

Exploring documentation of GNU Emacs and Org-mode

What these projects are doing right

Bastien Guerry – <https://bzg.fr>

March, 28th 2019

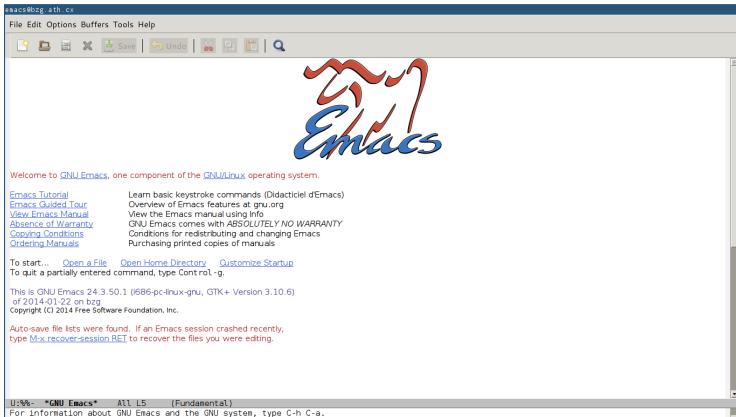
Emacs ancestor: TECO (1962)

- ▶ TECO : Tape/text Editor and COrrector
- ▶ An editor and a language to write text editing macros
- ▶ The language was interpreted and imperative
- ▶ Bad reputation as a "write-only" language (APL, ...)
- ▶ "TECO is not a text editor, it is a programming language"
- ▶ TINT: TINT Is Not TECO (very first recursive acronym)
- ▶ MUNG: MUNG Until No Good (cli for TECO)

Emacs (1976): Editing MACroS running on TECO

- ▶ RMS extends TECO: real-time full-screen mode, active keys
- ▶ First Emacs (1976) was written as a set of TECO macros
- ▶ Both an editor and an environment to run editing macros
- ▶ Modeless editing by default (vs Vi modal editing)
- ▶ Emacs code-base is using C and Emacs Lisp

Emacs initial screen

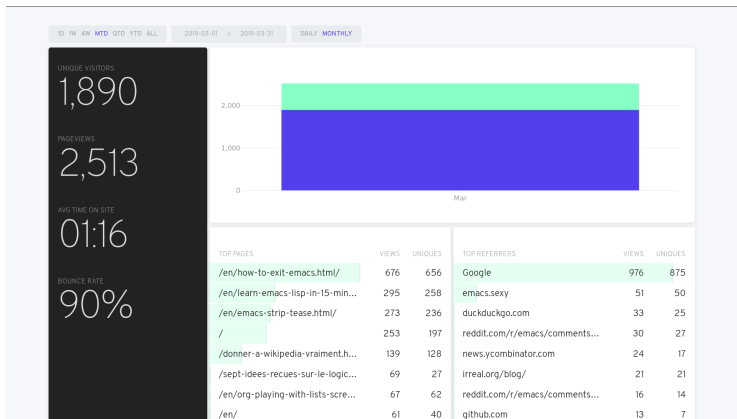


Emacs initial screen elements

- ▶ Link to a tutorial
- ▶ Link to a guided tour
- ▶ Link to the Emacs manual
- ▶ Link to ordering a manual
- ▶ Quick start: open a file/directory
- ▶ Quick help: recover a file

Anything missing?

(How to quit Emacs?)



(Not such a bad question after all)

The screenshot shows a Stack Overflow question page. At the top, the Stack Overflow logo and a search bar are visible. The question title is "How to exit the Vim editor?". The question body contains the text: "I'm stuck and cannot escape. It says: 'type :quit<Enter> to quit VIM' But when I type that it simply appears in the object body." The question has 3175 votes and 791 answers. It was asked 6 years, 7 months ago and viewed 1,881,807 times. The question is protected by Tim Gautier. The right sidebar shows featured questions and hot meta posts.

stackoverflow Search...

