Image Processing (GD)

Introduction

PHP is not limited to creating just HTML output. It can also be used to create and manipulate image files in a variety of different image formats, including gif, png, jpg, wbmp, and xpm. Even more convenient, PHP can output image streams directly to a browser. You will need to compile PHP with the GD library of image functions for this to work. GD and PHP may also require other libraries, depending on which image formats you want to work with.

You can use the image functions in PHP to get the size of JPEG, GIF, PNG, SWF, TIFF and JPEG2000 images.

With the exif extension, you are able to work with information stored in headers of JPEG and TIFF images. This way you can read meta data generated by digital cameras. The exif functions do not require the GD library.

Note
Read the requirements section about how to expand image capabilities to read, write and modify images. To read meta data of pictures taken by digital cameras you need the above mentioned exif extension.

Installing/Configuring

Requirements

If you have the GD library (available at » http://www.libgd.org/) you will also be able to create and manipulate images.

The format of images you are able to manipulate depend on the version of GD you install, and any other libraries GD might need to access those image formats. Versions of GD older than gd-1.6 support GIF format images, and do not support PNG, where versions greater than gd-1.6 and less than gd-2.0.28 support PNG, not GIF. GIF support was re-enabled in gd-2.0.28.

Note
Since PHP 4.3 there is a bundled version of the GD lib. This bundled version has some additional features like alpha blending, and should be used in preference to the external library since its codebase is better maintained and more stable.

Note
Support for GD 1.x has been removed as of PHP 6.0.0, which requires GD 2.0.33 or later.

You may wish to enhance GD to handle more image formats.

Supported image formats

Image format	Library to download	Notes
gif		Only supported in GD versions older than gd-1.6 and newer than gd-2.0.28. *Read-only* GIF support is available with PHP 4.3.0 and the bundled GD-library. *Write* support is available since PHP 4.3.9 and PHP 5.0.1.
jpeg-6b	» ftp://ftp.uu.net/graphics/jpeg/	When buliding the jpeg-v6b library (prior to building PHP) you must use the *--enable-shared* option in the configure step. If you do not,

		you will receive an error saying *libjpeg.(a|so) not found* when you get to the configure step of building PHP.
png	» http://www.libpng.org/pub/png/libpng.html	Only supported in GD versions greater than gd-1.6.
xpm	» ftp://metalab.unc.edu/pub/Linux/libs/X/!INDEX.html	It's likely you have this library already available, if your system has an installed X-Environment.

You may wish to enhance GD to deal with different fonts. The following font libraries are supported:

Supported font libraries

Font library	Download	Notes
FreeType 1.x	» http://www.freetype.org/	Support removed as of PHP 6.0.0
FreeType 2	» http://www.freetype.org/	
T1lib	» ftp://sunsite.unc.edu/pub/Linux/libs/graphics/)	Support for Postscript Type 1 fonts.

Installation

To enable GD-support configure PHP *--with-gd[=DIR]*, where DIR is the GD base install directory. To use the recommended bundled version of the GD library (which was first bundled in PHP 4.3.0), use the configure option *--with-gd*. GD library requires libpng and libjpeg to compile.

In Windows, you'll include the GD2 DLL *php_gd2.dll* as an extension in *php.ini*. The GD1 DLL *php_gd.dll* was removed in PHP 4.3.2. Also note that the preferred truecolor image functions, such as imagecreatetruecolor(), require GD2.

Enhance the capabilities of GD to handle more image formats by specifying the *--with-XXXX* configure switch to your PHP configure line.

Supported image formats

Image Format	Configure Switch

jpeg-6b	To enable support for jpeg-6b add *--with-jpeg-dir=DIR*.
png	To enable support for png add *--with-png-dir=DIR*. Note, libpng requires the zlib library, therefore add *--with-zlib-dir[=DIR]* to your configure line.
xpm	To enable support for xpm add *--with-xpm-dir=DIR*. If configure is not able to find the required libraries, you may add the path to your X11 libraries.

Note
When compiling PHP with libpng, you must use the same version that was linked with the GD library.

Enhance the capabilities of GD to deal with different fonts by specifying the *--with-XXXX* configure switch to your PHP configure line.

Supported font libraries

Font library	**Configure Switch**
FreeType 1.x	To enable support for FreeType 1.x add *--with-ttf[=DIR]*.
FreeType 2	To enable support for FreeType 2 add *--with-freetype-dir=DIR*.
T1lib	To enable support for T1lib (Postscript Type 1 fonts) add *--with-t1lib[=DIR]*.
Native TrueType string function	To enable support for native TrueType string function add *--enable-gd-native-ttf*.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

Image Configure Options

Name	Default	Changeable	Changelog
gd.jpeg_ignore_warning	"0"	PHP_INI_ALL	Available since PHP 5.1.3.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

gd.jpeg_ignore_warning bool
: Ignore warnings created by jpeg2wbmp() and imagecreatefromjpeg()

See also the exif configuration directives.

Warning
Image functions are very memory intensive. Be sure to set memory_limit high enough.

Resource Types

This extension defines two resource types: an image identifier and a font identifier.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

GD_VERSION (string)
The GD version PHP was compiled against. (Available as of PHP 5.2.4)

GD_MAJOR_VERSION (integer)
The GD major version PHP was compiled against. (Available as of PHP 5.2.4)

GD_MINOR_VERSION (integer)
The GD minor version PHP was compiled against. (Available as of PHP 5.2.4)

GD_RELEASE_VERSION (integer)
The GD release version PHP was compiled against. (Available as of PHP 5.2.4)

GD_EXTRA_VERSION (string)
The GD "extra" version (beta/rc..) PHP was compiled against. (Available as of PHP 5.2.4)

IMG_GIF (integer)
Used as a return value by imagetypes()

IMG_JPG (integer)
Used as a return value by imagetypes()

IMG_JPEG (integer)
Used as a return value by imagetypes()

Note
This constant has the same value as **IMG_JPG**

IMG_PNG (integer)
Used as a return value by imagetypes()

IMG_WBMP (integer)
Used as a return value by imagetypes()

IMG_XPM (integer)
Used as a return value by imagetypes()

IMG_COLOR_TILED (integer)
Special color option which can be used in stead of color allocated with imagecolorallocate() or imagecolorallocatealpha()

IMG_COLOR_STYLED (integer)

Special color option which can be used in stead of color allocated with imagecolorallocate() or imagecolorallocatealpha()

IMG_COLOR_BRUSHED (integer)
Special color option which can be used in stead of color allocated with imagecolorallocate() or imagecolorallocatealpha()

IMG_COLOR_STYLEDBRUSHED (integer)
Special color option which can be used in stead of color allocated with imagecolorallocate() or imagecolorallocatealpha()

IMG_COLOR_TRANSPARENT (integer)
Special color option which can be used in stead of color allocated with imagecolorallocate() or imagecolorallocatealpha()

IMG_ARC_ROUNDED (integer)
A style constant used by the imagefilledarc() function.

Note
This constant has the same value as **IMG_ARC_PIE**

IMG_ARC_PIE (integer)
A style constant used by the imagefilledarc() function.

IMG_ARC_CHORD (integer)
A style constant used by the imagefilledarc() function.

IMG_ARC_NOFILL (integer)
A style constant used by the imagefilledarc() function.

IMG_ARC_EDGED (integer)
A style constant used by the imagefilledarc() function.

IMG_GD2_RAW (integer)
A type constant used by the imagegd2() function.

IMG_GD2_COMPRESSED (integer)
A type constant used by the imagegd2() function.

IMG_EFFECT_REPLACE (integer)
Alpha blending effect used by the imagelayereffect() function.

IMG_EFFECT_ALPHABLEND (integer)
Alpha blending effect used by the imagelayereffect() function.

IMG_EFFECT_NORMAL (integer)
Alpha blending effect used by the imagelayereffect() function.

IMG_EFFECT_OVERLAY (integer)

Alpha blending effect used by the imagelayereffect() function.

IMG_FILTER_NEGATE (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_GRAYSCALE (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_BRIGHTNESS (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_CONTRAST (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_COLORIZE (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_EDGEDETECT (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_GAUSSIAN_BLUR (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_SELECTIVE_BLUR (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_EMBOSS (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_MEAN_REMOVAL (integer)
Special GD filter used by the imagefilter() function.

IMG_FILTER_SMOOTH (integer)
Special GD filter used by the imagefilter() function.

IMAGETYPE_GIF (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_JPEG (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_PNG (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_SWF (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_PSD (integer)

Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_BMP (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_WBMP (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_XBM (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_TIFF_II (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_TIFF_MM (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_IFF (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_JB2 (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_JPC (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_JP2 (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_JPX (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_SWC (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions.

IMAGETYPE_ICO (integer)
Image type constant used by the image_type_to_mime_type() and image_type_to_extension() functions. (Available as of PHP 5.3.0)

PNG_NO_FILTER (integer)

A special PNG filter, used by the imagepng() function.

PNG_FILTER_NONE (integer)
A special PNG filter, used by the imagepng() function.

PNG_FILTER_SUB (integer)
A special PNG filter, used by the imagepng() function.

PNG_FILTER_UP (integer)
A special PNG filter, used by the imagepng() function.

PNG_FILTER_AVG (integer)
A special PNG filter, used by the imagepng() function.

PNG_FILTER_PAETH (integer)
A special PNG filter, used by the imagepng() function.

PNG_ALL_FILTERS (integer)
A special PNG filter, used by the imagepng() function.

Examples

Example #1 - PNG creation with PHP

```
<?php

header("Content-type: image/png");
$string = $_GET['text'];
$im     = imagecreatefrompng("images/button1.png");
$orange = imagecolorallocate($im, 220, 210, 60);
$px     = (imagesx($im) - 7.5 * strlen($string)) / 2;
imagestring($im, 3, $px, 9, $string, $orange);
imagepng($im);
imagedestroy($im);

?>
```

This example would be called from a page with a tag like: <img src="button.php?text=text">. The above button.php script then takes this "text" string and overlays it on top of a base image which in this case is "images/button1.png" and outputs the resulting image. This is a very convenient way to avoid having to draw new button images every time you want to change the text of a button. With this method they are dynamically generated.

GD Functions

gd_info

gd_info -- Retrieve information about the currently installed GD library

Description

array **gd_info** (void)

Gets information about the version and capabilities of the installed GD library.

Return Values

Returns an associative array.

Elements of array returned by gd_info()

Attribute	**Meaning**
GD Version	string value describing the installed *libgd* version.
Freetype Support	boolean value. **TRUE** if Freetype Support is installed.
Freetype Linkage	string value describing the way in which Freetype was linked. Expected values are: 'with freetype', 'with TTF library', and 'with unknown library'. This element will only be defined if *Freetype Support* evaluated to **TRUE**.
T1Lib Support	boolean value. **TRUE** if *T1Lib* support is included.
GIF Read Support	boolean value. **TRUE** if support for *reading GIF* images is included.
GIF Create Support	boolean value. **TRUE** if support for *creating GIF* images is included.
JPG Support	boolean value. **TRUE** if *JPG* support is included.
PNG Support	boolean value. **TRUE** if *PNG* support is included.
WBMP Support	boolean value. **TRUE** if *WBMP* support is

	included.
XBM Support	boolean value. **TRUE** if *XBM* support is included.

Examples

Example #2 - Using gd_info()

```
<?php
var_dump(gd_info());
?>
```

The above example will output something similar to:

```
array(9) {
 ["GD Version"]=>
 string(24) "bundled (2.0 compatible)"
 ["FreeType Support"]=>
 bool(false)
 ["T1Lib Support"]=>
 bool(false)
 ["GIF Read Support"]=>
 bool(true)
 ["GIF Create Support"]=>
 bool(false)
 ["JPG Support"]=>
 bool(false)
 ["PNG Support"]=>
 bool(true)
 ["WBMP Support"]=>
 bool(true)
 ["XBM Support"]=>
 bool(false)
}
```

See Also

- imagepng()
- imagejpeg()
- imagegif()
- imagewbmp()
- imagetypes()

getimagesize

getimagesize -- Get the size of an image

Description

array **getimagesize** (string `$filename` [, array `&$imageinfo`])

The getimagesize() function will determine the size of any given image file and return the dimensions along with the file type and a height/width text string to be used inside a normal HTML `IMG` tag and the correspondant HTTP content type.

getimagesize() can also return some more information in *`imageinfo`* parameter.

Note
Note that JPC and JP2 are capable of having components with different bit depths. In this case, the value for "bits" is the highest bit depth encountered. Also, JP2 files may contain multiple JPEG 2000 codestreams. In this case, getimagesize() returns the values for the first codestream it encounters in the root of the file.

Note
The information about icons are retreived from the icon with the highest bitrate.

Parameters

`filename`
: This parameter specifies the file you wish to retrieve information about. It can reference a local file or (configuration permitting) a remote file using one of the supported streams.

`imageinfo`
: This optional parameter allows you to extract some extended information from the image file. Currently, this will return the different JPG APP markers as an associative array. Some programs use these APP markers to embed text information in images. A very common one is to embed » IPTC information in the APP13 marker. You can use the iptcparse() function to parse the binary APP13 marker into something readable.

Return Values

Returns an array with 5 elements.

Index 0 and 1 contains respectively the width and the height of the image.

Note
Some formats may contain no image or may contain multiple images. In these cases, getimagesize() might not be able to properly determine the image size. getimagesize() will return zero for width and height in these cases.

Index 2 is one of the *IMAGETYPE_XXX* constants indicating the type of the image.

