

CONTENTS

Unit 1:	Introduction to PHP	1
Unit 2:	Language Basics	25
Unit 3:	Flow Control Statements and PHP in Web Page	53
Unit 4:	Functions	75
Unit 5:	Strings	93
Unit 6:	Arrays	114
Unit 7:	Multidimensional Arrays	131
Unit 8:	Objects	158
Unit 9:	Web Techniques	177
Unit 10:	Database	207
Unit 11:	Graphic	228
Unit 12:	Portable Document Format	253
Unit 13:	Extensible Markup Language	274
Unit 14:	Security	302

SYLLABUS

Web Technologies - I

Objectives: To impart the skills needed for web programming, web administration and web site development. After studying this course student can develop static as well as dynamic web pages. Student will also learn how to process data stored in database using web pages.

S. No.	Topics
1.	Introduction to PHP: What Does PHP Do, A Brief History of PHP, Installing PHP, A Walk Through PHP Language Basics: Lexical Structure, Data Types, Variables, Expressions and Operators
2.	Flow-Control Statements: Including Code, Embedding PHP in Web Pages Functions: Calling a Function, Defining a Function, Variable Scope, Function Parameters, Return Values, Variable Functions
3.	Strings: Quoting String Constants, Printing Strings, Accessing Individual Characters, Cleaning Strings, Encoding and Escaping, Comparing Strings, Manipulating and Searching Strings, Regular Expressions
4.	Arrays: Indexed Versus Associative Arrays, Identifying Elements of an Array, Storing Data in Arrays, Multidimensional Arrays, Extracting Multiple Values, Converting Between Arrays and Variables, Traversing Arrays, Sorting, Acting on Entire Arrays, Using Arrays
5.	Objects: Terminology, Creating an Object, Accessing Properties and Methods, Declaring a Class, Introspection, Serialization
6.	Web Techniques: HTTP Basics, Variables, Server Information, Processing Forms, Setting Response Headers, Maintaining State, SSL
7.	Databases: Using PHP to Access a Database, Relational Databases and SQL, PEAR DB Basics, Advanced Database Techniques
8.	Graphics: Embedding an Image in a Page, The GD Extension, Basic Graphics Concepts, Creating and Drawing Images, Images with Text, Dynamically Generated Buttons, Scaling Images, Color Handling
9.	PDF: PDF Extensions, Documents and Pages, Text, Images and Graphics, Navigation. XML: Lightning Guide to XML, Generating XML, Parsing XML, Transforming XML with XSLT, Web Services
10.	Security: Global Variables and Form Data, Filenames, File Uploads, File Permissions, PHP Code, Shell Commands

Unit 1: Introduction to PHP

Notes

CONTENTS

Objectives

Introduction

- 1.1 What Does PHP Do?
 - 1.1.1 Server-side Scripting
 - 1.1.2 Command-line Scripting
 - 1.1.3 Graphical Scripting
- 1.2 A Brief History of PHP
 - 1.2.1 Evolution of PHP
 - 1.2.2 Growth of PHP
- 1.3 Installing PHP
 - 1.3.1 Installing with Unix/Linux PHP Distribution
 - 1.3.2 Windows
 - 1.3.3 Installing with PHP Windows Installer
 - 1.3.4 Go-Pear.Org
 - 1.3.5 Prerequisites
 - 1.3.6 Going PEAR
 - 1.3.7 Testing Installation
 - 1.3.8 Manual Installation
- 1.4 A Walk through PHP
 - 1.4.1 Configuration Page
 - 1.4.2 Forms
 - 1.4.3 Databases
 - 1.4.4 Graphics
 - 1.4.5 From the Shell
- 1.5 Summary
- 1.6 Keywords
- 1.7 Review Questions
- 1.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain What PHP Does
- Discuss the History of PHP

Notes

- Explain How to Install PHP in Your System
- Understand the Features of PHP

Introduction

PHP is a simple yet powerful language designed for creating HTML content. This unit covers essential background on the PHP language. It describes the nature and history of PHP, which platforms it runs on, and how to configure it. This unit ends by showing you PHP in action, with a quick walkthrough of several PHP programs that illustrate common tasks, such as processing form data, interacting with a database and creating graphics.

1.1 What Does PHP Do?

PHP can be used in three primary ways:

1.1.1 Server-side Scripting


PHP was originally designed to create dynamic web content, and it is still best suited for that task. To generate HTML, you need the PHP parser and a web server through which to send the coded documents. PHP has also become popular for generating XML documents, graphics, Flash animations, PDF files, and so much more.

1.1.2 Command-line Scripting

PHP can run scripts from the command line, much like Perl, awk, or the Unix shell. You might use the command-line scripts for system administration tasks, such as backup and log parsing; even some CRON job type scripts can be done this way (nonvisual PHP tasks).

1.1.3 Graphical Scripting

Using PHP-GTK, you can write full-blown, cross-platform GUI applications in PHP. In this book, however, we concentrate on the first item: using PHP to develop dynamic web content. PHP runs on all major operating systems, from Unix variants including Linux, FreeBSD, Ubuntu, Debian, and Solaris to Windows and Mac OS X. It can be used with all leading web servers, including Apache, Microsoft IIS, and the Netscape/iPlanet servers.



Notes The language itself is extremely flexible.



Example: You aren't limited to outputting just HTML or other text files – any document format can be generated. PHP has built-in support for generating PDF files, GIF, JPEG, and PNG images, and Flash movies.

One of PHP's most significant features is its wide-ranging support for databases. PHP supports all major databases (including MySQL, PostgreSQL, Oracle, Sybase, MS-SQL, DB2, and ODBC-compliant databases), and even many obscure ones. Even the more recent NoSQL-style databases like SQLite and MongoDB are also supported. With PHP, creating web pages with dynamic content from a database is remarkably simple. Finally, PHP provides a library of PHP code to perform common tasks, such as database abstraction, error handling, and so on, with the PHP

Extension and Application Repository (PEAR). PEAR is a framework and distribution system for reusable PHP components.

Notes

Self Assessment

State whether the following statements are true or false:

1. PHP was designed to create dynamic web content.
2. PHP can run scripts from the command line.
3. PHP runs on only one operating system.
4. PEAR is a framework and distribution system for reusable PHP components.

1.2 A Brief History of PHP

Rasmus Lerdorf first conceived of PHP in 1994, but the PHP that people use today is quite different from the initial version. To understand how PHP got where it is today, it is useful to know the historical evolution of the language. Here's that story, with ample comments and emails from Rasmus himself.

