# “I Wish I Knew How To Quit You”

Getting Started with Vim

# What *is* Vim anyway?

Vim is a **command-line editor**, meaning it gives you the ability to edit files right in the command line!

> Some other examples of command-line editors include:
> Pico, Nano, Emacs, Vi, Vim, and NeoVim

Nano is regarded as one of the easier command line editors to use, but *easy* doesn't mean *awesome.*

By learning Vim, you can look like a super cool programmer *and* get into weirdly long, passionate fights on StackOverflow about its superiority to eMacs*

*not covered in this talk

# Tell me more!

Vim is the new and improved version of the Vi editor from the 70s.

It retains the keybindings of Vi, but adds a ton of functionality and extensibility

*‘Fun’ fact: Vim is actually a portmanteau (of sorts) of **V**i **IM**proved*

Vim is also considered the *lingua franca* of System Administration, and is the editor most likely to be found on any *nix system (Unix, Linux...)

You can check to see if you have Vim on your computer (which you almost definitely do) by opening the terminal and typing:

```
vim
```

# Okay, very funny.
# Now how do I get out of VIM???

To quit Vim, type:

**`:q`**

If that doesn't work, try hitting `ESC`, then `:q`

or `:q!` to quit without saving

*We'll cover what these mean shortly, I promise.*
*But it is really that easy!*

# If you want to follow along...

I would *highly* recommend working in a ‘burner’ text file to play around in while we’re going through these commands, as it makes a lot more sense when you can see it on your screen.

You can make a file and open it immediately in Vim by typing something like:

```
vim my_vim_test.txt
```

*If you don’t want to follow along, that’s totally fine too! There’s some great online tutorials I’ll mention at the end, including one in Vim itself!*

# Vim ‘Modes’

Vim is also a *modal* text editor, meaning that the alphanumeric keyboard behaves differently in different modes.

Much like everything about Vim, people have a ***lot*** of feelings about how many modes Vim has, and what they’re called.

That being said, the most important/common modes I could identify are:

- **Command Mode** *(aka ‘default’ mode, ‘normal’ mode)*
- **Insert Mode** *(where we can type as we’d expect)*
- **Visual Mode** *(allows us to highlight text)*
- **Last-line mode** *(aka the mode we need to be in to quit Vim)*

# Command Mode

When we start Vim, we start in **Command Mode**. This is how we navigate through our files, and where a lot of confusion and frustration with Vim seems to come from.

See, in command mode, we use the alphanumeric keys to ***move*** through the text of a file. We cannot type as we normally would - instead, we have to use letters to move around. The most important keys here are **`h, j, k, l.`**

- **`h`** - moves to the left
- **`j`** - moves down one line
- **`k`** - moves up one line
- **`l`** - moves to the right


Note that lines don't *wrap* in Vim - we have to go up and down on a line once we reach the end!

# Command Mode

So, in Command mode, `h, j, k, l` become like our ‘arrow’ keys

*Note that you* can *use actual arrow keys, but you will be mocked mercilessly by people who use Vim*

But this isn’t very fast... and Vim is all about making it super fast to move through text! Here are some other incredibly useful and incredibly common ‘motions’ in Command Mode:

- `$` - Move cursor to the end of the line
- `0` - Move cursor to the beginning of the line
- `G` - move to the end of the file
- `gg` - move to the beginning of the file
- `w` - Move cursor to the *start* of the *next* word
- `e` - Move cursor to the *end* of the *current* word
- `b` - Move cursor to the *beginning* of the *current* word

# Insert Mode

Okay, so moving around a document is fun and all, but it's not much use if that's all we can do. Fortunately, there's more - **Insert Mode!**

To enter **Insert Mode**, just press **`i`**
while in a document open in Vim.

*Note: Make sure that the cursor is on top of the character* after *which you want to insert text.*

If you're following along, feel free to do this now and start typing - you'll see the keyboard behaves like normal!

When done with Insert Mode, press the **`esc`** key to get back to Command Mode.

# Insert Mode

Now, we can toggle between Command mode to navigate to a line in our file, and then enter Insert mode to start typing on that line! **Pretty cool!**

If we want to **add** a new line, press **o**.

This will make a new line below the current one.

If we want to just **add text** to the end of a line, press **A.**

This will *append* text to the end of the line.

