گرافیک بیتی پویا در php با استفاده از GD

سطح : مقدماتی

مؤلف : **Jacek Artymiak**

مترجم : مسعود امجدی

بخش اول - قبل از اینکه شروع کنید

در مورد این آموزش

اگر در کار یا پروژه تان نیاز به نوشتن برنامه های گرافیکی با php را داشته باشید، این آموزش می تواند شما را در تولید تصاویر بیتی پویا با استفاده از php و کتابخانه ی gd یاری کند.

یک عکس بهتر از هزار کلمه است، مخصوصا اگر از آن برای نشان دادن تجارت، امور مالی یا اطلاعات علمی استفاده شود. اگر شما بتوانید تصاویر را با توجه به درخواست ها ایجاد کنید، تاثیر آنها چندین برابر می شود و می تواند آخرین اطلاعات را در قالب تصاویر به تصویر بکشد.

اهداف

در پایان این آموزش شما قادر خواهید بود اسکریپت هایی را برای ایجاد تصاویر بیتی با php و کتابخانه ی gd بنویسید. شما خواهید توانست به راحتی اطلاعات تجاری ، مالی و علمی را به تصویر بکشید. به ویژه شما موارد زیر را می آموزید :

- اصول تنظیم سیستم gd

- نکات مخصوص سیستم عامل برای تسهیل تنظیم سیستم
- نکات مربوط به نصب php/gd
- مراحل آماده سازی canvas , color palette , background fill/image
- اصول اولیه ی طراحی با gd
- تمرین هایی برای کمک به ارائه ی نتایج تصویری داده ها
- نکاتی پیرامون انتخاب فرمت مناسب برای فایل ها
- دستورالعمل هایی برای جا دادن اسکریپت هایتان در کد های html

سیستم مورد نیاز

شما نیاز دارید تا به :

- یک سیستم عامل خوب دسترسی داشته باشید. من سیستم عامل های زیر را پیشنهاد می کنم :
  - Linux
  - Free/Net/OpenBSD
  - Microsoft Windows XP/7
  - Mac OS X

دیگر سیستم های unix پایه هم در صورتی که بتوانند Apache , Php , gd را اجرا کنند می توانند مورد استفاده قرار بگیرند.

- یک CPU با کلاک 100MHz یا بیشتر
- 32MB رم یا بیشتر
- 500MB هارد دیسک

علاوه بر این شما می توانید از یک HTTP server که php را ساپورت می کند استفاده کنید که در این صورت به کتابخانه های زیر نیز نیاز خواهید داشت:

- freetype2
- zlib
- libpng
- libjpeg6

ساخت ابزار مورد نیاز با استفاده از سورس کد آنها

اگر شما نتوانید بسته های کامپایل شده ی آماده را برای سیستم تان پیدا کنید مجبور خواهید شد خودتان آنها را با استفاده از یک کامپایلر ANSI C/C++ همانند GCC بسازید.

بسته های مورد نیاز را می توانید از لینک های زیر دریافت کنید :

- Apache: http://www.apache.org/dist/httpd/
- PHP: http://www.php.net/downloads.php
- gd: http://www.boutell.com/gd/
- libpng: http://www.libpng.org/pub/png/libpng.html
- zlib: http://www.gzip.org/zlib/
- libjpeg: http://www.ijg.org/
- freetype2 (optional, required for rendering TrueType fonts): http://www.freetype.org/
- t1lib (optional, required for rendering Type 1 fonts):

http://freshmeat.net/projects/t1lib/

برای اطلاعات بیشتر در مورد نحوه ی نصب و ساخت بسته های بالا می توانید به فایل های README یا INSTALL هر کدام از آنها مراجعه کنید. شما باید بسته ها را به ترتیب زیر نصب کنید:

- zlib
- libpng
- libjpeg
- freetype2
- t1lib
- gd
- Apache
- Php

اکثر این بسته ها را می توان با کد زیر نصب کرد :

```
$ ./configure
$ ./make
$ sudo ./make install
```

ابزار های اضافی

دیگر ابزاری که برای این کار نیاز دارید یک مرورگر وب و یک تکست ادیتور می باشد. برای مرورگر می توانید از IE , Chrome , Opera , Mozilla Firefox و ... استفاده کنید. برای text editor هم می توانید از BBEdit , emacs , vi برای لینوکس و Notepad , Wordpad , ... برای ویندوز استفاده کنید.

تست php/gd

زمانی که تمامی ابزار های مورد نیاز را نصب کردید می توانید با استفاده از دو اسکریپت ساده زیر مطمئن شوید که gd شما فعال شده است و کار می کند.

ابتدا یک صفحه ی ساده php ایجاد کرده و کد زیر را در آن بنویسید :

```
<?php
	phpinfo();
?>
```

اسکریپت بالا را با نام test.php در دایرکتوریه www خود ذخیره کرده و با مرورگر اجرا کنید.