Index 3 is a text string with the correct *height="yyy" width="xxx"* string that can be used directly in an IMG tag.

mime is the correspondant MIME type of the image. This information can be used to deliver images with correct the HTTP *Content-type* header:

Example #3 - getimagesize() and MIME types

```
<?php
$size = getimagesize($filename);
$fp = fopen($filename, "rb");
if ($size && $fp) {
   header("Content-type: {$size['mime']}");
   fpassthru($fp);
   exit;
} else {
   // error
}
?>
```

channels will be 3 for RGB pictures and 4 for CMYK pictures. *bits* is the number of bits for each color. However, for some image types, the presence of these values can be a bit confusing. As an example, GIF always uses 3 channels per pixel, but the number of bits per pixel cannot be calculated for an animated GIF with a global color table.

On failure, **FALSE** is returned.

Errors/Exceptions

If accessing the `filename` image is impossible, or if it isn't a valid picture, getimagesize() will generate an error of level **E_WARNING**. On read error, getimagesize() will generate an error of level **E_NOTICE**.

ChangeLog

Version	Description

5.3.0	Added icon support.
5.2.3	Read errors generated by this function downgraded to **E_NOTICE** from **E_WARNING**.
4.3.2	Support for JPC, JP2, JPX, JB2, XBM, and WBMP became available.
4.3.2	JPEG 2000 support was added for the *`imageinfo`* parameter.
4.3.0	*bits* and *channels* are present for other image types, too.
4.3.0	*mime* was added.
4.3.0	Support for SWC was added.
4.2.0	Support for TIFF was added.
4.0.5	URL support was added.

Examples

Example #4 - getimagesize (file)

```
<?php
list($width, $height, $type, $attr) = getimagesize("img/flag.jpg");
echo "<img src=\"img/flag.jpg\" $attr alt=\"getimagesize() example\" />";
?>
```

Example #5 - getimagesize (URL)

```
<?php
$size = getimagesize("http://www.example.com/gifs/logo.gif");

// if the file name has space in it, encode it properly
$size = getimagesize("http://www.example.com/gifs/lo%20go.gif");

?>
```

Example #6 - getimagesize() returning IPTC

```
<?php
$size = getimagesize("testimg.jpg", $info);
```

```
if (isset($info["APP13"])) {
   $iptc = iptcparse($info["APP13"]);
   var_dump($iptc);
}
?>
```

Notes

Note
The getimagesize() function does not require the GD image library.

See Also

- image_type_to_mime_type()
- exif_imagetype()
- exif_read_data()
- exif_thumbnail()

image_type_to_extension

image_type_to_extension -- Get file extension for image type

Description

string **image_type_to_extension** (int `$imagetype` [, bool `$include_dot`])

Returns the extension for the given *IMAGETYPE_XXX* constant.

Parameters

`imagetype`
: One of the *IMAGETYPE_XXX* constant.

`include_dot`
: Whether to prepend a dot to the extension or not. Default to **TRUE**.

Return Values

A string with the extension corresponding to the given image type.

image_type_to_mime_type

image_type_to_mime_type -- Get Mime-Type for image-type returned by getimagesize, exif_read_data, exif_thumbnail, exif_imagetype

Description

string **image_type_to_mime_type** (int `$imagetype`)

The image_type_to_mime_type() function will determine the Mime-Type for an IMAGETYPE constant.

Parameters

`imagetype`
: One of the *IMAGETYPE_XXX* constants

Return Values

The returned values are as follows

Returned values Constants

`imagetype`	**Returned value**
IMAGETYPE_GIF	*image/gif*
IMAGETYPE_JPEG	*image/jpeg*
IMAGETYPE_PNG	*image/png*
IMAGETYPE_SWF	*application/x-shockwave-flash*
IMAGETYPE_PSD	*image/psd*
IMAGETYPE_BMP	*image/bmp*
IMAGETYPE_TIFF_II (intel byte order)	*image/tiff*
IMAGETYPE_TIFF_MM (motorola byte order)	*image/tiff*
IMAGETYPE_JPC	*application/octet-stream*
IMAGETYPE_JP2	*image/jp2*
IMAGETYPE_JPX	*application/octet-stream*

IMAGETYPE_JB2	*application/octet-stream*
IMAGETYPE_SWC	*application/x-shockwave-flash*
IMAGETYPE_IFF	*image/iff*
IMAGETYPE_WBMP	*image/vnd.wap.wbmp*
IMAGETYPE_XBM	*image/xbm*
IMAGETYPE_ICO	*image/vnd.microsoft.icon*

Examples

Example #7 - image_type_to_mime_type (file)

```
<?php
header("Content-type: " . image_type_to_mime_type(IMAGETYPE_PNG));
?>
```

Notes

Note

This function does not require the GD image library.

See Also

- getimagesize()
- exif_imagetype()
- exif_read_data()
- exif_thumbnail()

image2wbmp

image2wbmp -- Output image to browser or file

Description

bool **image2wbmp** (resource `$image` [, string `$filename` [, int `$threshold`]])

image2wbmp() outputs or save a WBMP version of the given *`image`*.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filename`
: Path to the saved file. If not given, the raw image stream will be outputed directly.

`threshold`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #8 - image2wbmp() example

```
<?php

$file = 'php.png';
$image = imagecreatefrompng($file);

header('Content-type: ' . image_type_to_mime_type(IMAGETYPE_WBMP));
image2wbmp($image); // output the stream directly

?>
```

Notes

Note
WBMP support is only available if PHP was compiled against GD-1.8 or later.

See Also

- imagewbmp()

imagealphablending

imagealphablending -- Set the blending mode for an image

Description

bool **imagealphablending** (resource `$image`, bool `$blendmode`)

imagealphablending() allows for two different modes of drawing on truecolor images. In blending mode, the alpha channel component of the color supplied to all drawing function, such as imagesetpixel() determines how much of the underlying color should be allowed to shine through. As a result, gd automatically blends the existing color at that point with the drawing color, and stores the result in the image. The resulting pixel is opaque. In non-blending mode, the drawing color is copied literally with its alpha channel information, replacing the destination pixel. Blending mode is not available when drawing on palette images.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`blendmode`
Whether to enable the blending mode or not. Default to **FALSE**.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

imageantialias

imageantialias -- Should antialias functions be used or not

Description

bool **imageantialias** (resource `$image`, bool `$on`)

Activate the fast drawing antialiased methods for lines and wired polygons. It does not support alpha components. It works using a direct blend operation. It works only with truecolor images.

Thickness and styled are not supported.

Using antialiased primitives with transparent background color can end with some unexpected results. The blend method uses the background color as any other colors. The lack of alpha component support does not allow an alpha based antialiasing method.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`on`
Whether to enable antialiasing or not.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available if PHP is compiled with the bundled version of the GD library.

See Also

- imagecreatetruecolor()

imagearc

imagearc -- Draws an arc

Description

bool **imagearc** (resource `$image`, int `$cx`, int `$cy`, int `$width`, int `$height`, int `$start`, int `$end`, int `$color`)

imagearc() draws an arc of circle centered at the given coordinates.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`cx`
x-coordinate of the center

`cy`
y-coordinate of the center

`width`
The arc width

`height`
The arc height

`start`
The arc start angle, in degrees.

`end`
The arc end angle, in degrees. 0° is located at the three-o'clock position, and the arc is drawn clockwise.

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #9 - Drawing a circle with imagearc()

```
<?php

// create a 200*200 image
$img = imagecreatetruecolor(200, 200);

// allocate some colors
$white = imagecolorallocate($img, 255, 255, 255);
$red   = imagecolorallocate($img, 255,   0,   0);
$green = imagecolorallocate($img,   0, 255,   0);
$blue  = imagecolorallocate($img,   0,   0, 255);

// draw the head
imagearc($img, 100, 100, 200, 200,  0, 360, $white);
// mouth
imagearc($img, 100, 100, 150, 150, 25, 155, $red);
// left and then the right eye
imagearc($img,  60,  75,  50,  50,  0, 360, $green);
imagearc($img, 140,  75,  50,  50,  0, 360, $blue);

// output image in the browser
header("Content-type: image/png");
imagepng($img);

// free memory
imagedestroy($img);

?>
```

The above example will output something similar to:

See Also

- imagefilledarc()
- imageellipse()
- imagefilledellipse()

imagechar

imagechar -- Draw a character horizontally

Description

bool **imagechar** (resource `$image`, int `$font`, int `$x`, int `$y`, string `$c`, int `$color`)

imagechar() draws the first character of *`c`* in the image identified by *`image`* with its upper-left at *`x`*, *`y`* (top left is 0, 0) with the color *`color`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`font`
Can be 1, 2, 3, 4, 5 for built-in fonts in latin2 encoding (where higher numbers corresponding to larger fonts) or any of your own font identifiers registered with imageloadfont().

`x`
x-coordinate of the start

`y`
y-coordinate of the start

`c`
The character to draw

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #10 - imagechar() example

```
<?php

$im = imagecreate(100, 100);
```

```
$string = 'PHP';

$bg = imagecolorallocate($im, 255, 255, 255);
$black = imagecolorallocate($im, 0, 0, 0);

// prints a black "P" in the top left corner
imagechar($im, 1, 0, 0, $string, $black);

header('Content-type: image/png');
imagepng($im);

?>
```

The above example will output something similar to:

See Also

- imagecharup()
- imageloadfont()

imagecharup

imagecharup -- Draw a character vertically

Description

bool **imagecharup** (resource `$image`, int `$font`, int `$x`, int `$y`, string `$c`, int `$color`)

Draws the character *`c`* vertically at the specified coordinate on the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`font`
Can be 1, 2, 3, 4, 5 for built-in fonts in latin2 encoding (where higher numbers corresponding to larger fonts) or any of your own font identifiers registered with imageloadfont().

`x`
x-coordinate of the start

`y`
y-coordinate of the start

`c`
The character to draw

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #11 - imagecharup() example

```
<?php

$im = imagecreate(100, 100);
```

```
$string = 'Note that the first letter is a N';

$bg = imagecolorallocate($im, 255, 255, 255);
$black = imagecolorallocate($im, 0, 0, 0);

// prints a black "Z" on a white background
imagecharup($im, 3, 10, 10, $string, $black);

header('Content-type: image/png');
imagepng($im);

?>
```

The above example will output something similar to:

See Also

- imagechar()
- imageloadfont()

imagecolorallocate

imagecolorallocate -- Allocate a color for an image

Description

int **imagecolorallocate** (resource `$image`, int `$red`, int `$green`, int `$blue`)

Returns a color identifier representing the color composed of the given RGB components.

imagecolorallocate() must be called to create each color that is to be used in the image represented by *`image`*.

Note
The first call to imagecolorallocate() fills the background color in palette-based images - images created using imagecreate().

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
Value of red component

`green`
Value of green component

`blue`
Value of blue component

These parameters are integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

Return Values

A color identifier or **FALSE** if the allocation failed.

ChangeLog

Version	Description

Prior to 5.1.3	Returns -1 if the allocation failed.

Examples

Example #12 - imagecolorallocate() example

```
<?php

$im = imagecreate(100, 100);

// sets background to red
$background = imagecolorallocate($im, 255, 0, 0);

// sets some colors
$white = imagecolorallocate($im, 255, 255, 255);
$black = imagecolorallocate($im, 0, 0, 0);

// hexadecimal way
$white = imagecolorallocate($im, 0xFF, 0xFF, 0xFF);
$black = imagecolorallocate($im, 0x00, 0x00, 0x00);

?>
```

See Also

- imagecolorallocatealpha()
- imagecolordeallocate()

imagecolorallocatealpha

imagecolorallocatealpha -- Allocate a color for an image

Description

int **imagecolorallocatealpha** (resource `$image`, int `$red`, int `$green`, int `$blue`, int `$alpha`)

imagecolorallocatealpha() behaves identically to imagecolorallocate() with the addition of the transparency parameter `alpha`.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
: Value of red component

`green`
: Value of green component

`blue`
: Value of blue component

`alpha`
: A value between *0* and *127*. *0* indicates completely opaque while *127* indicates completely transparent.

The colors parameters are integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

Return Values

A color identifier or **FALSE** if the allocation failed.

ChangeLog

Version	Description
Prior to 5.1.3	Returns -1 if the allocation failed.

Examples

Example #13 - Example of using imagecolorallocatealpha()

```
<?php
$size = 300;
$image=imagecreatetruecolor($size, $size);

// something to get a white background with black border
$back = imagecolorallocate($image, 255, 255, 255);
$border = imagecolorallocate($image, 0, 0, 0);
imagefilledrectangle($image, 0, 0, $size - 1, $size - 1, $back);
imagerectangle($image, 0, 0, $size - 1, $size - 1, $border);

$yellow_x = 100;
$yellow_y = 75;
$red_x    = 120;
$red_y    = 165;
$blue_x   = 187;
$blue_y   = 125;
$radius   = 150;

// allocate colors with alpha values
$yellow = imagecolorallocatealpha($image, 255, 255, 0, 75);
$red    = imagecolorallocatealpha($image, 255, 0, 0, 75);
$blue   = imagecolorallocatealpha($image, 0, 0, 255, 75);

// drawing 3 overlapped circle
imagefilledellipse($image, $yellow_x, $yellow_y, $radius, $radius, $yellow);
imagefilledellipse($image, $red_x, $red_y, $radius, $radius, $red);
imagefilledellipse($image, $blue_x, $blue_y, $radius, $radius, $blue);

// don't forget to output a correct header!
header('Content-type: image/png');

// and finally, output the result
imagepng($image);
imagedestroy($image);
?>
```

The above example will output something similar to:

Notes

Note

This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

See Also

- imagecolorallocate()
- imagecolordeallocate()

imagecolorat

imagecolorat -- Get the index of the color of a pixel

Description

int **imagecolorat** (resource `$image`, int `$x`, int `$y`)

Returns the index of the color of the pixel at the specified location in the image specified by *`image`*.

