Глава 6. Графика, Flash, pdf

PHP позволяет выводить не только HTML-текст, но и создавать графические данные в различных форматах (gif, png, jpg, wbmp, xpm). Чтобы PHP мог работать с графикой, требуется библиотека GD. Возможно также потребуются дополнительные библиотеки для поддержки особенных форматов. Начиная с версии 1.6, GD не поддерживает работу с Gif форматом; вместо него используется формат png.

Библиотека GD library доступна по адресу http://www.boutell.com/gd/.

Для работы с рисунками jpeg используется библиотека jpeg-6b (ftp://ftp.uu.net/graphics/jpeg/), которую следует прикомпилировать с Gd, используя опцию компиляции PHP --with-jpeg-dir=/path/to/jpeg-6b. Для поддержки шрифтов Type 1, инсталлируйте t1lib (ftp://ftp.neuroinformatik.ruhr-uni-bochum.de/pub/software/t1lib/), используя опцию --with-t1lib[=dir].

Следующий пример (скрипт button.php) вызывается со страницы тегом вида: `<img src=«button.php?text»>` (в данном случает на рисунке «images/button1.png» будет выведена надпись «text» и затем этот рисунок будет отослан браузеру). Это более удобный способ, использующий готовый макет рисунка, не требующий его перерисовки с начала, но добавляющий в него дополнительные элементы.

```
<?php  // файл button.php
    Header("Content-type: image/png");
    $string=implode($argv," ");
    $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);
?>
```

ImageTypes. Определение графических форматов поддерживаемых PHP

```
int imagetypes(void);
```

Возвращает битовая маска графических форматов, поддерживаемых используемой библиотекой GD: `IMG_GIF | IMG_JPG | IMG_PNG | IMG_WBMP`.

```
<?php
   if (ImageTypes() & IMG_PNG) {  echo "формат PNG поддерживается"; }
?>
```

ImageCreate. Создание рисунка

```
int imagecreate (int x_size, int y_size)
```

Возвращает дескриптор ой картинки размером `x_size` на `y_size`.

```
<?php    header ("Content-type: image/png");
$im = @ImageCreate (50, 100) or die ("Нельзя создать GD image ");
$background_color = ImageColorAllocate ($im, 255, 255, 255);
$text_color = ImageColorAllocate ($im, 233, 14, 91);
ImageString ($im, 1, 5, 5,  "A Simple Text String", $text_color);
ImagePng ($im);  // передать рисунок браузеру
?>
```

ImageCreateFromGIF. Создание рисунка из файла GIF

```
int imagecreatefromgif (string filename)
```

Возвращает дескриптор рисунка `filename` (можно указать URL); или пустую строку при ошибке (с выдачей предупреждения).

```
function LoadGif ($imgname) {
    $im = @ImageCreateFromGIF ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im = ImageCreate (150, 30); /* Create a blank image */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        ImageString($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Так как GD начиная с версии 1.6 не поддерживает GIF, функция недоступна для этих версий.

ImageCreateFromJPEG. Создание рисунка из файла JPEG

```
int imagecreatefromjpeg (string filename)
```

Возвращает дескриптор рисунка `filename` (можно указать URL); или пустую строку при ошибке (с выдачей предупреждения).

ImageCreateFromPNG. Создание рисунка из файла PNG

```
int imagecreatefrompng (string filename)
```

Возвращает дескриптор рисунка `filename` (можно указать URL); или пустую строку при ошибке (с выдачей предупреждения).

GetImageSize. Получение свойств рисунка GIF, JPEG, PNG, SWF

```
array getimagesize (string filename [, array imageinfo])
```

Возвращает массив, содержащий информацию о графическом файле (4 элемента): [0] и [1]– ширина и высота рисунка в пикселях, [2] – тип файла (1 = GIF, 2 = JPG, 3 =

PNG, 4 = SWF), [3] – строка вида «height=xxx width=xxx», которая может использоваться непосредственно в HTML теге IMG.

```
<?php $size = GetImageSize ("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>
```

Указав необязательный массив imageinfo, в нем можно сохранить дополнительную информацию о файле, например различные маркеры JPG APP (внедренную информацию). Часто в маркер APP13 заносят информацию IPTC http://www.iptc.org/. Функция iptcparse() позволяет конвертировать эти данные APP13 в читаемый вид.

```
<?php     $size = GetImageSize ("testimg.jpg",&$info);
    if (isset ($info["APP13"])) {
        $iptc = iptcparse ($info["APP13"]);
        var_dump ($iptc);
    }
?>
```

Функция не требует библиотеки GD.

iptcparse. Интерпретация двоичного блока IPTC

```
array iptcparse (string iptcblock)
```

Функция извлекает данные из маркеров APP (обычно они содержаться в файлах интернет графики) и возвращает полученную информацию в массиве. См. пример в функции GetImageSize(). См. информацию по адресу: http://www.iptc.org/.

read_exif_data. Чтение заголовков EXIF из файла JPEG

```
array read_exif_data (string filename)
```

Возвращает ассоциативный массив, в котором индексами являются имена заголовков Exif. Заголовки Exif обычно хранят информацию цифровых камер (в разном виде).

```
<?php
 $exif = read_exif_data ('p0001807.jpg');
 while(list($k,$v)=each($exif)) {
   echo "$k: $v<br>\n";
 }
?>

Output:
FileName: p0001807.jpg
FileDateTime: 929353056
FileSize: 378599
CameraMake: Eastman Kodak Company
CameraModel: KODAK DC265 ZOOM DIGITAL CAMERA (V01.00)
DateTime: 1999:06:14 01:37:36
Height: 1024
```

```
Width: 1536
IsColor: 1
FlashUsed: 0
FocalLength:  8.0mm
RawFocalLength: 8
ExposureTime:  0.004 s (1/250)
RawExposureTime: 0.0040000001899898
ApertureFNumber: f/ 9.5
RawApertureFNumber: 9.5100002288818
FocusDistance: 16.66m
RawFocusDistance: 16.659999847412
Orientation: 1
ExifVersion: 0200
```

Функция доступна, только если PHP 4 компилировался с опцией --enable-exif. Библиотека GD не требуется.

ImageSX. Определение ширины рисунка

```
int imagesx (int im)
```

См. также: `ImageCreate()` и `ImageSY()`.

ImageSY. Определение высоты рисунка

```
int imagesy (int im)
```

См. также: `ImageCreate()` и `ImageSX()`.

ImageInterlace. Установка черезстрочности

```
int imageinterlace (int im [, int interlace])
```

Если `interlace` = 1, то это делает рисунок отображаемым черезстрочно, а 0 - последовательно. Возвращается текущая установка черезстрочности.

ImageGIF. Отсылка рисунка GIF браузеру или сохранение его в файле

```
int imagegif (int im [, string filename])
```

Создает файл GIF из рисунка `im`. Затем файл сохраняется под именем `filename`, или если имя не указано, то содержимое файла передается непосредственно браузеру (не забываете предварительно отослать браузеру заголовок, сообщающий, что передается графический файл). Файл имеет формат GIF87a, а если он был сделан прозрачным функцией `ImageColorTransparent()`, то формат будет GIF89a.

Так как GD, начиная с версии 1.6 не поддерживает GIF, функция недоступна для этих версий.

Следующий пример позволяет выводить рисунок в зависимости от того, какой формат поддерживается.

```
<?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);
  }
  else
    die("No image support in this PHP server");
?>
```

Используйте функцию `imagetypes()` вместо `function_exists()` для проверки поддержки различных форматов:

```
if (ImageTypes() & IMG_GIF) {
    Header("Content-type: image/gif");
    ImageGif($im);
}
elseif (ImageTypes() & IMG_JPG) {
... etc.
```

См. также: `ImagePng()`, `ImageJpeg()`, `ImageTypes()`.

ImagePNG. Отсылка рисунка PNG браузеру или сохранение его в файле

```
int imagepng (int im [, string filename])
```

Выводит GD рисунок `im` в формате PNG на стандартный поток вывода (обычно браузер), или если указано имя файла `filename`, то в этот файл.

```
<?php
$im = ImageCreateFromPng("test.png");
ImagePng($im);
?>
```

См. также: `ImageGif()`, `ImageJpeg()`, `ImageTypes()`.

ImageJPEG. Отсылка рисунка JPEG браузеру или сохранение его в файле

```
int imagejpeg (int im [, string filename [, int quality]])
```

Выводит рисунок im в формате JPEG на стандартный поток вывода (обычно браузер), или если указано имя файла filename, то сохраняет в этот файл. Аргументом quality можно указать качество изображение (обратно пропорционально размеру файла) от 0 до 100.

```
Imagejpeg($im,'',20);  // вывести рисунок
```

Поддержка JPEG была добавлена в GD-1.8 и более поздние версии.

См. также: ImagePng(), ImageGif(), ImageTypes().

ImageDestroy. Уничтожение рисунка

int **imagedestroy** (int im)

ImageCopy. Копирование части рисунка

int **ImageCopy** (int dst_im, int src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h)

Копирует прямоугольную область src_x, src_y, src_w (ширина), src_h (высота) рисунка src_im в рисунок dst_im со смещением dst_x, dst_y.

ImageCopyResized. Копирование части рисунка с масштабированием

int **imagecopyresized** (int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH)

Функция аналогична ImageCopy(), но область рисунка масштабируется, так что его ширина и высота становится равна dstW и dstH.

Цвета и палитра

ImageColorAllocate. Выделение цвета для рисунка

int **imagecolorallocate** (int im, int red, int green, int blue)

Возвращает индекс добавленного в палитру цвета, составленного из компонент: red (красный), green (зеленый), blue (синий).

```
$white = ImageColorAllocate ($im, 255, 255, 255); // белый
$black = ImageColorAllocate ($im, 0, 0, 0); // черный
```

ImageColorDeAllocate. Исключение цвета из палитры рисунка

int **imagecolordeallocate** (int im, int index)

```
$white = ImageColorAllocate($im, 255, 255, 255);
ImageColorDeAllocate($im, $white);
```

ImageColorSet. Замена цвета определенного элемента палитры

`bool` **`imagecolorset`** `(int im, int index, int red, int green, int blue)`

Устанавливает для элемента палитры `index` рисунка `im` значения компонент цвета: `red` (красный), `green` (зеленый), `blue` (синий). При этом все части рисунка закрашенные данным цветом также изменят свой оттенок.

См. также: `imagecolorat()`.

ImageColorsForIndex. Получение цвета элемента палитры

`array` **`imagecolorsforindex`** `(int im, int index)`

Возвращает ассоциативный массив трех элементов: `red`, `green`, `blue` содержащий значение компонент цветности указанного цвета `index`.

См. также: `imagecolorat()` и `imagecolorexact()`.

ImageColorExact. Получение индекса цвета палитры

`int` **`imagecolorexact`** `(int im, int red, int green, int blue)`

Если точно соответствующего цвета в палитре не имеется, то возвращается -1.

См. также: `imagecolorclosest()`.

ImageColorClosest. Получение цвета палитры наиболее близкого к указанному

`int` **`imagecolorclosest`** `(int im, int red, int green, int blue)`

Возвращает индекс цвета палитры рисунка. Близость вычисляется согласно RGB.

См. также: `imagecolorexact()`.

ImageColorResolve. Нахождение или создание указанного цвета

`int` **`imagecolorresolve`** `(int im, int red, int green, int blue)`

Возвращает индекс указанного цвета в палитре. Если такой цвет отсутствует, он создается. См. также: `imagecolorclosest()`.

ImageGammaCorrect. Применение гамма коррекции рисунка

`int` **`imagegammacorrect`** `(int im, double inputgamma, double outputgamma)`

ImageColorsTotal. Получение числа цветов в палитре

`int` **`imagecolorstotal`** `(int im)`

См. также: `imagecolorat()` и `imagecolorsforindex()`.

ImageColorTransparent. Определение цвета прозрачности

`int` **`imagecolortransparent`** `(int im [, int col])`

Устанавливает (возвращает) индекс цвета палитры, используемого как прозрачный. Возвращает индекс текущего (или установленного) «прозрачного» цвета.

ImageColorAt. Получение индекса цвета пикселя

`int` **`imagecolorat`** `(int im, int x, int y)`

См. также: `imagecolorset()` и `imagecolorsforindex()`.

Черчение фигур

ImageSetPixel. Установка пикселя

`int` **`imagesetpixel`** `(int im, int x, int y, int color)`

Верхний левый угол рисунка имеет координаты 0, 0.

См. также: `ImageCreate()` и `ImageColorAllocate()`.

ImageLine. Начертить линию

`int` **`imageline`** `(int im, int x1, int y1, int x2, int y2, int color)`

ImageDashedLine. Начертить прерывистую линию

`int` **`imagedashedline`** `(int im, int x1, int y1, int x2, int y2, int col)`

См. также: `ImageLine()`.

ImageRectangle. Начертить прямоугольник

`int` **`imagerectangle`** `(int im, int x1, int y1, int x2, int y2, int col)`

Левый верхний угол задается x1, y1, а правый нижний x2, y2.

ImageFilledRectangle. Зарисовка прямоугольной области

`int` **`imagefilledrectangle`** `(int im, int x1, int y1, int x2, int y2, int col)`

ImagePolygon. Начертить многоугольник

`int` **`imagepolygon`** `(int im, array points, int num_points, int col)`

Аргумент `points` должен содержать координаты углов в виде: points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, и т. д. `num_points` задает число углов.

См. также: `imagecreate()`.

ImageFilledPolygon. Закраска многоугольника

```
int imagefilledpolygon (int im, array points, int num_points, int col)
```

ImageArc. Нарисовать часть эллипса

```
int imagearc (int im, int cx, int cy, int w, int h, int s, int e, int col)
```

`cx, cy` задают центр эллипса, `w` и `h` – ширину и высоту, `s` и `e` – начальный и конечный угол в градусах, `col` - цвет.

ImageFill. Заливка цветом ограниченной области

```
int imagefill (int im, int x, int y, int col)
```

ImageFillToBorder. Заливка области ограниченной бордюром

```
int imagefilltoborder (int im, int x, int y, int border, int col)
```

Индекс цвета окантовки задается аргументом `border`, а заливки – `col`. Заливка начинается от точки с координатами (x,y).

Шрифты и надписи

ImageLoadFont. Загрузка шрифта

```
int imageloadfont (string file)
```

Загружает шрифт из файла `file` и возвращает назначенный ему идентификатор. Встроенные шрифты имеют идентификаторы 1-5 (меньший-больший).

Формат файла зависит от системы. В общем случае он имеет следующую структуру:

Байты	Тип	Описание
0-3	int	Число символов в файле
4-7	int	Код первого символа (обычно 32 – пробел)
8-11	int	Ширина символов в пикселях
12-15	int	Высота
16-…	int	Массив данных символов (байт на пиксель для каждого символа). Общий размер: (nchars*высота*ширина) байт.

См. также: `ImageFontWidth()` и `ImageFontHeight()`.

ImageFontHeight. Высота шрифта

```
int imagefontheight (int font)
```

См. также: `ImageFontWidth()` и `ImageLoadFont()`.

ImageFontWidth. Ширина шрифта

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

См. также: `ImageFontHeight()` и `ImageLoadFont()`.

ImageChar. Вывод символа горизонтально

`int` **`imagechar`** `(int im, int font, int x, int y, string c, int col)`

Шрифт указывается его номером `font` (1-5 – встроенные шрифты).

См. также: `imageloadfont()`.

ImageCharUp. Вывод символа вертикально

`int` **`imagecharup`** `(int im, int font, int x, int y, string c, int col)`

См. также: `imageloadfont()`.

ImageString. Вывод строки горизонтально

`int` **`imagestring`** `(int im, int font, int x, int y, string s, int col)`

См. также: `ImageLoadFont()`.

ImageStringUp. Вывод строки вертикально

`int` **`imagestringup`** `(int im, int font, int x, int y, string s, int col)`

См. также: `ImageLoadFont()`.

ImageTTFBBox. Расчет площади занимаемой строкой шрифта TypeType

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

Аргументы, от которых зависит результат функции:

`text` – Строка текста

`size` – Размер шрифта в пикселях.

`Fontfile` – Имя файла содержащего шрифт TrueType. (Можно указать URL)

`angle` – Угол наклона текста.

Возвращает массив 8 элементов, определяющих прямоугольник вывода текста:

0 – нижний левый угол, X координата

1 – нижний левый угол, Y координата

2 – нижний правый угол, X координата

3 – нижний правый угол, Y координата

4 – верхний правый угол, X координата

5 – верхний правый угол, Y координата

6 – верхний левый угол, X координата

7 – верхний левый угол, Y координата

Возвращаемые координаты зависят от угла наклона строки (точкой отчета служит нижний левый угол первого символа). Координаты могут иметь отрицательные значения. Функции требует библиотеки GD и FreeType.

См. также: `ImageTTFText()`.

ImageTTFText. Нарисовать текст шрифтом TrueType

`array` **`imagettftext`** `(int im, int size, int angle, int x, int y, int col, string fontfile, string text)`

Выводит строку `text` в рисунок `im`, в координаты `x, y` (верхний левый угол является началом координат), под углом `angle` (задается в градусах) цветом `col`, используя шрифт TrueType из файла `fontfile`.

Аргументами `x, y` задается базовая точка первого символа (нижний левый угол символа); что отличает данную функцию от `ImageString()`, где координаты x, y определяют верхний правый угол символа.

Строка текста может включать символьные последовательности UTF-8 (в виде: {) для вывода символов с кодами большими 255.

При использовании отрицательного значения индекса цвета отключается сглаживание шрифта (antialiasing).

Возвращает массив 8 элементов, определяющих прямоугольник вывода текста подобно функции `imagettfbbox()`.

Функции требует библиотеки GD и FreeType.

