presentations

Matthew M. Keeler

March 26, 2015

Emacs

A look at what I love about "The One True Editor" Matthew M. Keeler @keelerm84

My First Love

I have been a vim user for over 6 years. In fact, I still use vim occasionally so I don't get rusty. So why make the change?

An Informed Decision

Most people in the tech space know about the editor wars and the various points of conflict:

- vim or emacs
- modal editing or key chording
- thin editor or entire OS

My default reaction was that obviously vim is better. But considering I had never tried emacs, my opinion was little more than a defense mechanism.

I didn't want to feel like I had wasted all those years.

So What To Do

I figured if I was going to pick a side, it would be worthwile trying out both.

So I spent my nights and weekends playing around with emacs. Eventually, I had built up enough muscle memory and regained enough efficiency that I felt I could start using it during my day job.

My vim workflow was replaced entirely by emacs during this learning process.

After many months of emacs, I was finally able to decide on a winner, which was . . .

And the Winner Is?

Either. Neither. Both.

Having become quite effective in both editors led me to the conclusion that the editor war is pointless. emacs and vim focus and optimize for different experiences, so comparing them is pointless.

vim is all about having the most efficient editing capabilities possible.

emacs is all about having an integrated environment so your **workflow** is as efficient as possible.

But It is An Editor, right?

I'm sure we have all heard the joke:

> emacs is an operating system lacking only a decent editor.

While funny, I would argue that this statement is completely off base.

emacs' central tenet is that EVERYTHING should be treated as text. And we do mean everything – source files, database connections, shells.

Considering this, it would have been foolish to build it ontop of a sub par editor.

Editing

> How do you generate a random string? Put a first year undergrad
> in vim and tell them to quit.

One of the nice things about emacs is that it is accessible to people who aren't from a CS background. If you want to use it like notepad for some simple text edits, you can.

But of course the real power comes from being completely keyboard driven. Instead of relying on different modes, navigation and text manipulation is handled through key chords.

Escape Meta Alt Control Shift

The keychords in emacs have received a lot of slack as being difficult to press or hard to remember.

emacs navigation comes from the readline library since both were developed by the GNU Foundation. So if you're familiar with those, you already have a headstart in learning emacs.

The key combinations typically follow a pattern. `C-something` will perform a basic action. `M-something` is generally a stronger version of the control counterpart.

- `C-f` to move a character. `M-f` to move a word
- `C-t` will transpose a character. `M-t` will transpose a word

Some Editing Nicities

There is no point in going over all the editing commands as that will be boring and we don't have time for all of that. Instead, let's just talk about some of the features I really like

- Syntatical movement
  - Moving by sentences with `M-a` / `M-e`
  - Moving by defuns `C-M-a` / `C-M-e`
- Smart case find and replace
- Narrowing and widening
- The kill ring

A Window By Any Other Name

The terminology around emacs' file management stuff is a little different.

There is the concept of a buffer, which may or may not be associated with a file.

A window is the visible part of the screen. If you have splits, each section is a window. A window is only ever associated with one buffer at a time, though the same buffer can be associated with multiple windows.

The entire desktop application is held within a frame. There can be multiple frames opened at any time. Each of these frames can share the same set of buffers and can all connect to a central emacs daemon.

Learning About Emacs

New users starting out will really appreciate how easy it is to learn about emacs. The integrated help system is wonderful.

- Calling functions interactively with `M-x`
- Find out what a key is bound to with `C-h k`
- Show commands for the current mode with `C-h m`
- Find the value of variables with `C-h v`
- Read various help menus with `C-h i`

Let's Talk Customization

emacs is extensible in real-time because of the way it is written. Unlike other editors, emacs is a LISP machine running ontop of a tiny C based LISP intreperter.

This essentially means the editor is a large REPL.

Built-in functions and new features can be modified or developed, evaluated and injected into the current running environment without the need to restart.

Hooks and Advice

With third party and built-in packages, while you can modify the source, you don't want to. emacs provides the concept of advice and hooks which allows for arbitrary code to run during different stages of execution.

```
(defadvice kill-line (after say-ouch activate)
    (message "Ouch!"))
```

```
(add-hook 'go-mode-hook (lambda ()
                          (local-set-key (kbd "C-c C-r") 'go-remove-unused-imports)
                          (local-set-key (kbd "C-c i") 'go-goto-imports)
                          (local-set-key (kbd "M-.") 'godef-jump)
                          (go-eldoc-setup)
                          (ggtags-mode 0)
                          (if (not (string-match "go" compile-command))
                              (set (make-local-variable 'compile-command)
                                   "go build -v && go test -v && go vet"))))
```

Packages and Package Management

Newer versions of emacs come prepackaged with a tool, package.el This let's you connect to repositories of packages, download and install packages directly from within emacs.

el-get provides receipes for fetching and installing third party packages as well.