If PHP is compiled against GD library 2.0 or higher and the image is a truecolor image, this function returns the RGB value of that pixel as integer. Use bitshifting and masking to access the distinct red, green and blue component values:

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`x`
x-coordinate of the point

`y`
y-coordinate of the point

Return Values

Returns the index of the color.

Examples

Example #14 - Access distinct RGB values

```
<?php
$im = imagecreatefrompng("php.png");
$rgb = imagecolorat($im, 10, 15);
$r = ($rgb >> 16) & 0xFF;
$g = ($rgb >> 8) & 0xFF;
$b = $rgb & 0xFF;
?>
```

The above example will output something similar to:

```
int(119)
```

```
int(123)
int(180)
```

See Also

- imagecolorset()
- imagecolorsforindex()

imagecolorclosest

imagecolorclosest -- Get the index of the closest color to the specified color

Description

int **imagecolorclosest** (resource `$image`, int `$red`, int `$green`, int `$blue`)

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value.

The "distance" between the desired color and each color in the palette is calculated as if the RGB values represented points in three-dimensional space.

If you created the image from a file, only colors used in the image are resolved. Colors present only in the pallete are not resolved.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
: Value of red component

`green`
: Value of green component

`blue`
: Value of blue component

The colors parameters are integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

Return Values

Returns the index of the closest color, in the palette of the image, to the specified one

See Also

- imagecolorexact()

imagecolorclosestalpha

imagecolorclosestalpha -- Get the index of the closest color to the specified color + alpha

Description

int **imagecolorclosestalpha** (resource `$image`, int `$red`, int `$green`, int `$blue`, int `$alpha`)

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value and `alpha` level.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
Value of red component

`green`
Value of green component

`blue`
Value of blue component

`alpha`
A value between *0* and *127*. *0* indicates completely opaque while *127* indicates completely transparent.

The colors parameters are integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

Return Values

Returns the index of the closest color in the palette.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

See Also

- imagecolorexactalpha()

imagecolorclosesthwb

imagecolorclosesthwb -- Get the index of the color which has the hue, white and blackness

Description

int **imagecolorclosesthwb** (resource `$image`, int `$red`, int `$green`, int `$blue`)

Get the index of the color which has the hue, white and blacknessnearest the given color.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
Value of red component

`green`
Value of green component

`blue`
Value of blue component

Return Values

Returns an integer with the index of the color which has the hue, white and blackness nearest the given color.

Examples

Example #15 - Example of using imagecolorclosesthwb()

```
<?php
$im = imagecreatefromgif('php.gif');

echo 'HWB: ' . imagecolorclosesthwb($im, 116, 115, 152);

imagedestroy($im);
?>
```

The above example will output something similar to:

```
HWB: 33
```

Notes

Note
This function is not implemented on Windows platforms.

imagecolordeallocate

imagecolordeallocate -- De-allocate a color for an image

Description

bool **imagecolordeallocate** (resource `$image`, int `$color`)

De-allocates a color previously allocated with imagecolorallocate() or imagecolorallocatealpha().

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`color`
The color identifier

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #16 - Using imagecolordeallocate()

```
<?php
$white = imagecolorallocate($im, 255, 255, 255);
imagecolordeallocate($im, $white);
?>
```

See Also

- imagecolorallocate()
- imagecolorallocatealpha()

imagecolorexact

imagecolorexact -- Get the index of the specified color

Description

int **imagecolorexact** (resource `$image`, int `$red`, int `$green`, int `$blue`)

Returns the index of the specified color in the palette of the image.

If you created the image from a file, only colors used in the image are resolved. Colors present only in the pallete are not resolved.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
Value of red component

`green`
Value of green component

`blue`
Value of blue component

Return Values

Returns the index of the specified color in the palette, or -1 if the color does not exist.

See Also

- imagecolorclosest()

imagecolorexactalpha

imagecolorexactalpha -- Get the index of the specified color + alpha

Description

int **imagecolorexactalpha** (resource `$image`, int `$red`, int `$green`, int `$blue`, int `$alpha`)

Returns the index of the specified color+alpha in the palette of the image.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
Value of red component

`green`
Value of green component

`blue`
Value of blue component

`alpha`
A value between *0* and *127*. *0* indicates completely opaque while *127* indicates completely transparent.

The colors parameters are integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

Return Values

Returns the index of the specified color+alpha in the palette of the image, or -1 if the color does not exist in the image's palette.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

See Also

- imagecolorclosestalpha()

imagecolormatch

imagecolormatch -- Makes the colors of the palette version of an image more closely match the true color version

Description

bool **imagecolormatch** (resource `$image1`, resource `$image2`)

Warning
This function is currently not documented; only its argument list is available.

Parameters

`image1`
A truecolor image link resource

`image2`
A palette image link resource pointing to an image that has the same size as *`image1`*

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available if PHP is compiled with the bundled version of the GD library.

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

See Also

- imagecreatetruecolor()

imagecolorresolve

imagecolorresolve -- Get the index of the specified color or its closest possible alternative

Description

int **imagecolorresolve** (resource `$image`, int `$red`, int `$green`, int `$blue`)

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

If you created the image from a file, only colors used in the image are resolved. Colors present only in the pallete are not resolved.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
: Value of red component

`green`
: Value of green component

`blue`
: Value of blue component

Return Values

Returns a color index.

See Also

- imagecolorclosest()

imagecolorresolvealpha

imagecolorresolvealpha -- Get the index of the specified color + alpha or its closest possible alternative

Description

int **imagecolorresolvealpha** (resource `$image`, int `$red`, int `$green`, int `$blue`, int `$alpha`)

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`red`
Value of red component

`green`
Value of green component

`blue`
Value of blue component

`alpha`
A value between *0* and *127*. *0* indicates completely opaque while *127* indicates completely transparent.

The colors parameters are integers between 0 and 255 or hexadecimals between 0x00 and 0xFF.

Return Values

Returns a color index.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

See Also

- imagecolorclosestalpha()

imagecolorset

imagecolorset -- Set the color for the specified palette index

Description

void **imagecolorset** (resource `$image`, int `$index`, int `$red`, int `$green`, int `$blue`)

This sets the specified index in the palette to the specified color. This is useful for creating flood-fill-like effects in palleted images without the overhead of performing the actual flood-fill.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`index`
: An index in the palette

`red`
: Value of red component

`green`
: Value of green component

`blue`
: Value of blue component

Return Values

No value is returned.

See Also

- imagecolorat()

imagecolorsforindex

imagecolorsforindex -- Get the colors for an index

Description

array **imagecolorsforindex** (resource `$image`, int `$index`)

Gets the color for a specified index.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`index`

Return Values

Returns an associative array with red, green, blue and alpha keys that contain the appropriate values for the specified color index.

Examples

Example #17 - imagecolorsforindex() example

```
<?php

// open an image
$im = imagecreatefrompng('nexen.png');

// get a color
$start_x = 40;
$start_y = 50;
$color_index = imagecolorat($im, $start_x, $start_y);

// make it human readable
$color_tran = imagecolorsforindex($im, $color_index);

// what is it ?
print_r($color_tran);

?>
```

The above example will output something similar to:

```
Array
(
  [red] => 226
  [green] => 222
  [blue] => 252
  [alpha] => 0
)
```

See Also

- imagecolorat()
- imagecolorexact()

imagecolorstotal

imagecolorstotal -- Find out the number of colors in an image's palette

Description

int **imagecolorstotal** (resource `$image`)

Returns the number of colors in an image palette.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

Return Values

Returns the number of colors in the specified image's palette or 0 for truecolor images.

See Also

- imagecolorat()
- imagecolorsforindex()
- imageistruecolor()

imagecolortransparent

imagecolortransparent -- Define a color as transparent

Description

int **imagecolortransparent** (resource `$image` [, int `$color`])

Sets the transparent color in the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`color`
A color identifier created with imagecolorallocate()

Return Values

The identifier of the new (or current, if none is specified) transparent color is returned.

Notes

Note
Transparency is copied only with imagecopymerge() and true color images, not with imagecopy() or pallete images.

Note
The transparent color is a property of the image, transparency is not a property of the color. Once you have set a color to be the transparent color, any regions of the image in that color that were drawn previously will be transparent.

imageconvolution

imageconvolution -- Apply a 3x3 convolution matrix, using coefficient and offset

Description

bool **imageconvolution** (resource `$image`, array `$matrix`, float `$div`, float `$offset`)

Applies a convolution matrix on the image, using the given coefficient and offset.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`matrix`
A 3x3 matrix: an array of three arrays of three floats.

`div`
The divisor of the result of the convolution, used for normalization.

`offset`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #18 - Embossing the PHP.net logo

```
<?php
$image = imagecreatefromgif('http://www.php.net/images/php.gif');

$emboss = array(array(2, 0, 0), array(0, -1, 0), array(0, 0, -1));
imageconvolution($image, $emboss, 1, 127);

header('Content-Type: image/png');
imagepng($image, null, 9);
?>
```

The above example will output:

Example #19 - Gaussian blur

```
<?php
$image = imagecreatetruecolor(180,40);

// Writes the text and apply a gaussian blur on the image
imagestring($image, 5, 10, 8, 'Gaussian Blur Text', 0x00ff00);
$gaussian = array(array(1.0, 2.0, 1.0), array(2.0, 4.0, 2.0), array(1.0,
2.0, 1.0));
imageconvolution($image, $gaussian, 16, 0);

// Rewrites the text for comparison
imagestring($image, 5, 10, 18, 'Gaussian Blur Text', 0x00ff00);

header('Content-Type: image/png');
imagepng($image, null, 9);
?>
```

The above example will output:

Notes

Note

This function is only available if PHP is compiled with the bundled version of the GD library.

See Also

- imagefilter()

imagecopy

imagecopy -- Copy part of an image

Description

bool **imagecopy** (resource `$dst_im`, resource `$src_im`, int `$dst_x`, int `$dst_y`, int `$src_x`, int `$src_y`, int `$src_w`, int `$src_h`)

Copy a part of *`src_im`* onto *`dst_im`* starting at the x,y coordinates *`src_x`*, *`src_y`* with a width of *`src_w`* and a height of *`src_h`*. The portion defined will be copied onto the x,y coordinates, *`dst_x`* and *`dst_y`*.

Parameters

`dst_im`
: Destination image link resource

`src_im`
: Source image link resource

`dst_x`
: x-coordinate of destination point

`dst_y`
: y-coordinate of destination point

`src_x`
: x-coordinate of source point

`src_y`
: y-coordinate of source point

`src_w`
: Source width

`src_h`
: Source height

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imagecopymerge

imagecopymerge -- Copy and merge part of an image

Description

bool **imagecopymerge** (resource `$dst_im`, resource `$src_im`, int `$dst_x`, int `$dst_y`, int `$src_x`, int `$src_y`, int `$src_w`, int `$src_h`, int `$pct`)

Copy a part of *`src_im`* onto *`dst_im`* starting at the x,y coordinates *`src_x`*, *`src_y`* with a width of *`src_w`* and a height of *`src_h`*. The portion defined will be copied onto the x,y coordinates, *`dst_x`* and *`dst_y`*.

Parameters

`dst_im`
: Destination image link resource

`src_im`
: Source image link resource

`dst_x`
: x-coordinate of destination point

`dst_y`
: y-coordinate of destination point

`src_x`
: x-coordinate of source point

`src_y`
: y-coordinate of source point

`src_w`
: Source width

`src_h`
: Source height

`pct`
: The two images will be merged according to *`pct`* which can range from 0 to 100. When *`pct`* = 0, no action is taken, when 100 this function behaves identically to imagecopy() for pallete images, while it implements alpha transparency for true colour images.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imagecopymergegray

imagecopymergegray -- Copy and merge part of an image with gray scale

Description

bool **imagecopymergegray** (resource `$dst_im`, resource `$src_im`, int `$dst_x`, int `$dst_y` , int `$src_x`, int `$src_y`, int `$src_w`, int `$src_h`, int `$pct`)

imagecopymergegray() copy a part of *`src_im`* onto *`dst_im`* starting at the x,y coordinates *`src_x`*, *`src_y`* with a width of *`src_w`* and a height of *`src_h`*. The portion defined will be copied onto the x,y coordinates, *`dst_x`* and *`dst_y`*.

This function is identical to imagecopymerge() except that when merging it preserves the hue of the source by converting the destination pixels to gray scale before the copy operation.

Parameters

`dst_im`
Destination image link resource

`src_im`
Source image link resource

`dst_x`
x-coordinate of destination point

`dst_y`
y-coordinate of destination point

`src_x`
x-coordinate of source point

`src_y`
y-coordinate of source point

`src_w`
Source width

`src_h`
Source height

`pct`
The two images will be merged according to *`pct`* which can range from 0 to 100. When *`pct`* = 0, no action is taken, when 100 this function behaves identically to imagecopy() for pallete images, while it implements alpha transparency for true colour images.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imagecopyresampled

imagecopyresampled -- Copy and resize part of an image with resampling

Description

bool **imagecopyresampled** (resource `$dst_image`, resource `$src_image`, int `$dst_x`, int `$dst_y`, int `$src_x`, int `$src_y`, int `$dst_w`, int `$dst_h`, int `$src_w`, int `$src_h`)

imagecopyresampled() copies a rectangular portion of one image to another image, smoothly interpolating pixel values so that, in particular, reducing the size of an image still retains a great deal of clarity.

