vim

registers, macros, visual mode, and windows

vim Registers – The Default Register

- When you **delete**, **change**, or **yank** text, the text under the operation is stored in a **register**

- By default, the text is stored in *default register*

- When you **paste**, by default the text pasted is from the *default register*
    - p – Paste text after the cursor
    - Shift+P – Paste text before the cursor

vim Registers – Named Registers

- In many CLI apps a "register" is a variable whose name is a single character.

- You address registers with the double quote "
  - "a is register a
  - "b is register b
  - "" is register " (the default register)

- To place the text under the operation in a specific register, like variable assignment in programming, you first specify the register, then the operation which assigns to it:

  "ay$ - Assign to **register a** the **yanked** text to the end of the line. (**copy**)
  "bd$ - Assign to **register b** the text deleted to the end of the line. (cut)
  "zc$ - Assign to **register z** the changed text to the end of the line. (cut)
  "ap – Paste the contents of **register a.**

vim Registers – Historical Delete Stack

- Each time you *delete (d)* text it is pushed onto a historical register stack

- The stack has registers `"1` through `"9` where
  - `"1` is the last text you deleted
  - `"2` is the text deleted before `"1`
  - …
  - `"9` is the text deleted before `"8`

- Two primary uses:
  1. Accidentally deleted text a few operations ago? Check historical stack.
  2. Offers "multiple clipboards" without having to name registers

- View the contents of all registers, with `:registers` or `:reg`

vim Grammar - Registers

```
command      -> CURSOR_TO | operation | LINE_OPERATION | TO_INSERT_MODE | paste

operation    -> assign_to_register (N_TIMES? VERB CURSOR_TO | VERB TEXT_OBJECT)

paste         -> read_from_register ('p' | 'P')

assign_to_register    -> register

read_from_register    -> register

register              -> default_register | '"' register_name

default_register      -> ε

register_name         -> [a-z]
```

vim Macros
Record and Replay strings of commands

- To begin recording a vim macro, press the q key followed by a register name. For example:
  - **qm** – begin recording a macro in the a register
  - Notice the status bar tells you "recording **@m**"

- Then, enter your commands as you normally would.

- To stop recording a macro, press the **q** key again.

- To replay a macro, press the **@** symbol followed by the macro name. For example:
  - **@m** – replays the macro in register m

- Are these the *same* registers as what we cut and copy to? ***YES!!!***
  - You can *paste* your macro into the document!
  - You can also write your macro in your document and then copy it to a register for use as a macro!

vim Grammar - Macros

```
command_or_macro -> command | record_macro

command -> CURSOR_TO | OPERATION | … | replay_macro

record_macro       -> 'q' register_name command* 'q'

replay_macro       -> N_TIMES? ('@' register_name | replay_macro_again)

replay_macro_again -> '@' '@'

register_name      -> [a-z]
```

We now have a construct in our grammar that lets us ***compose*** commands together and allows us to define our own compound commands!

Composition is a superpower of languages.

vim Visual Mode 101

Like clicking and dragging your mouse around.

- **v** – Transition to **visual** mode. Select using *location_to* commands.
  - to_register? c – change
  - to_register? y – yank (copy)
  - to_register? d – delete (cut)

- **Shift+V** – Transition to **visual line** mode.
  - Verbs same as above
  - > - Indent
  - < - Unindent

- **Control+v** – Transition to **visual block** mode.
  - Shift+i – Insert in front of block.
    - Comment out block of code: Ctrl+v j j j Shift+i // Ctrl+[
  - Shift+a – Insert after block

vim Window Control

Ctrl+w, v- Split the window **V**ertically

Ctrl+w, s - **S**plit the window horizontally

Ctrl-w, w - Cycle between **w**indows

Ctrl-w (h|j|k|l) - Move to window
Move to left, down, up, right.

ZZ - Close split window (and save)

Ctrl-o - Open File Explorer

vim - A Few More Useful Keys in Normal Mode

- x - Delete the character under the cursor
- <Ctrl>+A – Increase the number under the cursor by 1
- ~ - Toggle the case of the letter under the cursor
- r<char> - Replace the character under the cursor and stay in normal mode
- Shift+J - Join the next line onto the end of the current line.