به صورت :

```
http://localhost/test.php
```

در این صفحه شما جدول بزرگی از تمامی موارد php خواهید دید. به قسمت gd رفته و بررسی کنید که آیا فعال شده است یا نه ؟

gd

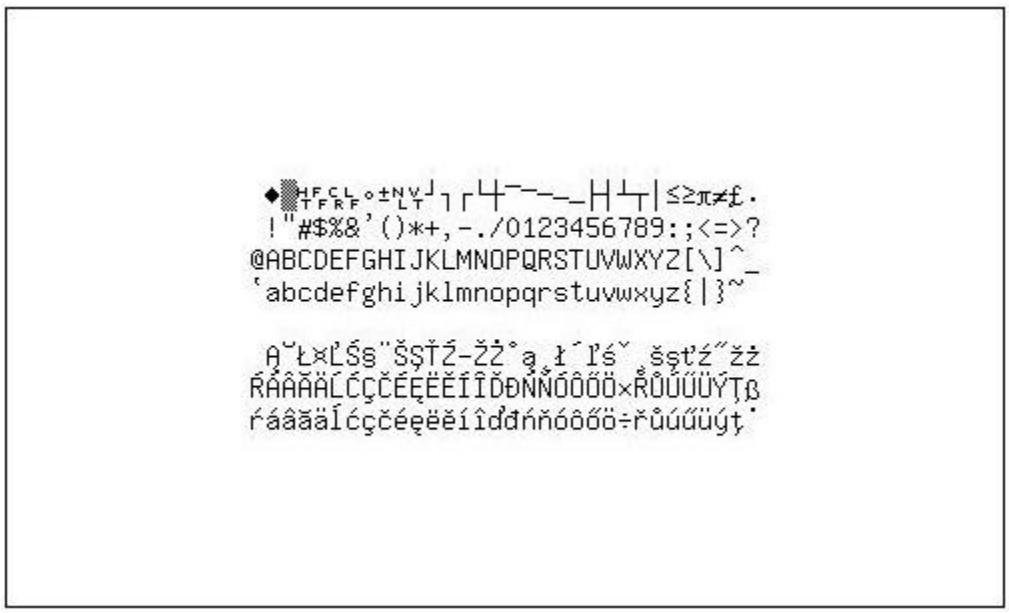
GD Support	enabled
GD Version	bundled (2.0.34 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.1.9
T1Lib Support	enabled
GIF Read Support	enabled
GIF Create Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled
XBM Support	enabled

سپس اسکریپت زیر را برای تست gd بنویسید:

```
<?php
	header ("Content-Type: iamge/jpeg");

	#set the dimentions of the canvas
	$cw = 500;
	$ch = 300;

	#create canvas
	$c = imagecreate ($cw , $ch);

	#generate color palette
	for ($n=0 ; $n <=255 ; $n++)
	{
		$cols[$n] = imagecolorallocate ($c , $n , $n , $n);
	}

	#create background
	imagefilledrectangle ($c , 0 , 0 , 600 , 400 , $cols[255]);
```

```
	#display all characters in a font
	$font = 4;
	$dx = imagefontwidth($font);
	$dy = imagefontheight($font);
	$x = ($cw / 2) - (16 * $dx);
	$y = ($ch / 2) - (4 * $dy);
	$m = 0;
	$z = 0;
	for ($n = 0; $n <= 7; $n++)
	{
	for ($m = 0; $m <= 31; $m++)
	{
	imagechar($c, $font, $x + $dx * $m, $y, chr($z), $cols[0]);
	$z++;
	}
	$y = $y + $dy;
	}

	#draw a nice thin border around the edges of the image
	imagerectangle($c, 0, 0, $cw-1, $ch-1, $cols[0]);

	#generate image
	imagejpeg($c);
?>
```

نتیجه ی کد بالا شکل زیر خواهد بود:



بخش دوم - برنامه ی خودتان را بنویسید: قدم های اول

با header درست شروع کنید

حالا که تست های اولیه را انجام دادید، وقت آن است که اسکریپت های gd خودتان را بنویسید.
برای این کار شما باید ساختار های عمومی یک اسکریپت gd را یاد بگیرید.
هر اسکریپتی که گرافیک ایجاد می کند باید با کد زیر شروع شود:

```
header ("Content-Type: image/jpeg");
```

با استفاده از تابع header() شما HTTP headers را که توسط سرویس دهنده ی HTTP (HTTP Server) به سرویس گیرنده HTTP (HTTP Client) ارسال میگردد را مقدار دهی می کنید. یکی از این header ها ، Content-Type می باشد که به HTTP Client اعلام می کند که چه نوع داده ای دریافت خواهد کرد. client از این اطلاعات برای تصمیم گیری در مورد نحوه ی اجرای داده های دریافتی استفاده می کند. سپس ممکن است client برای پردازش این داده ها از مبدل های خارجی استفاده کند یا پیغامی مبنی بر اینکه به تنهایی نمی تواند این اطلاعات را پردازش کند، صادر نماید.
برای مثال ، image/jpeg به client می گوید که یک عکس بیتی در فرمت jpeg را دریافت خواهد کرد. حالا client می تواند تصمیم بگیرد که آیا داده ی دریافتی را پردازش کند، از یک نمایشگر خارجی استفاده کند یا به کاربر بگوید که نمی تواند این نوع خاص تصویر را نمایش دهد.
بعد از اینکه Content-Type به مرورگر ارسال شد، اسکریپت شما می تواند کاری را که برای آن منظور طراحی شده است را انجام دهد. به خاطر بسپارید که تنها خروجی اسکریپت شما رشته های بیتی می باشد که تصویر را تشکیل می دهند. این امر باعث می شود تا نتوانید از دستوراتی مثل echo و print برای نوشتن کد های html در آن اسکریپت استفاده کنید.

اما نگران نباشید ، در ادامه ی همین آموزش نحوه ی افزودن کد های html را به یک اسکریپت gd بررسی خواهیم کرد. فعلا نحوه ی ساختن یک تصویر bitmap با gd را یاد بگیرید.