```
<?php Header ("Content-type: image/gif");
   $im = imagecreate (400, 30);
   $black = ImageColorAllocate ($im, 0, 0, 0);
   $white = ImageColorAllocate ($im, 255, 255, 255);
   ImageTTFText ($im, 20, 0, 10, 20, $white, "/path/arial.ttf",
              "Testing... Omega: &#937;");
   ImageGif ($im);
   ImageDestroy ($im);
?>
```

Следующий пример выводит строку по центру рисунка:

```
<?
$gi = ImageCreate(200,100);
$bg = ImageColorAllocate($gi,0,220,0);
$tx = ImageColorAllocate($gi,25,2,228);
```

```
$w=imagesx($gi); // ширина рисунка
$h=imagesy($gi); // высота
ImageFilledRectangle($gi,0,0,$w,$h,$bg);

$szf=20;  // размер шрифта
$ang=240; // угол поворота строки
$str="Heyou";  // текст строки
$font="symbol.ttf"; // файл шрифта
$sz=imagettfbbox($szf,$ang,$font,$str);
$sdx=$sz[4]/2;
$sdy=($sz[7]+$sz[3])/2;
imagettftext($gi,$szf,$ang,$w/2-$sdx,$h/2-$sdy,$tx,$font,$str);
//header("content-type: image/png");
ImagePng($gi,"n2.png");
?>
```

См. также: `ImageTTFBBox()`.

ImagePSLoadFont. Загрузка из файла шрифта PostScript Type 1

`int` **`imagepsloadfont`** `(string filename)`

Возвращает дескриптор загруженного шрифта, или `false` при ошибке (также выводится предупреждение). См. также: `ImagePSFreeFont()`.

ImagePSFreeFont. Выгрузка шрифта PostScript Type 1

`void` **`imagepsfreefont`** `(int fontindex)`

См. также: `ImagePSLoadFont()`.

ImagePSEncodeFont. Установка схемы перекодировки текста

`int` **`imagepsencodefont`** `(int font_index, string encodingfile)`

Загружает файл перекодировки `encodingfile` для шрифта `font_index`. Поскольку шрифты PostScript по умолчанию не используют символы с кодами большими 127, перекодировка требуется при необходимости использования не-английского языка. Формат файла описан в документации T1libs, также с библиотекой поставляются 2 готовых файла: IsoLatin1.enc и IsoLatin2.enc.

Если перекодировка используется постоянно, установите опцию `ps.default_encoding` в файле конфигурации со значением имени файла перекодировки, который будет загружаться автоматически.

ImagePsExtendFont. Масштабирование шрифт

`bool` **`imagepsextendfont`** `(int font_index, double extend)`

ImagePsSlantFont. Установка наклона шрифта

```
bool imagepsslantfont (int font_index, double slant)
```

ImagePSBBox. Расчет площади занимаемой строкой шрифта PostScript Type1

```
array imagepsbbox (string text, int font, int size [, int space [, int tightness [, float angle]]])
```

Расчеты производятся на основании аргументов: `Size` – размер шрифта в пикселях; `Space` – изменение размера пробелов по отношению к нормальному (может быть отрицательным); `Tightness` – промежутки между символами по отношению к нормальному (может быть отрицательным); `Angle` – угол наклона строки в градусах; Значения `space` и `tightness` измеряются в долях пробела (1/1000).

Аргументы `space, tightness, angle` не обязательны.

Результаты расчета не достаточно точны. Функция возвращает массив:

0 – нижний левый угол x-координата

1 – нижний левый угол y-координата

2 – верхний правый угол x-координата

3 – верхний правый угол y-координата

См. также: `imagepstext()`.

ImagePSText. Вывод текста поверх рисунка шрифтом PostScript Type1

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

Координаты `x, y` указывают левый нижний угол первого символа.

Аргументами `Foreground` и `Background` задаются цвета текста и фона (фон необходим только для сглаживания шрифта). Аргумент `Antialias_steps` позволяет указать число цветов используемых при сглаживании текста (допустимые значения 4 и 16). Для шрифтов размером меньше 20 используйте большее значение, так как это улучшает читабельность; для больших шрифтов используйте меньшее значение так как это увеличивает быстродействие.

Функция возвращает массив подобно `imagepsbbox()`. См. также: `imagepsbbox()`.

Shockwave Flash

PHP позволяет создавать файлы Shockwave Flash посредством библиотеки (созданной Paul Haeberli) libswf, которую можно загрузить с http://reality.sgi.com/grafica/flash/. Прикомпилировать libswf к PHP можно опцией **--with-**

swf[=DIR], где DIR. место размещение директорий include (содержит файл swf.h) и lib (содержит файл libswf.a).

Следующий пример демонстрирует создание Flash анимации:

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}

swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();
```

```
swf_closefile ();
?>
```

Просмотреть результат исполнения скрипта можно по адресу: http://www.designmultimedia.com/swfphp/test.swf.

libswf не поддерживается в Windows. Разработка библиотеки была прекращена, а исходные коды недоступны.

swf_openfile. Открытие нового файла Shockwave Flash

`void` **`swf_openfile`** `(string filename, float width, float height, float framerate, float r, float g, float b)`

Создает файл `filename` с анимацией (шириной `width`, высотой `height`, частотой кадров `framerate`, и цветом фона `R,G,B`).

Данная функция должна вызываться первой, иначе в скрипте возникнет ошибка «segfault». Если необходимо непосредственно отсылать файл браузеру, можно задать его имя как: «`php://stdout`».

swf_closefile. Закрытие текущего файла Shockwave Flash

`void` **`swf_closefile`** `([int return_file])`

Закрывает файл, открытый функцией `swf_openfile()`. При установке аргумента `return_file` функция возвращает содержимое SWF файла.

```
// создает flash файл; выводит его и сохраняет в БД
<?php

// $text - аргумент скрипта

// Параметры БД (для swf_savedata())
$DBHOST = "localhost";
$DBUSER = "user";
$DBPASS = "secret";

swf_openfile ("php://stdout", 256, 256, 30, 1, 1, 1);

    swf_definefont (10, "Ligon-Bold");
        swf_fontsize (12);
        swf_fontslant (10);

    swf_definetext (11, $text, 1);

    swf_pushmatrix ();
        swf_translate (-50, 80, 0);
        swf_placeobject (11, 60);
```

```
    swf_popmatrix ();

    swf_showframe ();

    swf_startdoaction ();
        swf_actionstop ();
    swf_enddoaction ();

$data = swf_closefile (1);

$data ?
  swf_savedata ($data) :
  die ("Невозможно сохранить SWF файл");

// сохранить файл в БД
function swf_savedata ($data)
{
    global $DBHOST, $DBUSER, $DBPASS;

    $dbh = @mysql_connect ($DBHOST, $DBUSER, $DBPASS);

    if (!$dbh) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    $stmt = "INSERT INTO swf_files (file) VALUES ('$data')";

    $sth = @mysql_query ($stmt, $dbh);

    if (!$sth) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }
    @mysql_free_result ($sth);
    @mysql_close ($dbh);
}
>
```

swf_labelframe. Пометка текущего кадра

```
void swf_labelframe (string name)
```

swf_showframe. Отображение текущего кадра

```
void swf_showframe (void);
```

swf_setframe. Переключение кадра

```
void swf_setframe (int framenumber)
```

swf_getframe. Получение номера текущего кадра

```
int swf_getframe (void);
```

swf_mulcolor. Установка множителя цвета rgba

```
void swf_mulcolor (float r, float g, float b, float a)
```

Функция устанавливает значения (они могут быть положительными или отрицательными), на которые будут умножаться цвета выводимых объектов. Этот цвет затем (косвенно) используется функциями `swf_placeobject()`, `swf_modifyobject()` и `swf_addbuttonrecord()`.

swf_addcolor. Установка слагаемого цвета rgba

```
void swf_addcolor (float r, float g, float b, float a)
```

Функция устанавливает значения (они могут быть положительными или отрицательными), которые будут добавляться к цвету выводимых объектов. Этот цвет затем (косвенно) используется функциями `swf_placeobject()`, `swf_modifyobject()` и `swf_addbuttonrecord()`.

swf_placeobject. Размещение объекта в текущем кадре

```
void swf_placeobject (int objid, int depth)
```

Размещает объект `objid` на уровне `depth`. Аргументы могут иметь значения от 1 до 65535. При этом используются текущие: множитель и слагаемое цвета (`swf_mulcolor()` и `swf_addcolor()`) и матрица позиционирования объекта.

Полностью поддерживаются цвета RGBA.

swf_modifyobject. Изменение объекта

```
void swf_modifyobject (int depth, int how)
```

Изменяет объект, расположенный на уровне `depth`, согласно аргументу `how`: если он имеет значение `MOD_MATRIX`, то изменяется положение объекта, если `MOD_COLOR`, то изменяется цвет. Эти константы можно комбинировать: (`MOD_MATRIX|MOD_COLOR`).

`MOD_COLOR` использует текущие: множитель и слагаемое цвета (`swf_mulcolor()` и `swf_addcolor()`); а `MOD_MATRIX` – матрицу позиционирования объекта.

swf_removeobject. Удаление объекта

```
void swf_removeobject (int depth)
```

Удаляет объект, расположенный на уровне `depth`.

swf_nextid. Получение идентификатора следующего свободного объекта

```
int swf_nextid (void);
```

swf_startdoaction. Начать описание списка действий текущего кадра

```
void swf_startdoaction (void);
```

Функцию следует вызывать после завершения определения действий.

swf_actiongotoframe. Проиграть кадр и остановиться

```
void swf_actiongotoframe (int framenumber)
```

swf_actiongeturl. Получение URL анимации Shockwave Flash

```
void swf_actiongeturl (string url, string target)
```

swf_actionnextframe. Перемещение вперед на один кадр

```
void swf_actionnextframe (void);
```

swf_actionprevframe. Перемещение назад на один кадр

```
void swf_actionprevframe (void);
```

swf_actionplay. Начать проигрывание анимации с текущего кадра

```
void swf_actionplay (void);
```

swf_actionstop. Остановка flash анимации текущего кадра

```
void swf_actionstop (void);
```

swf_actiontogglequality. Переключение между высоким / низким качеством

```
void swf_actiontogglequality (void);
```

swf_actionwaitforframe. Пропуск действия если кадр не был загружен

```
void swf_actionwaitforframe (int framenumber, int skipcount)
```

Функция проверяет видимость кадра `framenumber`, и если он не был загружен, пропускает указанное число действий `skipcount`. Полезно для анимаций типа «Loading...".

swf_actionsettarget. Установка контекста всех действий

`void` **`swf_actionsettarget`** `(string target)`

Используется для управления другими flash анимациями проигрываемыми в текущий момент.

swf_actiongotolabel. Отображение кадра с указанной меткой

`void` **`swf_actiongotolabel`** `(string label)`

После отображения проигрывание останавливается.

swf_enddoaction. Завершение текущего действия

`void` **`swf_enddoaction`** `(void);`

Завершает текущее действие, начатое функцией `swf_startdoaction()`.

swf_defineline. Создание линии

`void` **`swf_defineline`** `(int objid, float x1, float y1, float x2, float y2, float width)`

Рисует линию ((x1, y1) – (x2,y2)) шириной `width`.

swf_definerect. Создание прямоугольника

`void` **`swf_definerect`** `(int objid, float x1, float y1, float x2, float y2, float width)`

Верхний левый угол задается (x1, y1), левый нижний (x2, y2). Ширина `width` (если она равна 0.0 прямоугольник будет закрашен).

swf_definepoly. Создание многоугольника

`void` **`swf_definepoly`** `(int objid, array coords, int npoints, float width)`

Углы задаются координатами x, y в массиве `coords`. Число углов - `npoints`. Ширина `width` (если она равна 0.0 многоугольник будет закрашен).

swf_startshape. Начать сложную фигуру

`void` **`swf_startshape`** `(int objid)`

swf_shapelinesolid. Установка текущего стиля линии

`void` **`swf_shapelinesolid`** `(float r, float g, float b, float a, float width)`

Задаются параметры цвета `rgba` и ширина `width` (если указывается ширина 0.0, то линии рисоваться не будут).

swf_shapefilloff. Выключение заполнения текущей фигуры

```
void swf_shapefilloff (void);
```

swf_shapefillsolid. Установка стиля и цвета заполнения

```
void swf_shapefillsolid (float r, float g, float b, float a)
```

Устанавливает 100%-ый стиль заполнения цветом `rgba`.

swf_shapefillbitmapclip. Установка заполнения усеченной текстурой рисунка

```
void swf_shapefillbitmapclip (int bitmapid)
```

swf_shapefillbitmaptile. Установка заполнения размноженной текстурой рисунка

```
void swf_shapefillbitmaptile (int bitmapid)
```

swf_shapemoveto. Перемещение текущей позиции

```
void swf_shapemoveto (float x, float y)
```

swf_shapelineto. Начертить линию

```
void swf_shapelineto (float x, float y)
```

Чертит линию от текущей позиции до указанной.

swf_shapecurveto. Нарисовать квадратическую кривую безье

```
void swf_shapecurveto (float x1, float y1, float x2, float y2)
```

Чертит кривую от точки (x1, y1) до точки (x2,y2). Текущая позиция затем устанавливается в точке (x2,y2).

swf_shapecurveto3. Нарисовать кубическую кривую безье

```
void swf_shapecurveto3 (float x1, float y1, float x2, float y2, float x3, float y3)
```

Чертит кривую, используя точки (x1, y1) и (x2,y2) как внешне ориентирующие. А точку (x3,y3) как конечную. Текущая позиция затем устанавливается в точке (x3,y3).

swf_shapearc. Нарисовать дугу

```
void swf_shapearc (float x, float y, float r, float ang1, float ang2)
```

Начальный и конечный угол дуги задается `ang1` и `ang2`; центр (`x`, `y`); радиус `r`.

swf_endshape. Завершение определения текущей фигуры

```
void swf_endshape (void);
```

swf_definefont. Определение шрифта

```
void swf_definefont (int fontid, string fontname)
```

Задает шрифт `fontname` в качествен текущего и присваивает ему идентификатор `fontid`.

swf_setfont. Смена текущего шрифта

```
void swf_setfont (int fontid)
```

swf_fontsize. Смена размера текущего шрифта

```
void swf_fontsize (float size)
```

swf_fontslant. Установка наклона текущего шрифта

```
void swf_fontslant (float slant)
```

