A guided tour of hydra.el

Oleh Krehel

2017-07-24

hydra.el - Make bindings that stick around.

A package for GNU Emacs that can be used to tie related commands into a family of short bindings with a common prefix - a Hydra.

The simple example that started the project

I used to have something like this for adjusting the font size:

```
(global-set-key (kbd "<f2> g") 'text-scale-increase)
(global-set-key (kbd "<f2> l") 'text-scale-decrease)
```

Here's how a typical font size adjustment session would look like:

- `<f2> g <f2> g <f2> g <f2> l`.

Here's what I wanted it to look like instead:

- `<f2> g g g l`

And here's the code to do that with `hydra`:

```
(defhydra hydra-zoom (global-map "<f2>")
  ("g" text-scale-increase)
  ("l" text-scale-decrease))
```

Project beginning

It started when I described the code I had in my config to solve the above zooming problem on my blog https://oremacs.com/2015/01/14/repeatable-commands/.

```
(defun def-rep-command (alist)
  "Return a lambda that calls the first function of ALIST.
It sets the transient map to all functions of ALIST."
  (let ((keymap (make-sparse-keymap))
        (func (cdar alist)))
    (mapc (lambda (x) (define-key keymap (car x) (cdr x)))
          alist)
    (lambda (arg)
      (interactive "p")
      (funcall func arg)
      (set-transient-map keymap t))))

(global-set-key (kbd "<f2> g") (def-rep-command
                                '(("g" . text-scale-increase)
                                  ("l" . text-scale-decrease))))
(global-set-key (kbd "<f2> l") (def-rep-command
                                '(("l" . text-scale-decrease)
                                  ("g" . text-scale-increase))))
```

The addition of docstrings

Thanks to an early feature request from @immerrr (https://github.com/abo-abo/hydra/issues/2), hydra got one of its most attractive features: docstrings.
Let's see how the update code looks.

Before:

```
(defhydra hydra-zoom (global-map "<f2>")
  ("g" text-scale-increase)
  ("l" text-scale-decrease))
```

After:

```
(defhydra hydra-zoom (global-map "<f2>")
  "zoom"
  ("g" text-scale-increase "in")
  ("l" text-scale-decrease "out"))
```

The final hydra-zoom

Before:

```
(defhydra hydra-zoom (global-map "<f2>")
  "zoom"
  ("g" text-scale-increase "in")
  ("l" text-scale-decrease "out"))
```

After:

```
(defhydra hydra-zoom (global-map "<f2>")
  "zoom"
  ("g" text-scale-increase "in")
  ("l" text-scale-decrease "out")
  ("r" (text-scale-set 0) "reset")
  ("0" (text-scale-set 0) :bind nil :exit t)
  ("1" (text-scale-set 0) nil :bind nil :exit t))
```

Links / stats

The project development happens at:

- `https://github.com/abo-abo/hydra`.

Development project releases are at:

- `https://melpa.org/#/hydra`

Stable project releases are at:

- `http://elpa.gnu.org/packages/hydra.html`

	Initial	Current stable
version	0.1.0, Jan 2015	0.14.0, Jul 2017
loc	43	1173
commits	1	313
closed issues	0	204
closed PRs	0	24
contributors	1	13
packages	1	17

Community

Besides contributing issues and pull requests, you can make use of the Wiki pages: `https://github.com/abo-abo/hydra/wiki`.

- wiki pages are available for unrestricted editing.
- there are currently 41 pages on various topics by various authors.

Tip: it's possible to clone the wiki and write to it with `git`:

```
git clone https://github.com/abo-abo/hydra.wiki.git
```

Wiki example: helm-like-unite

```
(defhydra helm-like-unite (:hint nil
                           :color pink)
  "
Nav ^^^^^^^^^        Mark ^^          Other ^^       Quit
^^^^^^^^^^-----------^^--------------^^---------------------
_K_ ^ ^ _k_ ^ ^     _m_ark           _v_iew         _i_: cancel
^|^ _h_ ^+^ _l_     _t_oggle mark    _H_elp         _o_: quit
_J_ ^ ^ _j_ ^ ^     _U_nmark all     _d_elete
^^^^^^^^^^                           _f_ollow: %(helm-attr 'follow)
"
  ("h" helm-beginning-of-buffer)
  ("j" helm-next-line)
  ;; ...
  )
(define-key helm-map (kbd "C-o") 'helm-like-unite/body)
```

Resulting docstring:

```
Nav          Mark           Other        Quit
-------------------------------------------------
K   k        mark           view         i: cancel
↕ h ✥ l      toggle mark    Help         o: quit
J   j        Unmark all     delete
                            follow: nil

<escape>
pattern:
```

Diving into the code

What's in a hydra?

```
(defhydra hydra-zoom (global-map "<f2>")
  ("g" text-scale-increase)
  ("l" text-scale-decrease))
```

You can `macroexpand` the code to see what exactly it does:

- defvar `hydra-zoom/keymap`
- defvar `hydra-zoom/heads`
- defvar `hydra-zoom/hint`
- defun `hydra-zoom/text-scale-increase`
- defun `hydra-zoom/text-scale-decrease`
- defun `hydra-zoom/body`, the return result

Also, the following key bindings will take place:

```
(define-key global-map (kbd "<f2> g") 'hydra-zoom/text-scale-increase)
(define-key global-map (kbd "<f2> l") 'hydra-zoom/text-scale-decrease)
```

Examples from my config

Hydra ships with `hydra-examples.el`, which you can go through to see code of varying complexity and usefulness.
Let's examine some examples I use quite often:

- hydra-marked-items
- hydra-toggle
- hydra-vi
- hydra-k
- hydra-ivy
- lispy-x

See also this 9 minute video about `hydra` on Youtube:
`https://www.youtube.com/watch?v=_qZliI1BKzI`.

Thanks

- The slides will be available at `https://oremacs.com/download/london.pdf`.
- If there are no problems with my screen recorder (Kazaam), I'll post the video at `https://www.youtube.com/user/abo5abo/videos`.
- Questions?