آماده سازی canvas (پرده نقاشی)

قبل از اینکه چیزی بکشید مثل یک نقاش واقعی اول پرده نقاشی (canvas) را آماده کنید. این کار را می توانید با فراخوانی تابع imagecreate() انجام دهید:

```
#set the dimentions of the canvas
$cw = 500;
$ch = 300;

#create canvas
$c = imagecreate ($cw , $ch);
```

عددی که تابع imagecreate() بر میگرداند توسط تمامی توابعی که با عکس کار دارند مورد استفاده قرار می گیرد. این کار به شما این امکان را میدهد که بتوانید بیش از یک عکس را در یک اسکریپت ایجاد کنید. بهر حال، استفاده از این ویژگی محدودیت هایی هم دارد، چرا که اسکریپت می تواند تنها یک عکس را به HTTP Client ای که فراخوانی اش کرده، بفرستد.

از طرفی دیگر همان طور که بعدا خواهید دید، می توانید دو یا چند عکس را با هم ترکیب کنید و یا عکس ها را در local disk یا server ذخیره کنید.

توجه کنید که مقادیر dimension که طول و عرض عکس را مشخص می کنند باید با مقادیر width و height در تگ <img> برابر باشند. در غیر این صورت تصویر نمایش داده می شود، اما به هم می ریزد. حتما به این مورد دقت کنید.

ایجاد Palette (جعبه ی رنگ)

اکنون وقت آن رسیده که کمی با رنگ ها بازی کنید. اگر چه gd 2.x می تواند با palette چند میلیون رنگی کار کند اما بهتر است شما اندازه ی palette تان را 256 رنگ انتخاب کنید تا هم با gd 1.8.x هماهنگ باشد و هم با نسخه های بالاتر gd.

هر رنگ در palette با فراخوانی تابع imagecolorallocate() تولید می شود که این تابع چهار آرگومان دریافت میکند :

- جایگاه رنگ در palette
- مقدار مشخصه ی red عددی بین 0-255
- مقدار مشخصه ی green عددی بین 0-255
- مقدار مشخصه ی blue عددی بین 0-255

برای مثال کد زیر یک palette خاکستری ایجاد می کند:

```
# Generate color palette
for ($n = 0; $n <= 255; $n++)
{
      $cols[$n] = imagecolorallocate($c, $n, $n, $n);
}
```

اگرچه نیازی نیست شما کل palette را در ابتدا برنامه تعریف کنید و می توانید آن را بعدا در طول اسکریپت هم تغییر دهید، اما تعریف آن در ابتدای کد خیلی زیباتر است. برای دوباره تعریف کردن یک رنگ از تابع imagecolorallocate() با همان canvas ID و color ID اما با مقادیر متفاوتی از R , G , B ، استفاده کنید. شما همچنین می توانید یک رنگ را با استفاده از تابع imagecolordeallocate() از palette پاک کنید، مثل :

```
imagecolordeallocate($c , $cols[7]);
```

آماده سازی Background (پس زمینه)

اکنون شما آماده اید تا شاهکار خود را رسم کنید. شما آزادید تا از هر تکنیکی که می خواهید استفاده کنید، اما به یاد داشته باشید که اگر چیزی را رسم کردید نمی تواند ناقص بماند، بنابراین روشی که عکس خود را رسم می کنید مهم است.

بنابر این اولین کار برای انجام، رنگ کردن canvas با استفاده از یک رنگ ثابت و یا یک تصویر import شده است. مانند کد زیر :

```
# Create Background
ImageFilledRectangle($c, 0, 0, 499, 299, $cols[255]);
```

هیچ تابع خاصی پس زمینه را ایجاد نمی کند، بنابرین شما می توانید از تابع imagefilledrectangle() استفاده کنید که یک مستطیل رنگی را بر روی canvas رسم می کند. آرگومان اول این تابع ID مستطیلی است که رسم شده است. چهار آرگومان بعدی به ترتیب مختصات گوشه ی بالا سمت چپ و گوشه ی پایین سمت راست می باشد. آرگومان آخر هم ID رنگی است که به مستطیل زده شده است. البته می توانید از تابع imagefilledrectangle() برای رسم مستطیل در هر جای دیگری از اسکریپت استفاده کنید.

توجه کنید که تصاویر gd از گوشه ی بالا سمت چپ شروع شده و به سمت راست و پایین گسترش می یابد.

روش دوم استفاده از فایل های آماده به عنوان background می باشد. برای این منظور می توانید از توابع زیر استفاده کنید:

- ImageCreateFromGD()
- ImageCreateFromGD2()
- ImageCreateFromGD2Part()
- ImageCreateFromGIF()
- ImageCreateFromJPEG()
- ImageCreateFromPNG()
- ImageCreateFromString()
- ImageCreateFromWBMP()
- ImageCreateFromXBM()
- ImageCreateFromXPM()

- ImageCreateFromJPEG()

آرگومان های این توابع می توانند مسیر فایل های local و یا url فایل های اینترنتی باشند.
این روش دسترسی به فایل ها، امکان ویژه ای را برای نوشتن اسکریپت هایی فراهم می کند که تصاویر به روز شده مثل تصاویر ماهواره ای یا تصاویر خبری و ... را به صورت دینامیک دانلود می کند و شما می توانید اطلاعاتی را که خودتان می خواهید به آنها اضافه کرده و یا آنها را با تصاویر دیگر ترکیب کنید. اگر gd 2.x را نصب کرده باشید، این ویژگی ها بهتر کار خواهند کرد و شما از یک palette چند میلیون رنگی بهره مند خواهید بود.
زمانی که شما تصویری را وارد می کنید، ممکن است اندازه آن را لازم داشته باشید که می توانید آن را از خروجی تابع getimagesize() دریافت نمایید.

