

Emacs最前線

Emacs front line



pixiv Inc.
USAMI Kenta

pixiv

お前誰よ



- うさみけんた (@tadsan) / Zonu.EXE / にゃんだーすわん
- ピクシブ株式会社 pixiv事業本部 エンジニア
 - 最近はピクシブ百科事典(dic.pixiv.net)を開発しています
- Emacs Lisper, PHPer
 - Emacs PHP Modeを開発しています(2017年-)
- TechFeed公認PHPエキスパート

Modern editor - independent development environment for PHP

Japanese

Abstract

Traditional text editors have been favored as an antithesis against heavy IDE. In today's software development, however, the power of the IDE is increasingly recognized even in the project using a dynamic language, and sometimes it is recommended over bare text-editor. Considering that, I will introduce methods to add IDE-like language support to text-editor, especially I will introduce editor-agnostic tools for intelligent PHP development and explain its application to Emacs and Vim.

Speaker

USAMI Kenta



Vim user from 2011. Web programmer in pixiv Inc, and current maintainer of Emacs PHP Mode.

[Ticket](#)[Keynote
speakers](#)[What's
VimConf
2019](#)[Time
table](#)[Official
Blog](#)[Sponsors](#)[Individual
Sponsors](#)[Sessions](#)[Blogs](#)[Staff](#)[Code of
Conduct](#)[P](#)

Vim as a text processor

[Japanese](#)

Abstract

Vimは高機能なテキストエディタであるばかりか、その機能はVimをバッチモードで起動し、Vim scriptを通じて利用できます。つまりVimはヘッドレスなテキストエディタ、つまりテキストを操作するプロセッサでもあります。このLTではリファクタリングなどのテキスト整形に役立つVimの活用方法について手短にお伝えします。

Speaker

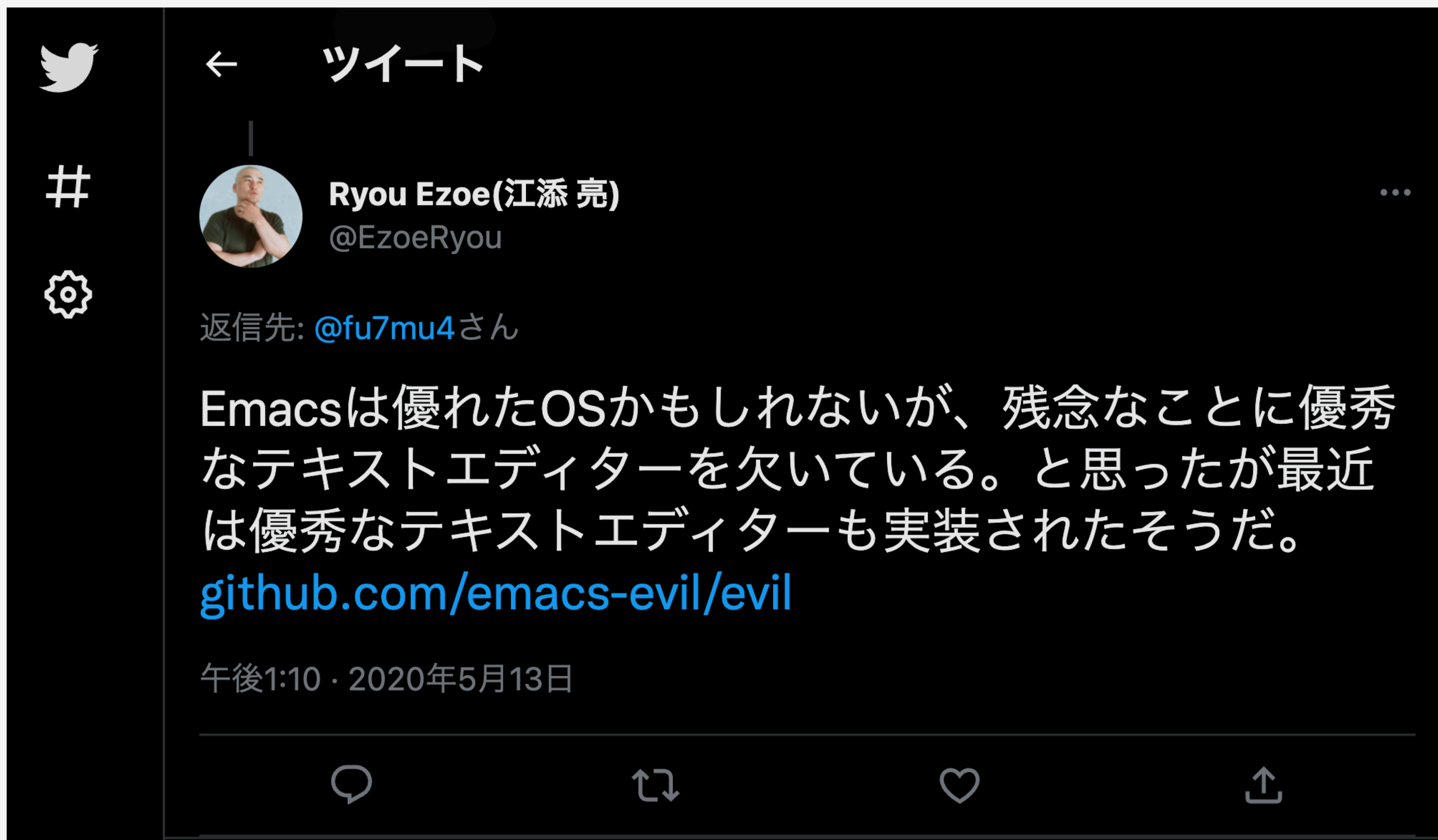
うさみけんた

私の立ち位置



さて

Vimはいいぞ！



Emacsの 本質とは何か

A modern Lisp machine

≡ LISPマシン

📄 15の言語版 ▼

ページ ノート

👁️ ソースを編集 履歴表示 ☆ その他 ▼

出典: フリー百科事典『ウィキペディア（Wikipedia）』

LISPマシンは、**LISP**を主要な**プログラミング言語**として効率的に実行することを目的として設計された汎用の**コンピュータ**である。ある意味では、最初の商用シングルユーザー**ワークステーション**とすることもできる。それほど数量的に大成功を収めたとはいえないが（1988年までに約7000台が出荷された^[1]）、その後よく使われることになる様々な技術を商用化する先駆けとなった。例えば、効率的**ガベージコレクション**、**レーザープリンター**、**ウィンドウシステム**、**コンピュータマウス**、高解像度**ビットマップ**グラフィックス、**CHAOSNet**（英語版）などのネットワーキングにおける技術革新などである。1980年代に**シンボリックス**（3600、3640、XL1200、MacIvoryなど）、LMI（**Lisp Machines Incorporated**、LMI Lambda）、**テキサス・インスツルメンツ**（**Explorer**、MicroExplorer）、**ゼロックス**（**InterLisp-D**搭載ワークステーション）といった企業がLISPマシンを製造販売した。**オペレーティングシステム**は **Lisp Machine Lisp**（英語版） や **Interlisp**（ゼロックスの場合）で書かれ、後に一部は **Common Lisp** で書かれた。

