>>> Technical writing using org-mode

Name: Jan Ypma
Date: October 19, 2021

>>> Outline

>>> 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

* 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

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
emacsconf2021-webserver-1     "/docker-entrypoint...."   webserver           running
```

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

```
<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 -->
```

- 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

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

>>> Package: ox-beamer

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

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

>>> Package: doom-modeline

A prettier mode line than the default.

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

>>> Package: org-superstar

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

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

>>> 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))))
```

>>> 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)
```

>>> 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
        ("<f8>" . org-tree-slide-mode))
```

>>> Other configuration

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
```