بخش سوم – اصول اولیه طراحی با gd

با پیکسل ها بازی کنید

پس از اینکه baclground آماده شد می توانید هر آنچه را که می خواهید با استفاده از توابع زیر رسم کنید:

- ImageSetPixel()
- ImageLine()
- ImageDashedLine()
- ImageRectangle()
- ImageFilledRectanlge()
- ImagePolygon()
- ImageFilledPolygon()
- ImageArc()
- ImageFilledArc()

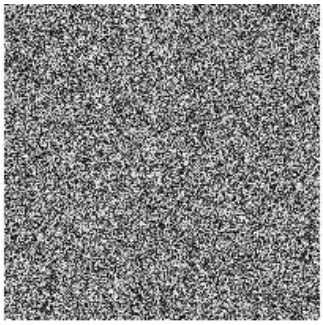
تابع اول، یعنی imagesetpixel() یک پیکسل تنها را می کشد. این تابع فقط به 4 آرگومان نیاز دارد :

- Canvas ID
- X Coordinates
- Y Coordinates
- Color ID

شما می توانید از این تابع برای افزودن جزئیات به تصاویرتان استفاده کنید و یا از تابع random() استفاده کنید، همانند مثال زیر :

```
<?php
	header("Content-Type: image/jpeg");

	#set the dimensions of the canvas
	$cw = 200;
	$ch = 200;

	#create canvas
	$c = imagecreate($cw, $ch);

	#generate color palette
	for ($n = 0; $n <= 255; $n++)
	{
	$cols[$n] = imagecolorallocate($c, $n, $n, $n);
	}

	#create background
	imagefilledrectangle($c, 0, 0, $cw, $ch, $cols[255]);

	#draw some pixels
	for ($n = 0; $n < $cw; $n++)
	{
	for ($m = 0; $m < $ch; $m++)
	{
	imagesetpixel($c, $n, $m, (int) rand(0, 255));
	}
	$m = 0;
	}

	#generate image
	imagejpeg($c);
?>
```

اسکریپت بالا تصویری به صورت زیر ایجاد خواهد کرد :



بخش چهارم – تصور داده های اولیه

ساخت یک ساعت با استفاده از خطوط و منحنی ها

در حالی که تصاویر random بسیار زیبا هستند، اما زیاد به درد نمی خورند (مگر برای طراحان گرافیکی که از این تصاویر برای ایجاد texture ها استفاده می کنند).

مثال زیر یک مثال بسیاز جذاب تر است که نحوه ی استفاده از gd برای به تصویر کشیدن داده های واقعی را نشان می دهد.

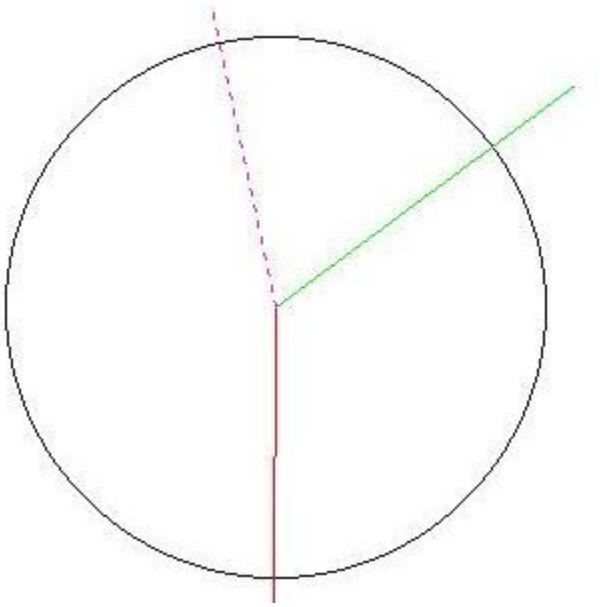
من از زمان کنونی برای ساخت ساعتی که زمان سرور را نشان می دهد استفاده می کنم. این ساعت مثال خوبی برای نشان دادن نحوه ی استفاده از توابع iamgearc() ، iamgeline() و imagedashedline() می باشد.

```
<?php
	header("Content-Type: image/jpeg");

	# Set the dimensions of the canvas
	$cw = 300;
	$ch = 300;

	# Create canvas
	$c = imagecreate($cw, $ch);

	# Generate color palette
	$cols[0] = imagecolorallocate($c, 0, 0, 0);
	$cols[1] = imagecolorallocate($c, 255, 255, 255);
	$cols[2] = imagecolorallocate($c, 255, 0, 0);
	$cols[3] = imagecolorallocate($c, 0, 255, 0);
	$cols[4] = imagecolorallocate($c, 255, 0, 255);
```

```
      # Create background
      imagefilledrectangle($c, 0, 0, $cw - 1,$ch - 1, $cols[1]);

      # Compute coordinates of the center of the image
      $x = (int) $cw / 2;
      $y = (int) $ch / 2;

      # Draw the circular border
      imagearc($c, $x, $y, (int) ($cw * .9),(int) ($ch * .9), 0,
360,$color[0]);

      # Read local time
      $t = localtime();

      # Initialize hour, minute, and second variables
      $th = $t[2];
      $tm = $t[1];
      $ts = $t[0];

      # Convert hour from 24- to 12-hour format
      if ($th > 11)
      {
      $th -= 11;
      }

      # Prepare some constants
      $hi = deg2rad(30);
      $mi = deg2rad(6);
      $adj = deg2rad(90);

      # Compute coordinates of the hour, minute, and second hands
      $th = $hi * $th - $adj;
      $tm = $mi * $tm - $adj;
      $ts = $mi * $ts - $adj;

      # Compute the length and the coordinates of
      # the hour hand
      $hhl = $cw * .6;
      $hhx = $hhl * cos($th);
      $hhy = $hhl * sin($th);

      # Compute the length and the coordinates of
      # the minute hand
      $mhl = $cw * .7;
      $mhx = $mhl * cos($tm);
      $mhy = $mhl * sin($tm);

      # Compute the length and the coordinates of
      # the second hand
      $shl = $cw * .8;
      $shx = $shl * cos($ts);
      $shy = $shl * sin($ts);

      # Draw the hour hand
      imageline($c, $x, $y, $x + $hhx, $y + $hhy, $cols[2]);

      # Draw the minute hand
```

```
        imageline($c, $x, $y, $x + $mhx, $y + $mhy, $cols[3]);

        # Draw the second hand
        imagedashedline($c, $x, $y, $x + $shx, $y + $shy,$cols[4]);

        # Generate image
        imagejpeg($c);
?>
```

