

```

(message-for 20 "foo")
; HyperTIES Storyboard mode
; Don Hopkins
(popmsg "foo") (message-for 10 "foobar...")

(declare-global "narrow-storyboards")
;(setq-default narrow-storyboards pop-up-frames)
(setq-default narrow-storyboards 0)
(declare-buffer-specific &ht-title)
(setq-default &ht-title "undefined title")

(defun (ht-lookup &hl-name
  (setq &hl-name (ht-canonicalize (arg 1 ": ht-lookup (name) ")))
  (save-window-excursion
    (switch-to-buffer "master-index")
    (beginning-of-file)
    (setq case-fold-search 1)
    (if (error-occurred (search-forward (concat "\n\"" &hl-name "\"")))
      ".unknown"
      (progn
        (end-of-line)
        (re-search-reverse "\"[\\t ]+")
        (region-around-match 0)
        (if (looking-at "\\.\/")
          (progn (forward-character)
                 (forward-character)
                 (set-mark)
                 (end-of-line)
                 (concat
                   (file-directory-name (current-file-name))
                   (region-to-string)))
          )
        (progn (set-mark)
                 (end-of-line)
                 (region-to-string))
        )
      )
    )
  )
)

))

(defun (create-storyboard &title &file &dir
  (setq &title (arg 1 ": create-storyboard (title) "))
  (if (!= (setq &file (ht-lookup &title)) ".unknown")
    (message &title " exists in " &file)
    (progn
      (setq &file (get-tty-file ": create-storyboard (file) "))
      (if (!= (substr &file -4 4) ".st0")
        (setq &file (concat &file ".st0"))
        )
      (save-window-excursion
        (switch-to-buffer "master-index")
        (setq &dir (file-directory-name (current-file-name)))
        (if (= (substr &file 0 (length &dir)) &dir)
          (setq &file
            (substr &file
              (+ (length &dir) 1)

```

```

                                (- (length &file)
                                   (length &dir)))
                                )
                                (if (& (!= (substr &file 1 1) "/")
                                     (!= (substr &file 1 2) "./")))
                                    (setq &file (concat "./" &file)))
                                (beginning-of-file)
                                (search-forward "----- ARTICLES -----")
                                (next-line)
                                (beginning-of-line)
                                (insert-string "\n\" &title "\"")
                                (while (< (current-column) 65)
                                    (insert-string "\t") ; tab
                                )
                                (insert-string &file "\n")
                                )
                                )
                                )
                                )
                                (articulate &title)
                                (widen-region)
                                (if (! (buffer-size))
                                    (progn (insert-string ".title\n" &title "\n\n")
                                           (insert-string ".synonyms\n\n")
                                           (insert-string ".definition\n\n\n")
                                           (insert-string ".contents\n\n")
                                    )
                                )
                                )
                                (narrow-storyboard)
                                (previous-window)
                                ))

; show-definition
; edit-definition

; show-title
; edit-title

; show-notes
; edit-notes

; show-synonyms
; edit-synonyms: lets you add as many as you want, then puts them in
; master-index. Able to delete them too.

(defun (ht-add-synonym &has-name &has-file &has-synonym
  (setq &has-name (ht-canonicalize (arg 1 ": ht-add-synonym (name) ")))
  (if (= (setq &has-file (ht-lookup &has-name)) ".unknown")
      (message "Can't find " &has-name " in master-index! " &has-file)
      (save-window-excursion
        (setq &has-synonym
              (ht-canonicalize (arg 2 ": ht-add-synonym (synonym) ")))
        (switch-to-buffer "master-index")
        (beginning-of-file)
        (setq case-fold-search 1)
        (search-forward (concat "\n\" &has-name "\""))
        (beginning-of-line)
        (search-forward "\n\n")
        (backward-character)

```

```

(insert-string "\"" &has-synonym "\"\n")
(visit-file &has-file)
(save-restriction
  (widen-region)
  (beginning-of-file)
  (if (error-occurred (search-forward ".synonyms"))
    (if (error-occurred
        (search-forward ".title")
        (search-forward "\n\n"))
      )
    (message "Can't find .synonym or .title field in "
      &has-file)
    (progn (beginning-of-line)
      (insert-string
        ".synonyms\n" &has-synonym "\n\n")
      )
    )
  (progn (end-of-line)
    (insert-string "\n" &has-synonym)
    )
  )
)
)
)
))