歴史 [\[ソースを編集\]](#)

背景 [\[ソースを編集\]](#)

そういう機械が
40年以上前にあった



図1 AI ワークステーション ELIS

Environment (cf. GNU's Not Unix) のことではないかと鋭く指摘してくれたが正鵠を突いている。

現在、ELIS は NTT、沖電気工業などが出資して作った NTT の子会社 NTT インテリジェントテクノロジー (NTT-IT) から約1000万円で販売されている。(この価格は Lisp マシンとしてはかなり安いと思うが、EWS として考えると高い。今後こういうマシンの真の競争相手は既存の Lisp マシンではなく UNIX ベースの EWS になるはずだから、マルチユーザで AI 向きという付加価値を加味してももう少し安いことが望まれる。もっとも、私は NTT-IT の経営に口をはさむ立場ではないので、これはあくまでも私の個人的願望だ。) というわけで、商品としての TAO/ELIS と我々 (主に TAO 周辺のソフトウェア担当者) の研究材料としての TAO/ELIS の切り分けが一時ちょっと微妙だったが、現在いろいろな関係が整理され、我々はまた研究材料としての TAO/ELIS に没頭できる状態になっている。今後の我々の研究の成果の中にもし「使える」ものがあれば、それを世に出すことのできるパイプが布設されたというわけである。

NTT ソフトウェア研究所の私の同僚たちは、これから TAO/ELIS の上に智能処理のための統合的プログラミング環境 NUE (New Unified Environment) を作り上げる研究を進めようとしている。NUE のキャッチフレーズは、「智能処理ソフトウェアの研究開発の場において研究者や開発技術者の創造性を最大限に発揮させる高性能・高機能のプログラミング環境」となかなか勇ましくて格好いいのだが、いまところ具体的なイメージは定まっていない。NUE (鵜) という名前からしてこれは当然かもしれない。ある人が NUE は NUE's Undefined

この写真は国産LISPマシンのELIS
なので本家Emacsの開発経緯とは直
接関係ないのだが内蔵されたZENと
いうEmacs風独自エディタの日本語
入力メソッドKanzzenが後のSKKKに
繋がりにIME史に影響を及ぼしている

それは置いて

AIならLisp
CGならLisp

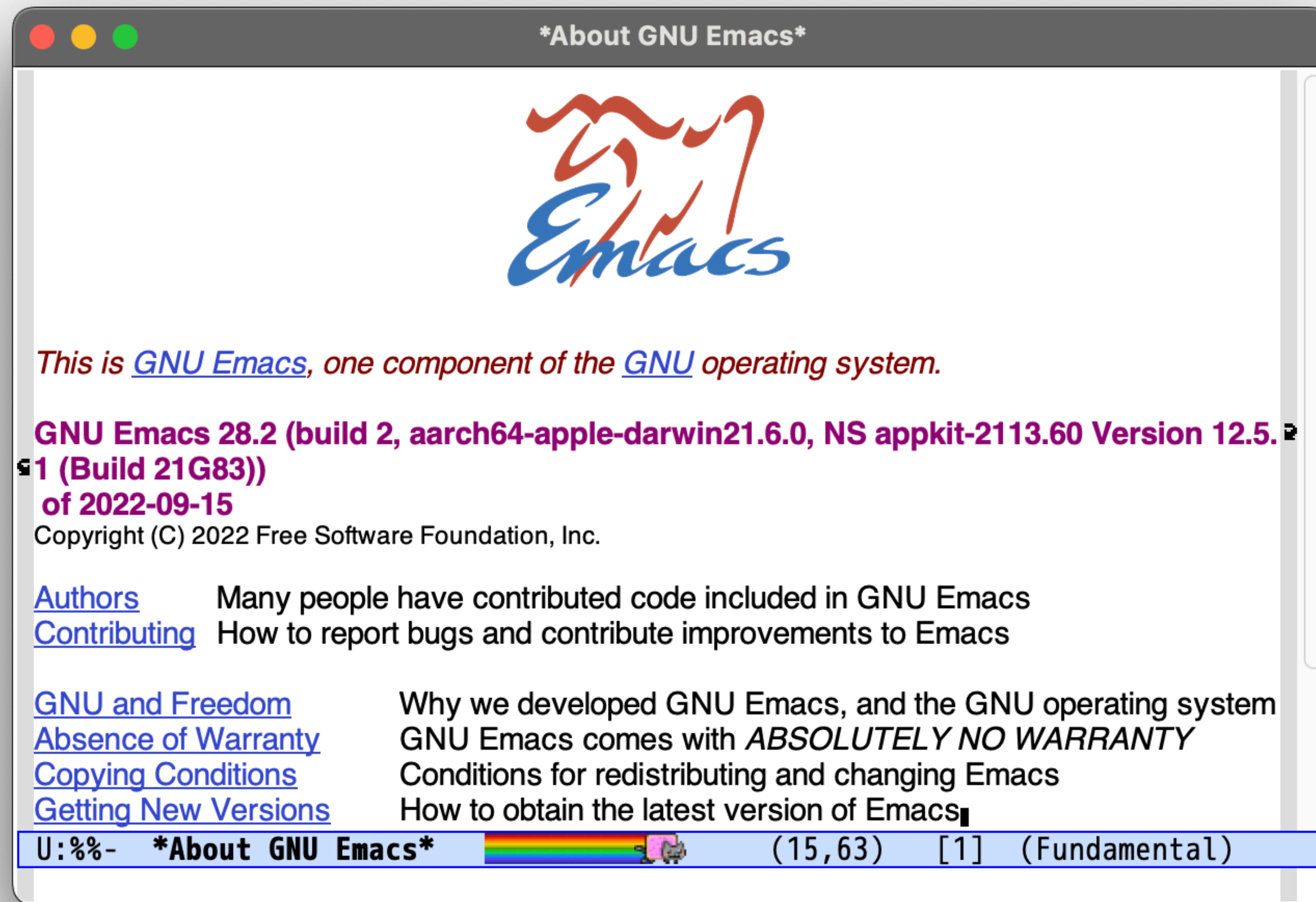
そういう時代もあった
(らしい)