1.2.1 Evolution of PHP

The development of PHP began long back in 1994 when Lerdorf created a bunch of Pearl scripts to enhance his personal homepage with features like recording the traffic and displaying his resume. He initially called this collection as "PERSONAL HOME PAGE TOOLS". Lerdorf announced the release of PHP in a Usenet discussion group on June 8, 1995 which was publicly called as PHP/FI - "Personal Home Page Tools (PHP Tools) version 1.0". After beta testing in team PHP/FI 2 was officially released in November 1997.

Than Zeev Suraski and Andi Gutmans the programmers at Technion IIT rewrote parser in 1997 and formed the base of PHP 3 and the name of PHP was changed to PHP: Hypertext Preprocessor.

Later various improvements kept on happening and finally on 14 June 2012 latest version PHP 5.4.4 was announced by PHP Team.



Did u know? PHP 3.0 is available for free download in source form and binaries for several platforms at <http://www.php.net/>.

1.2.2 Growth of PHP

Figures 1.1 and 1.2 show the growth of PHP as measured by the usage numbers collected by Netcraft since early 1998. Figure 1.1 shows the total number of unique IP addresses that report they are using Apache with the PHP module enabled. In November 2001, this number went beyond the one-million mark. The slight dip at the end of 2001 reflects the demise of a number of dot-coms that disappeared from the Web. The overall number of servers that Netcraft found also went down for the first time during this period.

Notes

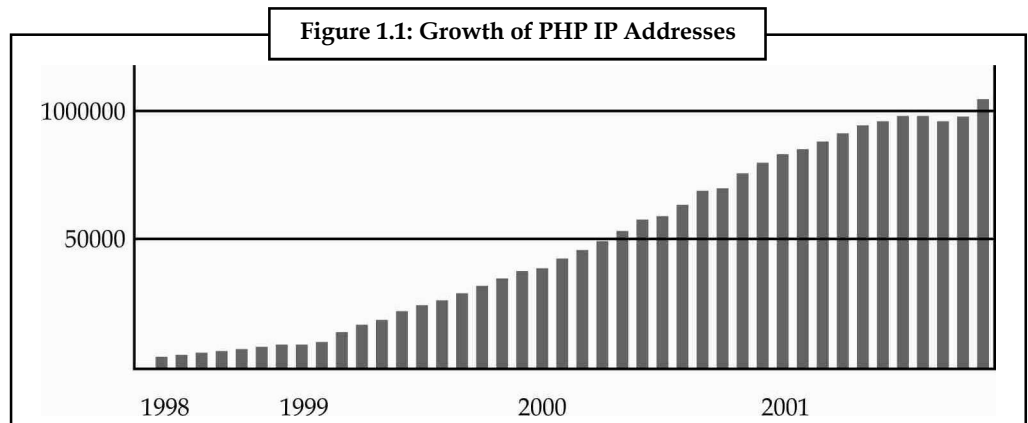
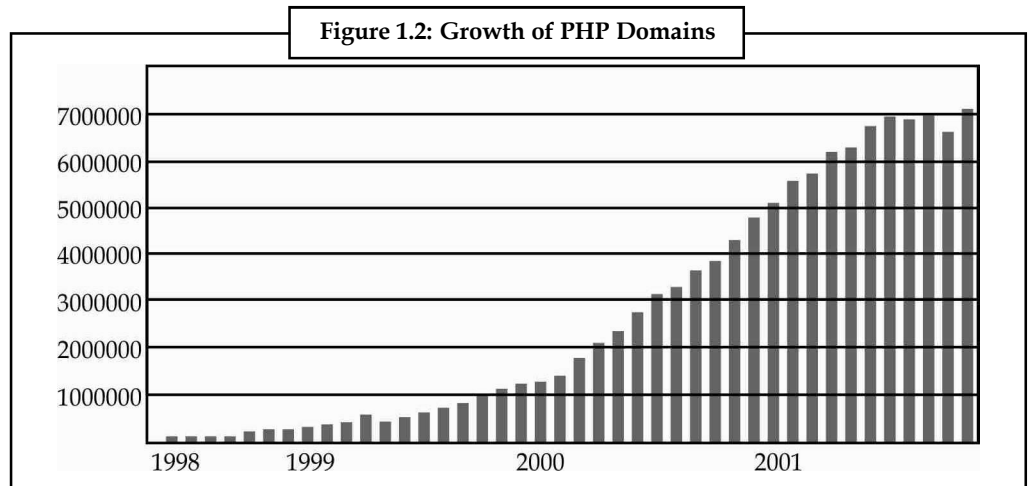


Figure 1.2 shows the number of actual domains that report they are using the PHP module. In November 2001, when Netcraft found 36,458,394 different domains 7,095,691 (just under 20%) of them were found to have PHP enabled. The domain figures represent the number of web sites using PHP, whereas IP addresses represent the number of physical servers running PHP.



Self Assessment

Fill in the blanks:

5. Rasmus Lerdorf first conceived of PHP in....., but the PHP that people use today is quite different from the initial version.
6. Zeev Suraski and Andi Gutmans the programmers at Technion IIT rewrote parser in

1.3 Installing PHP

As was mentioned above, PHP is available for many operating systems and platforms. From time to time, you may also want to change the way PHP is configured. To do that you will have to change the PHP configuration file and restart your Apache server. Each time you make a change to PHP's environment, you will have to restart the Apache server in order for those changes to take effect. PHP's configuration settings are maintained in a file called php.ini. The settings in this file control the behavior of PHP features, such as session handling and form processing. Later units refer to some of the php.ini options, but in general the code in this book does not require a customized configuration.



Notes In installing PHP, we learn how to install PEAR on your platform from a PHP distribution or through the go-pear.org website.

Notes

1.3.1 Installing with Unix/Linux PHP Distribution

It describes PEAR installation and basic usage that is specific for UNIX or UNIX-like platforms, such as Linux and Darwin. The installation of the PEAR Installer itself is somewhat OS-dependent, and because most of what you need to know about installation is OS-specific, you find that here. Using the installer is more similar on different platforms, so that is described in the next, with the occasional note about OS idiosyncrasies. As of PHP 4.3.0, PEAR with all its basic prerequisites is installed by default when you install PHP. If you build PHP from source, these configure options cause problems for PEAR:

– disable-pear

make install will neither install the PEAR installer or any packages.

– disable-cli

The PEAR Installer depends on a standalone version of PHP installed.