بعد از اجرای اسکریپت بالا، ساعت زیر را خواهید دید. حالا هر وقت مرورگر خود را refresh کنید، زمان local سرور را خواهید دید.



تصویر سازی اطلاعات عددی

مثال ساعت آنالوگ یک معرفی جالب برای تصویر سازی داده ها بود، اما استفاده از آن در مورد داده ها تجاری و علمی کمی محدودیت به همراه دارد. یک مثال کاربردی تر برای تصویر سازی داده ها می تواند نمودار نتایج یک تحقیق در نواحی مختلف جغرافیایی باشد.

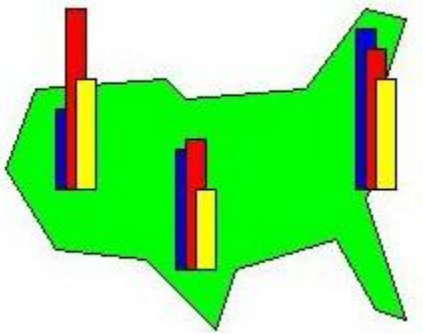
برای این مثال فرض کنید من از توسعه دهندگان کالیفورنیا، تگزاس و نیویورک خواسته ام تا به من بگویند که از کدام دسکتاپ استفاده می کنند، KDE، Gnome یا GNUStep . سپس من

نتایج را بر روی نقشه ی آمریکا نشان خواهم داد. این دقیقا کاری است که اسکریپت زیر انجام می دهد.

```
<?php
	header("Content-Type: image/jpeg");

	# Set the dimensions of the canvas
	$cw = 300;
	$ch = 300;

	# Define chart-drawing function
	function drawchart($c, $cx, $cy, $v1, $v2, $v3,$c0, $c1, $c2, $c3)
	{
	imagefilledrectangle($c, $cx, $cy - $v1,$cx + 10, $cy, $c1);
	imagerectangle($c, $cx, $cy - $v1,$cx + 10, $cy, $c0);
	imagefilledrectangle($c, $cx + 5, $cy - $v2,$cx + 15, $cy, $c2);
	imagerectangle($c, $cx + 5, $cy - $v2,$cx + 15, $cy, $c0);
	imagefilledrectangle($c, $cx + 10, $cy - $v3,$cx + 20, $cy, $c3);
	imagerectangle($c, $cx + 10, $cy - $v3,$cx + 20, $cy, $c0);
	}

	# Map displacement values
	$mdx = 0;
	$mdy = 0;

	# Make array of map points
	$usa[0] = 30 + $mdx; $usa[1] = 50 + $mdy;
	$usa[2] = 15 + $mdx; $usa[3] = 90 + $mdy;
	$usa[4] = 40 + $mdx; $usa[5] = 130 + $mdy;
	$usa[6] = 85 + $mdx; $usa[7] = 135 + $mdy;
	$usa[8] = 120 + $mdx; $usa[9] = 170 + $mdy;
	$usa[10] = 130 + $mdx; $usa[11] = 140 + $mdy;
	$usa[12] = 180 + $mdx; $usa[13] = 125 + $mdy;
	$usa[14] = 200 + $mdx; $usa[15] = 160 + $mdy;
	$usa[16] = 215 + $mdx; $usa[17] = 165 + $mdy;
	$usa[18] = 195 + $mdx; $usa[19] = 105 + $mdy;
	$usa[20] = 210 + $mdx; $usa[21] = 70 + $mdy;
	$usa[22] = 200 + $mdx; $usa[23] = 60 + $mdy;
	$usa[24] = 215 + $mdx; $usa[25] = 15 + $mdy;
	$usa[26] = 195 + $mdx; $usa[27] = 10 + $mdy;
	$usa[28] = 165 + $mdx; $usa[29] = 50 + $mdy;
	$usa[30] = 105 + $mdx; $usa[31] = 55 + $mdy;
	$usa[32] = 95 + $mdx; $usa[33] = 45 + $mdy;

	# Create canvas
	$c = imagecreate($cw, $ch);

	# Generate color palette
	$cols[0] = imagecolorallocate($c, 0, 0, 0);
	$cols[1] = imagecolorallocate($c, 255, 255, 255);
	$cols[2] = imagecolorallocate($c, 255, 0, 0);
	$cols[3] = imagecolorallocate($c, 0, 255, 0);
	$cols[4] = imagecolorallocate($c, 0, 0, 255);
	$cols[5] = imagecolorallocate($c, 255, 255, 0);
```

```
      # Create background
      imagefilledrectangle($c, 0, 0, $cw - 1,$ch - 1, $cols[1]);

      # Draw map
      imagefilledpolygon($c, $usa, 17, $cols[3]);

      # Draw map outline
      imagepolygon($c, $usa, 17, $cols[0]);
      drawchart($c, 190, 100, 80, 70,55,$cols[0], $cols[4],
$cols[2],$cols[5]);
      drawchart($c, 100, 140, 60, 65, 40,$cols[0], $cols[4],
$cols[2],$cols[5]);
      drawchart($c, 40, 100, 40, 90, 55,$cols[0], $cols[4],
$cols[2],$cols[5]);

      # Generate image
      imagejpeg($c);
?>
```