In other words, imagecopyresampled() will take an rectangular area from *`src_image`* of width *`src_w`* and height *`src_h`* at position (*`src_x`*, *`src_y`*) and place it in a rectangular area of *`dst_image`* of width *`dst_w`* and height *`dst_h`* at position (*`dst_x`*, *`dst_y`*).

If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if *`dst_image`* is the same as *`src_image`*) but if the regions overlap the results will be unpredictable.

Parameters

`dst_im`
: Destination image link resource

`src_im`
: Source image link resource

`dst_x`
: x-coordinate of destination point

`dst_y`
: y-coordinate of destination point

`src_x`
: x-coordinate of source point

`src_y`
: y-coordinate of source point

`dst_w`
: Destination width

`dst_h`
: Destination height

src_w
Source width

src_h
Source height

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #20 - Simple example

This example will resample an image to half its original size.

```
<?php
// The file
$filename = 'test.jpg';
$percent = 0.5;

// Content type
header('Content-type: image/jpeg');

// Get new dimensions
list($width, $height) = getimagesize($filename);
$new_width = $width * $percent;
$new_height = $height * $percent;

// Resample
$image_p = imagecreatetruecolor($new_width, $new_height);
$image = imagecreatefromjpeg($filename);
imagecopyresampled($image_p, $image, 0, 0, 0, 0, $new_width, $new_height,
$width, $height);

// Output
imagejpeg($image_p, null, 100);
?>
```

The above example will output something similar to:

Example #21 - Resampling an image proportionally

This example will display an image with the maximum width, or height, of 200 pixels.

```
<?php
```

```
// The file
$filename = 'test.jpg';

// Set a maximum height and width
$width = 200;
$height = 200;

// Content type
header('Content-type: image/jpeg');

// Get new dimensions
list($width_orig, $height_orig) = getimagesize($filename);

$ratio_orig = $width_orig/$height_orig;

if ($width/$height > $ratio_orig) {
  $width = $height*$ratio_orig;
} else {
  $height = $width/$ratio_orig;
}

// Resample
$image_p = imagecreatetruecolor($width, $height);
$image = imagecreatefromjpeg($filename);
imagecopyresampled($image_p, $image, 0, 0, 0, 0, $width, $height,
$width_orig, $height_orig);

// Output
imagejpeg($image_p, null, 100);
?>
```

The above example will output something similar to:

Notes

Note
There is a problem due to palette image limitations (255+1 colors). Resampling or filtering an image commonly needs more colors than 255, a kind of approximation is used to calculate the new resampled pixel and its color. With a palette image we try to allocate a new color, if that failed, we choose the closest (in theory) computed color. This is not always the closest visual color. That may produce a weird result, like blank (or visually blank) images. To skip this problem, please use a truecolor image as a destination image, such as one created by imagecreatetruecolor().

See Also

imagecopyresized()

imagecopyresized

imagecopyresized -- Copy and resize part of an image

Description

bool **imagecopyresized** (resource `$dst_image`, resource `$src_image`, int `$dst_x`, int `$dst_y`, int `$src_x`, int `$src_y`, int `$dst_w`, int `$dst_h`, int `$src_w`, int `$src_h`)

imagecopyresized() copies a rectangular portion of one image to another image. *`dst_image`* is the destination image, *`src_image`* is the source image identifier.

In other words, imagecopyresized() will take an rectangular area from *`src_image`* of width *`src_w`* and height *`src_h`* at position (*`src_x`*, *`src_y`*) and place it in a rectangular area of *`dst_image`* of width *`dst_w`* and height *`dst_h`* at position (*`dst_x`*, *`dst_y`*).

If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if *`dst_image`* is the same as *`src_image`*) but if the regions overlap the results will be unpredictable.

Parameters

`dst_im`
: Destination image link resource

`src_im`
: Source image link resource

`dst_x`
: x-coordinate of destination point

`dst_y`
: y-coordinate of destination point

`src_x`
: x-coordinate of source point

`src_y`
: y-coordinate of source point

`dst_w`
: Destination width

`dst_h`
: Destination height

`src_w`

Source width

src_h
Source height

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #22 - Resizing an image

This example will display the image at half size.

```
<?php
// File and new size
$filename = 'test.jpg';
$percent = 0.5;

// Content type
header('Content-type: image/jpeg');

// Get new sizes
list($width, $height) = getimagesize($filename);
$newwidth = $width * $percent;
$newheight = $height * $percent;

// Load
$thumb = imagecreatetruecolor($newwidth, $newheight);
$source = imagecreatefromjpeg($filename);

// Resize
imagecopyresized($thumb, $source, 0, 0, 0, 0, $newwidth, $newheight, $width,
$height);

// Output
imagejpeg($thumb);
?>
```

The above example will output something similar to:

The image will be output at half size, though better quality could be obtained using imagecopyresampled().

Notes

Note

There is a problem due to palette image limitations (255+1 colors). Resampling or filtering an image commonly needs more colors than 255, a kind of approximation is used to calculate the new resampled pixel and its color. With a palette image we try to allocate a new color, if that failed, we choose the closest (in theory) computed color. This is not always the closest visual color. That may produce a weird result, like blank (or visually blank) images. To skip this problem, please use a truecolor image as a destination image, such as one created by imagecreatetruecolor().

See Also

imagecopyresampled()

imagecreate

imagecreate -- Create a new palette based image

Description

resource **imagecreate** (int `$width`, int `$height`)

imagecreate() returns an image identifier representing a blank image of specified size.

We recommend the use of imagecreatetruecolor().

Parameters

`width`
The image width

`height`
The image height

Return Values

Returns an image resource identifier on success, **FALSE** on errors.

Examples

Example #23 - Creating a new GD image stream and outputting an image.

```
<?php
header("Content-type: image/png");
$im = @imagecreate(110, 20)
   or die("Cannot Initialize new GD image stream");
$background_color = imagecolorallocate($im, 0, 0, 0);
$text_color = imagecolorallocate($im, 233, 14, 91);
imagestring($im, 1, 5, 5,  "A Simple Text String", $text_color);
imagepng($im);
imagedestroy($im);
?>
```

The above example will output something similar to:

See Also

- imagedestroy()
- imagecreatetruecolor()

imagecreatefromgd2

imagecreatefromgd2 -- Create a new image from GD2 file or URL

Description

resource **imagecreatefromgd2** (string `$filename`)

Warning
This function is currently not documented; only its argument list is available.

Tip
A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Parameters

`filename`

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromgd2part

imagecreatefromgd2part -- Create a new image from a given part of GD2 file or URL

Description

resource **imagecreatefromgd2part** (string `$filename`, int `$srcX`, int `$srcY`, int `$width`, int `$height`)

Warning
This function is currently not documented; only its argument list is available.

Tip
A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Parameters

`filename`

`srcX`

`srcY`

`width`

`height`

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromgd

imagecreatefromgd -- Create a new image from GD file or URL

Description

resource **imagecreatefromgd** (string `$filename`)

Warning
This function is currently not documented; only its argument list is available.

Tip
A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Parameters

`filename`

Notes

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromgif

imagecreatefromgif -- Create a new image from file or URL

Description

resource **imagecreatefromgif** (string `$filename`)

imagecreatefromgif() returns an image identifier representing the image obtained from the given filename.

To ease debugging the following example will produce an error GIF:

Example #24 - Example to handle an error during creation

```
<?php
function LoadGif ($imgname)
{
   $im = @imagecreatefromgif ($imgname); /* Attempt to open */
   if (!$im) { /* See if it failed */
       $im = imagecreatetruecolor (150, 30); /* Create a blank image */
       $bgc = imagecolorallocate ($im, 255, 255, 255);
       $tc = imagecolorallocate ($im, 0, 0, 0);
       imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
       /* Output an errmsg */
       imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
   }
   return $im;
}
header("Content-Type: image/gif");
$img = LoadGif("bogus.image");
imagegif($img);
?>
```

The above example will output something similar to:

Tip

A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Parameters

`filename`

Path to the GIF image

Return Values

Returns an image resource identifier on success, **FALSE** on errors.

Notes

Note
GIF support was removed from the GD library in Version 1.6, and added back in Version 2.0.28. This function is not available between these versions.

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromjpeg

imagecreatefromjpeg -- Create a new image from file or URL

Description

resource **imagecreatefromjpeg** (string `$filename`)

imagecreatefromjpeg() returns an image identifier representing the image obtained from the given filename.

On failure imagecreatefromjpeg() outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error JPEG:

Example #25 - Example to handle an error during creation

```
<?php
function LoadJpeg($imgname)
{
   $im = @imagecreatefromjpeg($imgname); /* Attempt to open */
   if (!$im) { /* See if it failed */
       $im  = imagecreatetruecolor(150, 30); /* Create a black image */
       $bgc = imagecolorallocate($im, 255, 255, 255);
       $tc  = imagecolorallocate($im, 0, 0, 0);
       imagefilledrectangle($im, 0, 0, 150, 30, $bgc);
       /* Output an errmsg */
       imagestring($im, 1, 5, 5, "Error loading $imgname", $tc);
   }
   return $im;
}
header("Content-Type: image/jpeg");
$img = LoadJpeg("bogus.image");
imagejpeg($img);
?>
```

The above example will output something similar to:

Tip

A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Parameters

filename
Path to the JPEG image

Return Values

Returns an image resource identifier on success, **FALSE** on errors.

Notes

Note
JPEG support is only available if PHP was compiled against GD-1.8 or later.

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

imagecreatefrompng

imagecreatefrompng -- Create a new image from file or URL

Description

resource **imagecreatefrompng** (string `$filename`)

imagecreatefrompng() returns an image identifier representing the image obtained from the given filename.

imagecreatefrompng() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error PNG:

Example #26 - Example to handle an error during creation

```
<?php
function LoadPNG($imgname)
{
   $im = @imagecreatefrompng($imgname); /* Attempt to open */
   if (!$im) { /* See if it failed */
       $im  = imagecreatetruecolor(150, 30); /* Create a blank image */
       $bgc = imagecolorallocate($im, 255, 255, 255);
       $tc  = imagecolorallocate($im, 0, 0, 0);
       imagefilledrectangle($im, 0, 0, 150, 30, $bgc);
       /* Output an errmsg */
       imagestring($im, 1, 5, 5, "Error loading $imgname", $tc);
   }
   return $im;
}
header("Content-Type: image/png");
$img = LoadPNG("bogus.image");
imagepng($img);
?>
```

The above example will output something similar to:

Tip

A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Parameters

filename
Path to the PNG image

Return Values

Returns an image resource identifier on success, **FALSE** on errors.

Notes

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromstring

imagecreatefromstring -- Create a new image from the image stream in the string

Description

resource **imagecreatefromstring** (string `$data`)

imagecreatefromstring() returns an image identifier representing the image obtained from the given *`data`* These types will be automatically detected if your build of PHP supports them: JPEG, PNG, GIF, WBMP, and GD2.

Parameters

`image`
A string containing the image data

Return Values

An image resource will be returned on success. **FALSE** is returned if the image type is unsupported, the data is not in a recognised format, or the image is corrupt and cannot be loaded.

Examples

Example #27 - imagecreatefromstring() example

```
<?php
$data = 'iVBORw0KGgoAAAANSUhEUgAAABwAAAASCAMAAAB/2U7WAAAABl'
      . 'BMVEUAAAD///+l2Z/dAAAASUlEQVR4XqWQUQoAIAxC2/0vXZDr'
      . 'EX4IJTRkb7lobNUStXsB0jIXIAMSsQnWlsV+wULF4Avk9fLq2r'
      . '8a5HSE35Q3eO2XP1A1wQkZSgETvDtKdQAAAABJRU5ErkJggg==';
$data = base64_decode($data);

$im = imagecreatefromstring($data);
if ($im !== false) {
   header('Content-Type: image/png');
   imagepng($im);
}
else {
   echo 'An error occurred.';
}
?>
```

The above example will output something similar to:

See Also

- imagecreatefromjpeg()
- imagecreatefrompng()
- imagecreatefromgif()
- imagecreatetruecolor()

imagecreatefromwbmp

imagecreatefromwbmp -- Create a new image from file or URL

Description

resource **imagecreatefromwbmp** (string `$filename`)

imagecreatefromwbmp() returns an image identifier representing the image obtained from the given filename.

imagecreatefromwbmp() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error WBMP:

Example #28 - Example to handle an error during creation

```
<?php
function LoadWBMP($imgname)
{
   $im = @imagecreatefromwbmp($imgname); /* Attempt to open */
   if (!$im) { /* See if it failed */
       $im  = imagecreatetruecolor (20, 20); /* Create a blank image */
       $bgc = imagecolorallocate($im, 255, 255, 255);
       $tc  = imagecolorallocate($im, 0, 0, 0);
       imagefilledrectangle($im, 0, 0, 10, 10, $bgc);
       /* Output an errmsg */
       imagestring($im, 1, 5, 5, "Error loading $imgname", $tc);
   }
   return $im;
}
?>
```

Tip

A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Parameters

`filename`
Path to the WBMP image

Return Values

Returns an image resource identifier on success, **FALSE** on errors.

Notes

Note
WBMP support is only available if PHP was compiled against GD-1.8 or later.