– without-xml

PEAR requires the XML extension for parsing package information files. If you plan to install PHP on Linux or any other variant of Unix, then here is the list of prerequisites:

- The PHP source distribution
- The latest Apache source distribution
- A working PHP-supported database, if you plan to use one (For example, MySQL, Oracle etc.)
- Any other supported software to which PHP must connect (mail server, BCMath package, JDK, and so forth)
- An ANSI C compiler
- Gnu make utility

Now here are the steps to install Apache and PHP5 on your Linux or Unix machine. If your PHP or Apache versions are different then please take care accordingly.

- If you haven't already done so, unzip and untar your Apache source distribution. Unless you have a reason to do otherwise, /usr/local is the standard place.

```
gunzip -c apache_1.3.x.tar.gz
```

```
tar -xvf apache_1.3.x.tar
```

- Build the apache Server as follows

```
cd apache_1.3.x
```

```
./configure --prefix=/usr/local/apache --enable-so
```

```
make
```

```
make install
```

Notes

- Unzip and untar your PHP source distribution. Unless you have a reason to do otherwise, /usr/local is the standard place.

```
gunzip -c php-5.x.tar.gz
tar -xvf php-5.x.tar
cd php-5.x
```

- Configure and Build your PHP, assuming you are using MySQL database.

```
./configure --with-apxs=/usr/sbin/apxs \
            --with-mysql=/usr/bin/mysql
make
make install
```

- Install the php.ini file. Edit this file to get configuration directives:

```
cd ../../php-5.x
cp php.ini-dist /usr/local/lib/php.ini
```

- Tell your Apache server where you want to serve files from, and what extension(s) you want to identify PHP files. A .php is the standard, but you can use .html, .phtml, or whatever you want.

1. Go to your HTTP configuration files (/usr/local/apache/conf or whatever your path is)
2. Open httpd.conf with a text editor.
3. Search for the word DocumentRoot (which should appear twice), and change both paths to the directory you want to serve files out of (in our case, /home/httpd). We recommend a home directory rather than the default /usr/local/apache/htdocs because it is more secure, but it doesn't have to be in a home directory. You will keep all your PHP files in this directory.

- Add at least one PHP extension directive, as shown in the first line of code that follows. In the second line, we've also added a second handler to have all HTML files parsed as PHP

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php .html
```

- Restart your server. Every time you change your HTTP configuration or php.ini files, you must stop and start your server again.

```
cd ../bin
./apachectl start
```

- Set the document root directory permissions to world-executable. The actual PHP files in the directory need only be world-readable (644). If necessary, replace /home/httpd with your document root below:

```
chmod 755 /home/httpd/html/php
```

- Open a text editor. Type: <?php phpinfo(); ?>. Save this file in your Web server's document root as info.php.

- Start any Web browser and browse the file. You must always use an HTTP request (http://www.testdomain.com/info.php or http://localhost/info.php or http://127.0.0.1/info.php) rather than a filename (/home/httpd/info.php) for the file to be parsed correctly



Task Prepare a list of system requirement for installing PHP in PC.

1.3.2 Windows

It shows how to install PEAR on a Windows PHP installation. Start by just installing a binary distribution of PHP. If you go with the defaults, your PHP install will end up in C:\PHP,

Figure 1.3: Diagram of Windows



1.3.3 Installing with PHP Windows Installer

When you have PHP installed, you need to make sure that your `include_path` PHP setting is sensible. Some versions of the Windows PHP Installer use `c:\php4\pear` in the default include path, but this directory (`c:\php4`) is different from the one created by the PHP Windows Installer. So, edit your `php.ini` file (in `c:\winnt` or `c:\windows`, depending on your Windows version) and change this directory to `c:\php\pear`.

1.3.4 Go-Pear.Org

Go-pear.org is a website with a single PHP script that you can run to install the latest stable version of the PEAR Installer and the PHP Foundation Classes (PFC). Go-pear is cross-platform and can be run from the command line and from your web server. PHP distributions bundle a particular release of the PEAR Installer; on the other hand, go-pear gives you the newest stable PEAR releases. However, go-pear does know your directory layout, but really contorts itself to figure it out, and will try adapting your PEAR Installation to that. In this, you learn how to use go-pear from the command line and web server, and on UNIX and Windows.

1.3.5 Prerequisites

Because go-pear is written in PHP, you need a CGI or CLI version of PHP to execute it outside the web server. By default, the CLI version is installed along with your web server PHP module 1998–2004 Zend Technologies. By default, the `php` command is installed in the `/usr/local/bin` directory on UNIX, or `c:\php` on Windows. In Windows, the CLI version of PHP may also be called `php-cli`; in that case, you need to type `php-cli` for every example that says just `php`.

1.3.6 Going PEAR

If your PHP install did not include PEAR, you can use go-pear as a universal PEAR bootstrapper. All you need is a CLI or CGI version of PHP installed somewhere.

```
$ lynx source http://go-pear.org | php. This command simply takes the contents of http://
```

go-pear.org and sends it to PHP for execution. If you do not have lynx available on your system, try an alternative way of executing go-pear directly:

```
Using GNUS wget: $ wget nO- http://go-pear.org php  
Using fetch on FreeBSD: $ fetch no n http://go-pear.org php  
Using Perl LWPis GET utility: $ GET http://go-pear.org php
```

On Windows, there is no one to fetch this URL tool, but you may be able to use PHP in URL streams

```
C :> php-cli nr "include('http://go-pear.org');"
```

If none of this works open <http://go-pear.org> in your browser, save the contents as go-pear.php and simply run it from there:

```
C :> php go-pear.php The output will look like this:
```

Welcome to go-pear!

Go-pear will install the 'pear' command and all the files needed by it. This command is your tool for PEAR installation and maintenance. Go-pear will install the PEAR packages bundled with PHP: DB, Net_Socket, Net_SMTP, Mail, XML_Parser, PHPUnit.

This greeting tells you what you are about to start. Press Enter for the first real question: HTTP proxy (<http://user:password@proxy.myhost.com:port>), or Enter for none:

Go-pear checks your http_proxy environment variable and presents the value of that as the default value if http_proxy is defined.

Now, on to the interesting part: Below is a suggested file layout for your new PEAR installation. To change individual locations, type the number in front of the directory. Type 'all' to change all of them, or simply press Enter to accept these locations.

1. Installation prefix: /usr/local
2. Binaries directory: \$prefix/bin
3. PHP code directory: \$prefix/share/pear
4. Documentation base directory: \$php_dir/docs
5. Data base directory: \$php_dir/data
6. Tests base directory: \$php_dir/tests

Each setting is internally assigned to a variable (prefix, bin_dir, php_dir, doc_dir, data_dir and test_dir, respectively). You may refer to the value of other settings by referencing these variables, as shown previously. Let's take a look at each setting:

- **Installation prefix:** The root directory of your PEAR installation. It has no other effect than serving as a root for the next five settings, using \$prefix.
- **Binaries directory:** Where programs and PHP scripts from PEAR packages are installed. The pear executable ends up here. Remember to add this directory to your PATH.
- **PHP code directory:** Where PHP code is installed, this directory must be in your include_path when using the packages you install.

- **Documentation base directory:** This is the base directory for documentation. By default, it is `$php_dir/doc`, and the documentation files for each package are installed as `$doc_dir/Package/file`.
- **Database directory:** Where the PEAR Installer stores data files. Data files are just a catch-all category for anything that does not fit as PHP code, documentation, and so on. As with the documentation base directory, the package name is added to the path, so the data file `convert.xml` in `MyPackage` would be installed as `$data_dir/MyPackage/convert.xml`.
- **Tests base directory:** Where regression test scripts for the package are installed. The package name is also added to the directory. When you are satisfied with the directory layout, press Enter to proceed: The following PEAR packages are bundled with PHP: DB, Net_Socket. After you have compiled or installed PHP, you can still change its behavior with the `php.ini` file. On Linux/Unix systems, the default location for this file is `/usr/local/php/lib`, or the `lib` subdirectory of the PHP installation location you used at configuration time. On a Windows system, this file should be in the Windows directory.

Directives in the `php.ini` file come in two forms: values and flags. Value directives take the form of a directive name and a value separated by an equal sign. Possible values vary from directive to directive. Flag directives take the form of a directive name and a positive or negative term separated by an equal sign. Positive terms include 1, On, Yes, and True. Negative terms include 0, Off, No, and False. Whitespace is ignored.

You can change your `php.ini` settings at any time, but after you do, you will need to restart the server for the changes to take effect. At some point, take time to read through the `php.ini` file on your own to see the types of things that can be configured.



Did u know? PHP is running more than 20 million websites and 1 million web servers in the world till 2011.

1.3.7 Testing Installation

These instructions apply to PHPLIB running with CGI PHP. Most of them is valid for `mod_php` as well, though. This section offers an incremental approach to find installation problems, should the above installation process fail.

Checking that the Web Server is Up and Running

Make sure your web server is up and serving the virtual host you just set up. To do this, construct a small file `test1.html` in your DocumentRoot and access `test1.html` through your web server.

Checking that the Web Server is Executing CGI Programs

Make sure your web server is up and does run CGI. Check the current directory, the UID/GID it is running programs under and have a look at the environment variables. Install the following shell script (for most cases cutting and pasting)



Example:

```
#!/bin/sh -
echo "Content-Type: text/plain"
echo
id
```

Notes

```
echo
pwd
echo
env | sort
echo
```

in your cgi directory under the name of cgi-test/ and in your document root under the name of cgi-test.cgi. Make it executable.

What UID/GID are you running under, what is the output of pwd and what environment variables are set? What does QUERY_STRING look like? What does the PATH variable look like, what does the LD_LIBRARY_PATH variable look like and are all libraries needed by PHP accessible to PHP running in the CGI environment (check by running the Unixldd command on PHP).

In particular, if you built Oracle support into PHP and linked libclntsh dynamically: Can it be loaded from the CGI environment? If not, PHP will not come up later in the next step.

Checking that the PHP Interpreter is Running (Assuming CGI PHP)

Copy your PHP binary into the cgi binary directory (which should NOT be below DocumentRoot!) and make it executable. Copy php3.ini or php.ini into the same directory. In DocumentRoot, create a test2.php3 and put <?php phpinfo() ?> into it.

Are you running Apache? Add

```
Action          php3-script /cgi/php
AddHandler      php3-script .php3 .php
DirectoryIndex index.php3 index.php index.html index.htm
FancyIndexing  on
```

to your httpd.conf. This will map all requests to files ending in .php3, php to the php-script handler and define /cgi/php as the URL handling php-script requests internally.

Open test2.php3 in your browser and see that it is being executed. Make changes to your php.ini (preferable some color definitions) and reload. Are they reflected in the output of phpinfo()? If not, your php.ini is not being found and your are having a problem. Recompile with proper settings.

Check the output of phpinfo() carefully! Is your PHP version current? Are your database interfaces present in the output of phpinfo()? If not, recompile again.

Can you access test2.php3 under the URL /cgi/php/test2.php3 as well? If so, you did not compile your PHP interpreter with --enable-force-cgi-redirect. PHPLIB will not work with this interpreter. Recompile with the switch being set.

Checking that the PHP Interpreter is Running (Assuming mod_php)

Assuming your server is already correctly setup (don't forget to activate the PHP lines in httpd.conf!), enter the following file and save it as test2.php3 under your DocumentRoot.

```
<? phpinfo() ?>
```

If you access this using a web browser now, it should spit out much info about PHP, Apache and its environment.

Checking PHPLIB Inclusion

Notes

Does your PHP include PHPLIB properly? Check your `php.ini` file. It must include the following settings:

```
include_path = pathname to directory with all the .inc files
auto_prepend_file = path to prepend.php3
track_vars = On
```

If PHPLib is included properly by your setup, the following page will execute without errors:

```
<?php
$db = new DB_Example;
print "It works without error messages.<br>\n";
?>
```

Checking Database Connectivity

PHPLIB installation requires that you adapt `local.inc` properly. Particularly, the provided class `DB_Example` must be customized for your database connection. Test that your web server can access the database with the following example:



Example:

```
<?php
    include("table.inc"); // requires include_path to be functioning

    $db = new DB_Example;
    $db->query("select * from auth_user");