نتیجه اسکریپت بالا، تصویر زیر خواهد بود :



توجه کنید که در اسکریپت بالا من تابع drawchart() را برای رسم نمودار ها ایجاد کرده ام تا کارم راحت تر گردد. شما هم می توانید از این روش در کار هایتان استفاده کنید.

نقشه ی آمریکایی که در مثال بالا کشیده شده است، یک نقشه ی بسیار ساده می باشد که شامل آلاسکا و هاوایی نیست. این نقشه با استفاده از توابع imagepolygon() و imagefilledpolygon() کشیده شده است که هر دو تابع آرایه ای از اعداد را به عنوان مختصات نقاط چند ضلعی دریافت می کنند.

افزودن کمی متن

مثال قبلی کار خوبی از تبدیل مقادیر عددی به تصاویر بود، اما با استفاده از کمی متن می توان آن را مفید تر نیز کرد. با استفاده از تابع imagestring() می توانید متون افقی و با استفاده از تابع imagestringup() متون عمودی را در تصاویر خود بنویسید. هر دو تابع آرگومان های زیر را نیاز دارند :

- Canvas ID
- Font ID
- X Coordinates
- Y Coordinates
- Text string
- Color ID

Font ID یک مقدار صحیح در بازه ی 1-5 می باشد که یکی از 5 فونت توکار بیتی را مشخص می کند.

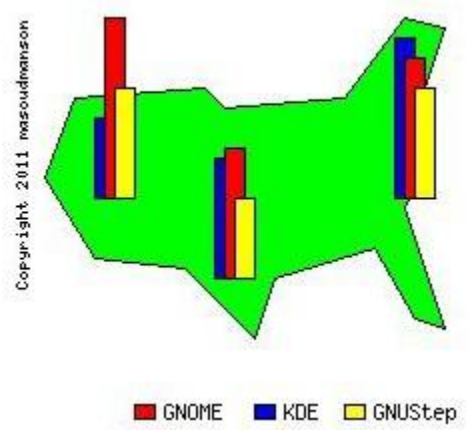
اسکریپت زیر نحوه ی استفاده از این توابع را در عمل به شما نشان می دهد. کد زیر توسعه یافته ی مثال قبل است.

```
<?php
      header("Content-Type: image/jpeg");

      # Set the dimensions of the canvas
      $cw = 300;
      $ch = 300;

      # Define chart-drawing function
      function drawChart($c, $cx, $cy, $v1, $v2, $v3,$c0, $c1, $c2, $c3)
      {
      ImageFilledRectangle($c, $cx, $cy - $v1,$cx + 10, $cy, $c1);
      ImageRectangle($c, $cx, $cy - $v1,$cx + 10, $cy, $c0);
      ImageFilledRectangle($c, $cx + 5, $cy - $v2,$cx + 15, $cy, $c2);
      ImageRectangle($c, $cx + 5, $cy - $v2,$cx + 15, $cy, $c0);
      ImageFilledRectangle($c, $cx + 10, $cy - $v3,$cx + 20, $cy, $c3);
      ImageRectangle($c, $cx + 10, $cy - $v3,$cx + 20, $cy, $c0);
      }

      # Map displacement values
      $mdx = 0;
      $mdy = 0;
```

```
# Make array of map points
$usa[0] = 30 + $mdx; $usa[1] = 50 + $mdy;
$usa[2] = 15 + $mdx; $usa[3] = 90 + $mdy;
$usa[4] = 40 + $mdx; $usa[5] = 130 + $mdy;
$usa[6] = 85 + $mdx; $usa[7] = 135 + $mdy;
$usa[8] = 120 + $mdx; $usa[9] = 170 + $mdy;
$usa[10] = 130 + $mdx; $usa[11] = 140 + $mdy;
$usa[12] = 180 + $mdx; $usa[13] = 125 + $mdy;
$usa[14] = 200 + $mdx; $usa[15] = 160 + $mdy;
$usa[16] = 215 + $mdx; $usa[17] = 165 + $mdy;
$usa[18] = 195 + $mdx; $usa[19] = 105 + $mdy;
$usa[20] = 210 + $mdx; $usa[21] = 70 + $mdy;
$usa[22] = 200 + $mdx; $usa[23] = 60 + $mdy;
$usa[24] = 215 + $mdx; $usa[25] = 15 + $mdy;
$usa[26] = 195 + $mdx; $usa[27] = 10 + $mdy;
$usa[28] = 165 + $mdx; $usa[29] = 50 + $mdy;
$usa[30] = 105 + $mdx; $usa[31] = 55 + $mdy;
$usa[32] = 95 + $mdx; $usa[33] = 45 + $mdy;

# Create canvas
$c = ImageCreate($cw, $ch);

# Generate color palette
$cols[0] = ImageColorAllocate($c, 0, 0, 0);
$cols[1] = ImageColorAllocate($c, 255, 255, 255);
$cols[2] = ImageColorAllocate($c, 255, 0, 0);
$cols[3] = ImageColorAllocate($c, 0, 255, 0);
$cols[4] = ImageColorAllocate($c, 0, 0, 255);
$cols[5] = ImageColorAllocate($c, 255, 255, 0);

# Create background
ImageFilledRectangle($c, 0, 0, $cw - 1, $ch - 1, $cols[1]);

# Draw map
ImageFilledPolygon($c, $usa, 17, $cols[3]);

# Draw map outline
ImagePolygon($c, $usa, 17, $cols[0]);
drawChart($c, 190, 100, 80, 70, 55,
$cols[0], $cols[4], $cols[2], $cols[5]);
drawChart($c, 100, 140, 60, 65, 40,
$cols[0], $cols[4], $cols[2], $cols[5]);
drawChart($c, 40, 100, 40, 90, 55,
$cols[0], $cols[4], $cols[2], $cols[5]);

# GNOME
ImageFilledRectangle($c, 60, 203, 70, 210, $cols[2]);
ImageRectangle($c, 60, 203, 70, 210, $cols[0]);
ImageString($c, 2, 75, 200, "GNOME", $cols[0]);

# KDE
ImageFilledRectangle($c, 120, 203, 130, 210, $cols[4]);
ImageRectangle($c, 120, 203, 130, 210, $cols[0]);
ImageString($c, 2, 135, 200, "KDE", $cols[0]);

# GNUStep
ImageFilledRectangle($c, 165, 203, 175, 210, $cols[5]);
```

```
        ImageRectangle($c, 165, 203, 175, 210, $cols[0]);
        ImageString($c, 2, 180, 200, "GNUStep", $cols[0]);

        # Copyright info
        ImageStringUp($c, 1, 0, 145, "Copyright 2011 masoudmanson", $cols[0]);

        # Generate image
        ImageJPEG($c);
?>
```