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromxbm

imagecreatefromxbm -- Create a new image from file or URL

Description

resource **imagecreatefromxbm** (string `$filename`)

imagecreatefromxbm() returns an image identifier representing the image obtained from the given filename.

Tip
A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

Parameters

`filename`
Path to the XBM image

Return Values

Returns an image resource identifier on success, **FALSE** on errors.

imagecreatefromxpm

imagecreatefromxpm -- Create a new image from file or URL

Description

resource **imagecreatefromxpm** (string `$filename`)

imagecreatefromxpm() returns an image identifier representing the image obtained from the given filename.

Tip
A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

Parameters

`filename`
Path to the XPM image

Return Values

Returns an image resource identifier on success, **FALSE** on errors.

Return Values

Note
This function is only available if PHP is compiled with the bundled version of the GD library.

Warning
Windows versions of PHP prior to PHP 4.3.0 do not support access of remote files via this function, even if allow_url_fopen is enabled.

imagecreatetruecolor

imagecreatetruecolor -- Create a new true color image

Description

resource **imagecreatetruecolor** (int `$width`, int `$height`)

imagecreatetruecolor() returns an image identifier representing a black image of the specified size.

Depending on your PHP and GD versions this function is defined or not. With PHP 4.0.6 through 4.1.x this function always exists if the GD module is loaded, but calling it without GD2 being installed PHP will issue a fatal error and exit. With PHP 4.2.x this behaviour is different in issuing a warning instead of an error. Other versions only define this function, if the correct GD version is installed.

Parameters

width
: Image width

height
: Image height

Return Values

Returns an image resource identifier on success, **FALSE** on errors.

Examples

Example #29 - Creating a new GD image stream and outputting an image.

```
<?php
header ("Content-type: image/png");
$im = @imagecreatetruecolor(120, 20)
     or die("Cannot Initialize new GD image stream");
$text_color = imagecolorallocate($im, 233, 14, 91);
imagestring($im, 1, 5, 5,  "A Simple Text String", $text_color);
imagepng($im);
imagedestroy($im);
?>
```

The above example will output something similar to:

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

Note
This function will not work with GIF file formats.

See Also

- imagedestroy()
- imagecreate()

imagedashedline

imagedashedline -- Draw a dashed line

Description

bool **imagedashedline** (resource `$image`, int `$x1`, int `$y1`, int `$x2`, int `$y2`, int `$color`)

This function is deprecated. Use combination of imagesetstyle() and imageline() instead.

imagedestroy

imagedestroy -- Destroy an image

Description

bool **imagedestroy** (resource `$image`)

imagedestroy() frees any memory associated with image *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imageellipse

imageellipse -- Draw an ellipse

Description

bool **imageellipse** (resource `$image`, int `$cx`, int `$cy`, int `$width`, int `$height`, int `$color`)

Draws an ellipse centered at the specified coordinates.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`cx`
x-coordinate of the center

`cy`
y-coordinate of the center

`width`
The ellipse width

`height`
The ellipse height

`color`
The color of the ellipse. A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #30 - imageellipse() example

```
<?php

// create a blank image
$image = imagecreatetruecolor(400, 300);

// fill the background color
```

```
$bg = imagecolorallocate($image, 0, 0, 0);

// choose a color for the ellipse
$col_ellipse = imagecolorallocate($image, 255, 255, 255);

// draw the ellipse
imageellipse($image, 200, 150, 300, 200, $col_ellipse);

// output the picture
header("Content-type: image/png");
imagepng($image);

?>
```

The above example will output something similar to:

Notes

Note
This function requires GD 2.0.2 or later.

See Also

- imagefilledellipse()
- imagearc()

imagefill

imagefill -- Flood fill

Description

bool **imagefill** (resource `$image`, int `$x`, int `$y`, int `$color`)

Performs a flood fill starting at the given coordinate (top left is 0, 0) with the given *`color`* in the *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`x`
x-coordinate of start point

`y`
y-coordinate of start point

`color`
The fill color. A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #31 - imagefill() example

```
<?php

$im = imagecreatetruecolor(100, 100);

// sets background to red
$red = imagecolorallocate($im, 255, 0, 0);
imagefill($im, 0, 0, $red);

header('Content-type: image/png');
imagepng($im);
imagedestroy($im);
?>
```

The above example will output something similar to:

See Also

- imagecolorallocate()

imagefilledarc

imagefilledarc -- Draw a partial ellipse and fill it

Description

bool **imagefilledarc** (resource `$image`, int `$cx`, int `$cy`, int `$width`, int `$height`, int `$start`, int `$end`, int `$color`, int `$style`)

Draws a partial ellipse centered at the specified coordinate in the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`cx`
x-coordinate of the center

`cy`
y-coordinate of the center

`width`
The arc width

`height`
The arc height

`start`
The arc start angle, in degrees.

`end`
The arc end angle, in degrees. 0° is located at the three-o'clock position, and the arc is drawn clockwise.

`color`
A color identifier created with imagecolorallocate()

`style`
A bitwise OR of the following possibilities:

- **IMG_ARC_PIE**
- **IMG_ARC_CHORD**
- **IMG_ARC_NOFILL**
- **IMG_ARC_EDGED**

IMG_ARC_PIE and **IMG_ARC_CHORD** are mutually exclusive; **IMG_ARC_CHORD** just connects the starting and ending angles with a straight line, while **IMG_ARC_PIE** produces a rounded edge. **IMG_ARC_NOFILL** indicates that the arc or chord should be outlined, not filled. **IMG_ARC_EDGED**, used together with **IMG_ARC_NOFILL**, indicates that the beginning and ending angles should be connected to the center - this is a good way to outline (rather than fill) a 'pie slice'.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #32 - Creating a 3D looking pie

```
<?php

// create image
$image = imagecreatetruecolor(100, 100);

// allocate some solors
$white    = imagecolorallocate($image, 0xFF, 0xFF, 0xFF);
$gray     = imagecolorallocate($image, 0xC0, 0xC0, 0xC0);
$darkgray = imagecolorallocate($image, 0x90, 0x90, 0x90);
$navy     = imagecolorallocate($image, 0x00, 0x00, 0x80);
$darknavy = imagecolorallocate($image, 0x00, 0x00, 0x50);
$red      = imagecolorallocate($image, 0xFF, 0x00, 0x00);
$darkred  = imagecolorallocate($image, 0x90, 0x00, 0x00);

// make the 3D effect
for ($i = 60; $i > 50; $i--) {
  imagefilledarc($image, 50, $i, 100, 50, 0, 45, $darknavy, IMG_ARC_PIE);
  imagefilledarc($image, 50, $i, 100, 50, 45, 75 , $darkgray, IMG_ARC_PIE);
  imagefilledarc($image, 50, $i, 100, 50, 75, 360 , $darkred, IMG_ARC_PIE);
}

imagefilledarc($image, 50, 50, 100, 50, 0, 45, $navy, IMG_ARC_PIE);
imagefilledarc($image, 50, 50, 100, 50, 45, 75 , $gray, IMG_ARC_PIE);
imagefilledarc($image, 50, 50, 100, 50, 75, 360 , $red, IMG_ARC_PIE);


// flush image
header('Content-type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

The above example will output something similar to:

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

imagefilledellipse

imagefilledellipse -- Draw a filled ellipse

Description

bool **imagefilledellipse** (resource `$image`, int `$cx`, int `$cy`, int `$width`, int `$height`, int `$color`)

Draws an ellipse centered at the specified coordinate on the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`cx`
x-coordinate of the center

`cy`
y-coordinate of the center

`width`
The ellipse width

`height`
The ellipse height

`color`
The fill color. A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #33 - imagefilledellipse() example

```
<?php

// create a blank image
$image = imagecreatetruecolor(400, 300);
```

```
// fill the background color
$bg = imagecolorallocate($image, 0, 0, 0);

// choose a color for the ellipse
$col_ellipse = imagecolorallocate($image, 255, 255, 255);

// draw the white ellipse
imagefilledellipse($image, 200, 150, 300, 200, $col_ellipse);

// output the picture
header("Content-type: image/png");
imagepng($image);

?>
```

The above example will output something similar to:

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

See Also

- imageellipse()
- imagefilledarc()

imagefilledpolygon

imagefilledpolygon -- Draw a filled polygon

Description

bool **imagefilledpolygon** (resource `$image`, array `$points`, int `$num_points`, int `$color`)

imagefilledpolygon() creates a filled polygon in the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`points`
An array containing the *x* and *y* coordinates of the polygons vertices consecutively

`num_points`
Total number of vertices, which must be bigger than 3

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #34 - imagefilledpolygon() example

```
<?php
// set up array of points for polygon
$values = array(
            40,  50,  // Point 1 (x, y)
            20,  240, // Point 2 (x, y)
            60,  60,  // Point 3 (x, y)
            240, 20,  // Point 4 (x, y)
            50,  40,  // Point 5 (x, y)
            10,  10   // Point 6 (x, y)
            );

// create image
$image = imagecreatetruecolor(250, 250);
```

```
// some colors
$bg   = imagecolorallocate($image, 200, 200, 200);
$blue = imagecolorallocate($image, 0, 0, 255);

// draw a polygon
imagefilledpolygon($image, $values, 6, $blue);

// flush image
header('Content-type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

The above example will output something similar to:

imagefilledrectangle

imagefilledrectangle -- Draw a filled rectangle

Description

bool **imagefilledrectangle** (resource `$image`, int `$x1`, int `$y1`, int `$x2`, int `$y2`, int `$color`)

Creates a rectangle filled with *`color`* in the given *`image`* starting at point 1 and ending at point 2. 0, 0 is the top left corner of the image.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`x1`
x-coordinate for point 1

`y1`
y-coordinate for point 1

`x2`
x-coordinate for point 2

`y2`
y-coordinate for point 2

`color`
The fill color. A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imagefilltoborder

imagefilltoborder -- Flood fill to specific color

Description

bool **imagefilltoborder** (resource `$image`, int `$x`, int `$y`, int `$border`, int `$color`)

imagefilltoborder() performs a flood fill whose border color is defined by *`border`*. The starting point for the fill is *`x`*, *`y`* (top left is 0, 0) and the region is filled with color *`color`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`x`
x-coordinate of start

`y`
y-coordinate of start

`border`
The border color. A color identifier created with imagecolorallocate()

`color`
The fill color. A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imagefilter

imagefilter -- Applies a filter to an image

Description

bool **imagefilter** (resource `$image`, int `$filtertype` [, int `$arg1` [, int `$arg2` [, int `$arg3` [, int `$arg4`]]]])

imagefilter() applies the given filter *`filtertype`* on the *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filtertype`
`filtertype` can be one of the following:

- **IMG_FILTER_NEGATE**: Reverses all colors of the image.
- **IMG_FILTER_GRAYSCALE**: Converts the image into grayscale.
- **IMG_FILTER_BRIGHTNESS**: Changes the brightness of the image. Use *`arg1`* to set the level of brightness.
- **IMG_FILTER_CONTRAST**: Changes the contrast of the image. Use *`arg1`* to set the level of contrast.
- **IMG_FILTER_COLORIZE**: Like **IMG_FILTER_GRAYSCALE**, except you can specify the color. Use *`arg1`*, *`arg2`* and *`arg3`* in the form of *`red`*, *`blue`*, *`green`* and *`arg4`* for the *`alpha`* channel. The range for each color is 0 to 255.
- **IMG_FILTER_EDGEDETECT**: Uses edge detection to highlight the edges in the image.
- **IMG_FILTER_EMBOSS**: Embosses the image.
- **IMG_FILTER_GAUSSIAN_BLUR**: Blurs the image using the Gaussian method.
- **IMG_FILTER_SELECTIVE_BLUR**: Blurs the image.
- **IMG_FILTER_MEAN_REMOVAL**: Uses mean removal to achieve a "sketchy" effect.
- **IMG_FILTER_SMOOTH**: Makes the image smoother. Use *`arg1`* to set the level of smoothness.

`arg1`

arg2

arg3

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ChangeLog

Version	Description
5.2.5	Alpha support for **IMG_FILTER_COLORIZE** was added.

Examples

Example #35 - imagefilter() **grayscale example**

```
<?php
$im = imagecreatefrompng('dave.png');
if ($im && imagefilter($im, IMG_FILTER_GRAYSCALE)) {
   echo 'Image converted to grayscale.';
   imagepng($im, 'dave.png');
} else {
   echo 'Conversion to grayscale failed.';
}

imagedestroy($im);
?>
```

Example #36 - imagefilter() **brightness example**

```
<?php
$im = imagecreatefrompng('sean.png');
if ($im && imagefilter($im, IMG_FILTER_BRIGHTNESS, 20)) {
   echo 'Image brightness changed.';
   imagepng($im, 'sean.png');
} else {
   echo 'Image brightness change failed.';
}

imagedestroy($im);
```

```
?>
```

Example #37 - imagefilter() colorize example

```
<?php
$im = imagecreatefrompng('philip.png');

/* R, G, B, so 0, 255, 0 is green */
if ($im && imagefilter($im, IMG_FILTER_COLORIZE, 0, 255, 0)) {
   echo 'Image successfully shaded green.';
   imagepng($im, 'philip.png');
} else {
   echo 'Green shading failed.';
}

imagedestroy($im);
?>
```

Notes

Note

This function is only available if PHP is compiled with the bundled version of the GD library.

See Also

- imageconvolution()

imagefontheight

imagefontheight -- Get font height

Description

int **imagefontheight** (int `$font`)

Returns the pixel height of a character in the specified font.

