# Emacs Lisp or Why Emacs' Extension Language Is Worth Another Look

Vasilij Schneidermann

August 24, 2014

# Outline

Section 1

# Introduction

# Speaker

- Vasilij Schneidermann, 22
- Information systems student
- Working at bevuta IT, Cologne
- v.schneidermann@gmail.com
- https://github.com/wasamasa

# Preliminary notes

- Pretty subjective at times
- Prepare for dogfooding

# What this talk will be about

- Emacs features
- Demonstrations of what Emacs can do
- The community

# What this talk will not be about

- Teaching you how to use Emacs
- Editor wars

Section 2

# How I got started with Emacs and Emacs Lisp

# How I got started with Emacs and Emacs Lisp

- Started out with switching text editors constantly
- Became curious, learned Vim
- Wanted more, tried Emacs
- Stuck with Emacs, didn't want to learn Emacs Lisp at first
- Curiosity took over, read sources of small packages
- Learned to prefer reading source over docs
- Small fixes at first, wrote own packages later
- Eventually dug in deep enough to hold a talk about it

Section 3

# Why I didn't want to learn Emacs Lisp at first

# It's a Lisp, Lisps are functional languages!

- Lisp doesn't mean it's a functional language
- Emacs Lisp itself is rather procedural
- dash.el helps if you want it to be more functional

# It's a Lisp, therefore it must be useless!

- Emacs is (probably) the largest open Lisp project out there
- There's a few thousand packages one can install

# So, there must be nothing useful left to write anymore!

- There's more than enough things lacking
- Add your own ideas and you'll have something useful to write

# I want to learn a real Lisp first!

- It is a real Lisp and a good starting point
- If you can't decide which one to go for, learn it first, then proceed depending on how much you like it

# I don't want to learn a completely different language just to customize a text editor!

- Starting out is very simple
- Transition to more complex code is gradual

# The existing tutorials and the manual are too intimidating, I want something more approachable!

- Introduction to reading code and customization: http://sachachua.com/blog/series/read-lisp-tweak-emacs/
- Minimal tutorial, REPL-centric: http://bzg.fr/learn-emacs-lisp-in-15-minutes.html
- More traditional introduction to concepts: http://harryrschwartz.com/2014/04/08/an-introduction-to-emacs-lisp.html
- Exactly what it says on the tin: http://steve-yegge.blogspot.com/2008/01/emergency-elisp.html

Section 4

# History

# History

- RMS disliked Unix, had the idea to create a completely free OS
- He started writing his own compiler, didn't like Vi
- He started writing an extensible editor that was able to do more than a mere text editor would
- He chose Lisp as the extension language everything apart the fundamentals would be implemented in
- He also made it free to distribute and added a clause that people had to contribute improvements back, way before they were using DVCS
- Later development moved from the cathedral to the bazaar style

Section 5

# Strengths

# Rich runtime

- Lots of Emacs Lisp tooling
- Serialization/Unserialization of XML, HTML, JSON
- Datetime/Calendar, Color, Unmarshaling
- File handling, recoding
- Numerical analysis, graphing
- Parsers, DBus, Terminal Emulation
- Wrappers for Mail, IRC, Printing, VCS, GPG, ...
- Network processes and access/requests
- Process control
- ...

# Event-driven

- Color selection with mouse (vivid-rodent.el)

# Event loop

- Play back frames with timeout, control playback (flipbook.el)

# Buffers are powerful

- State visualization (svg-2048.el, svg-2048-animation-demo.el)

# Complex UI is possible

- Trigger evaluation in different buffer with keyboard input (dial.el)
- Magit and makey, org-export UI

# More productivity

- Access often used functionality in a simpler way (helm-fkeys.el)

# Better workflow

- Switch window configurations in a simpler way (eyebrowse)

# Immediate feedback loop

- *commence fixing/writing code to make a more practical point* (svg-2048.el)

Section 6

# Weaknesses

# No APIs / Crufty APIs

- Very little or weird abstraction

# Speed

- Need to escape to external processes / FFI
- Byte-compilation helps a bit (with macros)

# Historical mistakes

- The C codebase is scary
- Complexity of the display engine
- No namespaces
- BZR
- Weird naming conventions

# There's still a lot to be fixed

**Just because it works, doesn't mean it's fixed.**
Volkswagen Genuine Parts.

Das Auto.

Section 7

# What do?

# Programmers

- Join the Mailing List, hang out on *#emacs* at Freenode
- Improve your Emacs Lisp skills
- Understand existing code, discuss and question it
- Write demos to find better approaches to a problem

# Designers & Writers

"Design is about pulling things apart." - Rich Hickey

- Gifcasts
- Clearer documentation
- Suggest (UI) ideas, discuss them
- Devise APIs and better abstractions

# Rewrite proponents

See Guile Emacs

# Possible stuff to hack on

- A “native” torrent client
- Guile Emacs and things using Guile bindings (graphical browser, video player, OpenGL, . . . )
- dired
- Window management library
- Input methods
- helm
- dash.el, s.el, f.el, b.el, . . .
- my stuff
- other people’s stuff (see next slide)

# Hackers to collaborate with

- Fuco1
- magnars
- skeeto
- chrisdone
- purcell
- thierryvolpiatto
- bbatsov
- technomancy
- dgutov
- ...

# Conclusion

- Emacs is pretty cool
- You should totally learn to mold it to your likings
- If you do, help out while you're at it
- There's more than enough to be fixed

# Questions?

“<technomancy> not making sense never stopped an intrepid elisper!”