - quit and not save
    - **e** <file-name> - edit a file (creates one if it does not exist)
    - <line-number> - goes to the line number (useful for debugging)
- You can also do **:terminal** to spawn a terminal in neovim

Find and Replace (Substitute)

```
:<range>s/<search pattern>/<replace string>/<options>
```

`:%s/hello/world/g` → replaces every “hello” with “world”

SPACE RANGER
LIGHTYEAR
BUT WAIT, THERE'S MORE

The Language

<verb><adjective><noun>

Verbs

- **v**isual
- **d**elete
- **c**hange
- **y**ank

Adjectives

- Numbers
- **t**ill
- **f**ind
- **i**nside
- **a**round

Nouns

- **hjkl**
- **w**ord
- **$ ^ %**
- **() [] {}**
- **“” ‘’**

Marks

- Marks are saved places that you can go to
- To save a mark, press m with a key (example a)
- To go to a mark, press `<key-used-to-mark> (e.g. `a)

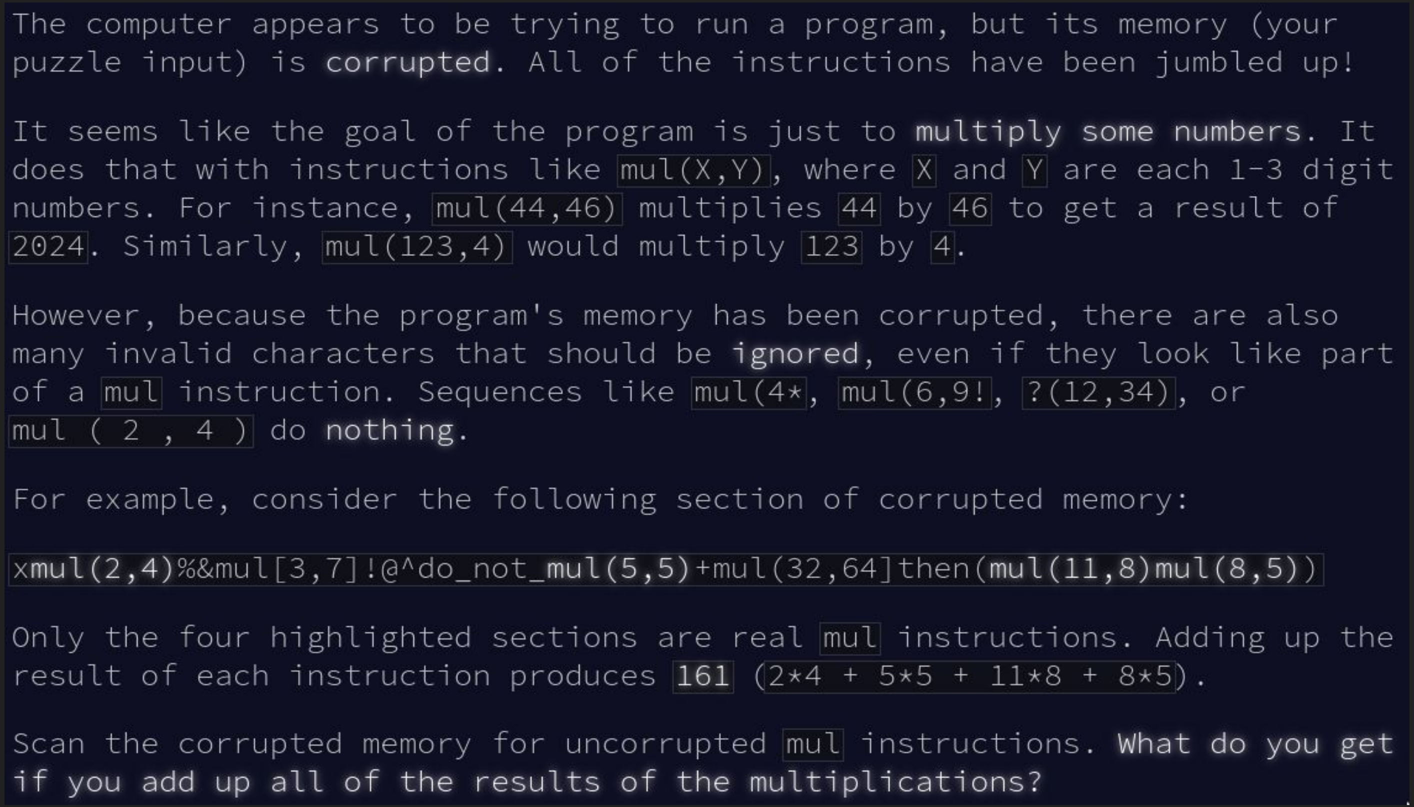
Macros

- Macros are sequences of motions squished together
- To start recording a macro, press q with a key (example a)
- To end the macro recording, simply press q again
- To apply the macro, press @<key-used-to-record> (e.g. @a)