Угол задается значением `slant` (положительное значение создает наклон вперед, отрицательное – назад).

swf_fonttracking. Установка интервала между буквами

```
void swf_fonttracking (float tracking)
```

Положительное значение увеличивает, а отрицательное – уменьшает интервал.

swf_getfontinfo. Высота букв

```
array swf_getfontinfo (void);
```

Возвращает ассоциативный массив двух элементов:

- `Aheight` – высота в пикселях заглавной буквы A.
- `xheight` – высота в пикселях маленькой буквы x.

swf_definetext. Создание строки текста

```
void swf_definetext (int objid, string str, int docenter)
```

Строка текста `str` создается с использованием текущего шрифта и его размера. Если в значении аргумента `docenter` указывается 1, то текст центрируется по горизонтали.

swf_textwidth. Расчет ширины строки

```
float swf_textwidth (string str)
```

Расчет использует размеры текущего шрифта.

swf_definebitmap. Определение рисунка

```
void swf_definebitmap (int objid, string image_name)
```

Добавляет рисунок GIF, JPEG, RGB или FI. (он автоматически конвертируется в формат Flash JPEG или Flash color map).

swf_getbitmapinfo. Получение информации о рисунке

```
array swf_getbitmapinfo (int bitmapid)
```

Возвращает массив с информацией о рисунке `bitmapid`, содержащий элементы:

- «size» – размер рисунка в байтах.
- «width» – ширина рисунка в пикселях.
- «height» – высота рисунка в пикселях.

swf_startsymbol. Определения символа

```
void swf_startsymbol (int objid)
```

Превращает объект в символ. Символы – это микро анимации flash которые могут проигрываться одновременно.

swf_endsymbol. Завершение определения символа

```
void swf_endsymbol (void);
```

Завершает определение символа, начатое функцией `swf_startsymbol()`.

swf_startbutton. Начать определения кнопки

```
void swf_startbutton (int objid, int type)
```

Аргументом `type` задается: может ли фокус перемещаться при нажатой кнопки мыши (`TYPE_MENUBUTTON`) или нет (`TYPE_PUSHBUTTON`).

swf_addbuttonrecord. Управление расположением, видом и активной областью текущей кнопки

```
void swf_addbuttonrecord (int states, int shapeid, int depth)
```

Аргументом `states` определяется, какие состояния может принимать кнопка: `BSHitTest`, `BSDown`, `BSOver`, `BSUp`; аргументом `shapeid` определяется

внешний вид кнопки (идентификатор объекта), а `depth` – уровень расположения кнопки.

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_oncondition. Назначение событие

```
void swf_oncondition (int transition)
```

Для кнопок `TYPE_MENUBUTTON` возможны опции:

- `IdletoOverUp`
- `OverUptoIdle`
- `OverUptoOverDown`
- `OverDowntoOverUp`
- `IdletoOverDown`
- `OutDowntoIdle`
- `MenuEnter (IdletoOverUp|IdletoOverDown)`
- `MenuExit (OverUptoIdle|OverDowntoIdle)`

Для кнопок `TYPE_PUSHBUTTON` возможны опции:

- `IdletoOverUp`
- `OverUptoIdle`
- `OverUptoOverDown`
- `OverDowntoOverUp`
- `OverDowntoOutDown`
- `OutDowntoOverDown`
- `OutDowntoIdle`
- `ButtonEnter (IdletoOverUp|OutDowntoOverDown)`
- `ButtonExit (OverUptoIdle|OverDowntoOutDown)`

swf_endbutton. Завершение определения текущей кнопки

```
void swf_endbutton (void);
```

swf_viewport. Выбор области для последующего рисования

```
void swf_viewport (double xmin, double xmax, double ymin, double ymax)
```

swf_ortho. Выбор объемной системы координат текущей области рисования

```
void swf_ortho (double xmin, double xmax, double ymin, double ymax, double zmin, double zmax)
```

swf_ortho2. выбор плоской системы координат текущей области рисования

```
void swf_ortho2 (double xmin, double xmax, double ymin, double ymax)
```

Для преобразований перспективы может использоваться функция `swf_perspective()`.

swf_perspective. Определение трансформации перспективы проекции

```
void swf_perspective (double fovy, double aspect, double near, double far)
```

Аргумент `fovy` задает угол зрения по отношению к оси y; `aspect` – масштаб текущей области рисования. Аргументы `near,far` определяют границы проекции: ближнюю и дальнюю. Поскольку Flash проигрыватели имеют только двумерную матрицу, возможны искажения проецирования.

swf_polarview. Установка позиции наблюдения в полярных координатах

```
void swf_polarview(double dist, double azimuth, double incidence, double twist)
```

`dist` задает расстояние от наблюдателя до начала системы координат; `azimuth` – угол азимута в плоскости x,y измеренный от оси y; `incidence` – угол обзора в плоскости y,z, измеренный относительно оси z; `twist` – вращение видимой области относительно линии наблюдения (по правилу правой руки).

swf_lookat. Установка трансформации наблюдения

```
void swf_lookat (double view_x, double view_y, double view_z, double reference_x, double reference_y, double reference_z, double twist)
```

Аргументами `view_x, view_y, view_z` задается позиция наблюдения; `reference_x, reference_y, reference_z` указывают наблюдаемую точку; а `twist` – угол поворота наблюдения относительно оси z.

swf_pushmatrix. Занесение текущей матрицы трансформации в стек

```
void swf_pushmatrix (void);
```

swf_popmatrix. Извлечение матрицы трансформации из стека

```
void swf_popmatrix (void);
```

swf_scale. Масштабирование текущей трансформации

```
void swf_scale (double x, double y, double z)
```

Масштабные коэффициенты задаются аргументами `x,y,z`.

swf_translate. Транслирование текущей трансформации

```
void swf_translate (double x, double y, double z)
```

swf_rotate. Поворот текущей трансформации

```
void swf_rotate (double angle, string axis)
```

Угол поворота `angle` задается относительно оси `axis`: 'x', 'y' или 'z'.

swf_posround. Разрешение или запрет округления координат объектов

```
void swf_posround (int round)
```

Значение 1 разрешает, а 0 запрещает округление.

Ming для Flash

Ming является библиотекой с открытым исходным кодом (LGPL), позволяющей создавать анимации SWF («Flash»). Ming поддерживает почти все возможности Flash 4, включая: фигуры (shapes), цветовые переходы (gradients), картинки (bitmaps: jpeg), превращения (morphs или «shape tweens»), текст, кнопки, действия, клипы (sprites), потоковую музыку mp3, и преобразования цветов; единственное чего не достает - звуковые события. Использование Ming предпочтительнее модуля libswf. Ming работает на большинстве платформ, включая Windows; использует PHP объекты для представления объектов SWF; и продолжает разрабатываться[1] (адрес разработчиков ming@opaque.net). Библиотеку Ming можно загрузить с сайта: http://www.opaque.net/ming/.

Все размеры указываются в единицах «твипс» (twips), что равно 1/20 пикселя. Но, масштаб может изменяться проигрывателем.
Ming использует PHP объекты тринадцати классов:

- `swfmovie()`
- `swfshape()`

---

[1] Модуль находится в стадии разработки (являясь для версии PHP 4.0.6 экспериментальным) и поэтому работает еще нестабильно. Также разработчики оставляют за собой право видоизменять программный интерфейс.

- swfdisplayitem()
- swfgradient()
- swfbitmap()
- swffill()
- swfmorph()
- swftext()
- swffont()
- swftextfield()
- swfsprite()
- swfbutton()
- swfaction()

SWFMovie – объект анимации SWF 4

SWFMovie->output. Вывод созданной анимации

void **swfmovie->output** (void)
Для того, чтобы браузер правильно воспринял передаваемый ему файл, используйте PHP команду отсылки заголовка до вывода анимации:

```
<?php header('Content-type: application/x-shockwave-flash');  ?>
```

См. также: swfmovie->save().

SWFMovie->save. Сохранение анимации в файле

void **swfmovie->save** (string filename)
См. также: output().

SWFMovie->add. Добавление компонент анимации

void **swfmovie->add** (resource instance)
Этим методом можно добавлять объекты различных типов: фигуры, текст, шрифты и т. п. Для отображаемых объектов (shape, text, button, sprite), возвращается дескриптор объекта SWFDisplayItem(), внесенный в список отображения. При многократном добавлении одного объекта каждый раз будет возвращаться другой дескриптор.
См. также: swfmovie->remove(), и пример в swfdisplayitem->rotateto().

SWFMovie->remove. Удаление объекта из списка отображения

void **swfmovie->remove** (resource instance)
Функция противоположна swfmovie->add().

SWFMovie->setbackground. Установка цвета фона

`void` **`swfmovie->setbackground`** `(int red, int green, int blue)`
Цвет задается значениями (0-255) компонент: красного, зеленого и синего.

SWFMovie->setdimension. Установка ширины и высоты анимации

`void` **`swfmovie->setdimension`** `(int width, int height)`

SWFMovie->setrate. Установка частоты кадров анимации

`void` **`swfmovie->setrate`** `(int rate)`
Значение `rate` число кадров в секунду. Анимация будет тормозиться, если проигрыватель не будет успевать перерисовывать кадры. Если одновременно проигрывается звук, то для его нормального воспроизведения частота перерисовки будет при необходимости снижаться.

SWFMovie->setframes. Установка общего числа кадров анимации

`void` **`swfmovie->setframes`** `(string number_of_frames)`

SWFMovie->nextframe. Переход к новому кадру анимации

`void` **`swfmovie->nextframe`** `(void)`

SWFMovie->streammp3. Воспроизведение потокового звука MP3

`void` **`swfmovie->streammp3`** `(string mp3FileName)`
Заметьте, что проигрываться будет не весь файл фонового звука целиком, а только та часть, которая успеет воспроизвестись за время воспроизведения установленного число кадров при текущей частоте кадров.

```
<?php
  $m = new SWFMovie();
  $m->setRate(10.0);  // кадров в секунду
  $m->streamMp3("distortobass.mp3");// use your own MP3

  // длительность проигрывания анимации: 100 / 10.0  = 10 секунд
  $m->setFrames(100); // кадров в анимации

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem – объект списка отображения

Используется для хранения компонентов анимации; после того, как объекты shape, text, sprite или button, *декларированы* (созданы экземпляры классов `swfshape()`, `swfbutton()`, `swftext()`, `swfsprite()`) и они *добавлены* в анимацию (методом `swfmovie->add()`), для каждого из них возвращается дескриптор (объект типа SWFDisplayItem), который можно использовать для перемещения, вращения, масштабирования и наклона объекта.

SWFDisplayItem->moveTo. Репозиционирование объекта

`void` **`swfdisplayitem->moveto`** `(int x, int y)`
Указываются абсолютные координаты.
См. также: `swfdisplayitem->move()`.

SWFDisplayItem->move. Смещение объекта

`void` **`swfdisplayitem->move`** `(int dx, int dy)`
Указываются координаты относительно текущей позиции.
См. также: `swfdisplayitem->moveto()`.

SWFDisplayItem->scaleTo. Задание новых размеров объекта

`void` **`swfdisplayitem->scaleto`** `(int x, int y)`
См. также: `swfdisplayitem->scale()`.

SWFDisplayItem->scale. Масштабирование объекта

`void` **`swfdisplayitem->scale`** `(int dx, int dy)`
Указываются коэффициенты масштабирования.
См. также: `swfdisplayitem->scaleto()`.

SWFDisplayItem->rotateTo. Установка абсолютного угла поворота объекта

`void` **`swfdisplayitem->rotateto`** `(double degrees)`
Угол указывается в градусах.

```
<?php // три вращающихся строки
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");

  $m = new SWFMovie();
  $m->setRate(24.0);
  $m->setDimension(2400, 1600);
  $m->setBackground(0xff, 0xff, 0xff);

  function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
  {
    global $f, $m;

    $t = new SWFText();
    $t->setFont($f);
    $t->setColor($r, $g, $b, $a);
    $t->setHeight(960);
    $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
    $t->addString($string);

    $i = $m->add($t);
    $i->x = $x;
```

```
    $i->y = $y;
    $i->rot = $rot;
    $i->s = $scale;
    $i->rotateTo($rot);
    $i->scale($scale, $scale);

    return $i;
 }

  function step($i)
  {
    $oldrot = $i->rot;
    $i->rot = 19*$i->rot/20;
    $i->x = (19*$i->x + 1200)/20;
    $i->y = (19*$i->y + 800)/20;
    $i->s = (19*$i->s + 1.0)/20;

    $i->rotateTo($i->rot);
    $i->scaleTo($i->s, $i->s);
    $i->moveTo($i->x, $i->y);

    return $i;
  }

  $i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
  $i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
  $i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

  for($i=1; $i<=100; ++$i)
  {
    $i1 = step($i1);
    $i2 = step($i2);
    $i3 = step($i3);

    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

См. также: `swfdisplayitem->rotate()`.

SWFDisplayItem->Rotate. Поворот объекта

`void` **`swfdisplayitem->rotate`** `(double ddegrees)`
Указывается относительный угол поворота в градусах.

См. также: swfdisplayitem->rotateto().

SWFDisplayItem->skewXTo. Установка наклона по X

void **swfdisplayitem->skewxto** (double d)
Указывается коэффициент смещения основания вправо, то есть 1.0, означает искажение в 45 градусов. Возможны отрицательные значения для смещения основания влево.
См. также: swfdisplayitem->skewx(), swfdisplayitem->skewy() и swfdisplayitem->skewyto().

SWFDisplayItem->skewX. Наклон объекта по X

void **swfdisplayitem->skewx** (double d)

SWFDisplayItem->skewYTo. Установка наклона по Y

void **swfdisplayitem->skewyto** (double degrees)
Указывается смещение правой стороны вниз.

SWFDisplayItem->skewY. Наклон объекта по Y

void **swfdisplayitem->skewy** (double ddegrees)

SWFDisplayItem->setDepth. Установка порядка отображения (z-order)

void **swfdisplayitem->setdepth** (double depth)
Размещает объект на указанном уровне. (При добавлении следующий объект размещается поверх предыдущего).

SWFDisplayItem->remove. Удаление объекта из анимации

void **swfdisplayitem->remove** (void)

SWFDisplayItem->setName. Присвоение объекту имя

void **swfdisplayitem->setname** (string name)
Используется для возможности обращения к объекту при написании программ действий. Полезно только для спрайтов.

