State of Doom Emacs 2023

Icon from @eccentric-j

Ellis Kenyő

March 8, 2023

Welcome everyone to my "State of Doom" talk. There's been a decent amount of discussion recently in Slack around Doom so I thought I'd do my best to fill everyone in on exactly what Doom is, what's happened recently and some exciting things coming soon.
There will also be a very short walkthrough of installing and running the ootb setup, assuming the demo Gods allow it to be.

What is Doom Emacs?

What is Doom Emacs?

- A modular config framework
- Similar to Spacemacs, Prelude, Centaur Emacs
- Batteries-included Emacs
- Sensible but (somewhat) opinionated defaults
- Package management *outside* Emacs and *declarative*
- Optimized around startup time

Well, put simply; Doom is a modular configuration framework built for Emacs. Similar in nature to Spacemacs, it aims to provide a more batteries-included experience for Emacs by providing what we've deemed as sensible but (somewhat, less so recently) opinionated defaults.
However, unlike other frameworks; Doom aims to keep the package management outside of Emacs (you don't want everything to download/update when you start Emacs) and also aims to be as declarative as possible (more on the future of this later) meaning that you can easily pin packages to specific versions and they'll only pull new updates when you update the pin.

A Brief History of Doom

- Created by Henrik Lissner (@hlissner)
- Started as his private config
- Initially structured like Prelude, shortly made to be modular

So where did this all come from? Doom wasn't created with a starter kit/config framework in mind, it just began as Henrik Lissner (the initial and current maintainer) decided to try and build his own config as an attempt to learn Emacs Lisp. Starting out a bit like Prelude, it was later migrated to be based around pluggable modules and incorporated things like Spacemacs' leader key from migrating users.

What is a module?

- Collection of packages with glue
- Grouped by a class and can include optional flags
- Looks like `:editor lsp +eglot`

Okay we've heard this term "modular" used quite a few times, but what exactly is it? Simply put, it's a collection of packages with some extra code to integrate them together. For example, the Clojure module currently pulls in `clojure-mode`, `paredit`, `clj-refactor` and `flycheck-clj-kondo` (when the syntax check module is enabled) and most of the configuration is moving the keybindings to be focused around the leader key (eg `SPC m '` for `cider-jack-in`), integrating with Doom's popup system and some LSP config (when the LSP module is enabled). Each module is grouped by a number of keywords and can include optional flags denoted by a plus. The example we have there is the LSP module with eglot support optionally enabled.

doom CLI

```
doom sync
```

```
> Tangling your literate config...
   Done tangling 5 file(s)!
> Synchronizing "default" profile...
  > Regenerating envvars file
     Generated ~/.config/emacs/.local/env
  > Installing packages...
    - No packages need to be installed
  > (Re)building packages...
    - No packages need rebuilding
  > Purging orphaned packages (for the emperor)...
    - Skipping builds
    - Skipping elpa packages
    - Skipping repos
    - Skipping regrafting
    - Skipping native bytecode
  > (Re)building profile in /home/elken/.config/emacs/.local/etc/@/...
    > Deleting old init files...
    > Generating 4 init files...
    > Byte-compiling ~/.config/emacs/.local/etc/@init.29.el...
     Built init.29.elc
  - Restart Emacs or use 'M-x doom/reload' for changes to take effect
```


I mentioned before the Doom offloads a lot to a CLI tool, but what exactly for? The primary purpose is to perform an operation called `sync` which will perform any defined package management options (and if you're using a literate org-based config, that will get tangled first) and also run through a number of internal steps such as building, clearing orphaned packages and something we'll touch shortly at the bottom for rebuilding the profile.

You can also do `doom sync -u` when you need to update all the package versions you have, or when you update the pin on a package or packages. `doom upgrade` to pull in the latest changes from Doom itself and also perform a `doom sync -u`.

doom CLI

```
(defcli! (publish) ()
  "Publish my doom config as a html page."
```

```
doom help | rg "publish"
```

```
publish    Publish my doom config as a html page.
```


And by far the *coolest* feature available is you can very easily extend this to provide your own CLI tools. This screenshot is just the macro from my Doom config to publish my config as HTML, borrowing a couple of other tecosaur packages to produce nicer output. The script is called `publish`, and the docstring is shown in the help menu. I can call this with `doom publish` so long as that cli.el file is present.

