

```

;;
;;   WEBIFY. Take a jpg (or an entire directory) and save it 'for the web'
;;   Output file is JPG 72dpi 800px long side (or at choice) with the
;;   provided compression factor.
;;
;;   At choice: (1) save for web
;;               (2) convert to grayscale and save for the web
;;               (3) Both of them (two files created)
;;
;; Copyright (C) 2008 Free Software Foundation
;;
;; This library is free software; you can redistribute it and/or
;; modify it under the terms of the GNU Lesser General Public
;; License as published by the Free Software Foundation; either
;; version 2.1 of the License.
;;
;; This library is distributed in the hope that it will be useful,
;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
;; Lesser General Public License for more details.
;;
;; You should have received a copy of the GNU Lesser General Public
;; License along with this library; if not, write to the Free Software
;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
;;
;; http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt

;;
; parse the file path and return it without extension
;
; @param filename String the file path
; @return the file path without the extension
(define (parse-name filename)
  (set! fparts (strbreakup filename "."))
  (set! y 0)
  (set! out "")
  (set! lngt (- (length fparts) 1))
  (while (< y lngt)
    (set! out (string-append out (nth y fparts)))
    (set! y (+ y 1))
    (if (< y lngt)
        (set! out (string-append out "."))
      )
  )
  (car (cons out '())))
)

;;
; retrieve the dimensions of the provided image and return the right width+height for the web
;
; @param img IMAGE the loaded image
; @return a list with at the first item the width and the last the height (width, height)
(define (get-dimension img long-side)
  (let*
    (
      (w (car (gimp-image-width img)))
      (h (car (gimp-image-height img)))
      (w2 0)
      (h2 0)
    )
    (cond
      ((> w h)
       (set! w2 long-side)
       (set! h2 (/ (* h long-side) w))
      )
      ((< w h)
       (set! w2 (/ (* w long-side) h))
       (set! h2 long-side)
      )
    )
    (list w2 h2)
  )
)

;;
; open a JPG image set his DPI to 72, scale it to 800x533 and save as JPG with no compression
; the image will be saved in the same directory as <file-without-extension>-72.jpg
;
; @param filename String the file path
;

```

```

(define (color filename jpg-quality long-side)
  (let*
    (
      (fname (parse-name filename)) ; getting the filepath without extension
      (scaled (string-append fname "-72.jpg"))
      (width long-side) ;image width
      (height 533) ;image height
      (dpi 72) ;image resolution
      (quality (/ jpg-quality 100))
      (image (car (gimp-file-load RUN-NONINTERACTIVE filename filename))) ;load image
      (drawable (car (gimp-image-get-active-layer image))) ;get drawable
      (display (car (gimp-display-new image))) ;display the image
      (dimensions (get-dimension image long-side))
    )
    (set! width (car dimensions))
    (set! height (cadr dimensions))
    (gimp-image-set-resolution image dpi dpi) ;setting image resolution
    (gimp-image-scale image width height) ; scale the image
    (file-jpeg-save 1 image drawable scaled scaled quality 0 0 0 " " 0 1 0 1)
    (gimp-display-delete display)
  )
)

;;
; open a JPG image set his DPI to 72, scale it to 800x533 and save as JPG with no compression
; and convert it to grayscale. Then save
; the image will be saved in the same directory as <file-without-extension>-72gray.jpg
;
; @param filename String the file path
;
(define (gray filename jpg-quality long-side)
  (let*
    (
      (fname (parse-name filename)) ; getting the filepath without extension
      (grayed (string-append fname "-72gray.jpg"))
      (width 800) ;image width
      (height 533) ;image height
      (dpi 72) ;image resolution
      (quality (/ jpg-quality 100))
      (image (car (gimp-file-load RUN-NONINTERACTIVE filename filename))) ;load image
      (drawable (car (gimp-image-get-active-layer image))) ;get drawable
      (display (car (gimp-display-new image))) ;display the image
      (dimensions (get-dimension image long-side))
    )
    (set! width (car dimensions))
    (set! height (cadr dimensions))
    (gimp-image-set-resolution image dpi dpi) ;setting image resolution
    (gimp-image-scale image width height) ; scale the image
    (if (= (car (gimp-drawable-is-gray drawable)) 0)
      (gimp-image-convert-grayscale image)
    )
    (file-jpeg-save 1 image drawable grayed grayed quality 0 0 0 " " 0 1 0 1)
    (gimp-display-delete display)
  )
)

;;
; take the for-web and gray action
;
; @param filename String the file path
;
(define (color-and-gray filename jpg-quality long-side)
  (color filename jpg-quality long-side)
  (gray filename jpg-quality long-side)
)

;;
; replace a character in a string with a new one
;
; @param string String the string to be parsed
; @param oldchar the character to be replaced
; @param newchar the new character
; @return the string with the replaced characters
(define (replace-char string oldchar newchar)
  (set! arr (strbreakup string oldchar))
  (set! out "")
  (while (not (null? arr))
    (set! out (string-append out (car arr)))
    (set! arr (cdr arr))
    (if (not (null? arr))