Fighting One Eyed Kirby

\(.*\)

Advent of Code 2024 Day 3



The computer appears to be trying to run a program, but its memory (your puzzle input) is **corrupted**. All of the instructions have been jumbled up!

It seems like the goal of the program is just to **multiply some numbers**. It does that with instructions like `mul(X,Y)`, where `X` and `Y` are each 1-3 digit numbers. For instance, `mul(44,46)` multiplies `44` by `46` to get a result of `2024`. Similarly, `mul(123,4)` would multiply `123` by `4`.

However, because the program's memory has been corrupted, there are also many invalid characters that should be **ignored**, even if they look like part of a `mul` instruction. Sequences like `mul(4*`, `mul(6,9!`, `?(12,34)`, or `mul ( 2 , 4 )` do **nothing**.

For example, consider the following section of corrupted memory:

```
xmul(2,4)%&mul[3,7]!@^do_not_mul(5,5)+mul(32,64]then(mul(11,8)mul(8,5))
```

Only the four highlighted sections are real `mul` instructions. Adding up the result of each instruction produces `161` (`2*4 + 5*5 + 11*8 + 8*5`).

Scan the corrupted memory for uncorrupted `mul` instructions. **What do you get if you add up all of the results of the multiplications?**

Plugins

Harpoon
Man
SNL
SUBSCRIBE

mason.nvim

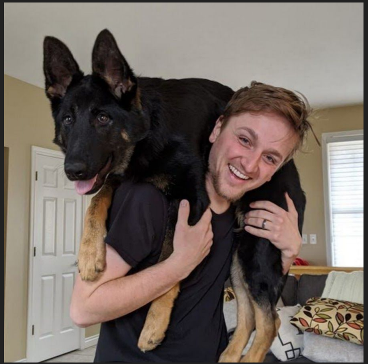
Complimentary Tools

zsh%

tmux

Resources

ThePrimeagen


TJ DeVries

Typecraft

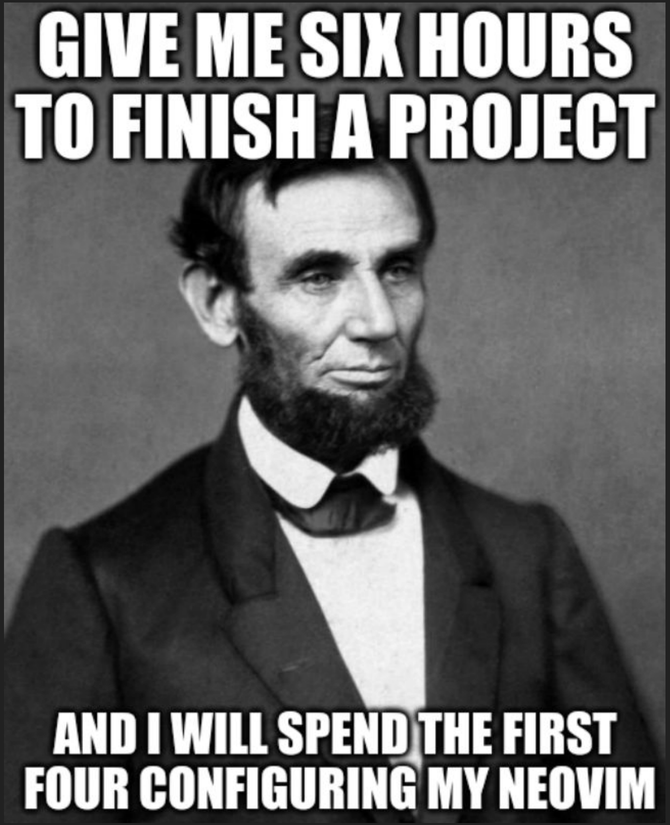
Folke

Special Thanks

Daniel Lindsay-Shad

Udit Samant

Q&A


GIVE ME SIX HOURS TO FINISH A PROJECT
AND I WILL SPEND THE FIRST FOUR CONFIGURING MY NEOVIM