みなさんご存じ(ない)通り
LISPマシンのような
高級言語専用マシンは
廃れている

199X年、世界は核の炎に
包まれた。海は枯れ、地は
裂け、全ての生物が死滅し
たかのように見えた。

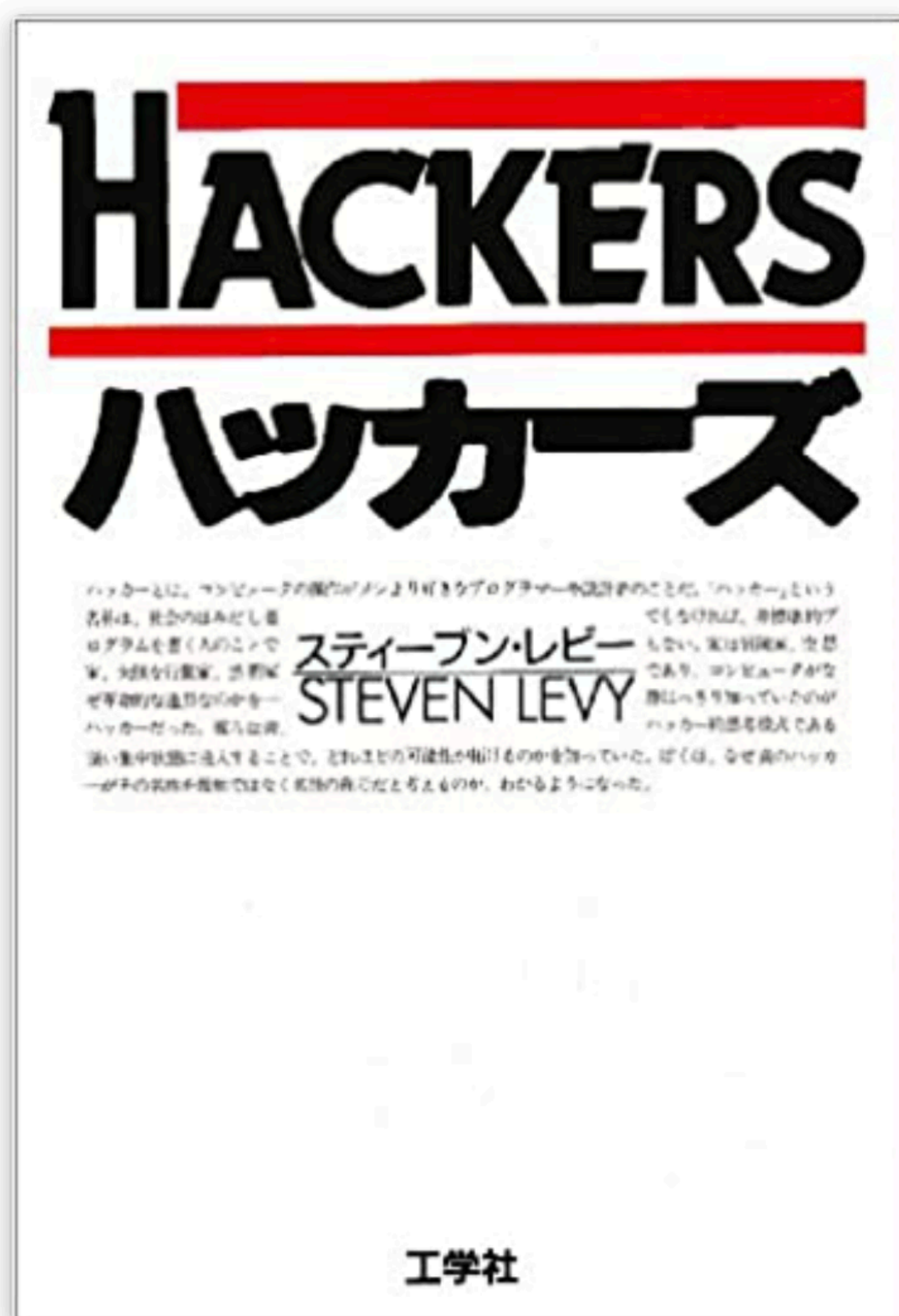
だが、Lispは
死滅していなかった！

世紀末救世主伝説Emacs



元祖Emacsは
TECOというエディタの
拡張マクロ集
(EditorMACroS)

なんだかんだありまして



ハッカーズ 単行本 – 1987/3/1

スティーブン・レビー (著), 松田 信子 (著), 古橋 芳恵 (著)

★★★★☆ 22個の評価

すべての形式と版を表示

単行本

¥2,750

獲得ポイント: 83pt ✓prime

¥1,253 より 14 中古品

¥2,750 より 13 新品

出版社



工学社

発売日



1987/3/1

現代一般に“Emacs”と
呼ばれるのは1985年に
発表されたGNU Emacs

かつてGNU Emacsは
「伽藍とバザール」
と例えられた

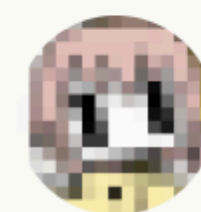
現代でもGitHubやGitLab
ではなくMLで開発されている
が、パッチを投げたら気軽
に取り込んでくれる雰囲気

むかしは歴史的経緯で
サードパーティのリポジト
リを追加しなければ
PHPサポートできなかった

スクリプトで拡張できる テキストエディタの元祖

現代でも健在

Emacsのバージョン



Author: zonuexe

Published: 2019-01-13

Last Modified: 2022-09-12

GitHub Source: [md](#)

最新バージョンについて

歴史的には多様なEmacsがありますが、今日においてEmacsと呼ばれるのは、もっぱら **GNU Emacs** です。

GNU Emacsの最新安定版は **28.2** (2022年9月12日リリース)です。

masterブランチでは次のメジャーバージョンとなる **29.0** 系統の開発が進行しています。

Languages



- **Emacs Lisp** 59.5%
- **Roff** 16.2%
- **C** 15.7%
- **Common Lisp** 4.7%
- **Objective-C** 0.6%
- **M4** 0.6%
- **Other** 2.7%