Demo

What's changed recently?

Profiles

- Largest recent change
- Still more changes to come
- Generational, similar to Nix

So what's new lately? The largest recent change has been the introduction of a profile system. If you remember from the output of `doom sync`, at the bottom there was mention of profiles being updated.
Powering something that will be talked about shortly (transactional package management), a profile system was introduced to act as an improved version of Chemacs (a tool for running multiple Emacs configs side-by-side).
Outside of the main profile which pertains to your config, you can place extras inside your private Doom config directory under a `profiles` folder.
The profiles system has a lot more changes coming soon, such as being able to generate profiles for use in the sandbox and the upcoming transactional package management changes.

New modules

- `:lang graphql`
- `:tools tree-sitter`
- More coming soon

A couple of new modules have been added, a simple graphql module added by me which provides a sane graphql experience and has improved support for a `.graphqlconfig` file. The LSP server is a bit buggy, thanks to the complexity of node packaging, but for the most part completion and running queries should work as expected. It also includes org-babel support for running graphql queries.
As well as that, predating Emacs 29 getting tree-sitter support (and as such needing a refactor to better integrate) support was added for tree-sitter and any modules that had grammars defined. This also included extra text objects for evil users, with proposed support for integrating tree-edit.

Docs progress

- https://docs.doomemacs.org/latest/
- Viewing in Emacs is the best way

Every developer's favourite thing; writing documentation. For a while, most of the knowledge not part of module READMEs was in Henrik's head making contribution in some cases difficult. This has been changing as more and more module maintainers have been introduced and the project shifts to being much more modular.
As such, a push to greatly improve the quality of the documentation is on-going; as well as improvements to viewing the docs inside Emacs, including things like conditionally highlighting modules and packages based on whether or not you've installed them.

What's coming next?

Doom v3 (this time, without The Rock)

- Splitting up into core, modules & community modules
- Transactional package management
- Generalizing the Doom CLI
- CI/CD for automating package bumping and tests
- On-the-fly profile generation
- On-the-fly module activation
- Recursive modules
- Managing external module libraries like you can packages
- An installation wizard for `doom install`
- Upcoming modules
  - `:completion corfu`
  - `:editor format` refactor
  - `:config tutorial`

First up, splitting up the core pieces of doom. People have been asking for some time now about having access to a few bits of Doom without having to include everything else, so everything will be split into core, the core modules and community modules.

Next we have transactional package management. Similar to Nix, each "change" to the packages in a profile will constitute a new generation that can be rolled back to at any point and managed similarly to any other profile.

Then we have some generalizing of the Doom CLI code to become a general-purpose tool for elisp development, providing things like linting, tests, etc. Similar to cask, eask or eldev.

Improved CI is self-explanatory, automate some of the more annoying tasks like bumping packages and add tests; all managed via a github action.

Profile generation on-the-fly relates to the sandbox, and being able to create a slightly less transient sandbox that you can even use Nix to deploy to specific Emacs versions.

Only-the-fly module activation is the simple act of the first time a file is opened that there is a module for, prompt the user asking if they wish to install said module.

Recursive modules solves a problem we hit recently where both the F# and C# modules have a shared dependency, but also different sets of dependencies that you won't need for either language, so the simplest way to solve this was to introduce a kind of "meta module" under `dotnet` but also allow you to selectively install just the languages you care about, without having to manage multiple sets of pins.

Then we have the ability to manage external modules as you do with packages, being able to pin, version, profile, etc.

Also on the cards is a simplified installation wizard during installation to let users tailor their installation easier.

And finally we have a few new modules working through, adding in corfu as a completion frontend with the view to replace company, a large refactor of all the `:editor format` components by me to use apheleia instead, and a module for creating interactive tutorials in Emacs by tecosaur.

Where you can find us

- https://doomemacs.org/discord
- https://discourse.doomemacs.org/
- https://github.com/doomemacs

That's just a taster though, if you're more interested there's a couple of links here to find us. There's been talk of moving away from Discord to Matrix, but until that materializes that's the best place to reach us in terms of following the project (outside of Github obviously). The recent addition of the Discourse server to better facilitate Q&A has also gone well, and is much more convenient for posting questions and getting answers to them, but the forum channels on the Discord also exist.

Any Questions?