    $t = new Table;
    $t->heading = "on";
    $t->show_result($db);
?>
```

When executing properly, this page will show you the user entry for kris, password test, permissions admin from the `auth_user` table. If this does not happen, your `DB_Example` definition in `local.inc` is broken.

Checking that Sessions Work

Access the page `index.php3` that has been provided with the distribution. This page will try to set a cookie in your browser. Allow that cookie to be set.

The page will display a headline with a counter. Reload that page. The counter must increment. If not, either your browser cannot deal properly with cookies or PHPLib cannot properly read or write the table `active_sessions` in your database. Check that the cookie is being set by viewing the output of `phpinfo()`. Check your database permissions with your database command line interface.

Notes

Checking that Authentication Works

Try loading showoff.php3 that has been provided with the distribution. This page will require a login. Login as kris, using a password of test. If the login is successful, you will see the per-session counter and a per-user counter again. Reload that page: The counters must increment.

If you can't login, you probably have a problem with cookies. Check again that your browser accepts and sends session cookies. Another problem may be access to theauth_user table. You must be able to SELECT on that table and there must be at an entry for the user you are trying to login.

1.3.8 Manual Installation

This section contains instructions for manually installing and configuring PHP on Microsoft Windows. For the instructions on how to use PHP installer to setup and configure PHP and a web server on Windows.

Selecting and Downloading the PHP Distribution Package


Download the PHP zip binary distribution from PHP for Windows: Binaries and Sources. There are several different versions of the zip package - choose the version that is suitable for the web server being used:

- If PHP is used with IIS then choose PHP 5.3 VC9 Non Thread Safe or PHP 5.2 VC6 Non Thread Safe;
- If PHP is used with IIS7 or greater and PHP 5.3+, then the VC9 binaries of PHP should be used.
- If PHP is used with Apache 1 or Apache 2 then choose PHP 5.3 VC6 or PHP 5.2 VC6.

VC9 Versions are compiled with the Visual Studio 2008 compiler and have improvements in performance and stability. The VC9 versions require you to have the Microsoft 2008 C++ Runtime (x86) or the Microsoft 2008 C++ Runtime (x64) installed.

PHP Package Structure and Content

Unpack the content of the zip archive into a directory of your choice, for example C:\PHP\. The directory and file structure extracted from the zip is shown in example below:

 **Example: PHP 5 Package Structure**

```

c:\php
|
|--dev
| |
| |-php5ts.lib      - php5.lib in non thread safe version
|
|--ext              - extension DLLs for PHP
| |
| |-php_bz2.dll
| |
| |-php_cpdf.dll
| |

```

```

| |-...
|
|--extras          - empty
|
|--pear           - initial copy of PEAR
|
|--go-pear.bat    - PEAR setup script
|
|--...
|
|--php-cgi.exe    - CGI executable
|
|--php-win.exe    - executes scripts without an opened command prompt
|
|--php.exe        - Command line PHP executable (CLI)
|
|--...
|
|--php.ini-development - default php.ini settings
|
|--php.ini-production  - recommended php.ini settings
|
|--php5apache2_2.dll   - does not exist in non thread safe version
|
|--php5apache2_2_filter.dll - does not exist in non thread safe version
|
|--...
|
|--php5ts.dll        - core PHP DLL ( php5.dll in non thread safe version)
|
|--...

```

Below is the list of the modules and executables included in the PHP zip distribution:

- go-pear.bat - the PEAR setup script.
- php-cgi.exe - CGI executable that can be used when running PHP on IIS via CGI or FastCGI.
- php-win.exe - the PHP executable for executing PHP scripts without using a command line window (for example, PHP applications that use Windows GUI).
- php.exe - the PHP executable for executing PHP scripts within a command line interface (CLI).
- php5apache2_2.dll - Apache 2.2.X module.
- php5apache2_2_filter.dll - Apache 2.2.X filter.

Changing the php.ini file

After the php package content has been extracted, copy the php.ini-production into php.ini in the same folder. If necessary, it is also possible to place thephp.ini into any other location of your choice but that will require additional configuration steps as described in PHP Configuration.

The php.ini file tells PHP how to configure itself, and how to work with the environment that it runs in. Here are a number of settings for the php.ini file that help PHP work better with Windows. Some of these are optional. There are many other directives that may be relevant to your environment - refer to the list of php.ini directives for more information.

Notes

Required Directives

- `extension_dir = <path to extension directory>` - The `extension_dir` needs to point to the directory where PHP extensions files are stored. The path can be absolute (i.e. "C:\PHP\ext") or relative (i.e. ".\ext"). Extensions that are listed lower in the `php.ini` file need to be located in the `extension_dir`.
- `extension = xxxxx.dll` - For each extension you wish to enable, you need a corresponding "extension=" directive that tells PHP which extensions in the `extension_dir` to load at startup time.
- `log_errors = On` - PHP has an error logging facility that can be used to send errors to a file, or to a service (i.e. syslog) and works in conjunction with the `error_log` directive below. When running under IIS, the `log_errors` should be enabled, with a valid `error_log`.
- `error_log = <path to the error log file>` - The `error_log` needs to specify the absolute, or relative path to the file where PHP errors should be logged. This file needs to be writable for the web server. The most common places for this file are in various TEMP directories, for example "C:\inetpub\temp\php-errors.log".
- `cgi.force_redirect = 0` - This directive is required for running under IIS. It is a directory security facility required by many other web servers. However, enabling it under IIS will cause the PHP engine to fail on Windows.
- `cgi.fix_pathinfo = 1` - This lets PHP access real path info following the CGI Spec. The IIS FastCGI implementation needs this set.
- `fastcgi.impersonate = 1` - FastCGI under IIS supports the ability to impersonate security tokens of the calling client. This allows IIS to define the security context that the request runs under.
- `fastcgi.logging = 0` - FastCGI logging should be disabled on IIS. If it is left enabled, then any messages of any class are treated by FastCGI as error conditions which will cause IIS to generate an HTTP 500 exception.

Optional Directives

- `max_execution_time = ##` - This directive tells PHP the maximum amount of time that it can spend executing any given script. The default for this is 30 seconds. Increase the value of this directive if PHP application take long time to execute.
- `memory_limit = ###M` - The amount of memory available for the PHP process, in Megabytes. The default is 128, which is fine for most PHP applications. Some of the more complex ones might need more.
- `display_errors = Off` - This directive tells PHP whether to include any error messages in the stream that it returns to the Web server. If this is set to "On", then PHP will send whichever classes of errors that you define with the `error_reporting` directive back to web server as part of the error stream. For security reasons it is recommended to set it to "Off" on production servers in order not to reveal any security sensitive information that is often included in the error messages.
- `open_basedir = <paths to directories, separated by semicolon>`, e.g. `openbasedir="C:\inetpub\wwwroot;C:\inetpub\temp"`. This directive specified the directory paths where PHP is allowed to perform file system operations. Any file operation outside of the specified paths will result in an error. This directive is especially useful for locking down the PHP installation in shared hosting environments to prevent PHP scripts from accessing any files outside of the web site's root directory.

- `upload_max_filesize = ###M` and `post_max_size = ###M` - The maximum allowed size of an uploaded file and post data respectively. The values of these directives should be increased if PHP applications need to perform large uploads, such as for example photos or video files.

PHP is now setup on your system. The next step is to choose a web server, and enable it to run PHP. Choose a web server from the table of contents.

In addition to running PHP via a web server, PHP can run from the command line just like a .BAT script.



Did u know? If you want to send emails using the PHP `mail()` function, enter the details of an SMTP server (your ISP's server should be suitable).

Self Assessment

State whether the following statements are true or false:

7. PEAR requires the HTML extension for parsing package information files.
8. When you have PHP installed, you need to make sure that your `Include_path` PHP setting is not sensible.
9. `Go-pear` is cross-platform and can be run from the command line and from your web server.
10. If your PHP install include PEAR then you can use `go-pear` as a universal PEAR bootstrapper.
11. Data files are just a catch-all category for anything that does not fit as PHP code.
12. In `Binaries` directory programs and PHP scripts from PEAR packages are installed.

1.4 A Walk through PHP

PHP pages are generally HTML pages with PHP commands embedded in them. This is in contrast to many other dynamic web page solutions, which are scripts that generate HTML. The web server processes the PHP commands and sends their output (and any HTML from the file) to the browser.



Example:

```
. hello_world.php
<html>
  <head>
    <title>Look Out World</title>
  </head>

  <body>
    <?php echo "Hello, world!"; ?>
</html>
<head>
  <title>Look Out World</title>
```

Notes

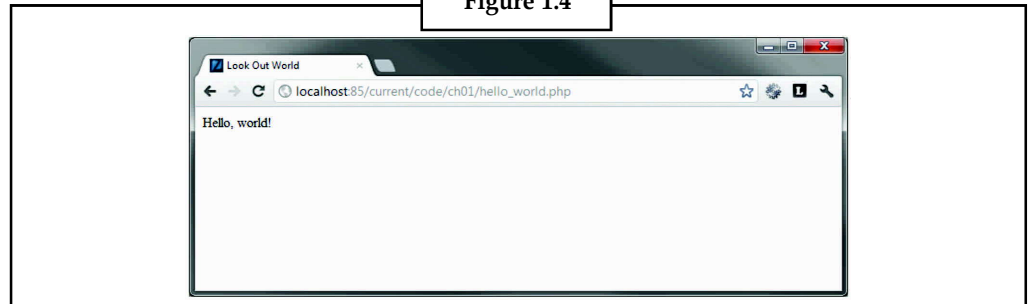
```

</head>

<body>
    <?php echo "Hello, world!"; ?>
</body>
</html>
    
```

The results appear in Figure 1.4.

Figure 1.4



The PHP echo command produces output (the string “Hello, world!” in this case) inserted into the HTML file. In this example, the PHP code is placed between the <?php and ?> tags.

1.4.1 Configuration Page

The PHP function phpinfo() creates an HTML page full of information on how PHP was installed and is currently configured. You can use it to see whether you have particular extensions installed, or whether the php.ini file has been customized.

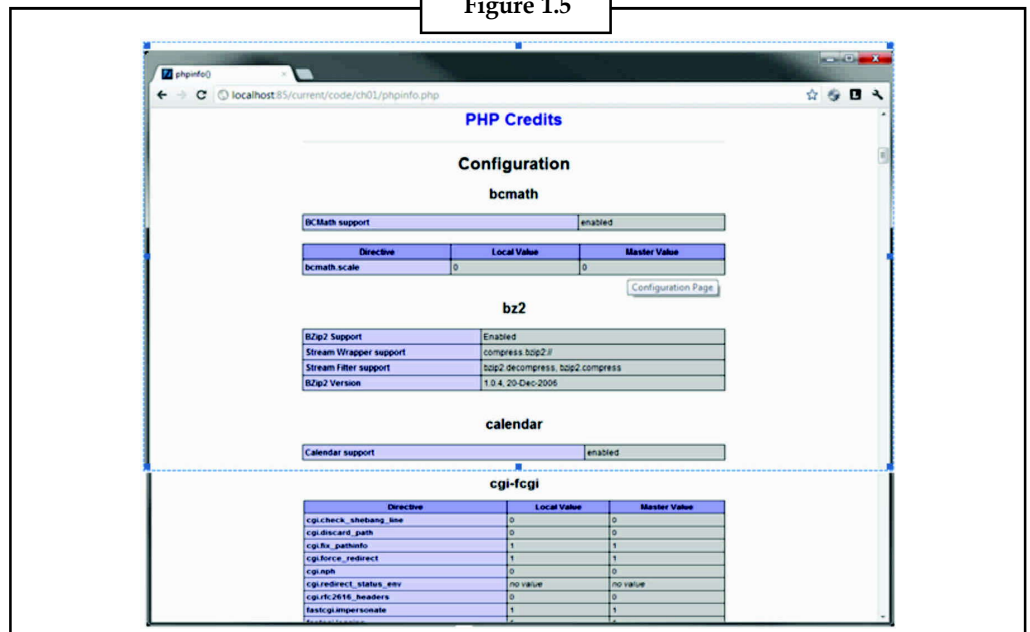


Example: Using phpinfo()

```

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

Figure 1.5



1.4.2 Forms

Notes

When the user submits the form, the information typed into the name field is sent back to this page. The PHP code tests for a name field and displays a greeting if it finds one.



Example:

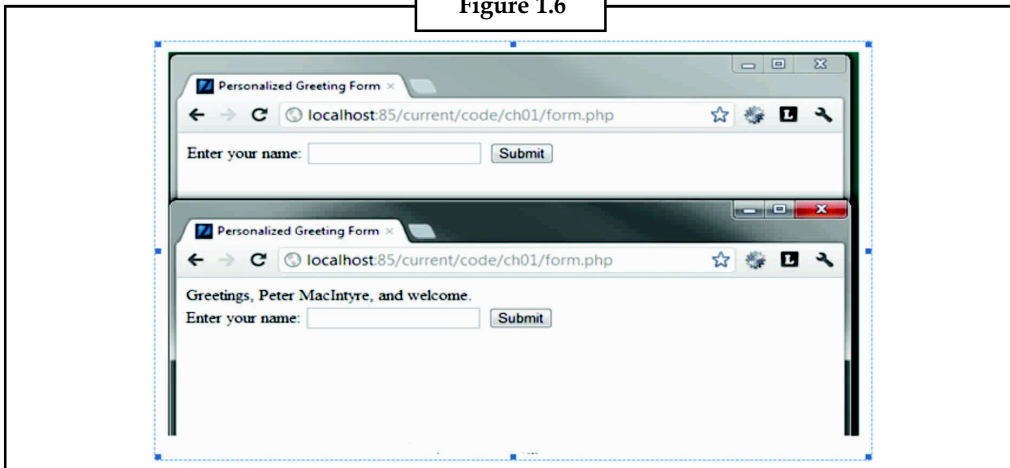
```
. Processing a form (form.php)
<html>
  <head>
    <title>Personalized Greeting Form</title>
  </head>

  <body>
    <?php if(!empty($_POST['name'])) {
      echo "Greetings, {$_POST['name']}, and welcome.";
    } ?>

    <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
      Enter your name: <input type="text" name="name" />
      <input type="submit" />
    </form>
  </body>
</html>
```

The form and the message are shown in Figure 1.6.