コードベースの
60%はLisp

EmacsコアはC言語で
書かれているが
割合は15%に過ぎない

(大言壮語ではなく)
あらゆるものをユーザー
がカスタマイズできる
エディタ

Emacsの設定は
init.elというファイルに
記述する

エディタの実装言語と
ユーザーのカスタマイズ
言語が同じ

ところで皆さん

ハッカーになりたいく
ありませんか

普通のやつらの上を行け ---Beating the Averages---

著者：Paul Graham

Copyright 2001 by Paul Graham

これは、Paul Graham: [Beating the Averages](#) を、原著者の許可を得て翻訳・公開するものです。

[プロジェクト杉田玄白](#)正式参加テキスト。

<著作権表示>

本和訳テキストの複製、変更、再配布は、この著作権表示を残す限り、自由に行って結構です。

(「この著作権表示」には上の文も含まれます。すなわち、再配布を禁止してはいけません)。

Copyright 2001 by Paul Graham

原文: <http://www.paulgraham.com/avg.html>

日本語訳: [Shiro Kawai](#) (shiro @ acm.org)

<著作権表示終り>

「普通のやつらの上を行け ---Beating the Averages---」より引用
<http://practical-scheme.net/trans/beating-the-averages-j.html>

(この記事は2001/3/25にケンブリッジで開催された*Franz Developer Symposium* で行った講演をベースにしている。2001/5/3版)

1995年の夏、私は友人のロバート・モリスとともにViawebというベンチャー企業を立ち上げた。エンドユーザーがオンラインストアを自分で作ることが出来るソフトウェアを書く、というのが計画だった。当時、このソフトウェアが新しかったのは、ソフトウェア自身を我々のサーバーで走らせて、通常のWebページをインタフェースにするという点だった。

もちろんたくさんの人達がこのアイディアを思い付いていたのだろうが、私の知る限り、Viawebは最初のWebベースアプリケーションだった。そのアイディアがとても斬新なものに思えたので、私達は自分の会社を そのように名付けたくらいだ：私達のソフトウェアはユーザのデスクトップではなく、webを通して (via web) 動く、というわけだ。

もうひとつ、私達のソフトウェアがユニークだったのは、それが主にLispと呼ばれるプログラミング言語で書かれていたからだ [\[注1\]](#)。それまで主として大学や研究所で使われていたLispの、事実上初めての大規模なエンドユーザ アプリケーションだった。そして、パワーにおいて劣る他の言語を使っている競争相手に対して、Lispは強力なアドバンテージとなった。

「普通のやつらの上を行け ---Beating the Averages---」より引用
<http://practical-scheme.net/trans/beating-the-averages-j.html>

言語にはパワーがある

「ほげ言語」のパラドックス

Lispの何がそんなに素晴らしいんだ？ それに、もしLispがそんなに良いのなら、 どうしてみんなそれを使わないんだ？ これは修辞疑問みたいに聞こえるが、 きちんとした答えのある質問だ。 Lispは、 信者のみが見ることができる魔法の特性があるから素晴らしいんじゃない。 単に、 今ある言語の中でもっともパワフルだからだ。 そして、 みんながそれを使わない理由は、 プログラミング言語とは単なる技術だけでなく、 心の習慣でもあり、 それは最も変化の遅いものであるというものだ。 もちろん、 この二つの答えはもっとよく説明されなければなるまい。

まず、 思いっきり物議を醸しそうな発言から始めよう。 プログラミング言語は、 その力において差がある。

少なくとも、 高級言語は機械語より力があるということに反対する人はほとんど居ないだろう。 こんにちではほとんどのプログラマーは、 通常、 機械語でプログラムを書きたいとは思わない ということに同意するだろう。 かわりに高級言語で書いて、 コンパイラにそれを機械語に変換させるべきなのだ。 この考えは既にハードウェアの設計に採り入れられている。 1980年代以降、 命令セットは人間のプログラマよりはコンパイラ向けに設計されるようになった。

あなたがプログラムの全てを機械語で書くと言い出したら、 誰もがそれは間違いだと言うだろう。 しかし、 同じ原理だが見過ごされがちな一般法則がある。 もしあなたがいくつかの言語を 選択することができて、 他の条件が同じ場合、 最も力のある言語以外を選択することは間違いなのだ [注3]。

「普通のやつらの上を行け ---Beating the Averages---」より引用
<http://practical-scheme.net/trans/beating-the-averages-j.html>

ということを
Yコンビネータ創始者の
ポールグレアムが
言ってる

(viawebで使っていたのはCommon Lispだよというツツコミは躲しつつ)

ハッカーになりたければ Emacs

Emacs Lispも
同じパワーを持ってる
(70%くらい^[要出典])