بعد از اجرای اسکریپت بالا شما تصویر زیر را خواهید داشت:



فونت های توکار بیتی (build-in bitmap fonts) همیشه در دسترس هستند، اما اگر شما بخواهید از فونت های دیگری استفاده کنید، باید کتابخانه ی freetype2 را نصب کرده و از تابع imagefttext() برای نمایش متن هایتان استفاده کنید. شما همچنین می توانید از Post Script Type 1 fonts نیز استفاده کنید که در این صورت به کتابخانه ی t1lib نیاز دارید و پس از نصب آن می توانید از توابع imagepsloadfont() و imagepstext() استفاده کنید.

در هر دو مورد بالا، باید php و gd این کتابخانه ها را ساپورت کنند. توجه کنید که Type 1 fonts زمان بیشتری را نسبت به Trur Type fonts برای رندر شدن نیاز دارند، و این امد ممکن است باعث timeout شدن اسکریپت شما گردد.

بخش پنجم – قبل از اینکه شاهکارتان را به تماشاگر بفرستید

تصمیم بگیرید چه نوع تصویری می خواهید ایجاد کنید؟

تا کنون شما تصاویر jpeg را ایجاد کرده اید، اما شاید شما بخواهید تصاویر PNG یا WBMP (در وسائلی که WAP-enabeld باشند کاربرد دارد، مثل موبایل ها) تولید کنید. برای این کار تابع imagejpeg() را با imagepng() یا imagewbmp() عوض کنید. اگر این کار را کردید فراموش نکنید که تغییرات زیر را نیز اعمال کنید، یعنی :

```
header ("Content-Type: image/jpeg");
```

را با

```
header ("Content-Type: image/png");
```

یا

```
header ("Content-Type: image/vnd.wap.wbmp");
```

جابجا کنید.