```

```

; Pass second argument to keep it from folding case
(defun (ht-canonicalize &string &fold
  (setq &string (arg 1 ": ht-canonicalize (string) "))
  (setq &fold (< (nargs) 2))
  (save-window-excursion
    (switch-to-buffer "*canonicalize*")
    (setq needs-checkpointing 0)
    (erase-buffer)
    (insert-string &string)
    (set-mark)
    (beginning-of-file)
    (if &fold (case-region-lower))
    (error-occurred
      (re-replace-string "[ \n\t<>]+" " "))
    (beginning-of-file)
    (if (looking-at " ") (delete-next-character))
    (end-of-file)
    (error-occurred
      (backward-character)
      (if (looking-at " ") (delete-next-character)))
    (buffer-to-string "*canonicalize*")
  )
)
)

```

```

(defun (gobble-expr &ge-str
  (while (looking-at "[ \t\n]") (forward-character))
  (if (looking-at "[<{}]")
    (progn (region-around-match 0)
      (set-mark)
      (backward-character)
      (if (error-occurred (forward-paren))
        ""

```

```

                (progn (backward-character)
                        (setq &ge-str (region-to-string))
                        (forward-character)
                        &ge-str
                )
        )
    )
    (if (looking-at "~")
        (progn
            (forward-character)
            (set-mark)
            (if (error-occurred (search-forward "~"))
                ""
                (progn (backward-character)
                        (setq &ge-str (region-to-string))
                        (forward-character)
                        &ge-str
                )
            )
        )
    )
    (progn
        (forward-word)
        (backward-word)
        (set-mark)
        (forward-word)
        (region-to-string)
    )
)
)
))

(defun (do-dot-command &command
  (setq &command (arg 1 " : do-dot-command "))
  (if (error-occurred
      (execute-mlisp-line (concat "(" &command ")")))
      (message (concat "Error: " &command))
  )
)
))

(defun (.button
  (press-button (gobble-expr) (gobble-expr) (gobble-expr))
)
)

(defun (.target &t-class &t-ref
  (setq &t-class (gobble-expr))
  (setq &t-ref (gobble-expr))
  (press-button &t-ref &t-class &t-ref)
)
)

(defun (press-button &b-label &b-class &b-ref
  (setq &b-label (arg 1 " : dot-button (label) "))
  (setq &b-class (arg 2 " : dot-button (class) "))
  (setq &b-ref (arg 3 " : dot-button (ref) "))
  ;(message-for 20 (concat ".button " &b-label " ; " &b-class " ; " &b-ref))
  (if (error-occurred
      (execute-mlisp-string (concat
          "(press-button-" &b-class " \" " &b-label "\" \" " &b-ref "\"")
      )
  )
)
)

```

```

    (save-excursion
      (save-window-excursion (articulate &b-label)))
  )
))

(defun (ht-button &b-dot &b-cmd
  (if (! (bit& (count-tildes-above) 1)) ; we're outside
    (progn (if (! (eobp)) (forward-character))
      (setq &b-dot (+ (dot)))
      (if (! (error-occurred
        (re-search-reverse
          "\\W\\(\\.\\.\\.^[0-9.]\\w*\\.\\.\\)[ \\n\\t]*"))
        (progn
          (region-around-match 1)
          (if (< (dot) &b-dot)
            ".background"
            (progn (setq &b-cmd (region-to-string))
              (region-around-match 0)
              &b-cmd
            )
          )
        )
      (progn ; .background
        ".background"
      )
    )
  )
  (progn (search-reverse "~")
    ".default-button"
  )
)
))

(defun (run-program &rp-dir &rp-pupw &rp-cmd
  (temp-use-buffer "master-index"
    (setq &rp-dir (file-directory-name (current-file-name))))
; (setq &rp-pupw pop-up-process-windows)
; (setq pop-up-process-windows 0)
  (setq &rp-cmd
    (concat "cd " &rp-dir " ; "
      (arg 1 ": run-program ")))
; (popmsg &rp-cmd)
  (execute-shell-command &rp-cmd "*cmd*")
; (setq pop-up-process-windows &rp-pupw)
))

; I don't want the popups!!
(defun (#report-process-status
; (if (! (buffer-is-visible (sending-process)))
; (popmsg (arg 1))
; )
))

(defun (.link &link-ref
  (setq &link-ref (gobble-expr))
  (press-button-link "link" &link-ref)
))