How to exit the Vim editor? [Ask Question](#)

I'm stuck and cannot escape. It says:

3175 "type :quit<Enter> to quit VIM"

But when I type that it simply appears in the object body.

vim vi

791

share edit flag

edited Nov 3 '16 at 20:26 Peter Mortensen 13.8k ● 19 ● 87 ● 113

asked Aug 6 '12 at 12:25 jclancy 15.3k ● 5 ● 21 ● 28

protected by [Tim Gautier](#) May 25 '17 at 22:25

This question is protected to prevent "thanks!", "me too!", or spam answers by new users. To answer it, you must have earned at least 10 [reputation](#) on this site (the [association bonus](#) does not count).

asked 6 years, 7 months ago
viewed 1,881,807 times
active 1 month ago

FEATURED ON META

- Updating the Hot Network Questions List - now with a bit more network and a...
- 2019 Community Moderator Election Results
- The Ask Question Wizard is Live!
- Should we burninate the [like] tag?

HOT META POSTS

18 In which Stack Overflow umbrella of sites should I ask a OO design question?

Fundamental Emacs design elements

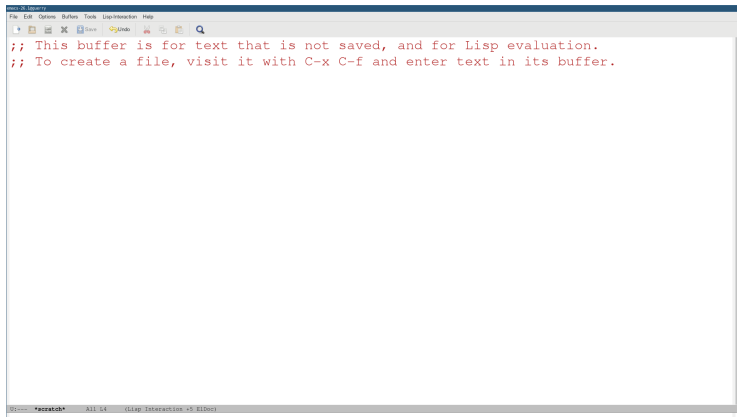
- ▶ menu-bar and tool-bar: make commands more accessible
- ▶ A buffer: the place to edit text
- ▶ A cursor: which state can give some information
- ▶ A modeline: a read-only place for quick info
- ▶ An echo area: a read/write place for quick interactions
- ▶ A fringe: where to display indicators
- ▶ A margin: for indicators (e.g. \ in the right margin)
- ▶ A header-line: for more modeline-like informations
- ▶ Transient regions: highlight selected text
- ▶ Scroll bars: visual clues on where you are
- ▶ Help text and tooltips: information on active text
- ▶ Text properties: for hints on syntax or actions
- ▶ Overlay properties: for more hints on actions

Beyond Emacs design elements

Emacs design **extensible** and **configurable** and place **context** at the heart of every interaction.

- ▶ `linum-mode`: display line numbers
- ▶ `hl-line-mode`: highlight current line
- ▶ `guide-key-mode`: display available keybindings
- ▶ `helm-mode` & `ido-mode`: contextual minibuffers
- ▶ `M-x doctor RET`: when you're really really lost

Example: Emacs scratch buffer



The screenshot shows an Emacs window titled "Emacs - 100x100". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Lisp-Interaction", and "Help". The toolbar contains icons for "Save", "Undo", "Redo", and "Search". The main text area contains two lines of red text: `;; This buffer is for text that is not saved, and for Lisp evaluation.` and `;; To create a file, visit it with C-x C-f and enter text in its buffer.` The status bar at the bottom shows "0:-- *scratch* All 14 (Lisp-Interaction > 110cc)".

```
;; This buffer is for text that is not saved, and for Lisp evaluation.  
;; To create a file, visit it with C-x C-f and enter text in its buffer.
```