えっじゃあ
現代にEmacsを使うのは
ハッカーワナビだけなの

そうだよ

~~そうだよ~~
違います

エディタには
パワーがある

実行時にあらゆるものを
再定義できる
最強の動的言語ランタイム

危険な再定義をせずに
hookという仕組みで
容易にカスタマイズ
できる設計が浸透している

Emacsは未完成品

ユーザーがinit.elを
記述することで完成する
フレームワーク

init.elに書かなくても
コードを書き換えて
Emacsの再起動なしに
挙動を変更できる

あらゆるものを
カスタマイズしたければ
Emacs

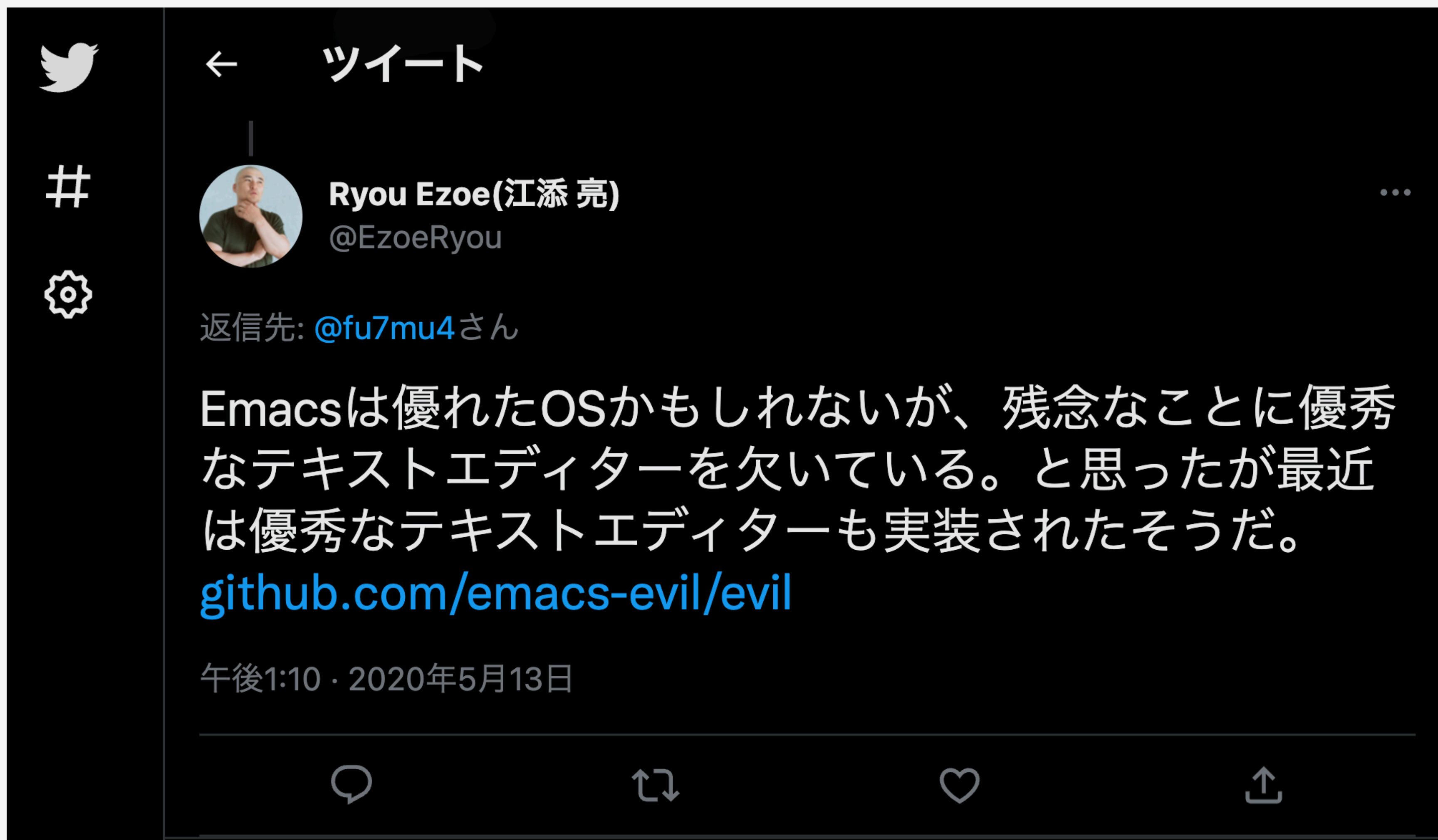
C言語を知らなくても
Lispがわかれば
カスタマイズできる!!!

さて

Emacsはなんでもできる

メーカーでもあり
RSSリーダーでもあり
EPUBやPDFも読める

エディタとして
使いにくい？





Evil is an **extensible vi** layer for [Emacs](#). It emulates the main features of [Vim](#), and provides facilities for writing custom extensions. Also see our page on [EmacsWiki](#).

Installation

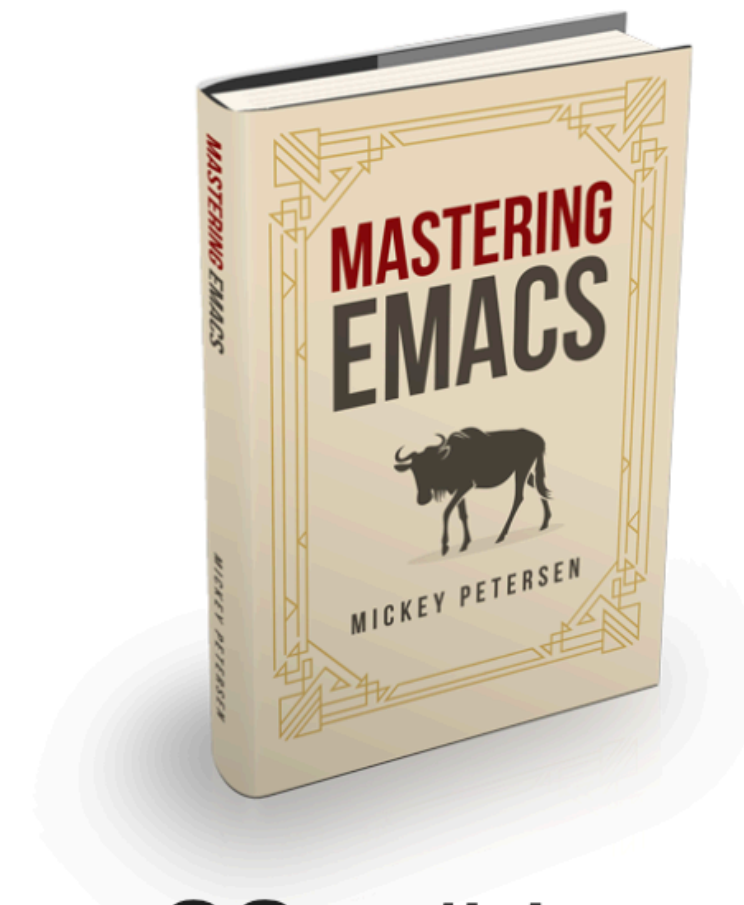
See the [official documentation](#) for installation instructions. We recommend using *package.el*.

As a quickstart, you can add the following code to your Emacs init file.

```
;; Set up package.el to work with MELPA
(require 'package)
(add-to-list 'package-archives
             '("melpa" . "https://melpa.org/packages/"))
(package-initialize)
(package-refresh-contents)
```


思考の速度で編集
したければEvilが使える

Welcome to
Mastering Emacs
mastering the world's best text editor



Emacs 28 edition out now!

[READ A FREE SAMPLE](#)

[LEARN MORE](#)

(私も翻訳の
お手伝いをしました)

Mastering Emacs is now available in Japanese

Thanks to the hard work of USAMI Kenta and AYANOKOJI Takesi you can now read Mastering Emacs in Japanese!

By *Mickey Petersen*
UPDATED FOR EMACS 28



If you're a Japanese speaker, you can now read my book, *Mastering Emacs*, in Japanese. I owe all this hard work to **AYANOKOJI Takesi** and **USAMI Kenta**, two legendary Emacs hackers and writers in the Japanese Emacs community. There's a large Emacs community in Japan but, unless you speak Japanese, you probably wouldn't know that!

It's easy to miss, but Emacs owes its superb Unicode and multilingual support to the Japanese **National Institute of Advanced Industrial Science and Technology (AIST)** who created (and maintained) a separate fork of Emacs called *Nihongo* ("Japanese") Emacs, first created in 1987. Later it was wrapped up into a separate package called **MULE** (**M-x find-library mule**.)