And then there is cask, which let's us download packages outside of emacs to help improve the startup process. This is what I use.

Some Fun Packages

Let's take a look at some fun packages emacs provides.

- dunnet
- doctor
- M-x butterfly

Some More Useful Packages

As I mentioned earlier, emacs is really about an efficient workflow. As such, there are packages to support all sorts of different tools you might want to use in your workflow.

I won't go into detail on all of these, but I thought I would include a small list of packages I use on a regular basis.

eshell	paredit	ace-jump	dired
multiple-cursors	helm	AucTex	tdd-mode
latex-preview-pane	undo-tree	magit	restclient
expand-region	yasnippet	org-mode	

eshell

eshell is a terminal written entirely in lisp. It is mostly compatible with bash, but it provides some additional features, like the ability to evaluate elisp directly in the shell.

It has some nice integrations with common tools, like grep and find.

yasnippet

Every editor needs a decent snippet mechanism. Yasnippet provides all the ability you would expect from such a package, with again the added benefit of being able to embed elisp directly into the snippet.

This means your snippets can be as arbitrarily complex as you want to make them. You can also generate them on the fly, like this awesome PHP package.

multiple-cursors

Sublime Text really made multiple cursor support very popular. So of course the emacs community adopted it.

Some nice things to notice:

- limit viewing to those lines with multiple cursors
- yank-ring per cursor

restclient

Postman is a cool like extension to Chrome to perform web requests. But there are a few things I don't like.

1. There isn't a convenient way to comment on what requests mean so I can refresh my memory when I return to a project later one. This means building up a multi-step workflow is a pain.

2. You can't easily share pre-saved requests.

3. Editing json bodies in the extension isn't the easiest.

Luckily for us, restclient addresses all of those problems.
An example restclient document

AucTex / latex-preview-pane

I don't do as much LaTeX as I once did, I still have an occasion to. For anyone that has ever worked with mathematical formulas in LaTeX can understand how hard it is to know that you got it right.
LaTeX with math equations
While I'm workin on a file, it is sometimes nice to be able to check on my progress and make sure everthing is coming out okay. Luckily, emacs has a nice package that let's me do just that.

dired

dired is the directory editor for emacs. There is tons of stuff we could talk about with dired, but what I really want to show is how treating everything like tests helps us rename files.

projectile

projectile is a project management tool. It let's you do

- jump between projects
- search for files within a project
- search within files within a project
- and so much more

helm

helm is THE package I use more than any other. It is an "incremental completion and selection narrowing framework".
It let's you do fuzzy matching for opening files, running command, consulting the man pages and potentially anything you want to do where multiple options might exist.

magit

When I was a full time vim user, I started using fugitive to interact with git from within vim. And I thought that was the bees knees.

When I switched to emacs, I tried out magit and realized how awful fugitive is in comparison. With the ability to easily interactively stage hunks, check the log, add remotes, make branches and do merging, magit makes working with git a breeze.

org-mode

It seems everyone these days is crazy for markdown. But markdown is child's play next to the power of org-mode.

org-mode started out as a note taking application built on top of outline-mode. It has since grown to include:

- an agenda
- clock in and out of tasks
- maintain check lists
- schedule and set deadlines for TODOs
- define code snippets
- export the file into various formats
- so much more

Some Recent Workflows

I made the assertion that emacs is all about efficient workflows. Now that we have seen some various packages I use, let's demo a couple of processes that I have found useful.

Reordering Database Records

At work, we have a table of sources that list where a user might have heard about our service. When we populated this list, we didn't put them in the order marketing prefers, so we need to fix that ordering.

Database seed file

View the Diff of a Change

There is a change in this file and I want to see which commit introduced it.

There are a couple of solutions. We can use git-time-machine or we can search the git log for it.

vim within emacs?

For those of you that might be interested in learning more about emacs but don't want to give up modal editing, there are a couple of solutions.

Wrap up

So that's it! If you have any questions, you can always contact me on twitter at @keelerm84

This presentation is available at `https://github.com/keelerm84/emacs-talk`.

You can view the org file directly since GitHub handles this format. But what if you prefer a reveal.js presentation? Or PDF?

Also, I should probably tweet out the link.