Parameters

`font`
: Can be 1, 2, 3, 4, 5 for built-in fonts in latin2 encoding (where higher numbers corresponding to larger fonts) or any of your own font identifiers registered with imageloadfont().

Return Values

Returns the height of the pixel.

See Also

- imagefontwidth()
- imageloadfont()

imagefontwidth

imagefontwidth -- Get font width

Description

int **imagefontwidth** (int `$font`)

Returns the pixel width of a character in font.

Parameters

`font`
: Can be 1, 2, 3, 4, 5 for built-in fonts in latin2 encoding (where higher numbers corresponding to larger fonts) or any of your own font identifiers registered with imageloadfont().

Return Values

Returns the width of the pixel

See Also

- imagefontheight()
- imageloadfont()

imageftbbox

imageftbbox -- Give the bounding box of a text using fonts via freetype2

Description

array **imageftbbox** (float `$size`, float `$angle`, string `$font_file`, string `$text` [, array `$extrainfo`])

Warning
This function is currently not documented; only its argument list is available.

Parameters

`size`

`angle`

`font_file`

`text`

`extrainfo`

Return Values

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

Note
This function is only available if PHP is compiled with freetype support (*--with-freetype-dir=DIR*)

ChangeLog

Version	Description
4.3.5	`extrainfo` was made optional.

imagefttext

imagefttext -- Write text to the image using fonts using FreeType 2

Description

array **imagefttext** (resource `$image`, float `$size`, float `$angle`, int `$x`, int `$y`, int `$color`, string `$font_file`, string `$text` [, array `$extrainfo`])

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`size`
The font size to use in points

`angle`
The angle in degrees, with 0 degrees being left-to-right reading text. Higher values represent a counter-clockwise rotation. For example, a value of 90 would result in bottom-to-top reading text.

`x`
The coordinates given by *`x`* and *`y`* will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the imagestring(), where *`x`* and *`y`* define the upper-left corner of the first character. For example, "top left" is 0, 0.

`y`
The y-ordinate. This sets the position of the fonts baseline, not the very bottom of the character.

`color`
The index of the desired color for the text, see imagecolorexact()

`font_file`
The full path to the font being used.

`text`
Text to be inserted into image.

`extrainfo`

Return Values

This function returns an array defining the four points of the box, starting in the lower left and moving counter-clockwise:

0	lower left x-coordinate
1	lower left y-coordinate
2	lower right x-coordinate
3	lower right y-coordinate
4	upper right x-coordinate
5	upper right y-coordinate
6	upper left x-coordinate
7	upper left y-coordinate

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

Note
This function is only available if PHP is compiled with freetype support (*--with-freetype-dir=DIR*)

ChangeLog

Version	**Description**
4.3.5	`extrainfo` was made optional.

imagegammacorrect

imagegammacorrect -- Apply a gamma correction to a GD image

Description

bool **imagegammacorrect** (resource `$image`, float `$inputgamma`, float `$outputgamma`)

Applies gamma correction to the given gd *`image`* given an input and an output gamma.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`inputgamma`
The input gamma

`outputgamma`
The output gamma

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imagegd2

imagegd2 -- Output GD2 image to browser or file

Description

bool **imagegd2** (resource `$image` [, string `$filename` [, int `$chunk_size` [, int `$type`]]])

Outputs a GD2 image to the given *`filename`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filename`
The path to save the file to. If not set or **NULL**, the raw image stream will be outputted directly.

`chunk_size`

`type`
Either **IMG_GD2_RAW** or **IMG_GD2_COMPRESSED**. Default is **IMG_GD2_RAW**.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

Note
The GD2 format is commonly used to allow fast loading of parts of images. Note that the GD2 format is only usable in GD2-compatible applications.

ChangeLog

Version	Description
4.3.2	*chunk_size* and *type* were added.

See Also

- imagegd()

imagegd

imagegd -- Output GD image to browser or file

Description

bool **imagegd** (resource `$image` [, string `$filename`])

Outputs a GD image to the given *`filename`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filename`
The path to save the file to. If not set or **NULL**, the raw image stream will be outputted directly.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
The GD format is commonly used to allow fast loading of parts of images. Note that the GD format is only usable in GD-compatible applications.

See Also

- imagegd2()

imagegif

imagegif -- Output image to browser or file

Description

bool **imagegif** (resource `$image` [, string `$filename`])

imagegif() creates the GIF file in filename from the image *`image`*. The *`image`* argument is the return from the imagecreate() or *imagecreatefrom** function.

The image format will be GIF87a unless the image has been made transparent with imagecolortransparent(), in which case the image format will be GIF89a.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filename`
: The path to save the file to. If not set or **NULL**, the raw image stream will be outputted directly.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note

Since all GIF support was removed from the GD library in version 1.6, this function is not available if you are using that version of the GD library. Support is expected to return in a version subsequent to the rerelease of GIF support in the GD library in mid 2004. For more information, see the » GD Project site.
 	
The following code snippet allows you to write more portable PHP applications by auto-detecting the type of GD support which is available. Replace the sequence *header ("Content-type: image/gif"); imagegif ($im);* by the more flexible sequence:
 	
`<?php`
`if (function_exists("imagegif")) {`
`  header("Content-type: image/gif");`
`  imagegif($im);`
`} elseif (function_exists("imagejpeg")) {`

```
   header("Content-type: image/jpeg");
   imagejpeg($im, "", 0.5);
} elseif (function_exists("imagepng")) {
   header("Content-type: image/png");
   imagepng($im);
} elseif (function_exists("imagewbmp")) {
   header("Content-type: image/vnd.wap.wbmp");
   imagewbmp($im);
} else {
   die("No image support in this PHP server");
}
?>
```

Note

As of version 3.0.18 and 4.0.2 you can use the function imagetypes() in place of function_exists() for checking the presence of the various supported image formats:

```
<?php
if (imagetypes() & IMG_GIF) {
   header ("Content-type: image/gif");
   imagegif ($im);
} elseif (imagetypes() & IMG_JPG) {
   /* ... etc. */
}
?>
```

See Also

- imagepng()
- imagewbmp()
- imagejpeg()
- imagetypes()

imagegrabscreen

imagegrabscreen -- Captures the whole screen

Description

resource **imagegrabscreen** (void)

Grabs a screenshot of the whole screen.

Return Values

Returns an image resource identifier on success, **FALSE** on failure.

Examples

Example #38 - imagegrabscreen() **example**

This example demonstrates how to take a screenshot of the current screen and save it as a png image.

```
<?php
$im = imagegrabscreen();
imagepng($im, "myscreenshot.png");
?>
```

Notes

Note

This function is only available on Windows.

See Also

- imagegrabwindow()

imagegrabwindow

imagegrabwindow -- Captures a window

Description

resource **imagegrabwindow** (int `$window` [, int `$client_area`])

Grabs a window or its client area using a windows handle (HWND property in COM instance)

Parameters

`window`
The HWND window ID

`client_area`
Include the client area of the application window

Return Values

Returns an image resource identifier on success, **FALSE** on failure.

Errors/Exceptions

E_NOTICE is issued if *`window_handle`* is invalid window handle. E_WARNING is issued if the Windows API is too old.

Examples

Example #39 - imagegrabwindow() **example**

Capture a window (IE for example)

```
<?php
$browser = new COM("InternetExplorer.Application");
$handle = $browser->HWND;
$browser->Visible = true;
$im = imagegrabwindow($handle);
$browser->Quit();
imagepng($im, "iesnap.png");
?>
```

Capture a window (IE for example) but with its content

```
<?php
```

```
$browser = new COM("InternetExplorer.Application");
$handle = $browser->HWND;
$browser->Visible = true;
$browser->Navigate("http://www.libgd.org");

/* Still working? */
while ($browser->Busy) {
   com_message_pump(4000);
}
$im = imagegrabwindow($handle, 0);
$browser->Quit();
imagepng($im, "iesnap.png");
?>
```

Notes

Note
This function is only available on Windows.

See Also

- imagegrabscreen()

imageinterlace

imageinterlace -- Enable or disable interlace

Description

int **imageinterlace** (resource `$image` [, int `$interlace`])

imageinterlace() turns the interlace bit on or off.

If the interlace bit is set and the image is used as a JPEG image, the image is created as a progressive JPEG.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`interlace`
If non-zero, the image will be interlaced, else the interlace bit is turned off.

Return Values

Returns 1 if the interlace bit is set for the image, 0 otherwise.

imageistruecolor

imageistruecolor -- Finds whether an image is a truecolor image

Description

bool **imageistruecolor** (resource `$image`)

imageistruecolor() finds whether the image *`image`* is a truecolor image.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

Return Values

Returns **TRUE** if the *`image`* is truecolor, **FALSE** otherwise.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

See Also

- imagecreatetruecolor()

imagejpeg

imagejpeg -- Output image to browser or file

Description

bool **imagejpeg** (resource `$image` [, string `$filename` [, int `$quality`]])

imagejpeg() creates a JPEG file from the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filename`
The path to save the file to. If not set or **NULL**, the raw image stream will be outputted directly. To skip this argument in order to provide the *`quality`* parameter, use **NULL**.

`quality`
`quality` is optional, and ranges from 0 (worst quality, smaller file) to 100 (best quality, biggest file). The default is the default IJG quality value (about 75).

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
JPEG support is only available if PHP was compiled against GD-1.8 or later.

Note
If you want to output Progressive JPEGs, you need to set interlacing on with imageinterlace().

See Also

- imagepng()
- imagegif()
- imagewbmp()
- imageinterlace()
- imagetypes()

imagelayereffect

imagelayereffect -- Set the alpha blending flag to use the bundled libgd layering effects

Description

bool **imagelayereffect** (resource `$image`, int `$effect`)

Set the alpha blending flag to use the bundled libgd layering effects.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`effect`
One of the following constants:
IMG_EFFECT_REPLACE
Use pixel replacement (equivalent of passing **TRUE** to imagealphablending())

IMG_EFFECT_ALPHABLEND
Use normal pixel blending (equivalent of passing **FALSE** to imagealphablending())

IMG_EFFECT_NORMAL
Same as **IMG_EFFECT_ALPHABLEND**.

IMG_EFFECT_OVERLAY
Overlay has the effect that black background pixels will remain black, white background pixels will remain white, but grey background pixels will take the colour of the foreground pixel.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available if PHP is compiled with the bundled version of the GD library.

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

imageline

imageline -- Draw a line

Description

bool **imageline** (resource `$image`, int `$x1`, int `$y1`, int `$x2`, int `$y2`, int `$color`)

imageline() draws a line between the two given points.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`x1`
x-coordinate for first point

`y1`
y-coordinate for first point

`x2`
x-coordinate for second point

`y2`
y-coordinate for second point

`color`
The line color. A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #40 - Drawing a thick line

```
<?php

function imagelinethick($image, $x1, $y1, $x2, $y2, $color, $thick = 1)
{
   /* this way it works well only for orthogonal lines
   imagesetthickness($image, $thick);
```

```
    return imageline($image, $x1, $y1, $x2, $y2, $color);
    */
    if ($thick == 1) {
        return imageline($image, $x1, $y1, $x2, $y2, $color);
    }
    $t = $thick / 2 - 0.5;
    if ($x1 == $x2 || $y1 == $y2) {
        return imagefilledrectangle($image, round(min($x1, $x2) - $t),
round(min($y1, $y2) - $t), round(max($x1, $x2) + $t), round(max($y1, $y2) +
$t), $color);
    }
    $k = ($y2 - $y1) / ($x2 - $x1); //y = kx + q
    $a = $t / sqrt(1 + pow($k, 2));
    $points = array(
        round($x1 - (1+$k)*$a), round($y1 + (1-$k)*$a),
        round($x1 - (1-$k)*$a), round($y1 - (1+$k)*$a),
        round($x2 + (1+$k)*$a), round($y2 - (1-$k)*$a),
        round($x2 + (1-$k)*$a), round($y2 + (1+$k)*$a),
    );
    imagefilledpolygon($image, $points, 4, $color);
    return imagepolygon($image, $points, 4, $color);
}

?>
```

See Also

- imagecreatetruecolor()
- imagecolorallocate()

imageloadfont

imageloadfont -- Load a new font

Description

int **imageloadfont** (string `$file`)

imageloadfont() loads a user-defined bitmap and returns its identifier.

Parameters

`file`
The font file format is currently binary and architecture dependent. This means you should generate the font files on the same type of CPU as the machine you are running PHP on.

Font file format

byte position	C data type	description
byte 0-3	int	number of characters in the font
byte 4-7	int	value of first character in the font (often 32 for space)
byte 8-11	int	pixel width of each character
byte 12-15	int	pixel height of each character
byte 16-	char	array with character data, one byte per pixel in each character, for a total of (nchars*width*height) bytes.

Return Values

The font identifier which is always bigger than 5 to avoid conflicts with built-in fonts or **FALSE** on errors.

Examples

Example #41 - Using imageloadfont

```
<?php
header("Content-type: image/png");
$im = imagecreatetruecolor(50, 20);
$black = imagecolorallocate($im, 0, 0, 0);
$white = imagecolorallocate($im, 255, 255, 255);
imagefilledrectangle($im, 0, 0, 49, 19, $white);
$font = imageloadfont("04b.gdf");
imagestring($im, $font, 0, 0, "Hello", $black);
imagepng($im);
?>
```

See Also

- imagefontwidth()
- imagefontheight()

imagepalettecopy

imagepalettecopy -- Copy the palette from one image to another

Description

void **imagepalettecopy** (resource `$destination`, resource `$source`)

imagepalettecopy() copies the palette from the *`source`* image to the *`destination`* image.