Figure 1.6



PHP programs access form values primarily through the `$_POST` and `$_GET` array variables.

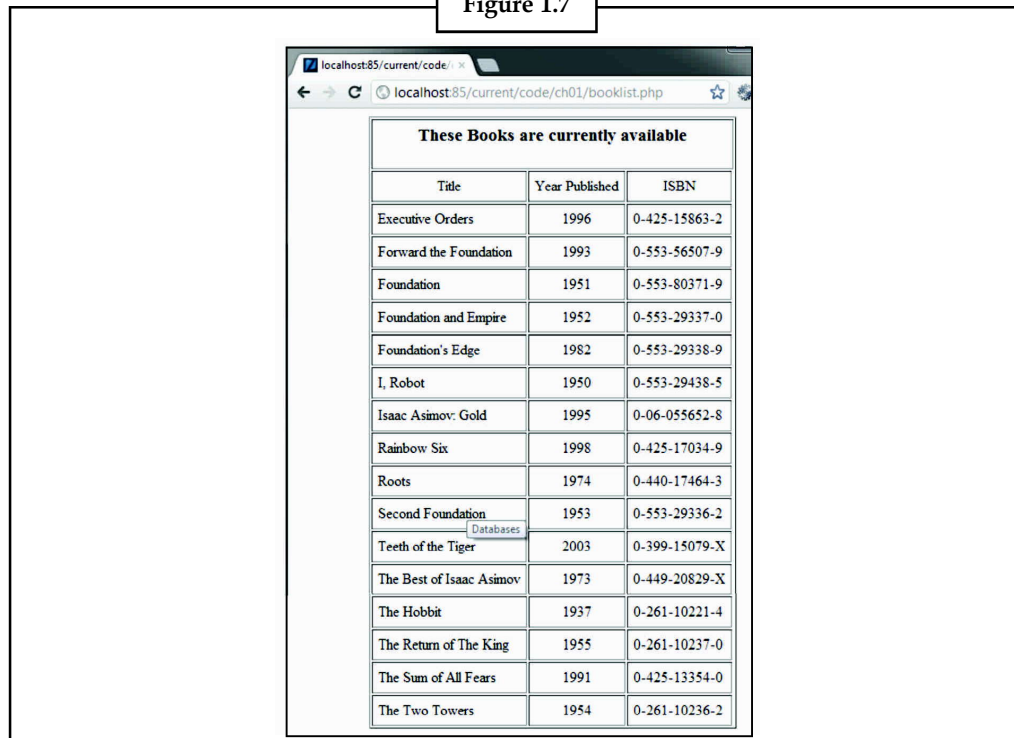
1.4.3 Databases

PHP supports all the popular database systems, including MySQL, PostgreSQL, Oracle, Sybase, SQLite, and ODBC-compliant databases. MySQL database query run through a PHP script showing the results of a book search on a book review site. This is showing the book title, the year the book was published, and the book's ISBN number.

Notes

The SQL code for this sample database is in the provided files called library.sql. The code connects to the database, issues a query to retrieve all available books (with the WHERE clause), and produces a table as output for all returned results through a while loop.

Figure 1.7



Example:

Querying the Books database (booklist.php)

```

<?php

$db = new mysqli("localhost", "petermac", "password", "library");

// make sure the above credentials are correct for your environment
if ($db->connect_error) {
    die("Connect Error ({$db->connect_errno}) {$db->connect_error}");
}

$sql = "SELECT * FROM books WHERE available = 1 ORDER BY title";
$result = $db->query($sql);

?>

<html>
<body>

<table cellSpacing="2" cellPadding="6" align="center" border="1">
    <tr>

```

Notes

```

    <td colspan="4">
        <h3 align="center">These Books are currently available</h3>
    </td>
</tr>

<tr>
    <td align="center">Title</td>
    <td align="center">Year Published</td>
    <td align="center">ISBN</td>
</tr>
<?php while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo stripslashes($row['title']); ?></td>
        <td align="center"><?php echo $row['pub_year']; ?></td>
        <td><?php echo $row['ISBN']; ?></td>
    </tr>
<?php } ?>
</table>

</body>
</html>

```

Database-provided dynamic content drives the news, blog, and e-commerce sites at the heart of the Web.

1.4.4 Graphics

With PHP, you can easily create and manipulate images using the GD extension. A text-entry field that lets the user specify the text for a button. It takes an empty button image file, and on it centers the text passed as the GET parameter 'message'. The result is then sent back to the browser as a PNG image.



Example:

```

Dynamic buttons (graphic_example.php)
<?php
if (isset($_GET['message'])) {
    // load font and image, calculate width of text
    $font = "times";
    $size = 12;
    $image = imagecreatefrompng("button.png");
    $tsize = imagettfbbox($size, 0, $font, $_GET['message']);

    // center
    $dx = abs($tsize[2] - $tsize[0]);
    $dy = abs($tsize[5] - $tsize[3]);
    $x = (imagesx($image) - $dx) / 2;

```

Notes

```
$y = (imagesy($image) - $dy) / 2 + $dy;

// draw text
$black = imagecolorallocate($im,0,0,0);
imagefttext($image, $size, 0, $x, $y, $black, $font, $_GET['message']);

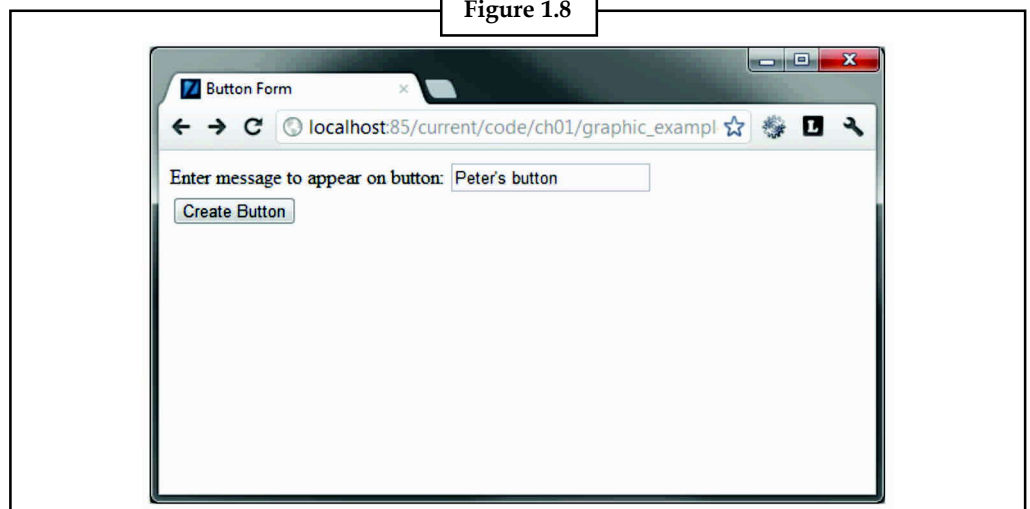
// return image
header("Content-type: image/png");
imagepng($image);

exit;
} ?>
<html>
<head>
<title>Button Form</title>
</head>

<body>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
  Enter message to appear on button:
  <input type="text" name="message" /><br />
  <input type="submit" value="Create Button" />
</form>
</body>
</html>
```