```

```

        (set! out (string-append out newchar))
    )
)
(car (cons out '()))
)

;;
; fix the pattern appending the right /*.jpg or \*.jpg depending on the
; operating system
;
; @param dir String the directory
; @param ext the pattern (extension of files)
; @return the fixed pattern like /home/davide/*.jpg or C:\\abc\\*.jpg
(define (fixpattern dir ext)
  (set! out dir)
  (if (> (length (strbreakup dir "\\")) 1)
      (set! out (string-append out "\\"))
      (set! out (string-append out "/")))
  )
  (set! out (string-append out ext))
  (car (cons out '()))
)

;;
; take the for-web action on every *.jpg files in a directory
;
; @param pattern String the file-glob pattern
(define (batch-color pattern jpg-quality long-side)
  (set! filelist (cadr (file-glob pattern 1)))
  (while (not (null? filelist))
    (set! filename (car filelist))
    (color filename jpg-quality long-side)
    (set! filelist (cdr filelist)))
  )
)

;;
; take the gray action on every *.jpg files in a directory
;
; @param pattern String the file-glob pattern
(define (batch-gray pattern jpg-quality long-side)
  (set! filelist (cadr (file-glob pattern 1)))
  (while (not (null? filelist))
    (set! filename (car filelist))
    (gray filename jpg-quality long-side)
    (set! filelist (cdr filelist)))
  )
)

;;
; take the color-and-gray action on every *.jpg files in a directory
;
; @param pattern String the file-glob pattern
(define (batch-color-gray pattern jpg-quality long-side)
  (set! filelist (cadr (file-glob pattern 1)))
  (while (not (null? filelist))
    (set! filename (car filelist))
    (color-and-gray filename jpg-quality long-side)
    (set! filelist (cdr filelist)))
  )
)

(define (webify-single path jpg-quality long-side elaboration)
  (let*
    (
      )
    (cond
      ((= elaboration 0) ;web
       (color path jpg-quality long-side)
      )
      ((= elaboration 1) ;gray
       (gray path jpg-quality long-side)
      )
      ((= elaboration 2) ; both
       (color-and-gray path jpg-quality long-side)
      )
    )
  )
)

```

```

(define (webify-batch path pattern jpg-quality long-side elaboration)
  (let* ()
    (cond
      ((= pattern 0) ;*.JPG
       (set! pattern "*.JPG")
      )
      ((= pattern 1) ;*.jpg
       (set! pattern "*.jpg")
      )
    )
    (set! pattern (fixpattern path pattern))
    (cond
      ((= elaboration 0) ;web
       (batch-color pattern jpg-quality long-side)
      )
      ((= elaboration 1) ;gray
       (batch-gray pattern jpg-quality long-side)
      )
      ((= elaboration 2) ; both
       (batch-color-gray pattern jpg-quality long-side)
      )
    )
  )
)

(script-fu-register "webify-single"
  "<Toolbox>/Xtns/Script-Fu/My Stuff/For Web/Single File"
  "Take the selected image and save it 'for the web'. 72dpi 800px long size jpg"
  "Davide Giannella"
  "(C) 2008 Free Software Foundation"
  "April 2008"
  ""
  SF-FILENAME "File Path" ""
  SF-ADJUSTMENT "JPG Quality (1 <= quality <= 100)" '(100 1 100 1 10 0 1)
  SF-VALUE "Long side (px)" "800"
  SF-OPTION "Elaboration" '("Color" "Gray" "Both")
)

(script-fu-register "webify-batch"
  "<Toolbox>/Xtns/Script-Fu/My Stuff/For Web/Directory"
  "Take the selected direcoty and save the jpgs in 'for the web'. 72dpi 800px long size jpg"
  "Davide Giannella"
  "(C) 2008 Free Software Foundation"
  "April 2008"
  ""
  SF-DIRNAME "Directory" ""
  SF-OPTION "Pattern" '("*.JPG" "*.jpg")
  SF-ADJUSTMENT "JPG Quality (1 <= quality <= 100)" '(100 1 100 1 10 0 1)
  SF-VALUE "Long side (px)" "800"
  SF-OPTION "Elaboration" '("Color" "Gray" "Both")
)

```