What is Vim

What is Vim?

- Vim is a declarative text editor by Bram Moolenaar
- Vim is an editing ideology.
- Vim is life.

Installation

On Mac:

- `brew install vim` or `brew install macvim` (for GUI)
- The built-in version of Vim for Mac is pretty out-of-date but should still work

For Windows:

- Download and install GVim
- OR download the VsVim plugin for Visual Studio (what I'll be using for this demonstation)

All the demos for this talk should work on any of the above.

Some Obligatory History

- Bill Joy created vi in 1976 for Unix
  - Continuation of the "ed" line editor
- Vim stands for "Vi IMproved" and was released in 1991
- Started as a basic port but has added a bunch of new features over vanilla vi
  - Default distribution with most flavors of Linux

Modes!

The part that scares everyone away from Vim is the concept of "modal editing". What does this mean?

- You can be in one of three different modes at any point
  - `i` or `a` puts you in insert mode
  - `v` puts you in visual (or "highlight") mode
  - Exit either of those modes by hitting escape. This puts you in Normal mode (which is also the default)
- It usually says on the bottom which one you're in.
- Makes learning a bit tricky, but also limits the need for modifier keys
- If you're ever scared, just hit escape. It will be OK.

Simple Movement

There is a relatively non-intuitive set of keys for moving around character-by-character

- `h` moves left
- `l` moves right
- `j` moves down
- `k` moves up

- You can also use the arrow keys

More advanced movement

I almost never use the `hjkl` commands to move around, and prefer to usually go by word or by paragraph.

- `w` moves you forward a word and puts your cursor in front of the next word.
- `e` moves you to the end of the current word or the end of the next one
- `b` jumps to the beginning of the current word (or the next one)
- } moves you to the end of the paragraph (or the next one)
- { moves you to the beginning of the paragraph (or the previous one)

I find it's useful to get into the pattern of going "by word" instead of "by character" as it makes writing macros a lot simpler

More advanced movement

You can repeat a movement by putting a number in front. For example, `20w` will jump forward 20 words

More advanced movement

- `0` jumps to the beginning of the line
- `$` jumps to the end of the line
- `I` jumps to the beginning of the line and puts you in insert mode
- `A` jumps to the end of the line and puts you in insert mode
- gg jumps to the beginning of the file
- `G` jumps to the end of the file

Actions

Movement is great, but the bread and butter of Vim is how it can compose with actions

- `d` begins the delete action
- `y` begins the yank (copy) action
- `v` begins the visual (highlight) action.
  - Visual can be an action **and** a mode

Any of these actions can be combined with the movements from before.

Note: Once an action has been started, `i` basically means "current"

Examples

As a demonstration, try these in Normal Mode

- Delete the next twenty words: `d20w`
- Delete the next two characters: `d2l`
- Copy the previous eight words: `y8b`
- Select until the end of the next four paragraphs: `v4}`
- Delete everything after this point on this line: `d$`

- Copy everything before this point on this line: `y0`
- Select everything from this point to the end of the file `vgg`
- Delete the current word: `diw`
- Copy everything in the paratheses: `yib`
  - Presumably to be annoying, after an action, `b` means "small block" (parentheses) in this context. `B` means "big block" and works with curly-braces.

Shortcuts

Some of these are so common that they have easier shortcuts

- `x` is basically equivalent to `dl` and deletes the current character
- `yy` is basically equivalent to `0y$` and copies the current line
- `dd` is basically equivalent to `0d$` and deletes the current line

Macros

Keeping your editing style declarative lends itself incredibly well for the use of macros

- begin recording a macro by hitting `q` and any other letter (for this we'll use `a`)
- Do a series of commands
- End the macro by hitting `q` again
- Execute the macro by hitting `@a` (substitute the `a` with whatever letter you chose)
- Repeat the macro by hitting `@@`

Example: Converting Hyperlinks to JSON

Here's an example

Example: Surround with delimiters and capitalize

Here's another Examples

Basic Macro Tips

Here are a few basic things I do when creating macros.

- Start by hitting 0 so that you can start at the beginning of the line
- Think as declaratively as possible.
  - Move by words instead of by character so that the macro is easier to reuse
  - Use the "jump to" features like "jump to end" liberally
  - Before writing the macro, look for bits that are easy to generalize and try and ignore the bits that aren't
  - When you're done making your edits, hit `j` to move to the next line, so that the macro is easier to repeat

Note: None of these are law, just rules of thumb for getting started.

To Learn More

- On Mac and Linux, type `vimtutor` into the command line.
  - On Windows, it comes with either Git Bash or Cygwin

To Learn More

- Avoid the temptation to live in Insert Mode
  - It's best to **only** use insert mode when you're typing new text.
  - Inserts count as only **one** layer of undo-history.
  - Editor is wholly unremarkable in insert mode

To Learn More

Experiment!

- Vim's interface is designed to be composable and guessable.
- It's entirely possible that you can discover a new and useful command that isn't explicitly documented
- It makes the editor a lot more fun to use

To Learn More

Feel free to bother me for questions!

- @tombert on Slack (the guy with the smiley-icon)
- github.com/tombert
- tom.gebert@jet.com