org-special-block-extras

Musa Al-hassy

April 17, 2020

Contents

1 Example Use

User type something along the lines of the following.

```
#+begin_red org
/This/
      *text*
             _is_
                  red!
#+end_red
```

Which generates *red* text when exported to HTML and LaTeX, **while supporting Org markup**.

This article may be read as a PDF or as HTML or as pure Org!

The remaining sections are implementation matter.

2 Core Utility

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Core utility

(defun org-special-block-extras--advice (backend blk contents _)
  "Invoke the appropriate custom block handler, if any.

A given custom block BLK has a TYPE extracted from it, then we
send the block CONTENTS along with the current export BACKEND to
the formatting function ORG-SPECIAL-BLOCK-EXTRAS/TYPE if it is
defined, otherwise, we leave the CONTENTS of the block as is."
  (let* ((type    (nth 1 (nth 1 blk)))
         (handler (intern (format "org-special-block-extras--%s" type))))
    (ignore-errors (apply handler backend contents nil))))
```

Figure 1: Write Org-markup once, generate for many backends ^_^

```
(advice-add #'org-html-special-block :before-until
            (-partial #'org-special-block-extras--advice 'html))

(advice-add #'org-latex-special-block :before-until
            (-partial #'org-special-block-extras--advice 'latex))
```

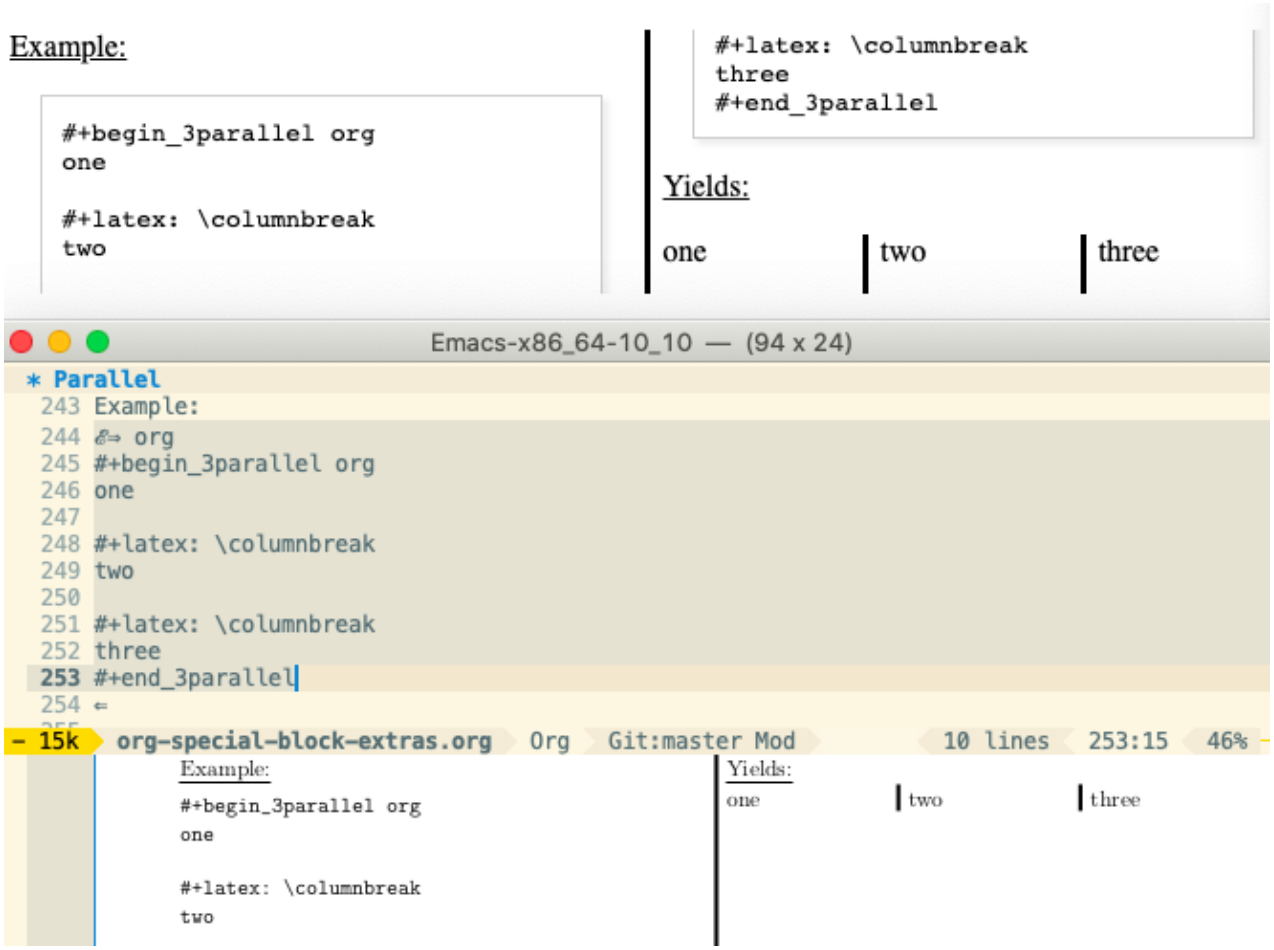

Figure 2: Displaying thoughts side-by-side ^_^

3 Colours

