Technical writing using org-mode

Jan Ypma

October 19, 2021

Contents

1 Introduction

1.1 This presentation

- Emacs for "Technical" writing
  - Developer guide
  - API reference documentation
  - Technical presentation
  - Live demo
- A little about me, Jan Ypma
  - `jan@ypmania.net`
  - Independent software developer

  - Scala / Java, C++ (embedded), a little Rust, and of course Lisp
  - Emacs for everything

1.2 Emacs and org-mode

- Emacs
  - Customizable text editor environment
- Org Mode
  - Defines text structures for headings, list, table, code blocks and others
  - Ideal for technical writing: API guides, code presentations, live demos
- Org Babel
  - Functionality in Org Mode to "execute" code blocks and capture results
  - Many languages supported, natively or through extensions

2 Demo

2.1 Developer guide

In this section, we'll demonstrate techniques for writing a developer guide. For example, imagine setting up something that requires us to run services using docker compose.

```
version: '3.1'

services:
  webserver:
    image: nginx
    volumes:
      - ".:/usr/share/nginx/html:ro"
    ports:
     - "8080:80"
```

The above code block is automatically copied into `docker-compose.yml` when this file is tangled (using `C-c C-v t`).

We can now run a shell script to start the docker containers:

```
docker-compose up -d
docker-compose ps
```

```
NAME                        COMMAND                  SERVICE             STATUS              PORTS
emacsconf2021-webserver-1   "/docker-entrypoint...."   webserver           running             0.0.
```

2.2 Rest API Guide

Let's we're documenting a REST API. Conveniently, we have an Nginx server running on port 8080 (see previous section).

Let's make sure we have an XML file to serve up:

```
<hello>
  This is XML!
</hello>
```

We can make an actual REST call from within Emacs. The mode for syntax highlighting in the response is automatically taken from the `Content-Type` header, if present.

```
GET http://localhost:8080/test.xml
```

```
<hello>
  This is XML!
</hello>

<!-- GET http://localhost:8080/test.xml -->
<!-- HTTP/1.1 200 OK -->
<!-- Server: nginx/1.21.3 -->
<!-- Date: Tue, 19 Oct 2021 17:51:46 GMT -->
<!-- Content-Type: text/xml -->
<!-- Content-Length: 32 -->
<!-- Last-Modified: Tue, 19 Oct 2021 17:51:43 GMT -->
<!-- Connection: keep-alive -->
<!-- ETag: "616f05af-20" -->
<!-- Accept-Ranges: bytes -->
<!-- Request duration: 0.003053s -->
```

Let's pretend to PUT a file (Nginx won't allow it)

```
PUT http://localhost:8080/test.xml
Content-Type: text/xml

<hello>
  Here's some XML!
</hello>
```

```
<html>
<head><title>405 Not Allowed</title></head>
<body>
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx/1.21.3</center>
</body>
</html>

<!-- PUT http://localhost:8080/test.xml -->
<!-- HTTP/1.1 405 Not Allowed -->
<!-- Server: nginx/1.21.3 -->
<!-- Date: Tue, 19 Oct 2021 17:51:53 GMT -->
<!-- Content-Type: text/html -->
<!-- Content-Length: 157 -->
<!-- Connection: keep-alive -->
<!-- Request duration: 0.001064s -->
```

2.3 Presentations

- Org-mode is also very suitable for making presentations (you're looking at one!).
- Fun to use `org-babel` for live coding / API demonstrations
- Presentations can be exported:
  - As plain PDF (`C-c C-e l p`), just like any other org file
  - As "beamer" PDF (`C-c C-e l P`), trying to make the PDF actually look like slides
    - Unfortunately, has a fairly rigid idea about heading structure

3 Packages and configuration

Let's go through some specific packages that help in the mentioned use cases (in addition to org and org-babel).

3.1 Package: ox-beamer

Export org-mode documents to Latex in Beamer style (PDF presentation handouts)

```
(require 'ox-beamer)
```

3.2 Package: doom-modeline

A prettier mode line than the default.

```
(use-package doom-modeline
  :ensure t
  :hook (after-init . doom-modeline-mode))
```

3.3 Package: org-superstar

Customizable way to show (or not) heading bullets in org-mode.

```
(use-package org-superstar
  :hook (org-mode . org-superstar-mode))
```

3.4 Package: restclient

Make REST calls by writing documents in Emacs.

```
(use-package restclient
  :config
  (org-babel-do-load-languages
   'org-babel-load-languages
   '((restclient . t))))
```

3.5 Package: ob-restclient

Makes REST calls from within org-mode as org-babel code block sections.

```
;; From https://github.com/alf/ob-restclient.el
(require 'ob-restclient)
```

