Image

The Image class offers a bunch of image processing features.

Namespace: `\`
File location: `lib/image.php`

Instantiation

```
$img = new Image ( [ string $file = NULL [, bool $flag = FALSE [, string $path = NULL
]]] )
```

You can create a new Image object from an existing image file like this:

```
$img = new Image('path/to/your/image.jpg'); // relative to UI search path
```

To create a new empty Image, i.e. for creating a captcha, just leave out the first `$file` argument:

```
$img = new Image();
```

The constructor also has a 2nd argument, the `$flag` option. Setting it to `TRUE` enables the file history (image#history), which can save additional states for the current file, allowing for example to revert (image#restore) to a specified state after having applied a filter.

If your image file is not located within one of the UI (quick-reference#ui) search paths, you must use the `$path` argument to specify its directory path. In particular, set `$path=''` if the provided file path is absolute.

Processing

invert

Invert image

```
$img->invert();
```

brightness

Adjust brightness

```
$img->brightness( int $level );
```

`$level` range: -255 to 255

contrast

Adjust contrast

```
$img->contrast( int $level );
```

`$level` range: -100 to 100

grayscale

Convert to grayscale

```
$img->grayscale();
```

smooth

Adjust smoothness

```
$img->smooth( int $level);
```

`$level` range: -8 to 8, but as its been used for a matrix operation, greater values are applyable too, but may lead to unusable results.

emboss

Emboss the image

```
$img->emboss();
```

sepia

Apply sepia effect

```
$img->sepia();
```

pixelate

Pixelate the image

```
$img->pixelate( int $size );
```

`$size` is the block size of a pixel, usually range: 0 - 100

blur

Blur the image using Gaussian filter

```
$img->blur( bool $selective );
```

Set `$selective to TRUE` to use a selective blur, otherwise a gaussian blur is used.

sketch

Apply sketch effect

```
$img->sketch();
```

hflip

Flip on horizontal axis

```
$img->hflip();
```

vflip

Flip on vertical axis

```
$img->vflip();
```

crop

Crop the image

```
$img->crop( int $x1, int $y1, int $x2, int $y2);
```

resize

Resize image (Maintain aspect ratio)

```
$img->resize( int $width [, int $height = NULL [, bool $crop = TRUE [, bool $enlarge
= TRUE ]]] );
```

If `$crop` is `TRUE` the image will be resized to fit with its smallest side into the resize box. The overflowing margins will be cropped relative to the center, so your resulting image will fully cover your desired width and height values.

If `$crop` is `FALSE` it will be resized to fit with its longest side into the resize box.

If `$enlarge` is `FALSE` the image will not be scaled up to fit in the resize box.

If either `$width` or `$height` is null, the other dimension is guessed in order to preserve the aspect ratio.

rotate

Rotate image

```
$img->rotate( int $angle );
```

overlay

Apply an image overlay

```
$img->overlay( Image $img [, int|array $align = NULL [, int $alpha = 100 ]] );
```

This is used to merge to images, i.e. for watermarks. You need to provide another Image object and can align that by the `$align` argument in a bitwise way or provide an (x,y) array.

Example:

```
$img = new \Image;
$img = new \Image('images/south-park.jpg');

$overlay = new \Image('images/watermark.png');
$overlay->resize(100,38)->rotate(90);

$img->overlay( $overlay, \Image::POS_Right | \Image::POS_Middle );
// or
$img->overlay( $overlay, array(200,100), 60);
```

Possible values for `$align` can be combined using this options:

x - align:

- POS_Left
- POS_Center
- POS_Right

y - align:

- POS_Top
- POS_Middle

- POS_Bottom

or just use an array containing the x and y values.

Use the `$alpha` argument to control the transparency of the overlay (0-100).

Rendering

identicon

Generate identicon

```
$img->identicon( string $str [, int $size = 64 [, int $blocks = 4 ]] );
```

This method renders a unique identicon (https://en.wikipedia.org/wiki/Identicon) based on the given `$str` . The `$size` argument defines the width and height of the resulting image. `$blocks` (range 2 - 7) describes the granularity of the pattern blocks inside the identicon.

captcha

Generate CAPTCHA image

```
$img->captcha( string $font [, int $size = 24 [, int $len = 5 [, string|bool $key = N
ULL [, string $path='' [, $foregroundcolor=0xFFF [, $backgroundcolor=0x000 ]]]]]] );
```

This renders a captcha image. Please have a look to this user guide section about rendering captcha images (plug-ins#captcha-images), to see a little example. If your font file is not located in the `UI` directory, you can then set its location with the `$path` argument.

Info

width

Return image width

```
$img->width();
```

height

Return image height

```
$img->height();
```

rgb

Convert RGB hex triad to array

```
$img->rgb( int | string $color );
```

You can pass an integer or a string. Even CSS shorthand notations are supported:

```
$img->rgb( 0xFF0033 ); // returns array( 255, 0, 51 );
$img->rgb( '#FF0033' ); // idem
$img->rgb( 'ff0033' ); // idem
$img->rgb( '#F03' ); // idem
$img->rgb( 'f03' ); // idem
```

Output

render

Output a raw image stream to the HTTP client

```
$img->render( [ string $imageformat = 'png' ] );
```

This method sends the image stream to the HTTP client. The image can be rendered in `png` , `jpeg` , `gif` or `wbmp` format. If not specified, the PNG image format will be used. For instance this example will output a PNG image:

```
$img->render(); // Send a [Content-Type: image/png] stream to the HTTP client
```

Extra arguments are allowed, depending on the requested image format:

PNG format (default)

```
$img->render( 'png' [, int $quality [, int $filters ]] );
```

`$quality` indicates the compression level from 0 to 9. The default quality seems (http://php.net/manual/en/function.imagepng.php#106093) to be 6.

`$filters` allows reducing the PNG file size. It is a bitmask field which may be set to any combination of the following constants: `PNG_FILTER_NONE` , `PNG_FILTER_SUB` , `PNG_FILTER_UP` , `PNG_FILTER_AVG` , `PNG_FILTER_PAETH` . `PNG_NO_FILTER` or `PNG_ALL_FILTERS` may also be used to respectively disable or activate all filters.

JPEG format

```
$img->render( 'jpeg' [, int $quality = 75 ] );
```

`$quality` ranges from 0 (worst quality, smaller file) to 100 (best quality, biggest file).

GIF format

```
$img->render( 'gif' );
```

WBMP format

```
$img->render( 'wbmp' [, int $foreground ] );
```

You can set the foreground color with `$foreground` by setting an identifier obtained from imagecolorallocate() (http://php.net/manual/en/function.imagecolorallocate.php). The default foreground color is black.

NB: internally this method is a wrapper for the following native PHP methods:

- imagepng (http://www.php.net/manual/en/function.imagepng.php)
- imagejpeg (http://www.php.net/manual/en/function.imagejpeg.php)
- imagegif (http://www.php.net/manual/en/function.imagegif.php)
- imagewbmp (http://www.php.net/manual/en/function.imagewbmp.php)

dump

Return image as a string

```
$img->dump();
```

This method accepts the same arguments as the `render()` (image#render) method above.

You can write the result of this method to a file:

```
$f3->write( '/path/to/file.png', $img->dump('png',9) );
```

History

The next methods only take effect when the `$flag` argument of the constructor (image#instantiation) was set to `TRUE`.

save

Save current state

```
$img->save();
```

This will create a new temporary image of the current state.

restore

Revert to specified state

```
$img->restore( [ int $state = 1 ] );
```

This fetches the original image state from the temp folder.

undo

Undo most recently applied filter

```
    $img->undo();
```