SWFDisplayItem->setRatio. Установка увеличения объекта

void **swfdisplayitem->setratio** (double ratio)
Полезно только для объектов SWFMorph ().
Пример ниже создает три концентрических изменяющихся кольца.

```
<?php
  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
  $g->addEntry(0.32, 0, 0, 0);
```

```
$g->addEntry(0.48, 0xff, 0xff, 0xff);
$g->addEntry(0.64, 0, 0, 0);
$g->addEntry(0.80, 0xff, 0xff, 0xff);
$g->addEntry(1.00, 0, 0, 0);

$s = $p->getShape1();
$f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
$f->scaleTo(0.05);
$s->setLeftFill($f);
$s->movePenTo(-160, -120);
$s->drawLine(320, 0);
$s->drawLine(0, 240);
$s->drawLine(-320, 0);
$s->drawLine(0, -240);

$g = new SWFGradient();
$g->addEntry(0.0, 0, 0, 0);
$g->addEntry(0.16, 0xff, 0, 0);
$g->addEntry(0.32, 0, 0, 0);
$g->addEntry(0.48, 0, 0xff, 0);
$g->addEntry(0.64, 0, 0, 0);
$g->addEntry(0.80, 0, 0, 0xff);
$g->addEntry(1.00, 0, 0, 0);

$s = $p->getShape2();
$f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
$f->scaleTo(0.05);
$f->skewXTo(1.0);
$s->setLeftFill($f);
$s->movePenTo(-160, -120);
$s->drawLine(320, 0);
$s->drawLine(0, 240);
$s->drawLine(-320, 0);
$s->drawLine(0, -240);

$m = new SWFMovie();
$m->setDimension(320, 240);
$i = $m->add($p);
$i->moveTo(160, 120);

for($n=0; $n<=1.001; $n+=0.01)
{
  $i->setRatio($n);
  $m->nextFrame();
}

header('Content-type: application/x-shockwave-flash');
```

```
  $m->output();
?>
```

SWFDisplayItem->addColor. Увеличение значений компонент цвета

```
void swfdisplayitem->addcolor ([integer red [, integer green [,
integer blue [, integer a]]]])
```

SWFDisplayItem->multColor. Умножение значений компонент цвета

```
void swfdisplayitem->multcolor ([integer red [, integer green [,
integer blue [, integer a]]]])
<?php
  $b = new SWFBitmap("backyard.jpg");
  // note use your own picture :-)
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for($n=0; $n<=20; ++$n)
  {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape – объект фигура

```
<?php
  $m = new SWFMovie();
  $m->setDimension(800, 800);     // размер рисунка
  $m->setbackground(50,100,200);  // цвет фона

  $s = new SWFShape();
  $s->setLine(160, 0x7f, 0, 0);   // стиль линии
```

```
  $s->setRightFill($s->addFill(0xff, 0, 0)); // заливка
  $s->movePenTo(200, 200);      // начальная точка
  $s->drawLineTo(620, 400);
  $s->drawLine(-200, 60);
  $s->drawCurveTo(400, 0, 200, 200);

  $m->add($s);
  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setLine. Установка типа линий фигуры

void **swfshape->setline** (int width [, integer red , integer green , integer blue [, integer a]])

Число аргументов может быть 1, 4 или 5 (но не 3 или 2). Для линии задается ширина width, цвет (значениями компонент красного, зеленого и синего) и альфа параметр.

```
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setRate(12.0);
  $i = $m->add($s);
  $i->moveTo(1500-$f->getWidth("!")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->addFill. Задание стиля заливки

void **swfshape->addfill** (integer red, integer green, integer blue [, integer a])

void **swfshape->addfill** (SWFbitmap bitmap [, integer flags])

void **swfshape->addfill** (SWFGradient gradient [, integer flags])

Возможны три стиля заливки (см. синтаксис): цветом, рисунком, градиентом.

Для рисунка задается дескриптор объекта swfbitmap(); и необязательная опция: SWFFILL_CLIPPED_BITMAP (обрезанный рисунок) или SWFFILL_TILED_BITMAP (размноженный рисунок, по ум.).

Для градиента задается дескриптор объекта `swfgradient()`; и необязательная опция: `SWFFILL_RADIAL_GRADIENT` или `SWFFILL_LINEAR_GRADIENT` (линейный, по ум.).
Возвращает дескриптор объекта `swffill()` для использования в функциях `swfshape->setleftfill()` и `swfshape->setrightfill()`.

```
<?php
  $p = new SWFMorph();
  $b = new SWFBitmap("alphafill.jpg");
      // используйте собственный рисунок
  $width = $b->getWidth();
  $height = $b->getHeight();

  $s = $p->getShape1();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  $f->moveTo(-$width/2, -$height/4);
  $f->scaleTo(1.0, 0.5);
  $s->setLeftFill($f);
  $s->movePenTo(-$width/2, -$height/4);
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);

  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // these two have no effect!
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
  $i->moveTo($width/2, $height/2);

  for($n=0; $n<1.001; $n+=0.03)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }
```

```
  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setLeftFill. Заливка фигуры налево

void **swfshape->setleftfill** (swfgradient fill)
void **swfshape->setleftfill** (int red, int green, int blue [, int a])
Если браузер отказывается закрашивать фигуру, попробуйте закраску с другой стороны.
Второй вариант является аббревиатурой swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));.
См. также: swfshape->setrightfill().

SWFShape->setRightFill. Заливка фигуры направо

void **swfshape->setrightfill** (swfgradient fill)
void **swfshape->setrightfill** (int red, int green, int blue [, int a])
См. также: swfshape->setleftfill().

SWFShape->movePenTo. Задание точки вывода

void **swfshape->movepento** (integer x, integer y)
Указывается точка от которой будет чертиться фигура.
См. также: swfshape->movepen(), swfshape->drawcurveto(), swfshape->drawlineto() and swfshape->drawline().

SWFShape->movePen. Смещение точки вывода

void **swfshape->movepen** (integer dx, integer dy)
См. также: swfshape->movepento(), swfshape->drawcurveto(), swfshape->drawlineto() and swfshape->drawline().

SWFShape->drawLineTo. Начертить линию

void **swfshape->drawlineto** (integer x, integer y)
Создает линию (используя текущий стиль) от текущей точки вывода до указанной.
См. также: swfshape->movepento(), swfshape->drawcurveto(), swfshape->movepen(), swfshape->drawline().

SWFShape->drawLine. Провести отрезок

void **swfshape->drawline** (integer dx, integer dy)
См. также: swfshape->movepento(), swfshape->drawcurveto(), swfshape->movepen() and swfshape->drawlineto().

SWFShape->drawCurveTo. Начертить кривую

`void` **`swfshape->drawcurveto`** `(int controlx, int controly, int endx, int endy)`

Создает квадратичную кривую (используя текущий стиль) от текущей точки вывода до указанной (`endx, endy`), используя точку (`endx, endy`) как ориентирующую.

См. также: `swfshape->drawlineto()`, `swfshape->drawline()`, `swfshape->movepento()` and `swfshape->movepen()`.

SWFShape->drawCurve. Провести кривую

`void` **`swfshape->drawcurve`** `(int controldx, int controldy, int dx, int dy)`

Координаты указываются относительные.

SWFGradient – объект градиента

Градиенты используются для заполнения фигур методом `swfshape->addfill()`.

This simple example will draw a big black-to-white gradient as background, and a redish disc in its center.

```
<?php
  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // first gradient- black to white
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.03);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // second gradient- radial gradient from red to transparent
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);
```

```
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFGradient->addEntry. Добавление экстремума градиента

`void` **`swfgradient->addentry`** `(double ratio, int red, int green, int blue [, int a])`

Аргумент `ratio` указывает позицию диапазона градиента (от 0 до 1) для которой указывается цвет. Необязательным аргументом `a` указывается прозрачность (0 – 255).

SWFBitmap – загрузить рисунок

`new` **`swfbitmap`** `(string filename [, integer alphafilename])`

Загружает рисунок из файла Jpeg или DBL-файла `filename`. Можно указать дополнительный рисунок `alphafilename`, для использования в качестве альфа маски. Jpeg рисунок должен быть в формате baseline (frame 0), другие форматы: baseline optimized и progressive scan jpeg не поддерживаются.

Импортировать рисунки png напрямую нельзя, хотя можно использовать утилиту png2dbl для конвертирования в формат dbl («define bits lossless»).

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap("p.dbl"));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
```

```
?>
```

Пример с использованием альфа маски:

```
<?php
  $s = new SWFShape();

  // .msk файл генерирован утилитой gif2mask
  $f = $s->addFill(new SWFBitmap("alphafill.jpg", "alphafill.msk"));
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);

  // нарисовать шахматный фон
  for($y=0; $y<480; $y+=40)  {
    for($x=0; $x<640; $x+=80)  {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
    $y+=40;
    for($x=40; $x<640; $x+=80)    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFBitmap->getWidth. Ширина рисунка в пикселях

```
int swfbitmap->getwidth (void)
```
См. также: `swfbitmap->getheight()`.

SWFBitmap->getHeight. Высота рисунка в пикселях

```
int swfbitmap->getheight (void)
```
См. также: `swfbitmap->getwidth()`.

SWFFill – объект заполнитель

С помощью этого объекта можно трансформировать (масштабировать, наклонять, вращать) рисунки и градиенты, используемые как заполнители. Этот объект создается методом `swfshape->addfill()`.

SWFFill->moveTo. Установка точки от которой начинается заполнение

```
void swffill->moveto (integer x, integer y)
```

SWFFill->scaleTo. Установка масштаба заполнителя

```
void swffill->scaleto (integer kx, integer ky)
```

SWFFill->rotateTo. Установка угла поворота заполнителя

```
void swffill->rotateto (double degrees)
```

SWFFill->skewXTo. Установка угла x-наклона заполнителя

```
void swffill->skewxto (double x)
```
Указывается коэффициент смещения основания вправо, то есть 1.0, означает искажение в 45 градусов. Возможны отрицательные значения для смещения основания влево.

SWFFill->skewYTo. Установка y-наклона заполнителя

```
void swffill->skewyto (double y)
```
Указывается смещение правой стороны вниз.

SWFMorph – объект превращение

Объект представляет собой превращение одной фигуры в другую («shape tween»).
В примере ниже, большой красный квадрат, вращаясь, превращается в маленький синий.

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0,0,0,0);

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
```

```
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,200,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for($r=0.0; $r<=1.0; $r+=0.1)  {
    $i->setRatio($r);
    $m->nextFrame();
}

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFMorph->getshape1. Начальная фигура превращения

mixed **swfmorph->getshape1** (void)
Возвращает объект swfshape().

SWFMorph->getshape2. Конечная фигура превращения

mixed **swfmorph->getshape2** (void)
Возвращает объект swfshape().

SWFText – объект текста

```
<?php
$f = new SWFFont("Techno.fdb");
$t = new SWFText();
$t->setFont($f);
$t->moveTo(200, 2400);
$t->setColor(0xff, 0xff, 0);
```

```
$t->setHeight(1200);
$t->addString("PHP generates Flash with Ming!!");

$m = new SWFMovie();
$m->setDimension(5400, 3600);

$m->add($t);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFText->setFont. Установка текущего шрифта

```
void swftext->setfont (font)
```

SWFText->setHeight. Установка высоты шрифта

```
void swftext->setheight (integer height)
```
По умолчанию 240.

SWFText->setSpacing. Установка расстояния между символами

```
void swftext->setspacing (double spacing)
```
По умолчанию 1.0. Значение 0 указывает, что следующий символ будет выводиться непосредственно за последним. Вычисления не отличаются точностью.

SWFText->setColor. Установка цвета шрифта

```
void swftext->setcolor (int red, int green, int blue [, int a])
```
По умолчанию черный.

SWFText->moveTo. Установка позиции вывода

```
void swftext->moveto (integer x, integer y)
```

SWFText->addString. Нарисовать строку

```
void swftext->addstring (string string)
```

SWFText->getWidth. Вычисление ширины строки

```
double swftext->addstring (string string)
```
Возвращает размер с учетом размеров шрифта, масштаба и т. п.

SWFFont – загрузить шрифт

```
new swffont (string filename)
```
В аргументе указывается имя FDB файла; или имя шрифта, поддерживаемого браузером. Формат FDB («font definition block») содержит описание шрифта,

наподобие «SWF DefineFont2 block». Файл FDB можно получить из файлов-шаблонов «SWT Generator» с помощью утилиты makefdb библиотеки ming.
Три специальных шрифта должны быть доступны всегда: _serif, _sans, и _typewriter.

```
<?php $f = newSWFFont("_sans"); ?>
```

Дескриптор шрифта, возвращенный `swffont()` используется как аргумент методов `SWFText->setFont()` и `SWFTextField->setFont()`.

swffont->getwidth. Вычисление ширины строки

`int` **`swffont->getwidth`** `(string string)`
Метод аналогичен методу `SWFText()`, но используется значение масштаба по умолчанию.

SWFTextField – объект текстового поля

`new` **`swftextfield`** `([int flags])`
Текстовые поля менее функциональны, нежели объекты `swftext()`: их нельзя вращать, непропорционально масштабировать, наклонять, но они могут использоваться для ввода в формах и использовать шрифты браузера.
Поведение поля можно изменить, указав необязательные флаги; следующие значения можно комбинировать, используя битовый оператор ИЛИ:

- `SWFTEXTFIELD_NOEDIT` – нередактируемое поле
- `SWFTEXTFIELD_PASSWORD` – скрывает вводимые данные
- `SWFTEXTFIELD_DRAWBOX` – чертит обрамление
- `SWFTEXTFIELD_MULTILINE` – многострочный режим
- `SWFTEXTFIELD_WORDWRAP` – позволяет автоматически разносить длинный текст на несколько строк
- `SWFTEXTFIELD_NOSELECT` – делает текст поля не выделяемым

Например следующая строка,

```
$t = newSWFTextField(SWFTEXTFIELD_NOSELECT | SWFTEXTFIELD_NOEDIT);
```

создает поле с невыделяемым и нередактируемым текстом.

SWFTextField->setFont. Установка шрифта поля

`void` **`swftextfield->setfont`** `(string font)`

SWFTextField->setbounds. Установка ширины и высоты поля

`void` **`swftextfield->setbounds`** `(int width, int height)`

SWFTextField->align. Установка выравнивание текста в поле

`void` **`swftextfield->align`** `(int alignement)`

Допустимые значения аргумента: SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER и SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setHeight. Установка высоты шрифта текстового поля

void **swftextfield->setheight** (int height)
Значение по умолчанию: 240.

SWFTextField->setLeftMargin. Установка ширины левого отступа поля

void **swftextfield->setleftmargin** (int width)
По умолчанию 0.

SWFTextField->setrightMargin. Установка ширины правого отступа поля

void **swftextfield->setrightmargin** (int width)
По умолчанию 0.

SWFTextField->setMargins. Установка ширины левого и правого отступов поля

void **swftextfield->setmargins** (int left, int right)
Это комбинация двух вышеописанных методов.

SWFTextField->setindentation. Установка ширины отступа первой строки

void **swftextfield->setindentation** (int width)

SWFTextField->setLineSpacing. Установка межстрочного расстояния

void **swftextfield->setlinespacing** (int height)
По умолчанию 40.

SWFTextField->setcolor. Установка цвета поля

void **swftextfield->setcolor** (int red, int green, int blue [, int a])
По умолчанию прозрачный.

SWFTextField->setname. Присвоение полю имени

void **swftextfield->setname** (string name)
Используется при отсылке данных формы и для выполнения действий.

SWFTextField->addstring. Добавление строки к тексту поля

void **swftextfield->addstring** (string string)

SWFSprite – создать клип (sprite)

Спрайт («movie clip») позволяет создавать анимации с собственной системой отчета времени. Поэтому спрайты функционируют подобно основным анимациям.

```
<?php
$s = new SWFShape();
```

```
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->movePenTo(-500,-500);
$s->drawLineTo(500,-500);
$s->drawLineTo(500,500);
$s->drawLineTo(-500,500);
$s->drawLineTo(-500,-500);

$p = new SWFSprite();
$i = $p->add($s);
for($j=0;$j<8;$j++){
$p->nextFrame();
$i->rotate(10);
}
$p->nextFrame();

$m = new SWFMovie();
$i = $m->add($p);
$i->moveTo(1500,1000);
$i->setName("blah");

$m->setDimension(3000,2000);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFSprite->add. Добавление объекта к спрайту

`void` **`swfsprite->add`** `(resource object)`
Этим методом можно добавлять объекты: `swfshape()`, `swfbutton()`, `swftext()`, `swfaction()`, `swfsprite()`. Метод возвращает дескриптор объекта в списке отображения.

SWFSprite->remove. Удаление объекта из спрайта

`void` **`swfsprite->remove`** `(resource object)`
Метод обратный предыдущему.

SWFSprite->setframes. Установка общего числа кадров

`void` **`swfsprite->setframes`** `(integer numberofframes)`

SWFSprite->nextframe. Переход к следующему кадру

`void` **`swfsprite->nextframe`** `(void)`

SWFbutton – объект кнопка

Пример отслеживает события кнопки.