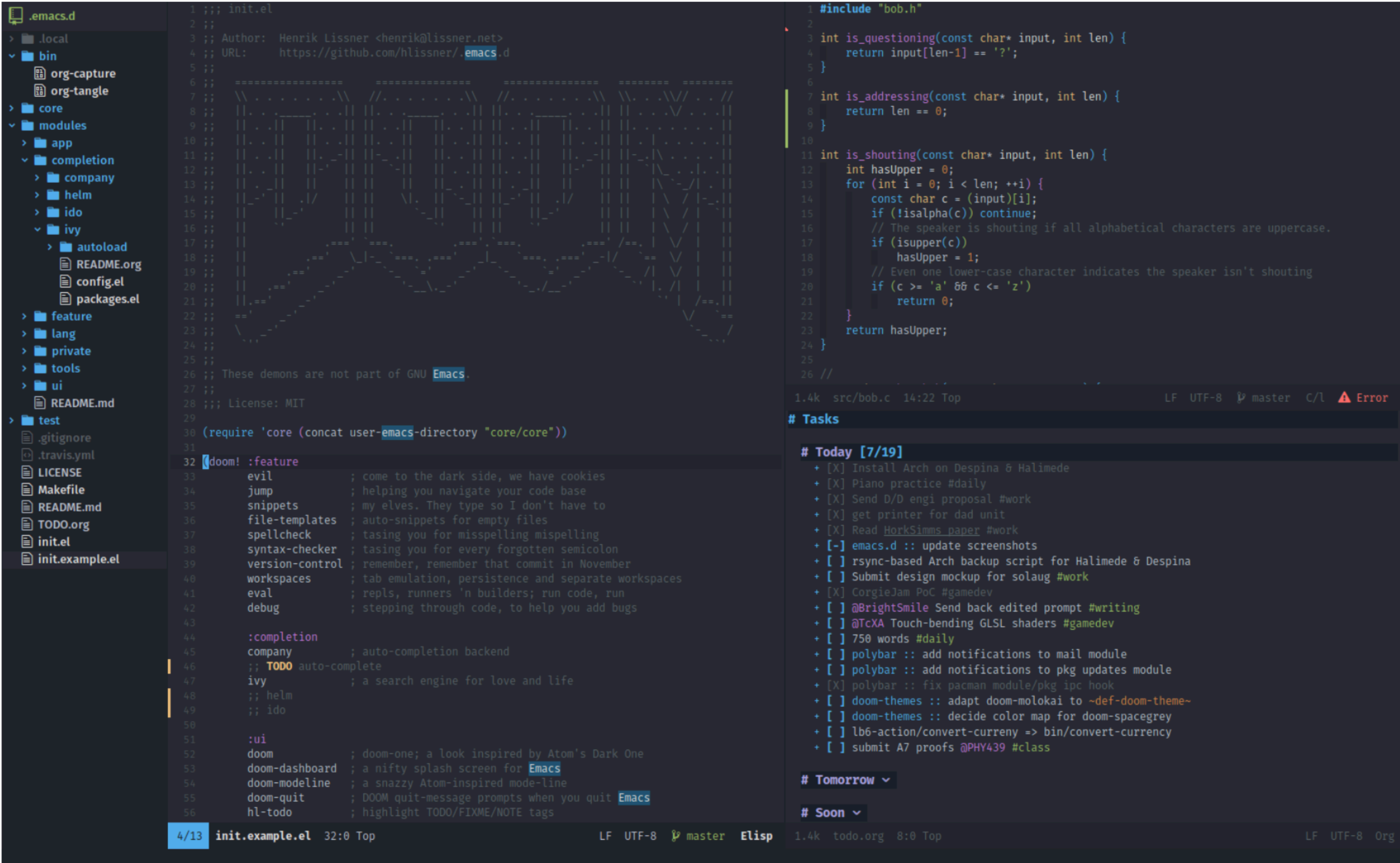
カスタマイズが
めんどくさい？

Doom Emacs

[Install](#) • [Documentation](#) • [FAQ](#) • [Screenshots](#) • [Contribute](#)

release **v2.0.9**  Supports Emacs 27.1-28.1 last commit december 2022 build <https://github.com/badges/shields/issues/8671>