Parameters

`destination`
: The destination image resource

`source`
: The source image resource

Return Values

No value is returned.

imagepng

imagepng -- Output a PNG image to either the browser or a file

Description

bool **imagepng** (resource `$image` [, string `$filename` [, int `$quality` [, int `$filters`]]])

Outputs or saves a PNG image from the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filename`
The path to save the file to. If not set or **NULL**, the raw image stream will be outputted directly.

Note
NULL is invalid if the *`quality`* and *`filters`* arguments are not used.

`quality`
Compression level: from 0 (no compression) to 9.

`filters`
Allows reducing the PNG file size. It is a bitmask field which may be set to any combination of the *PNG_FILTER_XXX* constants. **PNG_NO_FILTER** or **PNG_ALL_FILTERS** may also be used to respectively disable or activate all filters.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ChangeLog

Version	Description
5.1.3	Added the *`filters`* parameter.

5.1.2	Added the *quality* parameter.

Examples

```
<?php
$im = imagecreatefrompng("test.png");
imagepng($im);
?>
```

See Also

- imagegif()
- imagewbmp()
- imagejpeg()
- imagetypes()

imagepolygon

imagepolygon -- Draws a polygon

Description

bool **imagepolygon** (resource `$image`, array `$points`, int `$num_points`, int `$color`)

imagepolygon() creates a polygon in the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`points`
An array containing the polygon's vertices, i.e. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc.

`num_points`
Total number of points (vertices)

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #42 - imagepolygon() example

```
<?php
// create a blank image
$image = imagecreatetruecolor(400, 300);

// fill the background color
$bg = imagecolorallocate($image, 0, 0, 0);

// choose a color for the polygon
$col_poly = imagecolorallocate($image, 255, 255, 255);

// draw the polygon
imagepolygon($image, array (
```

```
        0,   0,
        100, 200,
        300, 200
    ),
    3,
    $col_poly);

// output the picture
header("Content-type: image/png");
imagepng($image);

?>
```

The above example will output something similar to:

See Also

- imagecreate()
- imagecreatetruecolor()

imagepsbbox

imagepsbbox -- Give the bounding box of a text rectangle using PostScript Type1 fonts

Description

array **imagepsbbox** (string `$text`, int `$font`, int `$size`)

array **imagepsbbox** (string `$text`, int `$font`, int `$size`, int `$space`, int `$tightness`, float `$angle`)

Gives the bounding box of a text rectangle using PostScript Type1 fonts.

The bounding box is calculated using information available from character metrics, and unfortunately tends to differ slightly from the results achieved by actually rasterizing the text. If the angle is 0 degrees, you can expect the text to need 1 pixel more to every direction.

Parameters

`text`

`font`
Can be 1, 2, 3, 4, 5 for built-in fonts in latin2 encoding (where higher numbers corresponding to larger fonts) or any of your own font identifiers registered with imageloadfont().

`size`
`size` is expressed in pixels.

`space`
Allows you to change the default value of a space in a font. This amount is added to the normal value and can also be negative. Expressed in character space units, where 1 unit is 1/1000th of an em-square.

`tightness`
`tightness` allows you to control the amount of white space between characters. This amount is added to the normal character width and can also be negative. Expressed in character space units, where 1 unit is 1/1000th of an em-square.

`angle`
`angle` is in degrees.

Return Values

Returns an array containing the following elements:

0	left x-coordinate
1	upper y-coordinate
2	right x-coordinate
3	lower y-coordinate

Notes

Note
This function is only available if PHP is compiled using *--with-t1lib[=DIR]*.

See Also

- imagepstext()

imagepsencodefont

imagepsencodefont -- Change the character encoding vector of a font

Description

bool **imagepsencodefont** (resource `$font_index`, string `$encodingfile`)

Loads a character encoding vector from a file and changes the fonts encoding vector to it. As a PostScript fonts default vector lacks most of the character positions above 127, you'll definitely want to change this if you use an other language than English.

If you find yourself using this function all the time, a much better way to define the encoding is to set ps.default_encoding in the configuration file to point to the right encoding file and all fonts you load will automatically have the right encoding.

Parameters

`font_index`

`encodingfile`

The exact format of this file is described in T1libs documentation. T1lib comes with two ready-to-use files, *IsoLatin1.enc* and *IsoLatin2.enc*.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available if PHP is compiled using *--with-t1lib[=DIR]*.

imagepsextendfont

imagepsextendfont -- Extend or condense a font

Description

bool **imagepsextendfont** (int `$font_index`, float `$extend`)

Extend or condense a font (*`font_index`*), if the value of the *`extend`* parameter is less than one you will be condensing the font.

Parameters

`font_index`

`extend`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available if PHP is compiled using *--with-t1lib[=DIR]*.

imagepsfreefont

imagepsfreefont -- Free memory used by a PostScript Type 1 font

Description

bool **imagepsfreefont** (resource `$fontindex`)

imagepsfreefont() frees memory used by a PostScript Type 1 font.

Parameters

`fontindex`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available if PHP is compiled using *--with-t1lib[=DIR]*.

See Also

- imagepsloadfont()

imagepsloadfont

imagepsloadfont -- Load a PostScript Type 1 font from file

Description

resource **imagepsloadfont** (string `$filename`)

Load a PostScript Type 1 font from the given *`filename`*.

Parameters

`filename`

Return Values

In the case everything went right, a valid font index will be returned and can be used for further purposes. Otherwise the function returns **FALSE**.

Examples

Example #43 - imagepsloadfont() **example**

```
<?php
header("Content-type: image/png");
$im = imagecreatetruecolor(350, 45);
$black = imagecolorallocate($im, 0, 0, 0);
$white = imagecolorallocate($im, 255, 255, 255);
imagefilledrectangle($im, 0, 0, 349, 44, $white);
$font = imagepsloadfont("bchbi.pfb"); // or locate your .pfb files on your
machine
imagepstext($im, "Testing... It worked!", $font, 32, $white, $black, 32,
32);
imagepsfreefont($font);
imagepng($im);
imagedestroy($im);
?>
```

Notes

Note
This function is only available if PHP is compiled using *--with-t1lib[=DIR]*.

See Also

- imagepsfreefont()

imagepsslantfont

imagepsslantfont -- Slant a font

Description

bool **imagepsslantfont** (resource `$font_index`, float `$slant`)

Slant a given font given.

Parameters

`font_index`

`slant`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available if PHP is compiled using *--with-t1lib[=DIR]*.

imagepstext

imagepstext -- Draws a text over an image using PostScript Type1 fonts

Description

array **imagepstext** (resource `$image`, string `$text`, resource `$font`, int `$size`, int `$foreground`, int `$background`, int `$x`, int `$y` [, int `$space` [, int `$tightness` [, float `$angle` [, int `$antialias_steps`]]]])

Draws a text on an image using PostScript Type1 fonts.

Refer to PostScript documentation about fonts and their measuring system if you have trouble understanding how this works.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`text`
The text to be written.

`font`
Can be 1, 2, 3, 4, 5 for built-in fonts in latin2 encoding (where higher numbers corresponding to larger fonts) or any of your own font identifiers registered with imageloadfont().

`size`
`size` is expressed in pixels.

`foreground`
The color in which the text will be painted.

`background`
The color to which the text will try to fade in with antialiasing. No pixels with the color *`background`* are actually painted, so the background image does not need to be of solid color.

`x`
x-coordinate for the lower-left corner of the first character

`y`
y-coordinate for the lower-left corner of the first character

`space`
Allows you to change the default value of a space in a font. This amount is added to

the normal value and can also be negative. Expressed in character space units, where 1 unit is 1/1000th of an em-square.

tightness
: *tightness* allows you to control the amount of white space between characters. This amount is added to the normal character width and can also be negative. Expressed in character space units, where 1 unit is 1/1000th of an em-square.

angle
: *angle* is in degrees.

antialias_steps
: Allows you to control the number of colours used for antialiasing text. Allowed values are 4 and 16. The higher value is recommended for text sizes lower than 20, where the effect in text quality is quite visible. With bigger sizes, use 4. It's less computationally intensive.

Return Values

This function returns an array containing the following elements:

0	lower left x-coordinate
1	lower left y-coordinate
2	upper right x-coordinate
3	upper right y-coordinate

Notes

Note
This function is only available if PHP is compiled using *--with-t1lib[=DIR]*.

See Also

- imagepsbbox()

imagerectangle

imagerectangle -- Draw a rectangle

Description

bool **imagerectangle** (resource `$image`, int `$x1`, int `$y1`, int `$x2`, int `$y2`, int `$color`)

imagerectangle() creates a rectangle starting at the specified coordinates.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`x1`
Upper left x coordinate

`y1`
Upper left y coordinate 0, 0 is the top left corner of the image.

`x2`
Bottom right x coordinate

`y2`
Bottom right y coordinate

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imagerotate

imagerotate -- Rotate an image with a given angle

Description

resource **imagerotate** (resource `$source_image`, float `$angle`, int `$bgd_color` [, int `$ignore_transparent`])

Rotates the *`source_image`* image using the given *`angle`* in degrees.

The center of rotation is the center of the image, and the rotated image is scaled down so that the whole rotated image fits in the destination image - the edges are not clipped.

Parameters

`source_image`
: The source image link

`angle`
: Rotation angle, in degrees.

`bgd_color`
: Specifies the color of the uncovered zone after the rotation

`ignore_transparent`
: If set and non-zero, transparent colors are ignored (otherwise kept).

Return Values

ChangeLog

Version	Description
5.1.0	*`ignore_transparent`* was added.

Examples

Example #44 - Rotate an image 180 degrees

This example rotates an image 180 degrees - upside down.

```
<?php
// File and rotation
$filename = 'test.jpg';
$degrees = 180;

// Content type
header('Content-type: image/jpeg');

// Load
$source = imagecreatefromjpeg($filename);

// Rotate
$rotate = imagerotate($source, $degrees, 0);

// Output
imagejpeg($rotate);
?>
```

The above example will output something similar to:

Notes

Note

This function is only available if PHP is compiled with the bundled version of the GD library.

imagesavealpha

imagesavealpha -- Set the flag to save full alpha channel information (as opposed to single-color transparency) when saving PNG images

Description

bool **imagesavealpha** (resource `$image`, bool `$saveflag`)

imagesavealpha() sets the flag to attempt to save full alpha channel information (as opposed to single-color transparency) when saving PNG images.

You have to unset alphablending (*imagealphablending($im, false)*), to use it.

Alpha channel is not supported by all browsers, if you have problem with your browser, try to load your script with an alpha channel compliant browser, e.g. latest Mozilla.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`saveflag`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

See Also

- imagealphablending()

imagesetbrush

imagesetbrush -- Set the brush image for line drawing

Description

bool **imagesetbrush** (resource `$image`, resource `$brush`)

imagesetbrush() sets the brush image to be used by all line drawing functions (such as imageline() and imagepolygon()) when drawing with the special colors **IMG_COLOR_BRUSHED** or **IMG_COLOR_STYLEDBRUSHED**.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`brush`
An image resource

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
You need not take special action when you are finished with a brush, but if you destroy the brush image, you must not use the **IMG_COLOR_BRUSHED** or **IMG_COLOR_STYLEDBRUSHED** colors until you have set a new brush image!

imagesetpixel

imagesetpixel -- Set a single pixel

Description

bool **imagesetpixel** (resource `$image`, int `$x`, int `$y`, int `$color`)

imagesetpixel() draws a pixel at the specified coordinate.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`x`
x-coordinate

`y`
y-coordinate

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #45 - imagesetpixel() example

A random drawing that ends with a regular picture.

```
<?php

$x = 200;
$y = 200;

$gd = imagecreatetruecolor($x, $y);

$corners[0] = array('x' => 100, 'y' =>  10);
$corners[1] = array('x' =>   0, 'y' => 190);
$corners[2] = array('x' => 200, 'y' => 190);
```

```
$red = imagecolorallocate($gd, 255, 0, 0);

for ($i = 0; $i < 100000; $i++) {
 imagesetpixel($gd, round($x),round($y), $red);
 $a = rand(0, 2);
 $x = ($x + $corners[$a]['x']) / 2;
 $y = ($y + $corners[$a]['y']) / 2;
}

header('Content-Type: image/png');
imagepng($gd);

?>
```

The above example will output something similar to:

See Also

- imagecreatetruecolor()
- imagecolorallocate()

imagesetstyle

imagesetstyle -- Set the style for line drawing

Description

bool **imagesetstyle** (resource `$image`, array `$style`)

imagesetstyle() sets the style to be used by all line drawing functions (such as imageline() and imagepolygon()) when drawing with the special color **IMG_COLOR_STYLED** or lines of images with color **IMG_COLOR_STYLEDBRUSHED**.