```
<?php
$f = new SWFFont("_serif");
$p = new SWFSprite();

function label($string) {
global $f;

$t = new SWFTextField();
$t->setFont($f);
$t->addString($string);
$t->setHeight(200);
$t->setBounds(3200,200);
return $t;
}
function addLabel($string) {
global $p;

$i = $p->add(label($string));
$p->nextFrame();
$p->remove($i);
}

$p->add(new SWFAction("stop();"));
addLabel("NO ACTION");
addLabel("SWFBUTTON_MOUSEUP");
addLabel("SWFBUTTON_MOUSEDOWN");
addLabel("SWFBUTTON_MOUSEOVER");
addLabel("SWFBUTTON_MOUSEOUT");
addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
addLabel("SWFBUTTON_DRAGOVER");
addLabel("SWFBUTTON_DRAGOUT");

function rect($r, $g, $b) {
$s = new SWFShape();
$s->setRightFill($s->addFill($r, $g, $b));
$s->drawLine(600,0);
$s->drawLine(0,600);
$s->drawLine(-600,0);
$s->drawLine(0,-600);

return $s;
}

$b = new SWFButton();
$b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
$b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
$b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);
```

```
$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
    SWFBUTTON_MOUSEUP);
$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
      SWFBUTTON_MOUSEDOWN);
$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
      SWFBUTTON_MOUSEOVER);
$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
      SWFBUTTON_MOUSEOUT);
$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
      SWFBUTTON_MOUSEUPOUTSIDE);
$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
      SWFBUTTON_DRAGOVER);
$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
      SWFBUTTON_DRAGOUT);

$m = new SWFMovie();
$m->setDimension(4000,3000);

$i = $m->add($p);
$i->setName("label");
$i->moveTo(400,1900);

$i = $m->add($b);
$i->moveTo(400,900);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Следующий пример позволяет перетаскивать кнопку по экрану.

```
<?php
$s = new SWFShape();
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->drawLine(1000,0);
$s->drawLine(0,1000);
$s->drawLine(-1000,0);
$s->drawLine(0,-1000);

$b = new SWFButton();
$b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP |
               SWFBUTTON_DOWN | SWFBUTTON_OVER);
$b->addAction(new SWFAction("startDrag('/test', 0);"),
      SWFBUTTON_MOUSEDOWN); // '0' указывает на неблокировку мыши
$b->addAction(new SWFAction("stopDrag();"),
      SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);
```

```
$p = new SWFSprite();
$p->add($b);
$p->nextFrame();

$m = new SWFMovie();
$i = $m->add($p);
$i->setName('test');
$i->moveTo(1000,1000);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFbutton->addShape. Назначение кнопке фигуры

`void` **`swfbutton->addshape`** `(resource shape, integer flags)`
Возможны флаги: `SWFBUTTON_UP`, `SWFBUTTON_OVER`, `SWFBUTTON_DOWN`, `SWFBUTTON_HIT`. Кнопка `SWFBUTTON_HIT` не отображается, а определяет область, реагирующую на нажатия.

SWFbutton->setUp. Синоним addShape(shape, SWFBUTTON_UP)

`void` **`swfbutton->setup`** `(resource shape)`
См. также: `swfbutton->addshape()` и `SWFAction()`.

SWFbutton->setOver. Синоним addShape(shape, SWFBUTTON_OVER)

`void` **`swfbutton->setover`** `(resource shape)`

SWFbutton->setdown. Синоним addShape(shape, SWFBUTTON_DOWN))

`void` **`swfbutton->setdown`** `(resource shape)`

SWFbutton->setHit. Синоним addShape(shape, SWFBUTTON_HIT)

`void` **`swfbutton->sethit`** `(resource shape)`

SWFbutton->addAction. Назначение кнопке действия

`void` **`swfbutton->addaction`** `(resource action, integer flags)`
Назначает обработчик событий кнопки (`action` – дескриптор объекта `swfaction()`) для следующих событий (`flags`): `SWFBUTTON_MOUSEOVER`, `SWFBUTTON_MOUSEOUT`, `SWFBUTTON_MOUSEUP`, `SWFBUTTON_MOUSEUPOUTSIDE`, `SWFBUTTON_MOUSEDOWN`, `SWFBUTTON_DRAGOUT` и `SWFBUTTON_DRAGOVER`.

SWFbutton->setAction. Синоним addAction(shape, SWFBUTTON_MOUSEUP)

`void` **`swfbutton->setaction`** `(resource action)`
Назначает кнопке действие, выполняемое при нажатии.

SWFAction – объект Действие

Объект компилирует скрипт в действие.
Синтаксис скриптов подобен упрощенному C. SWF bytecode machine не позволяет многих веще, например, достаточно сложно реализовать вызовы пользовательских функций.
Компилятор распознает следующие инструкции:

- `break`
- `for`
- `continue`
- `if`
- `else`
- `do`
- `while`

Все данные не имеют типа; значения SWF действий сохраняются как строки. Следующие функции могут использоваться в выражениях:

`time()`

возвращает число миллисекунд, прошедшее с начала проигрывания анимации

`random(seed)`

возвращает случайное число из диапазона (0-`seed`).

`length(expr)`

возвращает длину выражения

`int(number)`

округление до ближайшего целого

`concat(expr, expr)`

соединение строк

`ord(expr)`

возвращает ASCII код данного символа

`chr(num)`

возвращает символ с указанным ASCII кодом

`substr(string, location, length)`

возвращает часть строки `string` длинной length, начиная с позиции `location`

Также могут быть использованы дополнительные команды:

`duplicateClip(clip, name, depth)`

дублирует клип (sprite), присваивая ему имя `name` и располагая на уровне `depth`

`removeClip(expr)`

удаляет именованный клип анимации

`trace(expr)`

записывает сообщение в журнал отладки (бесполезно, пока)

`startDrag(target, lock, [left, top, right, bottom])`

начать перетаскивание объекта `target`. Аргумент `lock` определяет блокировать ли мышь: 0 (false) или 1 (true). Возможно указать область в которой осуществляется перетаскивание.

`stopDrag()`

прекратить перетаскивание

`callFrame(expr)`

вызвать именованный кадр как функцию

`getURL(url, target, [method])`

загрузить данный `url` в именованную область `target` (имя кадра или зарезервированное значение "_level0» для замены текущей анимации или "_level1» для загрузки анимации поверх текущей). В необязательном аргументе можно указать метод отсылки: post или get для отсылки серверу данных.

`loadMovie(url, target)`

подобие предыдущей команды

`nextFrame()`

перейти к следующему кадру

`prevFrame()`

перейти к предыдущему кадру

`play()`

начать проигрывание анимации

`stop()`

закончить проигрывание анимации

`toggleQuality()`

переключиться между низким и высоким качеством

`stopSounds()`

прекратить проигрывание звуков

`gotoFrame(num)`

перейти к кадру с указанным номером (нумерация начинается от 0)

`gotoFrame(name)`

перейти к именованному кадру

`setTarget(expr)`

установить контекст действия.

`frameLoaded(num)`

может использоваться в условных и циклических конструкциях для проверки того, был ли уже загружен кадр с указанным номером. Возможно использовать альтернативное выражение: `/:framesLoaded`.

Клипы и анимации имеют следующие свойства:

- `x`
- `y`
- `xScale` – масштаб по горизонтали
- `yScale` – масштаб по вертикали
- `currentFrame` – текущий кадр (только для чтения)
- `totalFrames` – общее число кадров (только для чтения)
- `alpha` – уровень прозрачности
- `visible` – видимость (1=on, 0=off)
- `width` - (только для чтения)
- `height` - (только для чтения)
- `rotation` – угол поворота
- `target` - (только для чтения)
- `framesLoaded` - (только для чтения)
- `name`
- `dropTarget` - (только для чтения)
- `url` - (только для чтения)
- `highQuality` – качество (1=высокое, 0=низкое)
- `focusRect`
- `soundBufTime`

Например, установить значение позиции спрайта можно выражением: `/box.x = 100;`. Поскольку flash сохраняет все компоненты анимации в древовидной структуре (наподобие файловой системы unix) начальный слеш указывает на массив объектов верхнего уровня; если спрайт box находится внутри спрайта biff, то выражение записывается как: `/box/biff.x = 100;`.

Следующий пример перемещает красный квадрат по экрану.

```
<?php
$s = new SWFShape();
$f = $s->addFill(0xff, 0, 0);
$s->setRightFill($f);

$s->movePenTo(-500,-500);
$s->drawLineTo(500,-500);
```

```
$s->drawLineTo(500,500);
$s->drawLineTo(-500,500);
$s->drawLineTo(-500,-500);

$p = new SWFSprite();
$i = $p->add($s);
$i->setDepth(1);
$p->nextFrame();

for($n=0; $n<5; ++$n) {
$i->rotate(-15);
$p->nextFrame();
}

$m = new SWFMovie();
$m->setBackground(0xff, 0xff, 0xff);
$m->setDimension(6000,4000);

$i = $m->add($p);
$i->setDepth(1);
$i->moveTo(-500,2000);
$i->setName("box");

$m->add(new SWFAction("/box.x += 3;"));
$m->nextFrame();
$m->add(new SWFAction(" gotoFrame(0); play(); "));
$m->nextFrame();

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Следующий пример отслеживает перемещение мыши по экрану.

```
<?php
$m = new SWFMovie();
$m->setRate(36.0);
$m->setDimension(1200, 800);
$m->setBackground(0, 0, 0);

/* пустой спрайт для отслеживания x и y координат мыши */

$i = $m->add(new SWFSprite());
$i->setName('mouse');

$m->add(new SWFAction(" startDrag('/mouse', 1); "));

// antialiasing можно выключить для квадратов.
```

```
$m->add(new SWFAction(" this.quality = 0; "));

$r = new SWFMorph();/* morphing box */
$s = $r->getShape1();
$s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
$s->movePenTo(-40, -40);
$s->drawLine(80, 0);
$s->drawLine(0, 80);
$s->drawLine(-80, 0);
$s->drawLine(0, -80);

$s = $r->getShape2();
$s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
$s->movePenTo(-1, -1);
$s->drawLine(2, 0);
$s->drawLine(0, 2);
$s->drawLine(-2, 0);
$s->drawLine(0, -2);

$box = new SWFSprite();
$box->add(new SWFAction(" stop(); "));
$i = $box->add($r);

for($n=0; $n<=20; ++$n) {
$i->setRatio($n/20);
$box->nextFrame();
}

/* спрайт контейнер позволяет многократно использовать действие */

$cell = new SWFSprite();
$i = $cell->add($box);
$i->setName('box');

$cell->add(new SWFAction("
setTarget('box');
/* x - родительская координата, i.e. (..).x */
dx = (/mouse.x + random(6)-3 - ...x)/5;
dy = (/mouse.y + random(6)-3 - ...y)/5;
gotoFrame(int(dx*dx + dy*dy));
"));

$cell->nextFrame();
$cell->add(new SWFAction(" gotoFrame(0); play(); "));

$cell->nextFrame();
```

```
/* добавим ячейки в анимацию */
for($x=0; $x<12; ++$x) {
for($y=0; $y<8; ++$y) {
$i = $m->add($cell);
$i->moveTo(100*$x+50, 100*$y+50);
}
}

$m->nextFrame();

$m->add(new SWFAction(" gotoFrame(1); play(); "));

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Тот же скрипт, но с цветными шариками...

```
<?php
$m = new SWFMovie();
$m->setDimension(11000, 8000);
$m->setBackground(0x00, 0x00, 0x00);

$m->add(new SWFAction("
      this.quality = 0;
      /frames.visible = 0;
      startDrag('/mouse', 1);
"));

// mouse tracking sprite
$t = new SWFSprite();
$i = $m->add($t);
$i->setName('mouse');

$g = new SWFGradient();
$g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
$g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
$g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
$g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

// gradient shape thing
$s = new SWFShape();
$f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
$f->scaleTo(0.03);
$s->setRightFill($f);
$s->movePenTo(-600, -600);
$s->drawLine(1200, 0);
$s->drawLine(0, 1200);
```

```
$s->drawLine(-1200, 0);
$s->drawLine(0, -1200);

// need to make this a sprite so we can multColor it
$p = new SWFSprite();
$p->add($s);
$p->nextFrame();

// put the shape in here, each frame a different color
$q = new SWFSprite();
$q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
$i = $q->add($p);

$i->multColor(1.0, 1.0, 1.0);
$q->nextFrame();
$i->multColor(1.0, 0.5, 0.5);
$q->nextFrame();
$i->multColor(1.0, 0.75, 0.5);
$q->nextFrame();
$i->multColor(1.0, 1.0, 0.5);
$q->nextFrame();
$i->multColor(0.5, 1.0, 0.5);
$q->nextFrame();
$i->multColor(0.5, 0.5, 1.0);
$q->nextFrame();
$i->multColor(1.0, 0.5, 1.0);
$q->nextFrame();

// finally, this one contains the action code
$p = new SWFSprite();
$i = $p->add($q);
$i->setName('frames');
$p->add(new SWFAction("
      dx = (/:mousex-/:lastx)/3 + random(10)-5;
      dy = (/:mousey-/:lasty)/3;
      x = /:mousex;
      y = /:mousey;
      alpha = 100;
"));
$p->nextFrame();
$p->add(new SWFAction("
      this.x = x; this.y = y; this.alpha = alpha;
      x += dx; y += dy; dy += 3; alpha -= 8;
"));
$p->nextFrame();
$p->add(new SWFAction("prevFrame(); play();"));
$p->nextFrame();
```

```
$i = $m->add($p);
$i->setName('frames');
$m->nextFrame();

$m->add(new SWFAction("
      lastx = mousex; lasty = mousey;
      mousex = /mouse.x; mousey = /mouse.y;
      ++num;

      if(num == 11) num = 1;

      removeClip('char' & num);
      duplicateClip(/frames, 'char' & num, num);
"));

$m->nextFrame();
$m->add(new SWFAction("prevFrame(); play();"));

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Следующий пример отслеживает нажатия клавиш. Необходимо, чтобы фокус окна был активен

```
<?php /* спрайт с буквой в каждом кадре */

$p = new SWFSprite();
$p->add(new SWFAction("stop();"));

$chars = "abcdefghijklmnopqrstuvwxyz".
    "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
"1234567890!@#$%^&*()_+-=/[]{}|;:,.<>?`~";

$f = new SWFFont("_sans");

for($n=0; $nremove($i);
$t = new SWFTextField();
$t->setFont($f);
$t->setHeight(240);
$t->setBounds(600,240);
$t->align(SWFTEXTFIELD_ALIGN_CENTER);
$t->addString($c);
$i = $p->add($t);
$p->labelFrame($c);
$p->nextFrame();
}
```

```
/* область нажатия для кнопки - кадр целиком */

$s = new SWFShape();
$s->setFillStyle0($s->addSolidFill(0, 0, 0, 0));
$s->drawLine(600, 0);
$s->drawLine(0, 400);
$s->drawLine(-600, 0);
$s->drawLine(0, -400);

/* кнопка проверяет нажатую клавишу и
          переходит к соответствующему кадру */

$b = new SWFButton();
$b->addShape($s, SWFBUTTON_HIT);

for($n=0; $naddAction(new SWFAction("
      setTarget('/char');gotoFrame('$c');"), SWFBUTTON_KEYPRESS($c));
}

$m = new SWFMovie();
$m->setDimension(600,400);
$i = $m->add($p);
$i->setName('char');
$i->moveTo(0,80);

$m->add($b);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

PDF документы

PDF функции позволяют PHP создавать PDF файлы с помощью библиотеки PDF, созданной Thomas Merz (http://www.pdflib.com/pdflib/index.html; также могут потребоваться библиотеки JPEG (ftp://ftp.uu.net/graphics/jpeg/) и TIFF (http://www.libtiff.org/).

С pdflib поставляется хорошая документация, описывающая возможности библиотеки. Имена функций и аргументы идентичны в библиотеке и PHP. Размеры и координаты измеряются в единицах Postscript (72 на дюйм), но это зависит от выбранного разрешения.

Аналогом библиотеки является ClibPDF (см. ниже).

Версии ниже 3.0 pdflib не поддерживаются в PHP4. Для компиляции PHP с библиотекой pdflib используйте опцию --enable-shared-pdflib.

