# corfu+yasnippet

Easier than I thought

Pedro A. Aranda Gutiérrez

`paaguti@gmail.com`

2025/12/07

## Motivation

- `yasnippet` is old … why not other template management packages?
  - I tried `tempel` but it wasn't my cup of tea
  - I have a nice base of `yasnippets`… migration comes at a cost.
- I've been using `company` as my c-a-p-f GUI for years
  - It was not easy to set it up for my needs…
- I had tried `corfu` + `eglot` some time ago but…
  1. You needed `corfu-terminal` in text mode (yet another package)
  2. I didn't find a quick way of getting rid of `company` to get `yasnippet` supported
  3. So … why try?
- I'm following `emacs-devel` and got interested when the support for `tty child frames` was announced for `master`
  - Less packages to download: Could I get `corfu` running without `corfu-terminal`?
  - They were actually mentioning this in the thread
- Let's give it a try

## My requirements

1. On a decently new emacs (master), I don't want `corfu-terminal`
2. I need `yasnippet`
   - *ol'dog don't learn new tricks*
   - Snippets must be *easy* and *quick* to configure
3. I don't want any reminiscence of `company` in my setup
4. Ahhh… and don't forget `eglot` integration

## Basic setup: `corfu` + `eglot`

- That was quick after looking for my previous tries:

```
(use-package corfu
  :ensure t
  :pin gnu
  :init
  (message "corfu init:")
  (global-corfu-mode))

(use-package eglot
  :ensure nil
  :defer t
  :hook ((python-mode . eglot-ensure)))
```

- Actually, I also set `corfu-auto` to `t`, although it is not recommended.

## Looking at `completion-at-point-functions`

- The information is somewhat scattered and cryptic
- At the end, I came up with

```
(defun yas-completion-at-point()
  (when-let* ((keywords (yas-kw-list))
              (bounds (bounds-of-thing-at-point 'word))
              (start (car bounds))
              (end (cdr bounds)))
    `(,start ,end ,keywords . ,completion-props)))
```

- And had to dig deep to create the `yas-kw-list` function and understand the `completion-props`
- Update:
  - I created a *better* ’thing’ that skips over non-blank characters.
    - `bounds-of-thing-at-point 'word` is confusing in LaTeX, because the backslash (for example `\texttt{}`) is not taken into account.

## Making my own (basic) `c-a-p-f` for `yasnippet`

### The completion properties

- The completion properties tell Emacs how to handle the information for a specific completion type

```
(defvar yas-completion-props
  (list ;; :annotation-function #'(lambda (_) " Snippet")
              :company-kind #'(lambda(_) 'snippet)
              :exclusive 'no))
```

- Which is basically read as:
  - This is a snippet keyword
  - Which should be added to other completions
- Note that `:company-kind` is a *function*
  - Used by `nerd-icons-corfu`
  - If you don’t use `nerd-icons-corfu`, use `:annotation-function`

## Getting `yas-kw-list` right

I - What do I want?

- Looking at pre-existing solutions, it looks a bit like *Mission Impossible*
  - The approach by all: get the *keys* and the *names* for the snippets associated to a specific major mode.
- However… do I really need *both*?
  - At the end, I use my own snippets and choose meaningful keys for them.
  - So, why not center everything around keys *only*? Can that help simplifying my code?

## Getting `yas-kw-list` right

II - Diving in yasnippet

- The `yasnippet` package has a lot of useful *semi*-hidden functions
- I discovered that getting the list of keys for a given mode is not that difficult
  1. Get all snippet tables used in a `major-mode`
  2. Get the list of keys for each table
     - sometimes, the list is empty (returning a `nil`)
     - when the snippets are structured, you need to filter out the non-strings

```
(defun yas-kw-list (mode)
  (let (result)
    (dolist (tab (yas--get-snippet-tables mode) result)
      (setq result
            (nconc result
                   (cl-remove-if-not #'stringp
                                     (yas--table-all-keys tab)))))))
```

## Fine-tuning

### Adding cape

- Just adding `yas-completion-at-point` to the `completion-at-point-functions` was not enough.
  - The `:exclusive no` didn't seem to work either
- Adding the cape package because `cape-capf-super` is the answer to the problem.
- Using `cape-capf-super`:
  1. Define an alias for the chain
     ```
     (defalias 'cape-lisp-mode
       (cape-capf-super
        #'yas-completion-at-point
        #'elisp-completion-at-point))
     ```
  2. Add the alias to the completion-at-point-functions list:
     ```
     (add-to-list 'completion-at-point-functions #'cape-lisp-mode))
     ```

## Fine-tuning II

### Automatic snippet expansion

- Adding the following code to the properties list is the key to having automatic expansion when a snippet is selected
  ```
  :exit-function #'(lambda (_ _)
                     (call-interactively 'yas-expand))))
  ```
- To avoid automatic selection being too *eager*, you will need to
  ```
  (setq corfu-on-exact-match nil)
  ```
  because otherwise you *always* get the snippet expanded,
  - suboptimal when a key can appear in a variable name, etc.
- I didn't need to add anything to my themes
  - The default faces in `corfu` adapt quite well to them