Example: guide-key-mode

```
C-x r-
[C-#] point-to-register
[ESC] Prefix Command
[SPC] point-to-register
[*] increment-register
[L] register-list
[M] bookmark-set-no-overwrite
[N] rectangle-number-lines
[b] bookmark-jump
[c] clear-rectangle
[d] delete-rectangle
[f] frameset-to-register
[g] insert-register
[i] insert-register
[j] jump-to-register
[k] kill-rectangle
[l] bookmark-bmenu-list
[m] bookmark-set
[n] number-to-register
[o] open-rectangle
[r] copy-rectangle-to-register
[s] copy-to-register
[t] string-rectangle
[w] window-configuration-to-register
[x] copy-to-register
[y] yank-rectangle
[C-SPC] point-to-register
[M-w] copy-rectangle-as-kill
```

Emacs documentation-related commands

- ▶ `C-h a` : search for symbol or command
- ▶ `C-h g` : open the HTML manual in a browser
- ▶ `C-h t` : open the Emacs tutorial
- ▶ `C-h k` : type a key and get the command
- ▶ `C-h f` : search for function or commands
- ▶ `C-h v` : search for variables or options
- ▶ `C-h C-d` : help for debugging Emacs
- ▶ ...
- ▶ `C-h C-h` : display more help commands

There is a **dedicated help-mode** to display help information and documentation-oriented commands like `info` and `man`.

Emacs documentation materials and tools

Documentation materials:

- ▶ The GNU Emacs manual and guided tour
- ▶ Many online tutorials and screencasts
- ▶ <https://www.emacswiki.org>
- ▶ <https://www.reddit.com/r/emacs/>

Documentation tools:

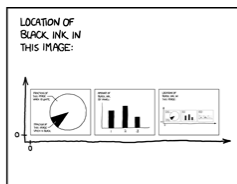
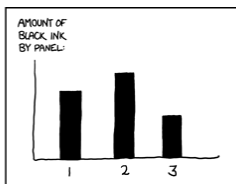
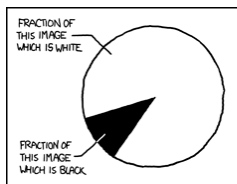
- ▶ GNU coding standards
- ▶ Recommendations on writing documentation
- ▶ M-x checkdoc RET: Emacs Lisp docstrings linter

Emacs: really "self-documenting"?

"Is Emacs better at documenting itself than Google?"

Terminology is still a blocker for beginners:

- ▶ yank => copy
- ▶ kill => cut
- ▶ window => pane
- ▶ frame => window
- ▶ kill buffer => close buffer



What about Org-mode?

```
#+TITLE: Example .org file
#+AUTHOR: Bastien
#+DATE: 2012-09-08 sam.

* Make a new website for orgmode.org

** TODO Make screenshots
   SCHEDULED: <2012-09-07 ven.>

** NEXT [#A] Add them to the website
   SCHEDULED: <2012-09-08 sam.>

** NEXT Publish the new website
   DEADLINE: <2012-09-09 dim.>

* Plain-text tables and spreadsheet...
* Working with source code █
```

```
--- website.org Top L27 [0] [(Org +5 Fill)] sam. sept. 8 08:15 0.30
(No changes need to be saved)
```

Eating our own dogfood

- ▶ Org-mode is both a **text editing/publishing** tool and a **todo list** manager
- ▶ Org-mode is used to **write documentation** (and README.org on Github)
- ▶ Org-mode is used to **track and display bugs** (M-x debugs-org RET)

TODO central file for Worg

[\(Back to Worg's index\)](#)

This file is the central page for inserting tasks related to Worg.

If you keep a local copy of worg in `~/org/Worg/`, then you can add `~/org/Worg/worg-todo.org` to the list of your agenda files.

Then a simple `git pull` will update the tasks in this file, and your agenda view will be populated with Worg-related tasks.

Tutorials

Gather examples of custom agenda views from the mailing list

Tutorial about links (export and abbrev)

Tutorial about the LaTeX exporter

General tasks

TODO Create a "user" directory where users can have their own pages

TODO Build the FAQ from [the survey](#)

TODO Define column views for important files in Worg

TODO Start a FAQ for Org from the mailing list

TODO Make a better .css file for the HTML export?



Org-mode and the Joel test

Do you use source control?	Yes, Git
Can you make a build in one step?	Yes
Do you fix bugs before writing new code?	Generally
Do you have a spec?	For elements
Do you use the best tools money can buy?	Yes, Emacs
Do you have testers?	Yes, users
Do you do hallway usability testing?	Not enough
<hr/>	
Do you have a bug database?	NO (Well, yes.)