```
#+begin_blue
This text is blue!
#+end_blue

#+begin_brown
This text is brown!
#+end_brown

#+begin_cyan
This text is cyan!
#+end_cyan

#+begin_darkgray
This text is darkgray!
#+end_darkgray

#+begin_gray
This text is gray!
#+end_gray

#+begin_green
This text is green!
#+end_green

#+begin_lightgray
This text is lightgray!
#+end_lightgray

#+begin_lime
```

This text is blue!
This text is brown!
This text is cyan!
This text is darkgray!
This text is gray!
This text is green!
This text is lightgray!
This text is lime!
This text is magenta!
This text is olive!
This text is orange!
This text is pink!
This text is purple!
This text is red!
This text is teal!
This text is violet!

This text is black!
This text is blue!
This text is brown!
This text is cyan!
This text is darkgray!
This text is gray!
This text is green!
This text is lightgray!
This text is lime!
This text is magenta!
This text is olive!
This text is orange!
This text is pink!
This text is purple!
This text is red!
This text is teal!
This text is violet!

This text is yellow!

This text is black!
This text is blue!
This text is brown!
This text is cyan!
This text is darkgray!
This text is gray!
This text is green!
This text is lightgray!
This text is lime!
This text is magenta!
This text is olive!
This text is orange!
This text is pink!
This text is purple!
This text is red!
This text is teal!
This text is violet!

This text is yellow!

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Load support for 19 colour custom blocks

(defvar org-special-block-extras--colors
  '(black blue brown cyan darkgray gray green lightgray lime
          magenta olive orange pink purple red teal violet white
          yellow)
  "Colours that should be available on all systems.")

(loop for colour in org-special-block-extras--colors
      do (eval (read (format
                      "(defun org-special-block-extras--%s (backend contents)
                      (format (pcase backend
                      (`latex \"\\\\begingroup\\\\color{%s}%%s\\\\endgroup\")
                      (`html  \"<div style=\\\"color:%s;\\\">%%s</div>\")
                      (t      \"org-special-block-extras: Unsupported backend\"))
                      contents))"
                      colour colour colour))))
```

4 Parallel

Example:

```
#+begin_3parallel org
one

#+latex: \columnbreak
two

#+latex: \columnbreak
three
#+end_3parallel
```

Yields:

one two three

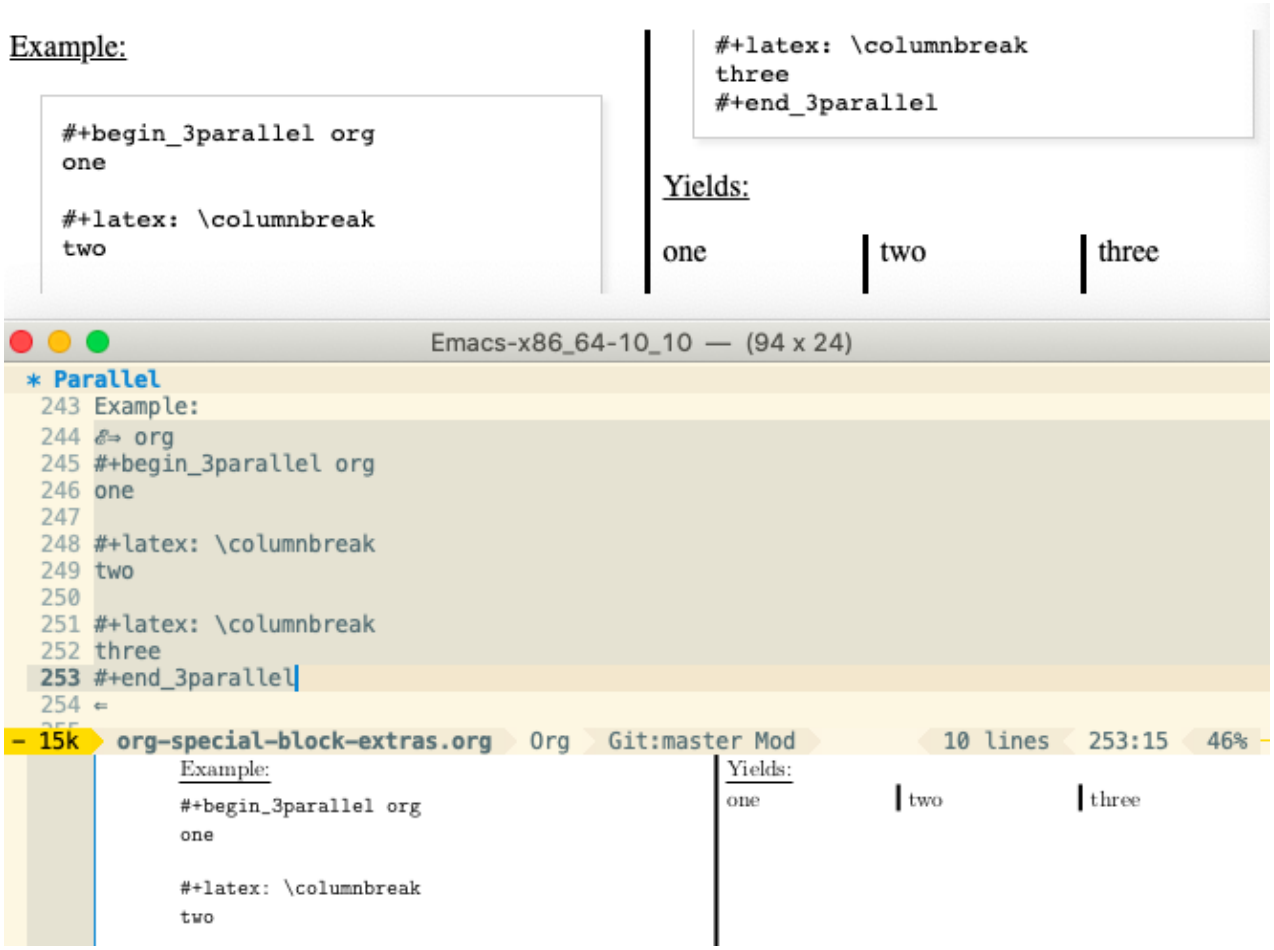

Figure 3: Displaying thoughts side-by-side ^_^

I initially used the names `paralell<n>` but names ending with a number did not inherit highlighting, so I shifted the number to being a prefix instead.

- For LaTeX, new lines are used to suggest opportunities for column breaks and are needed even if explicit columnbreaks are declared.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Parallel blocks: parallel<n>[NB] for n:2..5, optionally with ‘N’o ‘b’ar
;; in-between the columns.
;;
;; Common case is to have three columns, and we want to avoid invoking the
;; attribute via org, so making this.

(loop for cols in '("1" "2" "3" "4" "5")
      do (loop for rule in '("solid" "none")
      do (eval (read (concat
"(defun org-special-block-extras--" cols "parallel"
(if (equal rule "solid") "" "NB")
"(backend contents)"
"(format (pcase backend"
"(‘html \"<div style=\\\"column-rule-style:" rule ";column-count:" cols ";\\\"%s</div>\")"
"(‘latex \"\\\\par \\\\setlength{\\\\columnseprule}{" (if (equal rule "solid") "2" "0") "pt}"
"          \\\\begin{minipage}[t]{\\\\linewidth}"
"          \\\\begin{multicols}{" cols "}"
"          %s"
"          \\\\end{multicols}\\\\end{minipage}\")) contents))")))))

(defalias #'org-special-block-extras--parallel   #'org-special-block-extras--2parallel)
(defalias #'org-special-block-extras--parallelNB #'org-special-block-extras--2parallelNB)
```

(The Emacs Web Wowser, `M-x eww`, does not display `parallel` environments as desired.)