**O** (capital O) will make a new line *above* our current line

**a** will append text after the *cursor* position.

Note that both of these commands will enter us into Insert Mode, so we'll need to hit **esc** to get back to Command Mode when we're done.

# Shoot... I f#&ked up! How do I fix it???

Don’t worry! Deleting words, fixing typos, and undoing changes is actually pretty easy in Vim.

To **delete** a *single* character,
move the cursor over the character and press **x**.

To **undo** a change,
press **u** to undo the last command.

Or, if you’ve made a lot of mistakes,
typing **U** will undo changes to a whole *line*.

Ctrl + R will *redo* (or undo an undo)

# It's just a typo, though, no need to delete everything...

To **replace** a single character while in Command mode:

Move the cursor over the letter we want to replace and **press** `r`, followed by the letter we want to update it to.

> So if we typoed the word 'type' to 'tupe', we'd move the cursor over the 'u' and then press **ry** to *replace* the u with a y.

If we want to replace a bunch of characters, we can type **R**, which will put us in **'Replace mode'**

> **Replace mode** is very similar to *Insert Mode*, except that every character we type *deletes* the existing character in its place.

# Whoa! Tell me more!

Need to **delete a whole word**?

Move the cursor to the start of the word you want to delete and **type `dw`**

Tip: If you want to delete a whole line, just type **`dd`**

Note that the ‘w’ here is the same ‘motion’ that we saw earlier - it deletes until the *start* of the next word!

You can also try **`d$`** to delete to the end of the line, or **`de`** to delete to the end of the current word...

This is a common pattern with a lot of the ‘commands’ in Vim - the first letter will be the *command* and the second letter will be the *motion* in which to act on.

If you’re feeling bold, try out **`ce`** (to *change* to the *end* of the word)

# But I wanna go *faster!*

As we saw with deleting and changing, actions in Vim seem to follow the pattern of:

*command + motion*

But what if we want to, say, delete three or four words?

Super easy! To perform an action on multiple words, we just add a number between the command and motion to indicate how many times we want to perform that action.

*command + number + motion*

*So typing **d3w** will delete the next three words, **c4b** will change the previous four...*

To move quickly through text, we can simply prefix the motion with a number!

***6k** will move the cursor up by six lines, **5w** will move over five words, **10b** will move back 10 words.*

# Last-Line Mode

To enter last-line mode, **press `:`**

Last-line mode is basically like a ‘toolbar’ in a traditional GUI app. This is where we can save, quit, edit, read, and get help.

Here are some common commands

- **`:w`** - save
- **`:q`** - quit
- **`:wq`** - save and quit
- **`:q!`** - quit without saving

as well as some neat ones to play around with:

- **`:r`** *`<filename>`* - insert the contents of the specified file into the current file
- **`:e`** *`<filepath>`* - open a file in Vim to start editing it
- **`:help`** - open the help menu

# Visual Mode

To enter visual mode, **press v**

Now you can use those motion command keys to move and highlight chunks of text. Neat-o!

If we want to *copy* a chunk of text,
**press y** while we have something highlighted
and then **press p** to paste it!

When you delete text (with **d**),
it actually acts as a *cut* rather than a delete,
and we can paste that deleted text somewhere else by using **p** also!

# Still not convinced? Check out how easy it is to search!

With the cursor on an opening parenthetical, use **%** to jump to its matching pair.

**f** will find the next occurrence of a character, and **F** will find the previous one.

**fo** will find the next 'o', and **3fq** will find the 3rd occurrence of q

To **search** a document, press **/** followed by the phrase you want to search for!

Then we can use **n** to search for the same phrase further down the document, or **N** to search in the opposite direction

To do a 'find and replace' on a line, use **:s/old/new/g**
(which will replace *'new'* with *'old'*)

To replace that word *everywhere* in the file, do **:%s/old/new/g**

# Alright, you win! I want to learn more!

Fortunately, there's a *ton* of fun resources online to practice VIM, including a great tutorial within Vim itself!

Typing `vimtutorial` in the command line will open a tutorial... It takes about 20-30 minutes or so, and covers most of this talk and more!

Want to play a game while learning?
Check out **vim-adventure.com**

Worried about memorizing all these shortcuts?
Check out **vimsheet.com** for a quick reference guide!

See, Vim isn't so scary!

Plus, when you use it, you'll always feel like...