```
<?php
$fp = fopen("test.pdf", "w");
$pdf = pdf_open($fp);
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Test for PHP PDFlib");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Testing");
pdf_begin_page($pdf, 595, 842);
pdf_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
pdf_end_page($pdf);
pdf_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php>finished</A>";
?>

<?php // скрипт getpdf.php, просто возвращает документ pdf
$fp = fopen("test.pdf", "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

Пример из комплекта pdflib создает документ, состоящий из страниц с изображением часов (каждая страница показывает текущее время создания):

```
<?php
$pdffilename = "clock.pdf";    // имя файла
$radius = 200;             // радиус циферблата
$margin = 20;                // поля страницы
$pagecount = 40;             // число страниц

$fp = fopen($pdffilename, "w");
$pdf = pdf_open($fp);

while($pagecount-- > 0) {
pdf_begin_page($pdf, 2 * ($radius + $margin),
                     2 * ($radius + $margin));

pdf_set_parameter($pdf, "transition", "wipe");
pdf_set_value($pdf, "duration", 0.5);

pdf_translate($pdf, $radius + $margin, $radius + $margin);
```

```
pdf_save($pdf);
pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

/* минутные метки */
pdf_setlinewidth($pdf, 2.0);
for ($alpha = 0; $alpha < 360; $alpha += 6) {
pdf_rotate($pdf, 6.0);
pdf_moveto($pdf, $radius, 0.0);
pdf_lineto($pdf, $radius-$margin/3, 0.0);
pdf_stroke($pdf);
}

pdf_restore($pdf);
pdf_save($pdf);

/* 5 минутные метки */
pdf_setlinewidth($pdf, 3.0);
for ($alpha = 0; $alpha < 360; $alpha += 30) {
pdf_rotate($pdf, 30.0);
pdf_moveto($pdf, $radius, 0.0);
pdf_lineto($pdf, $radius-$margin, 0.0);
pdf_stroke($pdf);
}

$ltime = getdate();

/* часовая стрелка */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['minutes']/60.0)+
                        $ltime['hours']-3.0)*30.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius/2, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* минуная стрелка */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['seconds']/60.0)+
                     $ltime['minutes']-15.0)*6.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius * 0.8, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);
```

```
/* секундная стрелка */
pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
pdf_setlinewidth($pdf, 2);
pdf_save($pdf);
pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
pdf_moveto($pdf, -$radius/5, 0.0);
pdf_lineto($pdf, $radius, 0.0);
pdf_stroke($pdf);
pdf_restore($pdf);

/* центр */
pdf_circle($pdf, 0, 0, $radius/30);
pdf_fill($pdf);

pdf_restore($pdf);

pdf_end_page($pdf);
}

$pdf = pdf_close($pdf);
fclose($fp);
$fp = fopen($pdffilename, "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

pdf_set_info. Заполнение поля информации документа

`void` **`pdf_set_info`** `(int pdf_document, string fieldname, string value)`

Возможные поля `fieldname`: 'Subject', 'Title', 'Creator', 'Author', 'Keywords' и одно определяемое пользователем. Функция должна вызываться до создания страниц.

```
<?php
$fd = fopen("test.pdf", "w");
$pdfdoc = pdf_open($fd);
pdf_set_info($pdfdoc, "Author", "Имя автора");
pdf_set_info($pdfdoc, "Creator", "Название создателя");
pdf_set_info($pdfdoc, "Title", "Заголовок");
pdf_set_info($pdfdoc, "Subject", "Тема");
pdf_set_info($pdfdoc, "Keywords", "Ключевые, слова");
pdf_set_info($pdfdoc, "CustomField", "Чего-то еще");
pdf_begin_page($pdfdoc, 595, 842);
pdf_end_page($pdfdoc);
pdf_close($pdfdoc);
```

```
?>
```

Эта функция заменяет собой: `pdf_set_info_keywords()`, `pdf_set_info_title()`, `pdf_set_info_subject()`, `pdf_set_info_creator()`, `pdf_set_info_sybject()`.

pdf_open. Открытие нового документа pdf

`int` **`pdf_open`** `(int file)`

Функция делает файл, открытый функцией `fopen()` документом pdf. Если не указывать дескриптор файла, он создается в памяти, и затем может выводиться на стандартный поток вывода или отсылаться браузеру. Функция возвращает дескриптор документа, который следует указывать в последующих pdf функциях.

См. также: `fopen()`, `pdf_close()`.

pdf_close. Закрытие документа pdf

`void` **`pdf_close`** `(int pdf_document)`

См. также: `pdf_open()`, `fclose()`.

pdf_begin_page. Начать новую страницу

`void` **`pdf_begin_page`** `(int pdf_document, double width, double height)`

Аргументы `height` и `width` задают высоту и ширину страницы. После внесения на страницу информации ее следует закрыть функцией `pdf_end_page()`.

См. также: `pdf_end_page()`.

pdf_end_page. Завершение страницы

`void` **`pdf_end_page`** `(int pdf_document)`

После этого модификация этой страницы невозможна.

См. также: `pdf_begin_page()`.

pdf_show. Вывод текста в текущую позицию

`void` **`pdf_show`** `(int pdf_document, string text)`

Для вывода используется текущая позиция и текущий шрифт.

См. также: `pdf_show_xy()`, `pdf_show_boxed()`, `pdf_set_text_pos()`, `pdf_set_font()`.

pdf_show_boxed. Вывод текста в прямоугольную область

`int` **`pdf_show_boxed`** `(int pdf_document, string text, double x, double y, double width, double height, string mode [, string feature])`

Левый нижний угол области вывода задается (`x,y`); высота и ширина: `height`, `width`. Аргумент `mode` определяет выравнивание текста: если высота и ширина равна нулю, то возможны значения: «left», «right» или «center», если они не равны нулю, то - «justify» или «fulljustify».

Если аргумент `feature` содержит значение «blind», текст не отображается.

Возвращает число символов, которое не поместилось в указанный прямоугольник.

См. также: `pdf_show()`, `pdf_show_xy()`.

pdf_show_xy. Вывод текста в указанную позицию

`void` **`pdf_show_xy`** `(int pdf_doc, string text, double x, double y)`

См. также: `pdf_show()`, `pdf_show_boxed()`.

pdf_set_font. Выбор шрифта, его размера и кодировки

`void` **`pdf_set_font`** `(int pdf_document, string font_name, double size, string encoding [, int embed])`

Аргумент вида кодировки `encoding` может принимать значения: 'winansi' (по ум.), 'builtin', 'host', 'macroman' и т. п. Если для последнего аргумента задано значение 1, шрифт будет внедрен в документ pdf (иначе нет). Если шрифт распространен, внедрять его не следует из-за увеличения размера документа.

Функция должна вызываться после `pdf_begin_page()`.

pdf_set_leading. Установка промежутка между строками текста

`void` **`pdf_set_leading`** `(int pdf_document, double distance)`

Используется при выводе текста функцией `pdf_continue_text()`.

См. также: `pdf_continue_text()`.

pdf_set_parameter. Установка строкового значения параметра pdflib

`void` **`pdf_set_parameter`** `(int pdf_document, string name, string value)`

См. также: `pdf_get_value()`, `pdf_set_value()`, `pdf_get_parameter()`.

pdf_get_parameter. Получение строкового значения параметра pdflib

`string` **`pdf_get_parameter`** `(int pdf_doc, string name [, double modifier])`

Аргумент `modifier` используется при необходимости.

См. также: `pdf_get_value()`, `pdf_set_value()`, `pdf_set_parameter()`.

pdf_set_value. Установка численного значения параметра pdflib

`void` **`pdf_set_value`** `(int pdf_document, string name, double value)`

См. также: `pdf_get_value()`, `pdf_get_parameter()`, `pdf_set_parameter()`.

pdf_get_value. Получение численного значения параметра pdflib

`double` **`pdf_get_value`** `(int pdf_document, string name [, double modifier])`

Аргумент `modifier` используется при необходимости.

См. также: `pdf_set_value()`, `pdf_get_parameter()`, `pdf_set_parameter()`.

pdf_set_text_rendering. Установка метода вывода текста

`void` **`pdf_set_text_rendering`** `(int pdf_document, int mode)`

Устарела: используйте `PDF_set_value()`.

pdf_set_horiz_scaling. Установка масштабирования текста по горизонтали

`void` **`pdf_set_horiz_scaling`** `(int pdf_document, double scale)`

pdf_set_text_rise. Установка подъема текста

`void` **`pdf_set_text_rise`** `(int pdf_document, double rise)`

pdf_set_text_matrix. Установка матрицы преобразования шрифта

`void` **`pdf_set_text_matrix`** `(int pdf_document, array matrix)`

Начиная с версии pdflib 2.3 эта функция не доступна.

pdf_set_text_pos. Установка позиции текста

`void` **`pdf_set_text_pos`** `(int pdf_document, double x-coor, double y-coor)`

Устанавливает позицию вывода текста последующим вызовом `pdf_show()`.

См. также: `pdf_show()`, `pdf_show_xy()`.

pdf_set_char_spacing. Установка интервала между символами

`void` **`pdf_set_char_spacing`** `(int pdf_document, double space)`

Устарела: используйте `PDF_set_value()`.

См. также: `pdf_set_word_spacing()`, `pdf_set_leading()`.

pdf_set_word_spacing. Установка интервала между символами

`void` **`pdf_set_word_spacing`** `(int pdf_document, double space)`

Устарела: используйте `PDF_set_value()`.

См. также: `pdf_set_char_spacing()`, `pdf_set_leading()`.

pdf_skew. Поворот системы координат

`void` **`pdf_skew`** `(int pdf_document, double alpha, double beta)`

Угол поворота в градусах указывается относительно осей `alpha` (x) и `beta` (y). Углы не могут принимать значения 90 или 270 градусов.

pdf_continue_text. Вывод текста со следующей строки

`void` **`pdf_continue_text`** `(int pdf_document, string text)`

Расстояние между строками может быть установлено функцией `pdf_set_leading()`.

См. также: `pdf_show_xy()`, `pdf_set_leading()`, `pdf_set_text_pos()`.

pdf_stringwidth. Вычисление ширины текста

`double` **`pdf_stringwidth`** `(int pdf_document, string text)`

При вычислении длины строки используется текущий шрифт. Предварительно шрифт должен быть установлен с помощью `pdf_set_font()`.

См. также: `pdf_set_font()`.

pdf_save. Сохранение текущих установок

`void` **`pdf_save`** `(int pdf_document)`

Действует подобно команде postscript gsave. Полезна при необходимости масштабировать или развернуть объект, не воздействуя на другие объекты. `pdf_save()` требует, чтобы затем была вызвана функция `pdf_restore()`.

См. также: `pdf_restore()`.

pdf_restore. Восстановление ранее сохраненных установок

`void` **`pdf_restore`** `(int pdf_document)`

Восстанавливает установки, сохраненные `pdf_save()`. Действует подобно команде postscript grestore.

```
<?php pdf_save($pdf);
// всякие вращения и трансформации ...
pdf_restore($pdf) ?>
```

См. также: `pdf_save()`.

pdf_translate. Установка начала системы координат

`void` **`pdf_translate`** `(int pdf_document, double x, double y)`

Координаты указываются относительно текущей точки отчета. Затем требуется установить текущую точку, до начала рисования объектов.

```
<?php pdf_moveto($pdf, 0, 0);
pdf_lineto($pdf, 100, 100);
pdf_stroke($pdf);
pdf_translate($pdf, 100, 100);
pdf_moveto($pdf, 0, 0);
pdf_lineto($pdf, 100, 100);
pdf_stroke($pdf);
?>
```

pdf_scale. Установка масштабирования

`void` **`pdf_scale`** `(int pdf_document, double x_scale, double y_scale)`

```
<?php    pdf_scale($pdf, 72.0, 72.0);
         pdf_lineto($pdf, 1, 1); // на дюйм
         pdf_stroke($pdf);
?>
```

pdf_rotate. Установка угла вращения в градусах

`void` **`pdf_rotate`** `(int pdf_document, double angle)`

pdf_setflat. Установка равномерности

`void` **`pdf_setflat`** `(int pdf_document, double value)`

Возможные значения параметра от 0 до 100.

pdf_setlinejoin. Установка параметра linejoin

`void` **`pdf_setlinejoin`** `(int pdf_document, long value)`

Возможные значения параметра от 0 до 2.

pdf_setlinecap. Установка параметра linecap

`void` **`pdf_setlinecap`** `(int pdf_document, int value)`

Возможные значения параметра от 0 до 2.

pdf_setmiterlimit. Установка параметра miter limit

`void` **`pdf_setmiterlimit`** `(int pdf_document, double value)`

Возможные значения параметра: 1 и более.

pdf_setlinewidth. Установка ширины строк

`void` **`pdf_setlinewidth`** `(int pdf_document, double width)`

pdf_setdash. Установка вида штриховки

`void` **`pdf_setdash`** `(int pdf_document, double white, double black)`

Устанавливает размер белых и черных полос. Если оба аргумента равны 0, то линия будет непрерывной.

pdf_moveto. Установка текущей точку

`void` **`pdf_moveto`** `(int pdf_document, double x, double y)`

pdf_curveto. Начертить кривую

`void` **`pdf_curveto`** `(int pdf_document, double x1, double y1, double x2, double y2, double x3, double y3)`

Чертит кривую Безье от текущей точки до (`x3,y3`), используя точки (`x1,y1`) и (`x2,y2`) как ориентирующие.

См. также: `pdf_moveto()`, `pdf_lineto()`, `pdf_stroke()`.

pdf_lineto. Начертить отрезок

`void` **`pdf_lineto`** `(int pdf_document, double x, double y)`

Чертит линию от текущей точки до указанной (`x,y`).

См. также: `pdf_moveto()`, `pdf_curveto()`, `pdf_stroke()`.

pdf_circle. Начертить окружность

`void` **`pdf_circle`** `(int pdf_document, double x, double y, double radius)`

См. также: `pdf_arc()`, `pdf_stroke()`.

pdf_arc. Начертить дугу

`void` **`pdf_arc`** `(int pdf_document, double x, double y, double radius, double start, double end)`

Начальный и конечный угол задаются `start` и `end`.

См. также: `pdf_circle()`, `pdf_stroke()`.

pdf_rect. Начертить прямоугольник

`void` **`pdf_rect`** `(int pdf_document, double x, double y, double width, double height)`

Левый нижний угол задается (`x`, `y`); высота и ширина `height`, `width`.

См. также: `pdf_stroke()`.

pdf_closepath. Завершить текущий путь

`void` **`pdf_closepath`** `(int pdf_document)`

Чертит линию от текущей точки до точки, где начиналась первая линия. Многие функции, например, `pdf_moveto()`, `pdf_circle()`, `pdf_rect()` начинают новый путь.

pdf_stroke. Заштриховка пути

`void` **`pdf_stroke`** `(int pdf_document)`

Текущий путь – это совокупность всех линий. Без этой функции линии начерчены не будут.

См. также: `pdf_closepath()`, `pdf_closepath_stroke()`.

pdf_closepath_stroke. Прочертить и закрыть путь

`void` **`pdf_closepath_stroke`** `(int pdf_document)`

Это комбинация `pdf_closepath()` и `pdf_stroke()`.

См. также: `pdf_closepath()`, `pdf_stroke()`.

pdf_fill. Заполнение пути цветом

`void` **`pdf_fill`** `(int pdf_document)`

См. также: pdf_closepath(), pdf_stroke(), pdf_setgray_fill(), pdf_setgray(),pdf_setrgbcolor_fill(),pdf_setrgbcolor().

pdf_fill_stroke. Заполнение пути цветом и закрытие его

void **pdf_fill_stroke** (int pdf_document)

См. также: pdf_closepath(), pdf_stroke(), pdf_fill(), pdf_setgray_fill(), pdf_setgray(), pdf_setrgbcolor_fill(), pdf_setrgbcolor().

pdf_closepath_fill_stroke. Начертить, закрасить и закрыть путь

void **pdf_closepath_fill_stroke** (int pdf_document)

См. также: pdf_closepath(), pdf_stroke(), pdf_fill(), pdf_setgray_fill(), pdf_setgray(), pdf_setrgbcolor_fill(), pdf_setrgbcolor().

pdf_endpath. Завершение пути без его закрытия

void **pdf_endpath** (int pdf_document)

См. также: pdf_closepath().

pdf_clip. Прикрепление всех линий к текущему пути

void **pdf_clip** (int pdf_document)

pdf_setgray_fill. Установка заполнения серым цветом

void **pdf_setgray_fill** (int pdf_document, double gray_value)

См. также: pdf_setrgbcolor_fill().

pdf_setgray_stroke. Установка штриховки серым цветом

void **pdf_setgray_stroke** (int pdf_document, double gray_value)

См. также: pdf_setrgbcolor_stroke().

pdf_setgray. Установка заполнения и штриховки серым цветом

void **pdf_setgray** (int pdf_document, double gray_value)

См. также: pdf_setrgbcolor_stroke(),pdf_setrgbcolor_fill().

pdf_setrgbcolor_fill. Установка заполнения цветом rgb

`void` **`pdf_setrgbcolor_fill`** `(int pdf_document, double red_value, double green_value, double blue_value)`

См. также: `pdf_setrgbcolor_fill()`.

pdf_setrgbcolor_stroke. Установка штриховки цветом rgb

`void` **`pdf_setrgbcolor_stroke`** `(int pdf_document, double red_value, double green_value, double blue_value)`

См. также: `pdf_setrgbcolor_stroke()`.

pdf_setrgbcolor. Установка заполнения и штриховки серым цветом rgb

`void` **`pdf_setrgbcolor`** `(int pdf_document, double red_value, double green_value, double blue_value)`

См. также: `pdf_setrgbcolor_stroke()`, `pdf_setrgbcolor_fill()`.

pdf_add_outline. Добавление закладки для текущей страницы

`int` **`pdf_add_outline`** `(int pdf_document, string text [, int parent [, int open]])`

Название закладки определяется аргументом `text`. Она становится дочерним объектом объекта `parent` и по умолчанию открыта (если аргумент `open` не равен 0). Возвращается идентификатор закладки, который может использоваться как родительский для других закладок.

pdf_set_transition. Установка режима перехода между страницами

`void` **`pdf_set_transition`** `(int pdf_document, int transition)`

Используйте функцию `PDF_set_parameter()` с параметром «transition».

См. также: `pdf_set_duration()`.

pdf_set_duration. Установка интервала между страницами

`void` **`pdf_set_duration`** `(int pdf_document, double duration)`

См. также: `pdf_set_transition()`.

pdf_open_gif. Открытие рисунка GIF

`int` **`pdf_open_gif`** `(int pdf_document, string filename)`

Используйте функцию `pdf_open_image_file()`.