See *The Joel Test: 12 Steps to Better Code*

Basic facts about org-mode development

- ▶ There are **22087 commits** as of 2019-03-28
- ▶ We don't have a roadmap (but good willing users)
- ▶ We don't use Github (but code.orgmode.org)
- ▶ We don't have a bug tracker (but a mailing list)
- ▶ We have a single mailing list for developers and users

Github made it easy to report issues and to start projects: it does not mean this "default" widely used interface is not questionable.

Facts about org-mode and its documentation

- ▶ We have both a manual and a "compact" guide
- ▶ We have a book version of the manual for Org 7.0
- ▶ We started Worg, a git-based collaborative documentation
- ▶ The Org manual, guide and worg docs are .org files
- ▶ We taught users how to [give useful feedback](#) in the manual
- ▶ We promote the notion of "ECM" (complete minimal example)
- ▶ The mailing list is welcoming, a place to learn

For the Org 7.0 book, we received the help of a professional editor, which taught us a lot.

Worg: collaborative documentation

Hello Worg, the Org-Mode Community!

[Sitemap](#) and [index](#)

Introduction to Org-Mode and Worg

Org-mode is a powerful system for organizing your complex life with simple plain-text files. It integrates all your notes, mindmaps, TODO lists, calendar, day planner, and project schedule that can be easily searched (e.g. by `grep`), encrypted (e.g. by `GnuPG`), backed up and sync imported/exported, and [accessed on the go](#) (e.g. on an iPhone or Android smartphone). It is used for authoring web pages and documents.

Check out some [screenshots of Org-mode](#) in action. See [what people have to say](#) about Org-mode, and read a few [user stories](#)!

Org-mode is distributed as part of the popular `Emacs` text editor and runs wherever Emacs runs, including on GNU-Linux, Windows, and Mac. Written by Carsten Dominik, it is currently maintained by Bastien Guerry and used by many helpful people who, like you, are desperate to get and stay organized.

The page you're reading is part of `Worg`, a section of the [Org-mode web site](#) that is written by a volunteer community of Org-mode fans. It includes tutorials, ideas, code snippets, etc., shared to make your introduction and customization of Org-mode as easy as possible. Worg is maintained by Matthew Lundin, a group of [Worg contributors](#), and maybe [you](#).

Table of Contents

- [Introduction to Org-Mode and Worg](#)
- [Org-Mode Resources](#)
 - [Learn Org-Mode](#)
 - [Library of Babel](#)
 - [Get Help with Org-Mode](#)
 - [Use Org-Mode Effectively](#)
 - [Dive Deeper into Org-Mode](#)
 - [Check Out Third-Party Contributions](#)
- [Join the Community](#)
 - [Help out](#)
- [Maintenance of Worg](#)
 - [Latest changes to the Worg git repository](#)
- [Git'r done!](#)



A mailing list as a bug tracker, are you insane?

- ▶ Bugs get a very large exposure
- ▶ It promotes a collective sense of responsibility
- ▶ Each bug is discussed in a unique place (a thread)
- ▶ It is easy to refer to bugs with a simple URL
- ▶ Patches are all discussed on the list

Yes, we can do better

- ▶ Enhance documents about Org syntax and elements
- ▶ Fix obsolete resources on Worg
- ▶ Test Org-mode with beginners
- ▶ Test Org documentation with beginners
- ▶ Publish documentation for the Org stable **and** unstable
- ▶ ...

Resources

- ▶ fr.wikipedia.org/wiki/TECO
- ▶ fr.wikipedia.org/wiki/Emacs
- ▶ Where does the name "Emacs" come from?
- ▶ Stack Overflow: Helping One Million Developers Exit Vim
- ▶ How do I undo the most recent commits in Git?
- ▶ How to effectively use the self-documenting system of Emacs?
- ▶ The Joel Test: 12 Steps to Better Code

The XKCD drawing is published [here](#) under the CC-by-nd 2.5 license.