## My check-list

- Going back to my requirements:

  1. On a decently new emacs (master), I don't want `corfu-terminal` ✅
  2. I need `yasnippet` (ol'dog don't learn new tricks) ✅
  3. I don't want any reminiscence of `company` in my setup ✅

  Note: (`company-kind` isn't defined in `company`)

  4. Ahhh… and don't forget `eglot` integration ✅

## Takeaways

- Accepting the *extra-burden* of `corfu-terminal` on `emacs<31`, it was not too difficult to get the setup running
- `corfu` was easier to configure/integrate than `company`
  - with the help of `cape`
- I learnt a lot about `c-a-p-f`'s in the process
- `nerd-icons-corfu` makes the overall look-and-feel nice

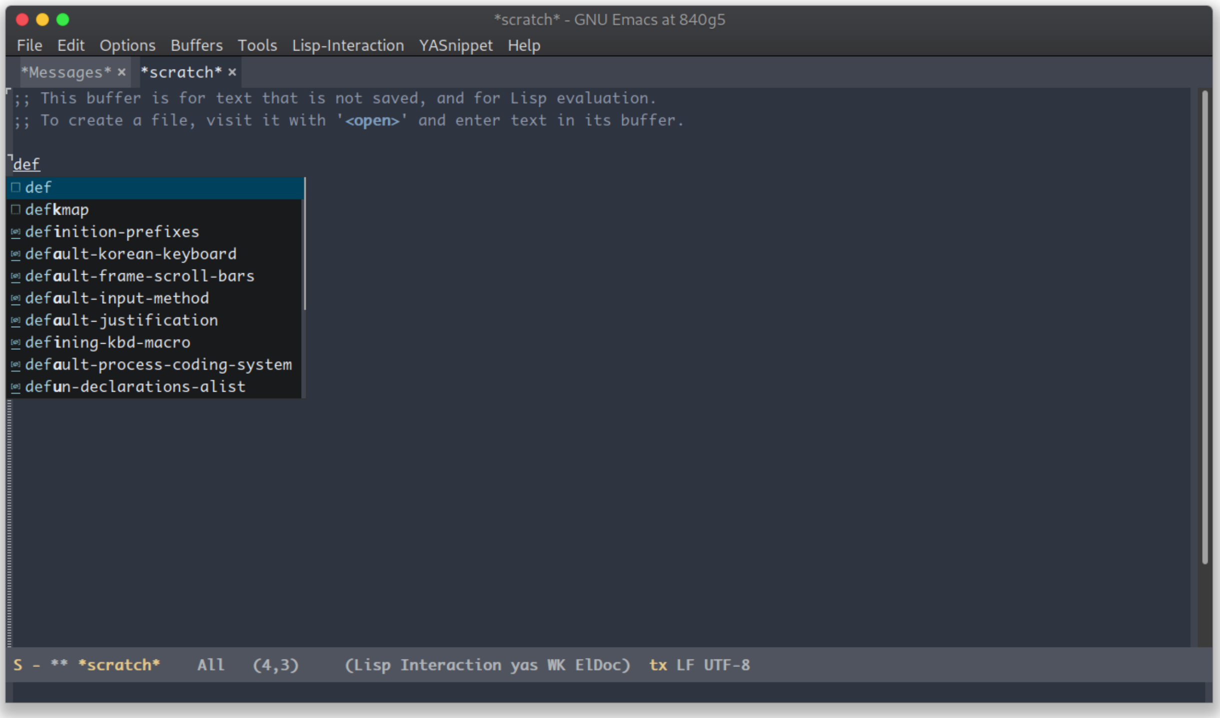
## Requests (to whom it may concern):

- `cape` has nice features that could be integrated into Emacs
- `corfu` is a ”*really* nice-to-have”
  - Could it make its way into Emacs?
- Please keep `yasnippet` alive
- P.S. if you are on `master` and using semantic highlighting for Emacs Lisp, you may need to

```
(add-hook 'snippet-mode-hook
          #'(lambda()
              (setq-local elisp-fontify-semantically nil)))
```

  if you feel the faces a bit too ”*pushy*”

## How this looks in *real life* on my Emacs


*scratch* - GNU Emacs at 840g5
File Edit Options Buffers Tools Lisp-Interaction YASnippet Help
*Messages* ×
*scratch* ×

```
;; This buffer is for text that is not saved, and for Lisp evaluation.
;; To create a file, visit it with '<open>' and enter text in its buffer.

def
def
defkmap
definition-prefixes
default-korean-keyboard
default-frame-scroll-bars
default-input-method
default-justification
defining-kbd-macro
default-process-coding-system
defun-declarations-alist

S - ** *scratch*    All   (4,3)     (Lisp Interaction yas WK ElDoc)  tx LF UTF-8
```