```
<?php
$im = pdf_open_gif($pdf, "test.gif");
```

```
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

pdf_open_png. Открытие рисунка PNG

`int` **`pdf_open_png`** `(int pdf, string png_file)`

Используйте функцию `pdf_open_image_file()`.

pdf_open_jpeg. Открытие рисунка JPEG

`int` **`pdf_open_jpeg`** `(int pdf_document, string filename)`

Используйте функцию `pdf_open_image_file()`.

pdf_open_tiff. Открытие рисунка TIFF

`int` **`pdf_open_tiff`** `(int PDF-document, string filename)`

Используйте функцию `pdf_open_image_file()`.

pdf_open_image_file. Чтение рисунка из файла

`int` **`pdf_open_image_file`** `(int PDF_document, string format, string filename)`

Загружает рисунок формата `format` из файла `filename` и возвращает его идентификатор. Возможные форматы: 'png', 'tiff', 'jpeg' и 'gif'.

```
<?php
   $pim = pdf_open_image_file($pdf, "png", "picture.png");
   pdf_place_image($pdf, $pim, 100, 100, 1);
   pdf_close_image($pdf, $pim);
?>
```

Функция заменяет: `pdf_open_jpeg()`, `pdf_open_gif()`, `pdf_open_tiff()`, `pdf_open_png()`. См. также: `pdf_close_image()`, `pdf_execute_image()`, `pdf_place_image()`, `pdf_put_image()`.

pdf_open_memory_image. Открытие рисунка созданного графическими функциями PHP

`int` **`pdf_open_memory_image`** `(int pdf_document, int image)`

Функция принимает дескриптор рисунка, созданного PHP и делает его доступным для документа pdf. Функция возвращает идентификатор рисунка pdf.

```
<?php // Example: Including a memory image
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
```

```
ImageFill($im, 10, 10, $col);
$pim = pdf_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

См. также: pdf_close_image(), pdf_execute_image(), pdf_place_image(), pdf_put_image().

pdf_close_image. Закрытие рисунка

void **pdf_close_image** (int image)

Закрывает рисунок, открытый функциями pdf_open_ ().

См. также: pdf_open_jpeg(), pdf_open_gif(), pdf_open_memory_image().

pdf_get_image_height. Высота рисунка в пикселях

string **pdf_get_image_height** (int pdf_document, int image)

См. также: pdf_open_image_file(), pdf_open_memory_image(), pdf_get_image_width().

pdf_get_image_width. Ширина рисунка в пикселях

string **pdf_get_image_width** (int pdf_document, int image)

См. также: pdf_open_image_file(), pdf_open_memory_image(), pdf_get_image_height().

pdf_place_image. Размещение рисунка на странице

void **pdf_place_image** (int pdf_doc, int image, double x, double y, double scale)

Позиция размещения задается (x, y); масштаб scale.

См. также: pdf_put_image().

pdf_put_image. Сохранение рисунка в PDF для дальнейшего использования

void **pdf_put_image** (int pdf_document, int image)

Внедряет рисунок в документ без его отбражения. Затем рисунок может быть размещен на странице функцией pdf_execute_image() необходимое число раз. Полезно при многократной вставке рисунка (уменьшается размер файла).

Начиная с версии 2.01 pdflib, функция бесполезна и выводит только предупреждение.

См. также: pdf_place_image(), pdf_execute_image().

pdf_execute_image. Размещение сохраненного рисунка на странице

void **pdf_execute_image** (int pdf_document, int image, double x, double y, double scale)

Отображает рисунок, внедренный функцией pdf_put_image().

Начиная с версии 2.01 pdflib, функция бесполезна и выводит только предупреждение.

Example 1. Multiple show of an image

```
<?php
$im = ImageCreate(100, 100);
$col1 = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col1);
$pim = pdf_open_memory_image($pdf, $im);
pdf_put_image($pdf, $pim);
pdf_execute_image($pdf, $pim, 100, 100, 1);
pdf_execute_image($pdf, $pim, 200, 200, 2); // 200 %
pdf_close_image($pdf, $pim);
?>
```

pdf_add_annotation. Добавление примечания

void **pdf_add_annotation** (int pdf_document, double llx, double lly, double urx, double ury, string title, string content)

Примечание располагается в левом нижнем углу (llx, lly), верхний правый угол (urx, ury).

pdf_set_border_style. Установка стиля обрамления примечаний и гиперссылок

void **pdf_set_border_style** (int pdf_document, string style, double width)

Аргумент style может принимать значения: 'solid' или 'dashed'. Ширина задается аргументом width.

См. также: pdf_set_border_color(), pdf_set_border_dash().

pdf_set_border_color. Установка цвета обрамления ссылок и примечаний

void **pdf_set_border_color** (int pdf_document, double red, double green, double blue)

Три компонента цвета могут принимать значения из диапазона от 0.0 до 1.0.

См. также: pdf_set_border_style(), pdf_set_border_dash().

pdf_set_border_dash. Установка стиля окантовки ссылок и примечаний

`void` **`pdf_set_border_dash`** `(int pdf_document, double black, double white)`

Устанавливает длину черных и белых полос прерывистых линий.

См. также: `pdf_set_border_style()`, `pdf_set_border_color()`.

ClibPDF

Библиотека ClibPDF позволяет PHP создавать документы PDF. Она не является полностью свободно распространяемой. Она действует подобно pdflib, но создает документы меньшего размера с большей скоростью.

Воспользуйтесь документацией поставляемой с ClibPDF при необходимости.

Все функции за исключением `cpdf_open()` принимают в качестве своего первого аргумента дескриптор открытого документа.

В настоящее время, ClibPDF в отличии от pdflib не позволяет одновременно работать с несколькими документами. Но, она способна создавать документы в памяти, не используя временный файл. Возможно также модифицировать любую страницу.

```
<?php
define("PPM", 2.83464566929); // поинтов в миллиметре
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842, PPM);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_begin_text($cpdf);
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_end_text($cpdf);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

Пример из комплекта pdflib можно использовать для сравнения двух библиотек.

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;

$pdf = cpdf_open(0);
```

```
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Analog Clock");

while($pagecount-- > 0) {
cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius
+ $margin), 1.0);

cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0); /* wipe */

cpdf_translate($pdf, $radius + $margin, $radius + $margin);
cpdf_save($pdf);
cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

/* minute strokes */
cpdf_setlinewidth($pdf, 2.0);
for ($alpha = 0; $alpha < 360; $alpha += 6)
{
cpdf_rotate($pdf, 6.0);
cpdf_moveto($pdf, $radius, 0.0);
cpdf_lineto($pdf, $radius-$margin/3, 0.0);
cpdf_stroke($pdf);
}

cpdf_restore($pdf);
cpdf_save($pdf);

/* 5 minute strokes */
cpdf_setlinewidth($pdf, 3.0);
for ($alpha = 0; $alpha < 360; $alpha += 30)
{
cpdf_rotate($pdf, 30.0);
cpdf_moveto($pdf, $radius, 0.0);
cpdf_lineto($pdf, $radius-$margin, 0.0);
cpdf_stroke($pdf);
}

$ltime = getdate();

/* draw hour hand */
cpdf_save($pdf);
cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) *
30.0);
cpdf_moveto($pdf, -$radius/10, -$radius/20);
cpdf_lineto($pdf, $radius/2, 0.0);
cpdf_lineto($pdf, -$radius/10, $radius/20);
cpdf_closepath($pdf);
cpdf_fill($pdf);
cpdf_restore($pdf);
```

```
/* draw minute hand */
cpdf_save($pdf);
cpdf_rotate($pdf, -(($ltime['seconds']/60.0)
                    + $ltime['minutes'] - 15.0) * 6.0);
cpdf_moveto($pdf, -$radius/10, -$radius/20);
cpdf_lineto($pdf, $radius * 0.8, 0.0);
cpdf_lineto($pdf, -$radius/10, $radius/20);
cpdf_closepath($pdf);
cpdf_fill($pdf);
cpdf_restore($pdf);

/* draw second hand */
cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
cpdf_setlinewidth($pdf, 2);
cpdf_save($pdf);
cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
cpdf_moveto($pdf, -$radius/5, 0.0);
cpdf_lineto($pdf, $radius, 0.0);
cpdf_stroke($pdf);
cpdf_restore($pdf);

/* draw little circle at center */
cpdf_circle($pdf, 0, 0, $radius/30);
cpdf_fill($pdf);

cpdf_restore($pdf);

cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
cpdf_save_to_file($pdf,"e:\\clock.pdf");
cpdf_close($pdf);
?>
```

cpdf_global_set_document_limits. Установка ограничения для всех документов

`void` **`cpd8f_global_set_document_limits`** `(int maxpages, int maxfonts, int maximages, int maxannotations, int maxobjects)`

Функцию следует вызывать до `cpdf_open()`.

См. также: `cpdf_open()`.

cpdf_set_creator. Заполнение поля «создатель документа»

```
void cpdf_set_creator (string creator)
```

См. также: `cpdf_set_subject()`, `cpdf_set_title()`, `cpdf_set_keywords()`.

cpdf_set_title. Заполнение поля «заголовок документа»

```
void cpdf_set_title (string title)
```

См. также: `cpdf_set_subject()`, `cpdf_set_creator()`, `cpdf_set_keywords()`.

cpdf_set_subject. Заполнение поля «тема документа»

```
void cpdf_set_subject (string subject)
```

См. также: `cpdf_set_title()`, `cpdf_set_creator()`, `cpdf_set_keywords()`.

cpdf_set_keywords. Заполнение поля «ключевые слова документа»

```
void cpdf_set_keywords (string keywords)
```

См. также: `cpdf_set_title()`, `cpdf_set_creator()`, `cpdf_set_subject()`.

cpdf_open. Открытие нового документа

```
int cpdf_open (int compression [, string filename])
```

Первым аргументом можно указать необходимость сжатия документа (если он не равен 0). Если указывается второй аргумент, то документ будет создаваться в файле, а не в памяти. Указание имени файла "-" указывает на стандартный поток вывода (это пока не работает с apache).

Возвращаемый дескриптор используется всеми последующими функциями.

См. также: `cpdf_close()`, `cpdf_output_buffer()`, `cpdf_save_to_file()`.

cpdf_close. Закрытие документа pdf