 Discord 1383 online  Discourse 2.4k users



Emacs vs 重力

Emacsは動的言語インタ
プリタでもあるので
GILからは逃れられない

Emacs28では
Lispをネイティブコンパ
イルできるようになった

Lispで言語ごとの
開発環境を
実装し続けるの大変

言語とEmacsLisp
両方わかるひとが
メンテし続けないと
いけない

動的言語Lispで
重厚な処理を
すると重い

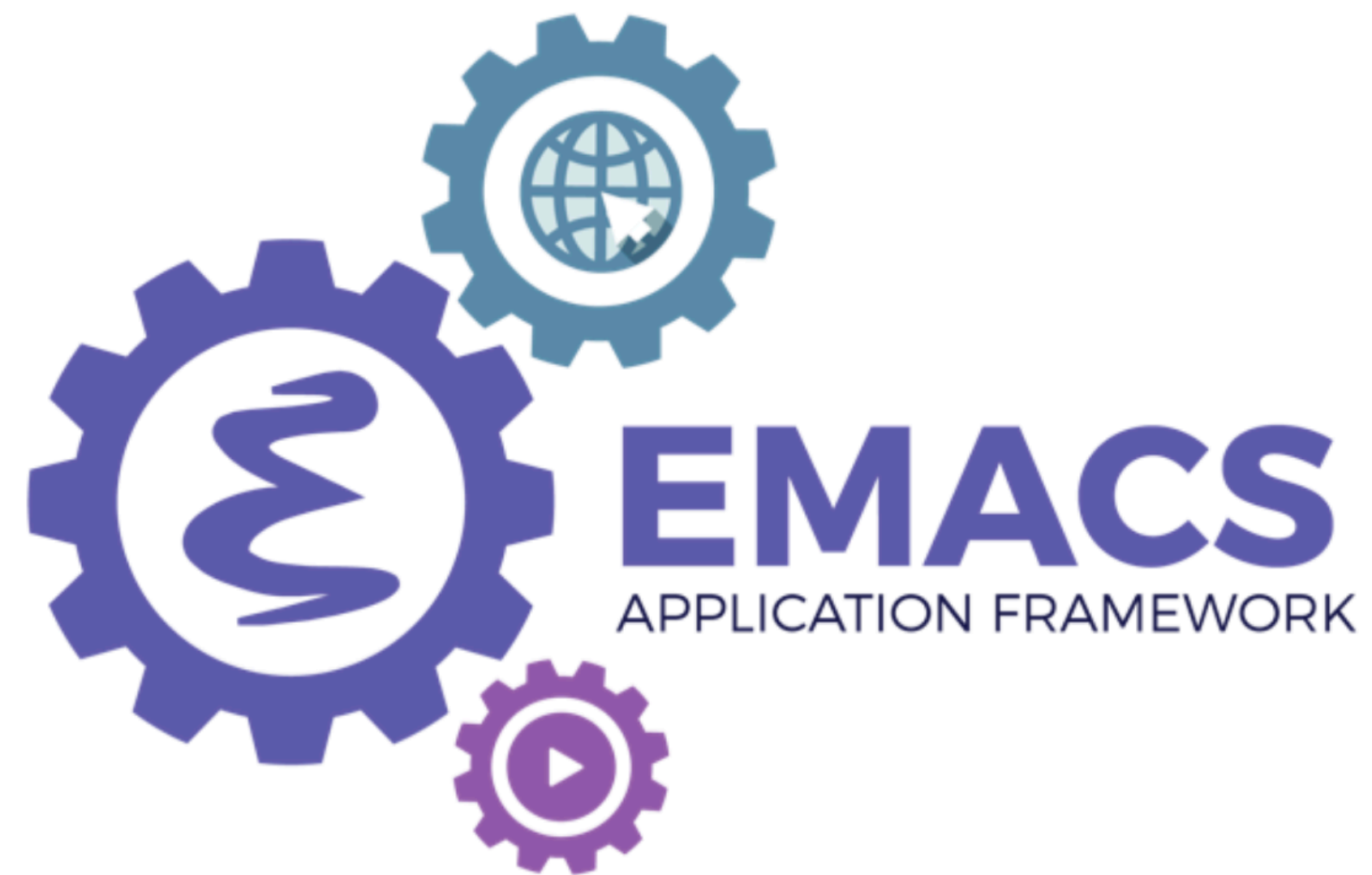
重い関数は
C言語などで実装して
ダイナミックモジュール化

Emacs 29では Tree-sitterベースの 言語モードを導入

問題意識を持って
高速な非同期UIを
実装している人も居る

☰ README.md

English | [简体中文](#)

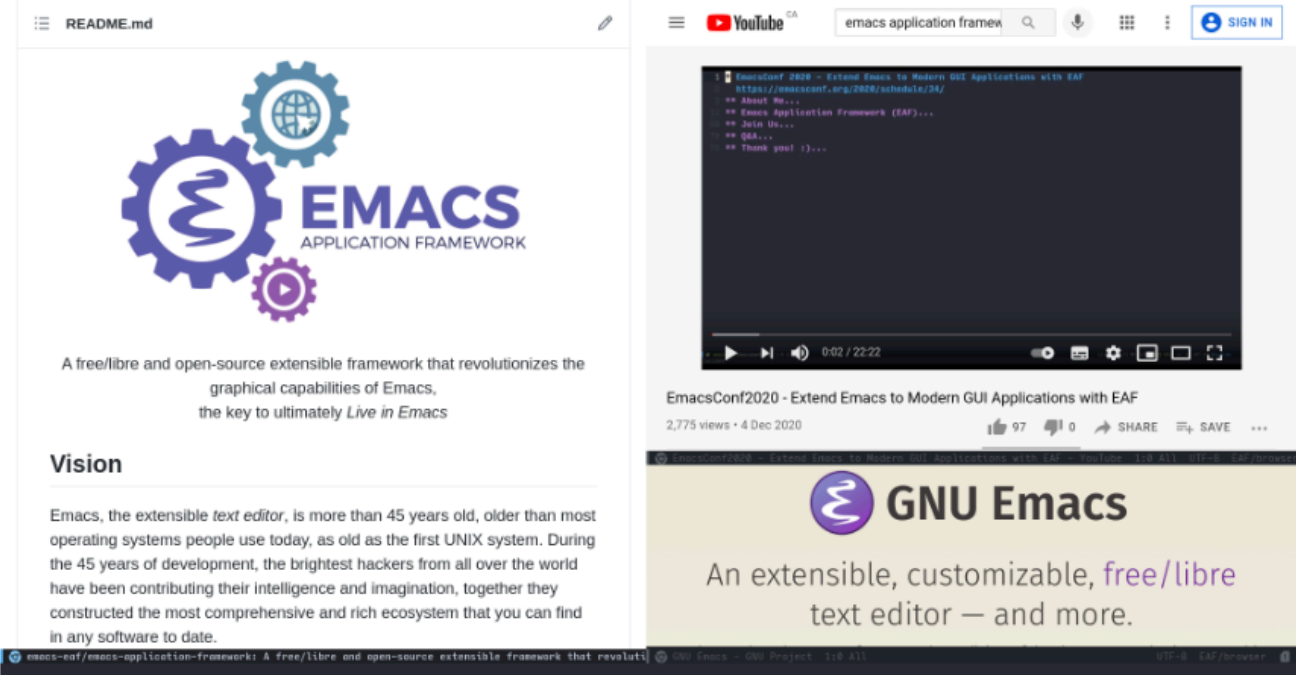


A free/libre and open-source extensible framework that revolutionizes the graphical capabilities of Emacs.
The key to ultimately *Live in Emacs*

Features

EAF is very extensible. We ship a lot of applications, feel free to choose anything you find interesting to install:

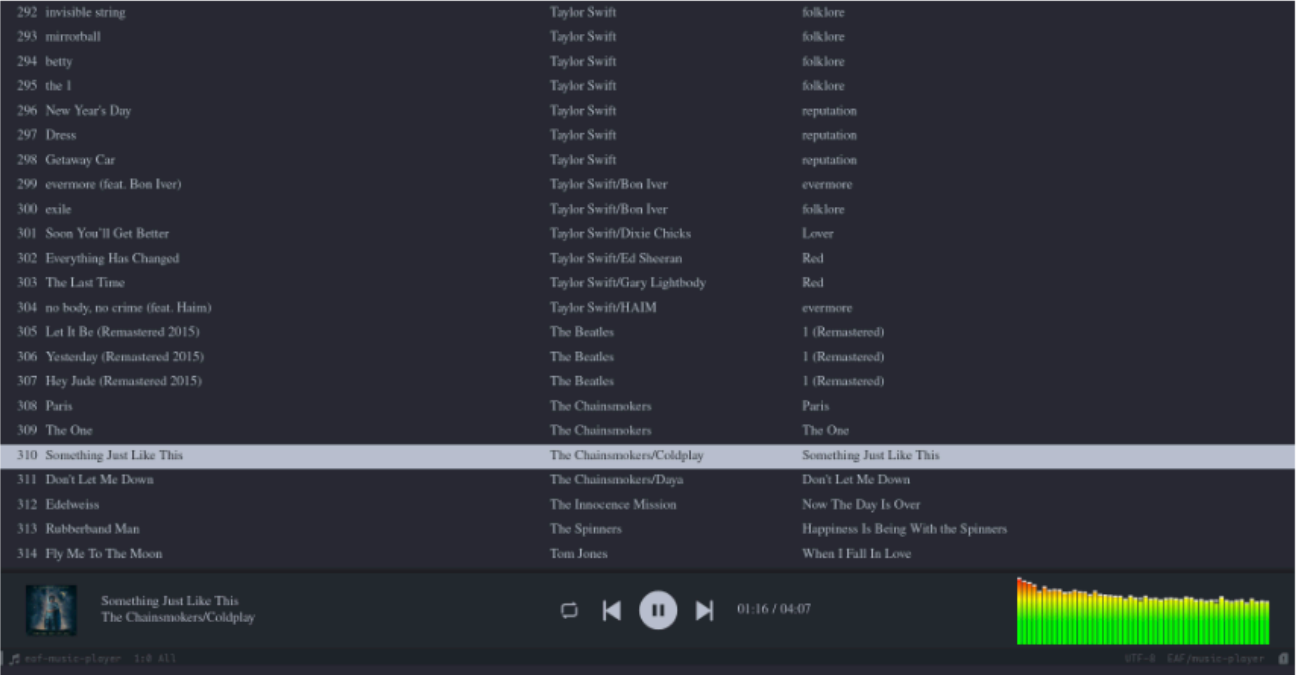
Browser



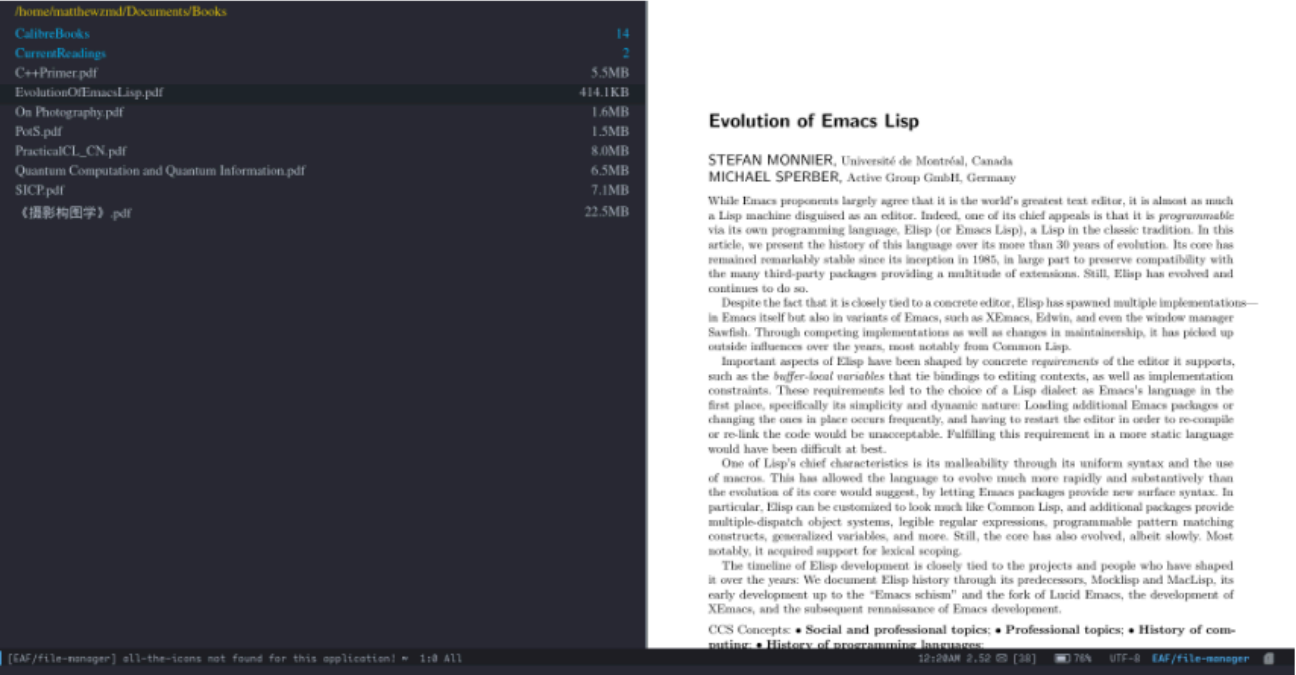
PDF Viewer



Music Player



File Manager



Browser: A modern, customizable and extensible browser in Emacs

Lsp-bridge

Lsp-bridge's goal is to become the fastest LSP client in Emacs.

Lsp-bridge uses python's threading technology to build caches that bridge Emacs and LSP server. Lsp-bridge will provide smooth completion experience without compromise to slow down emacs' performance.

```
class LspServer:
    def __init__(self, message_queue, project_path, server_info, server_name):
        self.message_queue = message_queue
        self.project_path = project_path
        self.server_info = server_info
        self.initialize_id = generate_request_id()
        self.server_name = server_name
        self.request_dict: Dict[int, Handler] = dict()
        self.root_path = self.project_path

    self.p
    (x) p Variable
    # (x) parse_document_uri Method parse_document_uri: (filepath: Unknown, external_file_link: Unknown) -> (Unknown | str)
    s (x) project_path Variable
    s (x) completion_resolve_provider Variable
    s (x) completion_trigger_characters Variable
    (x) get_capabilities Method
    # (x) get_server_workspace_change_co ... Method
    s (x) handle_workspace_configuration ... Method
    (x) lsp_message_dispatcher Method
    # (x) rename_prepare_provider Variable
    message_emacs("Start LSP server ({} for {})..."format(self.server_info["name"], self.root_path))

    # Two separate thread (read/write) to communicate with LSP server.
    self.receiver = LspServerReceiver(self.p)
    self.receiver.start()

    self.sender = LspServerSender(self.p)
    self.sender.start()

    # All LSP server response running in ls_message_thread.
    self.ls_message_thread = threading.Thread(target=self.lsp_message_dispatcher)
    self.ls_message_thread.start()

    self.files: Dict[str, "FileAction"] = dict()
```

コンピュータサイエンス
の問題に立ち向かって
最強の環境を作りたい人
はEmacsを

コミュニティは
Emacs JPのSlackに
(不定期にオンラインミートアップも
やってるよ)