```

```

(defun (press-button-link
  (save-excursion (save-window-excursion (articulate (arg 2))))
  (concat ".link <" (arg 2) ">")
))

(defun (.default-button
  (.link)
))

(defun (.psh &psh-ref
  (setq &psh-ref (gobble-expr))
  (press-button-psh "psh" &psh-ref)
))

(defun (press-button-psh
  (message-for 1 (arg 1) " : Loading PS file " (arg 2))
  (run-program (concat "psh " (arg 2)))
  (concat ".psh <" (arg 2) ">")
))

(defun (.popmsg &popmsg-ref
  (setq &popmsg-ref (gobble-expr))
  (press-button-popmsg "popmsg" &popmsg-ref)
))

(defun (press-button-popmsg
  (popmsg (arg 2))
  (message-for 1 (arg 2))
  (concat ".popmsg <" (arg 2) ">")
))

(defun (.psview &psview-ref
  (setq &psview-ref (gobble-expr))
  (press-button-psview "psview" &psview-ref)
))

(defun (press-button-psview
  (message-for 1 (arg 1) " : Viewing PS file " (arg 2))
  (run-program (concat "psview " (arg 2)))
  (concat ".psview <" (arg 2) ">")
))

(defun (.paper &paper-ref
  (setq &paper-ref (gobble-expr))
  (press-button-paper "paper" &paper-ref)
))

(defun (press-button-paper
  (message-for 1 (arg 1) " : Previewing PS file " (arg 2))
  (run-program (concat "paper " (arg 2)))
  (concat ".paper <" (arg 2) ">")
))

(defun (.fork &fork-ref
  (setq &fork-ref (gobble-expr))
  (press-button-fork "fork" &fork-ref)
))

```

```

(defun (press-button-fork
  (message-for 5 (arg 1) " : Forking program " (arg 2))
  (run-program (arg 2))
  (concat ".fork <" (arg 2) ">"))
))

(defun (.file &file-ref
  (setq &file-ref (gobble-expr))
  (press-button-file "file" &file-ref)
))

(defun (press-button-file &b-file &b-dir
  (setq &b-file (arg 2 ": press-button-file (file) "))
  (message-for 1 (arg 1) " : Editing file " &b-file)
  (if (= (substr &b-file 1 1) "/")
    (setq &b-dir "")
    (temp-use-buffer "master-index"
      (setq &b-dir (file-directory-name (current-file-name))))))
  (save-excursion
    (save-window-excursion
      (visit-file (concat &b-dir &b-file))
      (frame-to-top)
    ))
  (concat ".file " &b-file)
))

(defun (.directory &directory-ref
  (setq &directory-ref (gobble-expr))
  (press-button-directory "directory" &directory-ref)
))

(defun (press-button-directory &b-file &b-dir
  (setq &b-file (arg 2 ": press-button-directory (dir) "))
  (message-for 19 (arg 1) " : Editing directory " &b-file)
  (if (= (substr &b-file 1 1) "/")
    (setq &b-dir "")
    (temp-use-buffer "master-index"
      (setq &b-dir (file-directory-name (current-file-name))))))
  (save-excursion
    (save-window-excursion
      (dired (concat &b-dir &b-file))
      (frame-to-top)
    ))
  (concat ".directory " &b-file)
))

(defun (count-tildes-above &tildes
  (save-excursion
    (setq &tildes 0)
    (while (! (error-occurred (search-reverse "~")))
      (++ &tildes)
      (error-occurred
        (backward-character)
        (if (looking-at "\\") ; emacs ")--:
          (-- &tildes)
          (forward-character)
        )
      )
    )
  )
)

```

```

        )
    )
    &tildes
)
))

(defun (articulate &ea-name &ea-file
  (setq &ea-name (arg 1 ": articulate (name) "))
  (if (= (setq &ea-file (ht-lookup &ea-name)) ".unknown")
    (message (concat &ea-file " <" (ht-canonicalize &ea-name) ">"))
    (progn (message "Found " &ea-name " in " &ea-file)
      (find-file &ea-file)
      (narrow-storyboard)
      (frame-to-top)
      (save-command-context)
      (mouse-to-frame)
    )
  )
)
))

```