```
void cpdf_close (int pdf_document)
```

Эту функцию следует вызывать последней, даже после `cpdf_finalize()`, `cpdf_output_buffer()` или `cpdf_save_to_file()`.

См. также: `cpdf_open()`.

cpdf_page_init. Начать новую страницу

`void` **`cpdf_page_init`** `(int pdf_document, int page_number, int orientation, double height, double width [, double unit])`

Параметры новой страницы: `page_number` – ее номер, `orientation` – ориентация (0 – вертикальная, 1 – горизонтальная), высота `height` и ширина `width`; необязательный аргумент `unit` указывает разрешение в точках на дюйм (по ум. 72).

См. также: `cpdf_set_current_page()`.

cpdf_finalize_page. Завершение указанной страницу

`void` **`cpdf_finalize_page`** `(int pdf_document, int page_number)`

После этого страница не может модифицироваться.

См. также: `cpdf_page_init()`.

cpdf_finalize. Завершение документа

`void` **`cpdf_finalize`** `(int pdf_document)`

Поле этого необходимо вызвать `cpdf_close()`

См. также: `cpdf_close()`.

cpdf_output_buffer. Вывод документа из буфера памяти

`void` **`cpdf_output_buffer`** `(int pdf_document)`

Выводит документ на стандартный поток вывода (отсылает браузеру). Для этого документ должен быть создан в памяти (а не файле). См. также: `cpdf_open()`.

cpdf_save_to_file. Запись документа в файл

`void` **`cpdf_save_to_file`** `(int pdf_document, string filename)`

Функция не требуется, если документ изначально создавался в файле.

См. также: `cpdf_output_buffer()`, `cpdf_open()`.

cpdf_set_current_page. Установка текущей страницы

`void` **`cpdf_set_current_page`** `(int pdf_document, int page number)`

Все последующие операции будут проводиться на данной странице. Между страницами можно переключаться до вызова `cpdf_finalize_page()`.

См. также: `cpdf_finalize_page()`.

cpdf_begin_text. Начать текстовый раздел

`void` **`cpdf_begin_text`** `(int pdf_document)`

Раздел должен быть завершен функцией `cpdf_end_text()`.

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

См. также: `cpdf_end_text()`.

cpdf_end_text. Завершение текстового раздела

`void` **`cpdf_end_text`** `(int pdf_document)`

Завершает раздел, начатый `cpdf_begin_text()`.

См. также: `cpdf_begin_text()`.

cpdf_show. Вывод текста в текущую позицию

`void` **`cpdf_show`** `(int pdf_document, string text)`

См. также: `cpdf_text()`, `cpdf_begin_text()`, `cpdf_end_text()`.

cpdf_show_xy. Вывод текста в указанную позицию

`void` **`cpdf_show_xy`** `(int pdf_doc, string text, double x, double y [, int mode])`

Аргументом `mode` можно указать используемое разрешение (если указывается 0, то используется разрешение по умолчанию).

Функция идентична `cpdf_text()` без необязательных аргументов.

См. также: `cpdf_text()`.

cpdf_text. Вывод текста с параметрами

`void` **`cpdf_text`** `(int pdf_document, string text, double x, double y [, int mode [, double orientation [, int alignmode]]])`

Параметром `orientation` задается вращение строки в градусах, а `alignmode` – выравнивание текста. См. документацию ClibPdf относительно возможных значений.

См. также: `cpdf_show_xy()`.

cpdf_set_font. Выбор текущего шрифта

`void` **`cpdf_set_font`** `(int pdf_doc, string font_name, double size, string encoding)`

Выбирает текущий шрифт, его размер и кодировку. В настоящее время поддерживаются только стандартные шрифты postscript. Кодировка может быть указана значениями: «MacRomanEncoding», «MacExpertEncoding», «WinAnsiEncoding», и «NULL» (использовать встроенную).

См. также: документацию ClibPDF.

cpdf_set_leading. Установка межстрочного расстояния

`void` **`cpdf_set leading`** `(int pdf_document, double distance)`

Используется при выводе текста функцией `cpdf_continue_text()`.

См. также: `cpdf_continue_text()`.

cpdf_set_text_rendering. Установка режима вывода текста

`void cpdf_set_text_rendering (int pdf_document, int mode)`

Возможные значения аргумента `mode`: 0 (текст с заполнением), 1 (контурный текст), 2 (заполненный контурный текст), 3 (невидимый), 4 (заполненный текст, прикрепленный к пути), 5 (контурный текст, прикрепленный к пути), 6 (заполненный контурный текст, прикрепленный к пути), 7 (текст, прикрепленный к пути).

cpdf_set_horiz_scaling. Установка горизонтального разрешения

`void` **`cpdf_set_horiz_scaling`** `(int pdf_document, double scale)`

cpdf_set_text_rise. Установка подъема текста

`void` **`cpdf_set_text_rise`** `(int pdf_document, double value)`

cpdf_set_text_matrix. Установка матрицы преобразования шрифта

`void` **`cpdf_set_text_matrix`** `(int pdf_document, array matrix)`

cpdf_set_text_pos. Установка позиции текста

`void` **`cpdf_set_text_pos`** `(int pdf_document, double x, double y [, int mode])`

Устанавливает позицию вывода текста последующим вызовом `cpdf_show()`.

Параметром `mode` можно указать разрешение. См. также: `cpdf_show()`, `cpdf_text()`.

cpdf_set_char_spacing. Установка межсимвольного интервала

`void` **`cpdf_set_char_spacing`** `(int pdf_document, double space)`

См. также: `cpdf_set_word_spacing()`, `cpdf_set_leading()`.

cpdf_set_word_spacing. Установка интервала между словами

`void` **`cpdf_set_word_spacing`** `(int pdf_document, double space)`

См. также: `cpdf_set_char_spacing()`, `cpdf_set_leading()`.

cpdf_continue_text. Вывод текста со следующей строки

`void` **`cpdf_continue_text`** `(int pdf_document, string text)`

См. также: `cpdf_show_xy()`, `cpdf_text()`, `cpdf_set_leading()`, `cpdf_set_text_pos()`.

cpdf_stringwidth. Вычисление ширины строки используя текущий шрифт

`double` **`cpdf_stringwidth`** `(int pdf_document, string text)`

См. также: `cpdf_set_font()`.

cpdf_save. Сохранение текущих установок

`void` **`cpdf_save`** `(int pdf_document)`

См. также: `cpdf_restore()`.

cpdf_restore. Восстановление ранее сохраненных установок

`void` **`cpdf_restore`** `(int pdf_document)`

Восстанавливает установки, сохраненные `cpdf_save()`.

```
<?php
cpdf_save($pdf);
// всякие вращения и трансформации ...
cpdf_restore($pdf)
?>
```

См. также: `cpdf_save()`.

cpdf_translate. Установка начала системы координат

`void` **`cpdf_translate`** `(int pdf_doc, double x, double y [, int mode])`

cpdf_scale. Установка масштабирования

```
void cpdf_scale (int pdf_document, double x-scale, double y-scale)
```

cpdf_rotate. Установка угла вращения в градусах

```
void cpdf_rotate (int pdf_document, double angle)
```

cpdf_setflat. Установка равномерности

```
void cpdf_setflat (int pdf_document, double value)
```

Возможные значения параметра от 0 до 100.

cpdf_setlinejoin. Установка параметра linejoin

```
void cpdf_setlinejoin (int pdf_document, long value)
```

Возможные значения параметра от 0 до 2. (0 = miter, 1 = round, 2 = bevel).

cpdf_setlinecap. Установка параметра linecap

```
void cpdf_setlinecap (int pdf_document, int value)
```

Возможные значения параметра от 0 до 2. (0 = butt end, 1 = round, 2 = projecting square).

cpdf_setmiterlimit. Установка параметра miter limit

```
void cpdf_setmiterlimit (int pdf_document, double value)
```

Возможные значения параметра: 1 и более.

cpdf_setlinewidth. Установка ширины строк

```
void cpdf_setlinewidth (int pdf_document, double width)
```

cpdf_setdash. Установка вида штриховки

```
void cpdf_setdash (int pdf_document, double white, double black)
```

Устанавливает размер белых и черных полос. Если оба аргумента равны 0, то линия будет непрерывной.

cpdf_newpath. Начать новый путь

```
void cpdf_newpath (int pdf_document)
```

cpdf_moveto. Установка текущей точки

```
void cpdf_moveto (int pdf_document, double x, double y [, int mode])
```

cpdf_rmoveto. Смещение текущей точки

`void` **`cpdf_rmoveto`** `(int pdf_doc, double x, double y [, int mode])`

См. также: `cpdf_moveto()`.

cpdf_curveto. Начертить кривую

`void` **`cpdf_curveto`** `(int pdf_document, double x1, double y1, double x2, double y2, double x3, double y3 [, int mode])`

Чертит кривую Безье от текущей точки до (`x3,y3`), используя точки (`x1,y1`) и (`x2,y2`) как ориентирующие.

См. также: `cpdf_moveto()`, `cpdf_rmoveto()`, `cpdf_rlineto()`, `cpdf_lineto()`.

cpdf_lineto. Начертить линию

`void` **`cpdf_lineto`** `(int pdf_document, double x, double y [, int mode])`

Чертит линию от текущей точки до указанной (`x,y`).

См. также: `cpdf_moveto()`, `cpdf_rmoveto()`, `cpdf_curveto()`.

cpdf_rlineto. Начертить отрезок

`void` **`cpdf_rlineto`** `(int pdf_document, double x, double y [, int mode])`

Координаты (`x,y`) указываются относительно текущей точки.

См. также: `cpdf_moveto()`, `cpdf_rmoveto()`, `cpdf_curveto()`.

cpdf_circle. Начертить окружность

`void` **`cpdf_circle`** `(int pdf_document, double x, double y, double radius [, int mode])`

См. также: `cpdf_arc()`.

cpdf_arc. Начертить дугу

`void` **`cpdf_arc`** `(int pdf_document, double x-coor, double y-coor, double radius, double start, double end [, int mode])`

Начальный и конечный угол задаются `start` и `end`.

См. также: `cpdf_circle()`.

cpdf_rect. Начертить прямоугольник

`void` **`cpdf_rect`** `(int pdf_document, double x, double y, double width, double height [, int mode])`

Левый нижний угол задается (x, y); высота и ширина height, width.

cpdf_closepath. Завершить текущий путь

```
void cpdf_closepath (int pdf_document)
```

cpdf_stroke. Заштриховать путь

```
void cpdf_stroke (int pdf_document)
```

См. также: cpdf_closepath(), cpdf_closepath_stroke().

cpdf_closepath_stroke. Прочертить и закрыть путь

```
void cpdf_closepath_stroke (int pdf_document)
```

Это комбинация cpdf_closepath() и cpdf_stroke().

См. также: cpdf_closepath(), cpdf_stroke().

cpdf_fill. Заполнение пути текущим цветом

```
void cpdf_fill (int pdf_document)
```

См. также: cpdf_closepath(), cpdf_stroke(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_fill_stroke. Заполнение пути цветом и закрытие его

```
void cpdf_fill_stroke (int pdf_document)
```

См. также: cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_closepath_fill_stroke. Начертить, закрасить и закрыть путь

```
void cpdf_closepath_fill_stroke (int pdf_document)
```

См. также: cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_clip. Прикрепение всех линий к текущему пути

```
void cpdf_clip (int pdf_document)
```

cpdf_setgray_fill. Установка заполнения серым цветом

```
void cpdf_setgray_fill (int pdf_document, double value)
```

См. также: cpdf_setrgbcolor_fill().

cpdf_setgray_stroke. Установка штриховки серым цветом

void **cpdf_setgray_stroke** (int pdf_document, double gray value)

См. также: cpdf_setrgbcolor_stroke().

cpdf_setgray. Установка заполнения и штриховки серым цветом

void **cpdf_setgray** (int pdf_document, double gray value)

См. также: cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_setrgbcolor_fill. Установка заполнения цветом rgb

void **cpdf_setrgbcolor_fill** (int pdf_doc, double red, double green, double blue)

См. также: cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor().

cpdf_setrgbcolor_stroke. Установка штриховки цветом rgb

void **cpdf_setrgbcolor_stroke** (int pdf_document, double red value, double green value, double blue value)

См. также: cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_setrgbcolor. Установка заполнения и штриховки серым цветом rgb

void **cpdf_setrgbcolor** (int pdf_document, double red_value, double green_value, double blue_value)

См. также: cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_add_outline. Добавление закладки для текущей страницы

void **cpdf_add_outline** (int pdf_document, string text)

Название закладки определяется аргументом text.

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
// ...
// some drawing
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
```

```
cpdf_close($cpdf);
?>
```

cpdf_set_page_animation. Установка режима перехода между страницами

`void` **`cpdf_set_page_animation`** `(int pdf_document, int transition, double duration)`

Аргумент `transition` определяет метод перехода, а `duration` его длительность в сек.

cpdf_import_jpeg. Открытие рисунка JPEG

`int` **`cpdf_import_jpeg`** `(int pdf_doc, string filename, double x, double y, double angle, double width, double height, double x-scale, double y-scale [, int mode])`

Открывает рисунок из файла `filename` и размещает его в позиции (`x,y`) на текущей странице. Для рисунка можно указать угол поворота в градусах, и масштабирование.

См. также: `cpdf_place_inline_image()`.

cpdf_place_inline_image. Размещение сгенерированного рисунка на странице

`void` **`cpdf_place_inline_image`** `(int pdf_document, int image, double x, double y, double angle, double width, double height [, int mode])`

Внедряет рисунок, созданный графическими функциями PHP.

См. также: `cpdf_import_jpeg()`.

cpdf_add_annotation. Добавление примечания

`void` **`cpdf_add_annotation`** `(int pdf_document, double llx, double lly, double urx, double ury, string title, string content [, int mode])`

Примечание располагается в левом нижнем углу (`llx, lly`), верхний правый угол (`urx, ury`).

Forms Data Format

Функции форматирования данных форм (Forms Data Format, FDF) предназначены для работы с формами Pdf документов. Для предварительного ознакомления ознакомьтесь с документом http://partners.adobe.com/asn/developer/acrosdk/forms.html.

Основная идея FDF сходна с формами HTML. Отличие состоит в передаче (отправке) данных, поскольку передаются как сами данные (FDF), так и документ формы (PDF). В обычном случае, FDF форма с данными создается на основе PDF документа, используемого как шаблон формы. То есть, сначала создается FDF документ

(fdf_create()), и устанавливаются значения для каждого поля ввода (fdf_set_value()), а затем ассоциируется форма PDF (fdf_set_file()). После чего, происходит отправка FDF формы браузеру (с заголовком: MimeType application/vnd.fdf). Надстройка браузера (Acrobat reader plugin) распознает MimeType, и загружает ассоциированную PDF форму, заполняя ее данными FDF документа.

Следующий пример иллюстрирует обработку данных формы.

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
// pdf form содержит значения полей: date, comment,
// publisher, и checkbox: show_publisher.
$fdf = fdf_open("test.fdf");

$date = fdf_get_value($fdf, "date");
echo "The date field has the value '<B>$date</B>'<BR>";

$comment = fdf_get_value($fdf, "comment");
echo "The comment field has the value '<B>$comment</B>'<BR>";

if(fdf_get_value($fdf, "show_publisher") == "On") {
$publisher = fdf_get_value($fdf, "publisher");
echo "The publisher field has the value '<B>$publisher</B>'<BR>";
} else
echo "Publisher shall not be shown.<BR>";

fdf_close($fdf);
?>
```

fdf_open. Открытие документа FDF

```
int fdf_open (string filename)
```

Открываемый файл должен содержать данные, возвращенные формой PDF. В настоящее время файл необходимо создавать вручную, используя fopen() и записывая данные HTTP_FDF_DATA функцией fwrite().

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);
```

```
// Open temp file and evaluate data
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

См. также: `fdf_close()`.

fdf_close. Закрытие документа FDF

`boolean` **`fdf_close`** `(int fdf_document)`

См. также: `fdf_open()`.

fdf_create. Создание нового документа FDF

`int` **`fdf_create`** `(void)`

Функция необходима для установки значений полей.

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "volume", $volume, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

См. также: `fdf_close()`, `fdf_save()`, `fdf_open()`.

fdf_save. Сохранение документа FDF

`int` **`fdf_save`** `(string filename)`

Если имя файла указано как '.', то документ будет выводиться на стандартный поток вывода (отсылаться браузеру). Это не действует, если PHP является модулем apache (для вывода используйте `fpassthru()`).

См. также: `fdf_close()` и пример в `fdf_create()`.

fdf_get_value. Получение значения поля

`string` **`fdf_get_value`** `(int fdf_document, string fieldname)`

См. также: `fdf_set_value()`.

fdf_set_value. Установка значения поля

`bool` **`fdf_set_value`** `(int fdf_document, string fieldname, string value, int isName)`

Последний аргумент определяет, будет ли значение поля преобразовываться в PDF Name (`isName` = 1), или в PDF String (`isName` = 0).

См. также: `fdf_get_value()`.

fdf_next_field_name. Получение имени следующего поля

`string` **`fdf_next_field_name`** `(int fdf_document, string fieldname)`

Возвращает имя первого поля (если аргумент `fieldname` имеет значение NULL), или имя поля, следующего за полем `fieldname`.

См. также: `fdf_set_field()`, `fdf_get_field()`.

fdf_set_ap. Установка отображения поля

`bool` **`fdf_set_ap`** `(int fdf_document, string field_name, int face, string filename, int page_number)`

Устанавливается значение ключа /AP. Возможные значения аргумента `face`: 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_status. Установка ключа /STATUS

`bool` **`fdf_set_status`** `(int fdf_document, string status)`

См. также: `fdf_get_status()`.

fdf_get_status. Получение значения ключа /STATUS

`string` **`fdf_get_status`** `(int fdf_document)`

См. также: `fdf_set_status()`.

fdf_set_file. Назначение документа формы, установить ключ /F

`bool` **`fdf_set_file`** `(int fdf_document, string filename)`

Ключ /F – это ссылка на документ PDF, который заполняется данными, обычно задаваемый URL (например: http:/testfdf/resultlabel.pdf).

См. также: `fdf_get_file()` and example for `fdf_create()`.

fdf_get_file. Получение значения ключа /F

`string` **`fdf_get_file`** `(int fdf_document)`

См. также: `fdf_set_file()`.

fdf_set_flags. Установка флага поля

```
bool fdf_set_flags (int fdf_doc, string fieldname, int whichFlags,
int newFlags)
```

См. также: `fdf_set_opt()`.

fdf_set_opt. Установка опции поля

```
bool fdf_set_opt (int fdf_document, string fieldname, int element,
string str1, string str2)
```

См. также: `fdf_set_flags()`.

fdf_set_submit_form_action. Установка действия javascript для формы

```
bool fdf_set_submit_form_action (int fdf_document, string fieldname,
int trigger, string script, int flags)
```

См. также: `fdf_set_javascript_action()`.

fdf_set_javascript_action. Установка действия javascript для поля

```
bool fdf_set_javascript_action (int fdf_document, string fieldname,
int trigger, string script)
```

См. также: `fdf_set_submit_form_action()`.