3.6 Package: org-tree-slide

Present an org-mode document, one heading at a time.

```
(defun my/presentation-setup ()
  (shell-command "dunstctl set-paused true")
  (flyspell-mode 0)
  (setq text-scale-mode-amount 3)
  (org-display-inline-images)
  (text-scale-mode 1)
  (font-lock-flush)
  (font-lock-ensure))

(defun my/presentation-end ()
  (shell-command "dunstctl set-paused false")
  (flyspell-mode 1)
  (text-scale-mode 0)
  (org-remove-inline-images)
  (font-lock-flush)
  (font-lock-ensure))

(use-package org-tree-slide
  ;; Load immediately, since it messes with org-mode faces
  :demand
  :hook
  ((org-tree-slide-play . my/presentation-setup)
   (org-tree-slide-stop . my/presentation-end))
  :bind
  (:map org-mode-map
        ("<f6>" . org-tree-slide-mode))
  :custom
  (org-image-actual-width nil)
  )
```

3.7 Other configuration

3.7.1 Customize ellipsis display

Makes hide-show mode a bit more pretty (helps in presentations).

```
;; customize the face as well
(defface hs-ellipsis
  '((((class color) (background light)) (:underline t))
    (((class color) (background dark)) (:underline t))
    (t (:underline t)))
  "Face for ellipsis in hideshow mode.")

;; Use this in whitespace-mode
(defun whitespace-change-ellipsis ()
  "Change ellipsis when used with `whitespace-mode'."
  (when buffer-display-table
    (set-display-table-slot buffer-display-table
                            'selective-display
                            ;;(string-to-vector " ... ")
                            (let ((face-offset (* (face-id 'hs-ellipsis) (lsh 1 22))))
                              (vconcat (mapcar (lambda (c) (+ face-offset c)) " ... ")))
                            )))
(add-hook 'whitespace-mode-hook #'whitespace-change-ellipsis)

;; Use this in non-whitespace modes
(set-display-table-slot
 standard-display-table
 'selective-display
 (let ((face-offset (* (face-id 'hs-ellipsis) (lsh 1 22))))
   (vconcat (mapcar (lambda (c) (+ face-offset c)) " ... "))))
```

3.7.2 Show emphasis markers at point

This makes the bold, italic, etc. markers in org-mode disappear, *except* when you're within them.

```
;;https://www.reddit.com/r/orgmode/comments/43uuck/temporarily_show_emphasis_markers_wh
↪  en_the_cursor/
;; (adapted to also show verbatim markers)
(defun my/org-show-emphasis-markers-at-point ()
  (save-match-data
    (if (and (or (org-in-regexp org-emph-re 2) (org-in-regexp org-verbatim-re 2))
             (>= (point) (match-beginning 3))
             (<= (point) (match-end 4))
             (member (match-string 3) (mapcar 'car org-emphasis-alist)))
        (with-silent-modifications
          (setq my/org-show-emphasis-hidden t)
          (remove-text-properties
           (match-beginning 3) (match-beginning 5)
           '(invisible org-link)))
      (if my/org-show-emphasis-hidden
          (progn
            ;; Add about 100 characters extra, in case we're moving lines.
            (apply 'font-lock-flush (list (- (match-beginning 3) 100) (+
            ↪   (match-beginning 5) 100)))
            (setq my/org-show-emphasis-hidden nil))))))
```

3.7.3 Image animation

Animate an image when hovering over it and pressing `a`.

```
(defun my/image-animate ()
    "Starts to animate the image under the cursor"
    (interactive)
    (image-animate (image--get-imagemagick-and-warn)))

(define-key image-map (kbd "a") 'my/image-animate)
```

3.7.4 Org mode startup

```
(defun my/org-mode-setup ()
  (whitespace-mode -1)

  ;; https://orgmode.org/list/87pn8huuq2.fsf@iki.fi/t/
  (electric-indent-local-mode -1)

  ;; Shorten some text
  (setq prettify-symbols-alist
        (map-merge 'list prettify-symbols-alist
                   `(
                     ("#+name:" . "")
                     ("#+NAME:" . "")
                     ("#+BEGIN_SRC" . "")
                     ("#+BEGIN_EXAMPLE" . "")
                     ("#+END_SRC" . "")
                     ("#+END_EXAMPLE" . "")
                     ("#+RESULTS:" . "")
                     )))
  (prettify-symbols-mode 0)
  (prettify-symbols-mode)

  ;; Auto-wrap lines
  (visual-line-mode)
  (setq adaptive-wrap-extra-indent 2)

  (variable-pitch-mode)
  ;; from https://lepisma.xyz/2017/10/28/ricing-org-mode/
  ;; A little bit of space in the left/right margins:
  (setq left-margin-width 2)
  (setq right-margin-width 2)
  (set-window-buffer nil (current-buffer))

  (flyspell-mode 1)
  (ws-butler-mode 1)

  (defvar-local my/org-show-emphasis-hidden nil)
  (add-hook 'post-command-hook
            'my/org-show-emphasis-markers-at-point nil t))
```

3.7.5 Miscellaneous

```
;; fontify inside org mode
(setq org-src-fontify-natively t)

;; Don't indent org documents
(setq org-startup-indented nil)
```