Parameters

image
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

style
: An array of pixel colors. You can use the **IMG_COLOR_TRANSPARENT** constant to add a transparent pixel.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Following example script draws a dashed line from upper left to lower right corner of the canvas:

Example #46 - imagesetstyle() example

```
<?php
header("Content-type: image/jpeg");
$im  = imagecreatetruecolor(100, 100);
$w   = imagecolorallocate($im, 255, 255, 255);
$red = imagecolorallocate($im, 255, 0, 0);

/* Draw a dashed line, 5 red pixels, 5 white pixels */
$style = array($red, $red, $red, $red, $red, $w, $w, $w, $w, $w);
imagesetstyle($im, $style);
imageline($im, 0, 0, 100, 100, IMG_COLOR_STYLED);

/* Draw a line of happy faces using imagesetbrush() with imagesetstyle */
$style = array($w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $red);
imagesetstyle($im, $style);

$brush =
```

```
imagecreatefrompng("http://www.libpng.org/pub/png/images/smile.happy.png");
$w2 = imagecolorallocate($brush, 255, 255, 255);
imagecolortransparent($brush, $w2);
imagesetbrush($im, $brush);
imageline($im, 100, 0, 0, 100, IMG_COLOR_STYLEDBRUSHED);

imagejpeg($im);
imagedestroy($im);
?>
```

The above example will output something similar to:

See Also

- imagesetbrush()
- imageline()

imagesetthickness

imagesetthickness -- Set the thickness for line drawing

Description

bool **imagesetthickness** (resource `$image`, int `$thickness`)

imagesetthickness() sets the thickness of the lines drawn when drawing rectangles, polygons, ellipses etc. etc. to *`thickness`* pixels.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`thickness`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

imagesettile

imagesettile -- Set the tile image for filling

Description

bool **imagesettile** (resource `$image`, resource `$tile`)

imagesettile() sets the tile image to be used by all region filling functions (such as imagefill() and imagefilledpolygon()) when filling with the special color **IMG_COLOR_TILED**.

A tile is an image used to fill an area with a repeated pattern. *Any* GD image can be used as a tile, and by setting the transparent color index of the tile image with imagecolortransparent(), a tile allows certain parts of the underlying area to shine through can be created.

Note
You need not take special action when you are finished with a tile, but if you destroy the tile image, you must not use the **IMG_COLOR_TILED** color until you have set a new tile image!

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`tile`
The image resource to be used as a tile

Return Values

Returns **TRUE** on success or **FALSE** on failure.

imagestring

imagestring -- Draw a string horizontally

Description

bool **imagestring** (resource `$image`, int `$font`, int `$x`, int `$y`, string `$string`, int `$color`)

Draws a *`string`* at the given coordinates.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`font`
Can be 1, 2, 3, 4, 5 for built-in fonts in latin2 encoding (where higher numbers corresponding to larger fonts) or any of your own font identifiers registered with imageloadfont().

`x`
x-coordinate of the upper left corner

`y`
y-coordinate of the upper left corner

`string`
The string to be written

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #47 - imagestring() example

```
<?php
// create a 100*30 image
$im = imagecreate(100, 30);
```

```
// white background and blue text
$bg = imagecolorallocate($im, 255, 255, 255);
$textcolor = imagecolorallocate($im, 0, 0, 255);

// write the string at the top left
imagestring($im, 5, 0, 0, "Hello world!", $textcolor);

// output the image
header("Content-type: image/png");
imagepng($im);
?>
```

The above example will output something similar to:

See Also

- imageloadfont()
- imagettftext()

imagestringup

imagestringup -- Draw a string vertically

Description

bool **imagestringup** (resource `$image`, int `$font`, int `$x`, int `$y`, string `$string`, int `$color`)

Draws a *`string`* vertically at the given coordinates.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`font`
Can be 1, 2, 3, 4, 5 for built-in fonts in latin2 encoding (where higher numbers corresponding to larger fonts) or any of your own font identifiers registered with imageloadfont().

`x`
x-coordinate of the upper left corner

`y`
y-coordinate of the upper left corner

`string`
The string to be written

`color`
A color identifier created with imagecolorallocate()

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- imageloadfont()

imagesx

imagesx -- Get image width

Description

int **imagesx** (resource `$image`)

Returns the width of the given *`image`* resource.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

Return Values

Return the width of the *`image`* or **FALSE** on errors.

Examples

Example #48 - Using imagesx()

```
<?php

// create a 300*200 image
$img = imagecreatetruecolor(300, 200);

echo imagesx($img); // 300

?>
```

See Also

- imagecreatetruecolor()
- getimagesize()
- imagesy()

imagesy

imagesy -- Get image height

Description

int **imagesy** (resource `$image`)

Returns the height of the given *`image`* resource.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

Return Values

Return the height of the *`image`* or **FALSE** on errors.

Examples

Example #49 - Using imagesy()

```
<?php

// create a 300*200 image
$img = imagecreatetruecolor(300, 200);

echo imagesy($img); // 200

?>
```

See Also

- imagecreatetruecolor()
- getimagesize()
- imagesx()

imagetruecolortopalette

imagetruecolortopalette -- Convert a true color image to a palette image

Description

bool **imagetruecolortopalette** (resource `$image`, bool `$dither`, int `$ncolors`)

imagetruecolortopalette() converts a truecolor image to a palette image. The code for this function was originally drawn from the Independent JPEG Group library code, which is excellent. The code has been modified to preserve as much alpha channel information as possible in the resulting palette, in addition to preserving colors as well as possible. This does not work as well as might be hoped. It is usually best to simply produce a truecolor output image instead, which guarantees the highest output quality.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`dither`
Indicates if the image should be dithered - if it is **TRUE** then dithering will be used which will result in a more speckled image but with better color approximation.

`ncolors`
Sets the maximum number of colors that should be retained in the palette.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function requires GD 2.0.1 or later (2.0.28 or later is recommended).

imagettfbbox

imagettfbbox -- Give the bounding box of a text using TrueType fonts

Description

array **imagettfbbox** (float `$size`, float `$angle`, string `$fontfile`, string `$text`)

This function calculates and returns the bounding box in pixels for a TrueType text.

Parameters

`size`
The font size in pixels

`angle`
Angle in degrees in which *`text`* will be measured

`fontfile`
The name of the TrueType font file (can be a URL). Depending on which version of the GD library that PHP is using, it may attempt to search for files that do not begin with a leading '/' by appending '.ttf' to the filename and searching along a library-defined font path

`text`
The string to be measured

Return Values

imagettfbbox() returns an array with 8 elements representing four points making the bounding box of the text:

0	lower left corner, X position
1	lower left corner, Y position
2	lower right corner, X position
3	lower right corner, Y position
4	upper right corner, X position
5	upper right corner, Y position
6	upper left corner, X position
7	upper left corner, Y position

The points are relative to the *text* regardless of the `angle`, so "upper left" means in the top left-hand corner seeing the text horizontally.

See Also

Note
This function requires both the GD library and the » FreeType library.

See Also

- imagettftext()

imagettftext

imagettftext -- Write text to the image using TrueType fonts

Description

array **imagettftext** (resource `$image`, float `$size`, float `$angle`, int `$x`, int `$y`, int `$color`, string `$fontfile`, string `$text`)

Writes the given *`text`* into the image using TrueType fonts.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`size`
The font size. Depending on your version of GD, this should be specified as the pixel size (GD1) or point size (GD2)

`angle`
The angle in degrees, with 0 degrees being left-to-right reading text. Higher values represent a counter-clockwise rotation. For example, a value of 90 would result in bottom-to-top reading text.

`x`
The coordinates given by *`x`* and *`y`* will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the imagestring(), where *`x`* and *`y`* define the upper-left corner of the first character. For example, "top left" is 0, 0.

`y`
The y-ordinate. This sets the position of the fonts baseline, not the very bottom of the character.

`color`
The color index. Using the negative of a color index has the effect of turning off antialiasing. See imagecolorallocate()

`fontfile`
The path to the TrueType font you wish to use. Depending on which version of the GD library PHP is using, *when `fontfile` does not begin with a leading / then.ttf will be appended* to the filename and the library will attempt to search for that filename along a library-defined font path. When using versions of the GD library lower than 2.0.18, a *space* character, rather than a semicolon, was used as the 'path separator' for different font files. Unintentional use of this feature will result in the warning message: *Warning: Could not find/open font*. For these affected versions, the only solution is moving the

font to a path which does not contain spaces. In many cases where a font resides in the same directory as the script using it the following trick will alleviate any include problems.

```
<?php
// Set the enviroment variable for GD
putenv('GDFONTPATH=' . realpath('.'));
      The y-ordinate. This sets the position of the fonts baseline, not
      the very bottom of the character.
// Name the font to be used (note the lack of the .ttf extension)
$font = 'SomeFont';
?>
```

text

The text string in UTF-8 encoding. May include decimal numeric character references (of the form: €) to access characters in a font beyond position 127. The hexadecimal format (like ©) is supported as of PHP 5.2.0. Strings in UTF-8 encoding can be passed directly. Named entities, such as ©, are not supported. Consider using html_entity_decode() to decode these named entities into UTF-8 strings (html_entity_decode() supports this as of PHP 5.0.0). If a character is used in the string which is not supported by the font, a hollow rectangle will replace the character.

Return Values

Returns an array with 8 elements representing four points making the bounding box of the text. The order of the points is lower left, lower right, upper right, upper left. The points are relative to the text regardless of the angle, so "upper left" means in the top left-hand corner when you see the text horizontally.

Examples

Example #50 - imagettftext() example

This example script will produce a white PNG 400x30 pixels, with the words "Testing..." in black (with grey shadow), in the font Arial.

```
<?php
// Set the content-type
header("Content-type: image/png");

// Create the image
$im = imagecreatetruecolor(400, 30);

// Create some colors
$white = imagecolorallocate($im, 255, 255, 255);
$grey = imagecolorallocate($im, 128, 128, 128);
$black = imagecolorallocate($im, 0, 0, 0);
imagefilledrectangle($im, 0, 0, 399, 29, $white);

// The text to draw
$text = 'Testing...';
```

```
// Replace path by your own font path
$font = 'arial.ttf';

// Add some shadow to the text
imagettftext($im, 20, 0, 11, 21, $grey, $font, $text);

// Add the text
imagettftext($im, 20, 0, 10, 20, $black, $font, $text);

// Using imagepng() results in clearer text compared with imagejpeg()
imagepng($im);
imagedestroy($im);
?>
```

The above example will output something similar to:

Notes

Note
This function requires both the GD library and the » FreeType library.

See Also

- imagettfbbox()

imagetypes

imagetypes -- Return the image types supported by this PHP build

Description

int **imagetypes** (void)

Returns the image types supported by the current PHP installation.

Return Values

Returns a bit-field corresponding to the image formats supported by the version of GD linked into PHP. The following bits are returned, **IMG_GIF** | **IMG_JPG** | **IMG_PNG** | **IMG_WBMP** | **IMG_XPM**.

Examples

Example #51 - Checking for PNG support

```
<?php
if (imagetypes() & IMG_PNG) {
   echo "PNG Support is enabled";
}
?>
```

imagewbmp

imagewbmp -- Output image to browser or file

Description

bool **imagewbmp** (resource `$image` [, string `$filename` [, int `$foreground`]])

imagewbmp() outputs or save a WBMP version of the given *`image`*.

Parameters

`image`
An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filename`
The path to save the file to. If not set or **NULL**, the raw image stream will be outputted directly.

`foreground`
You can set the foreground color with this parameter by setting an identifier obtained from imagecolorallocate(). The default foreground color is black.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
WBMP support is only available if PHP was compiled against GD-1.8 or later.

See Also

- image2wbmp()
- imagepng()
- imagegif()
- imagejpeg()
- imagetypes()

imagexbm

imagexbm -- Output XBM image to browser or file

Description

bool **imagexbm** (resource `$image`, string `$filename` [, int `$foreground`])

Outputs or save an XBM version of the given *`image`*.

Parameters

`image`
: An image resource, returned by one of the image creation functions, such as imagecreatetruecolor().

`filename`
: The path to save the file to. If not set or **NULL**, the raw image stream will be outputted directly.

`foreground`
: You can set the foreground color with this parameter by setting an identifier obtained from imagecolorallocate(). The default foreground color is black.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is only available if PHP is compiled with the bundled version of the GD library.

iptcembed

iptcembed -- Embed binary IPTC data into a JPEG image

Description

mixed **iptcembed** (string `$iptcdata`, string `$jpeg_file_name` [, int `$spool`])

Warning
This function is currently not documented; only its argument list is available.

Parameters

`iptcdata`

`jpeg_file_name`

`spool`

iptcparse

iptcparse -- Parse a binary IPTC block into single tags.

Description

array **iptcparse** (string `$iptcblock`)

Parses an » IPTC block into its single tags.

Parameters

`iptcblock`
A binary IPTC block

Return Values

Returns an array using the tagmarker as an index and the value as the value. It returns **FALSE** on error or if no IPTC data was found.

See Also

- getimagesize() for an example

jpeg2wbmp

jpeg2wbmp -- Convert JPEG image file to WBMP image file

Description

bool **jpeg2wbmp** (string `$jpegname`, string `$wbmpname`, int `$dest_height`, int `$dest_width`, int `$threshold`)

Converts a JPEG file into a WBMP file.

Parameters

`jpegname`
Path to JPEG file

`wbmpname`
Path to destination WBMP file

`dest_height`
Destination image height

`dest_width`
Destination image width

`threshold`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
WBMP support is only available if PHP was compiled against GD-1.8 or later.

Note
JPEG support is only available if PHP was compiled against GD-1.8 or later.

See Also

- png2wbmp()

png2wbmp

png2wbmp -- Convert PNG image file to WBMP image file

Description

bool **png2wbmp** (string `$pngname`, string `$wbmpname`, int `$dest_height`, int `$dest_width`, int `$threshold`)

Converts a PNG file into a WBMP file.

Parameters

`pngname`
: Path to PNG file

`wbmpname`
: Path to destination WBMP file

`dest_height`
: Destination image height

`dest_width`
: Destination image width

`threshold`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
WBMP support is only available if PHP was compiled against GD-1.8 or later.

See Also

- jpeg2wbmp()