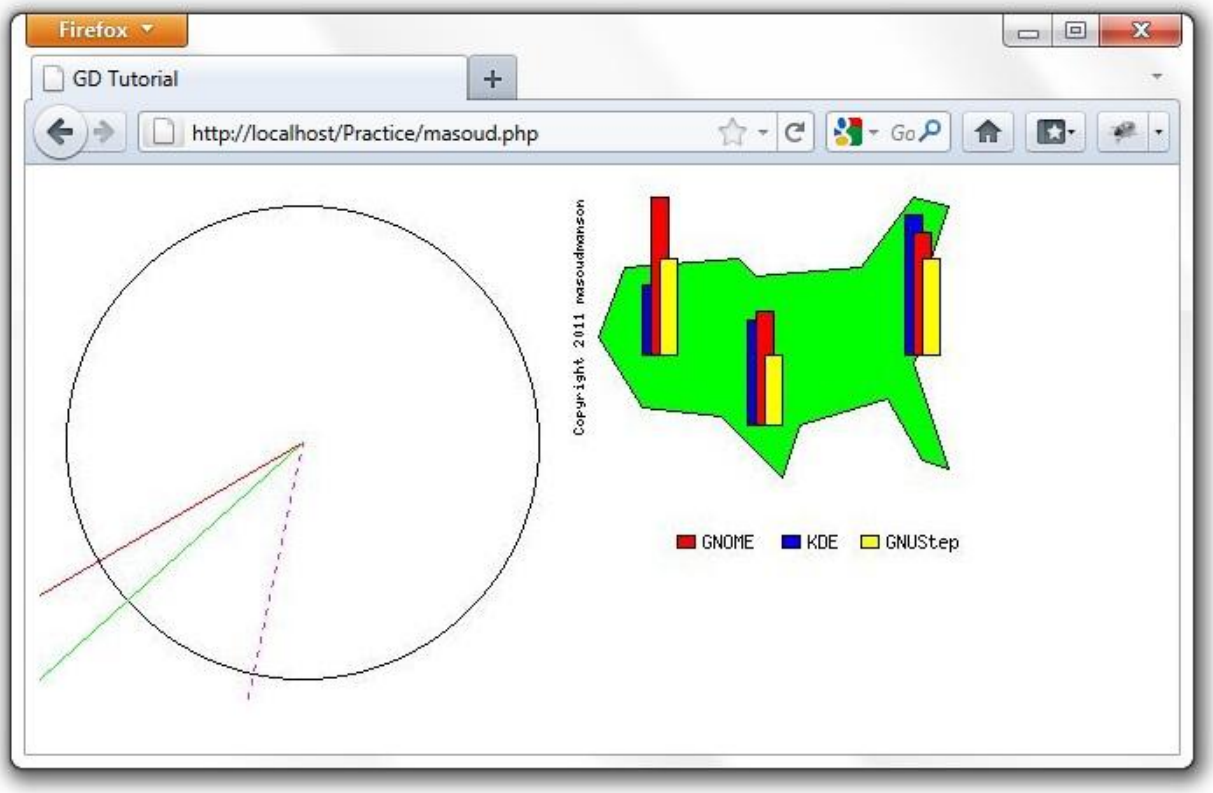
بخش ششم – تعبیه کردن در بین کد های html

تا کنون شما برای استفاده از اسکریپت های این آموزش آنها را به صورت مستقیم فراخوانی می کردید، در حالی که این مناسب ترین روش برای دسترسی به تصاویر تولیدی توسط اسکریپت ها نیست. اگر شما بخواهید از اسکریپتتان در یک صفحه ی html استفاده کنید، چه؟

این کار بسیار راحت است. تمام کاری که شما باید بکنید قرار دادن URL اسکریپتی که تصاویر را می سازد در آرگومان src تگ ها ی <img> می باشد:

```
<html>
<head>
    <title>GD Tutorial</title>
</head>
<body>
    <img src="clock.php">
    <img src="chart.php">
</body>
</html>
```

شکل زیر خروجی کد بالا می باشد :



About the author

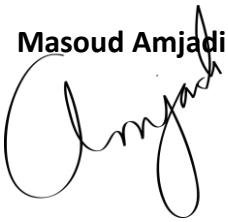
Jacek Artymiak works as a freelance consultant, developer, and writer. Since 1991 he's been developing software for many commercial and free variants of the UNIX and BSD operating systems (AIX, HP-UX, IRIX, Solaris, Linux, FreeBSD, NetBSD, OpenBSD, and others), as well as for the MS-DOS, Microsoft Windows, Mac OS, and Mac OS X operating systems. Jacek specializes in business and financial application development, Web design, network security, computer graphics, animation, and multimedia. He's a prolific writer on technology subjects and the coauthor of *Install, Configure, and CustomizeSlackware Linux* (Prima Tech, 2000) and *StarOffice for Linux Bible* (IDG Books, 2000). Find many of Jacek's software projects at SourceForge.net.

About the interpreter

Since 2009 I'm studing Information Technology in Institute for Advanced Studies in Basic Sciences (IASBS).

E-mail : masoudmanson@gmail.com
E-mail (2) : m.amjadi@iasbs.ac.ir
Cellular phone : 09148401824
Blog : iasbs-it.mihanblog.com

Masoud Amjadi



References

1- Installation manuals for gd 1.8.x (http://www.boutell.com/gd/manual1.8.4.html) and gd2.x (http://www.boutell.com/gd/manual2.0.1.html).

2- Installation manual for Apache (http://httpd.apache.org/docs/install.html).

3- Additional information for developers using PHP in *Programming PHP* , Rasmus Lerdorf and Kevin Tatroe, O'Reilly and Associates, 2002 (http://www.oreilly.com/catalog/progphp/).

4- For programming Web applications with PHP, see *Web Database Applications with PHP and MySQL* , Hugh E. Williams and David Lane, O'Reilly and Associates, 2002 (http://www.oreilly.com/catalog/webdbapps/).

5- For a general discussion of information visualization, read the Edward Tufte trilogy at Graphics Press: *The Visual Display of Quantitative Information*, 2001; *Envisioning Information*, 1990; and *Visual Explanations: Images and Quantities, Evidence and Narrative*, 1997 (http://www.edwardtufte.com/1028032219/tufte/).

6- Apple offers instructions on adding PHP for Mac OS X (http://developer.apple.com/internet/macosx/php.html).