You can use GD to dynamically resize images, produce graphs, and much more. PHP also has several extensions to generate documents in Adobe’s popular PDF format.

Figure 1.8



Now that you’ve had a taste of what is possible with PHP, you are ready to learn how to run a program in PHP. We start with the basic structure of the language, with special focus given to user-defined functions, string manipulation, and object-oriented programming. Then we move to specific application areas such as the Web, databases, graphics, XML, and security. We finish with quick references to the built-in functions and extensions.



Caution Do not compile PHP without specifying a specific web server type; otherwise you will get a PHP interpreter as a program on the place of a web server module.

1.4.5 From the Shell

PHP Shell is a shell wrapped in a PHP script. It is a tool you can use to execute arbitrary shell-commands or browse the filesystem on your remote webserver. This replaces, to a degree, a normal telnet connection, and to a lesser degree a SSH connection.

You use it for administration and maintenance of your website, which is often much easier to do if you can work directly on the server. For example, you could use PHP Shell to unpack and move big files around. All the normal command line programs like ps, free, du, df, etcâ€¦ can be used.

Self Assessment

Fill in the blanks:

13. The web server processes the PHP commands and sends theirto the browser.
14. The PHP functioncreates an HTML page full of information on how PHP was installed and is currently configured.
15.database query run through a PHP script showing the results of a book search on a book review site.
16. PHP Shell is a shell wrapped in a PHP.....



Case Study

CppCMS vs PHP

Benchmark

In this benchmark two technologies were compared: CppCMS and PHP. This written using CppCMS technology was compared to WordPress blog powered by PHP5.

Set-up

I had compared two blog systems: this one and WordPress 2.5 with a patched WP-Cache-2 add on. I used following configuration:

1. Web Server lighttpd 1.4.13
2. Interface FastCGI
3. PHP 5.2
4. Bytecode cacher: XCache 1.2.1
5. Database MySQL 5.0
6. Caching for WP: WP-Cache-2 with an additional performance patch that caches already deflated versions of page in order to reduce requirement of re-compressing on cache hit.

Contd...

Notes

7. Hardware: AMD Athlon XP 64bit, 1G RAM

8. OS: Linux, Debian Etch 64bit.

I prepared two blogs that were filled up with 1000 articles each. Each article had 10 comments, all the articles were organized in 10 categories in each blog.

Tests

First I run load tests with disabled caching system. Then the cache was enabled and cleaned before each test run. Each time, the cache was “warmed up” with 100 requests of different pages. Then CMS was loaded by http_load with 1000 requests from 5 concurrent connection. The client was patched in order to send a header: Accept-Encoding: gzip, deflate in order to fetch compressed pages: the real live situation. For each run, the cache included a certain percent of new pages (in order to achieve a correct hit/miss ratio) and the rest were taken from the 100 “warm pages”.

Results

CMS No cache fetches per second ratio Warm up time

WordPress: 7.6 13.0 s

CppCMS: 310 0.28 s

Miss ratio (%) WordPress CppCMS

0 711 2300

1 370 2160

3 176 1940

5 118 1790

10 64 1450

15 41 1210

20 33 1075

30 20 795

40 13 570

50 13 570

Requests per second for different hit/miss ratio

Questions

1. How many blog made in CppCMS technology explains.
2. Explain the testing process of CppCMS technology blog.

1.5 Summary

- Constants value does not change, whereas loop control structures are used for repeating certain tasks.
- After you have compiled or installed PHP, you can still change its behavior with the php.ini file.
- One of PHP’s most significant features is its wide-ranging support for databases.

- PHP supports all major databases (including MySQL, PostgreSQL, Oracle, Sybase, and ODBC compliant databases), and even many obscure ones.
- PHP pages are HTML pages with PHP commands embedded in them. This is in contrast to many other dynamic web-page solutions, which are scripts that generate HTML.
- PHP supports all the popular database systems, including MySQL, PostgreSQL, Oracle, Sybase, and ODBC-compliant databases.

1.6 Keywords

Binaries Directory: Where programs and PHP scripts from PEAR packages are installed. The pear executable ends up here.

Database Directory: These are the directory where the PEAR Installer stores data files.

Hypertext Preprocessor (PHP): It is a scripting language originally designed for web development to produce web pages.

Installation Prefix: It is the root directory of PEAR installation. It has no other effect than serving as a root for the next five settings, using and prefix.

PHP Code Directory: This directory exists where PHP code is installed. This directory is the include N path when using the packages you install.

Tests Base Directory: These are the directory where regression test scripts for the package are installed.

1.7 Review Questions

1. What was the main purpose to develop the PHP? How was it used?
2. Give a brief history of PHP.
3. Write the steps how to install PHP in your PC.
4. What are the minimum requirements to install the PHP in your PC?
5. Write the main features of PHP.
6. Write a PHP program to process a form.
7. Discuss which databases are mainly used in PHP. What will be the program to bond the data base in PHP?
8. What is the use of graphics in PHP? Explain with example.
9. Develop a shell-based PHP program to create a dropdown list.
10. Write a code for configure PHP as an Apache module.

Answers: Self Assessment

- | | |
|----------|----------|
| 1. True | 2. True |
| 3. False | 4. True |
| 5. 1994 | 6. 1997 |
| 7. False | 8. False |

Notes

- | | |
|------------|---------------|
| 9. True | 10. False |
| 11. True | 12. True |
| 13. Output | 14. phpinfo() |
| 15. MySQL | 16. Script |

1.8 Further Readings



Books

Bangia, Ramesh (2008). *Web Technology (including HTML, CSS, XML, ASP, JAVA)*. Firewall Media.

Jackson (2007). *Web Technologies: A Computer Science Perspective*. Pearson Education India.

Kamal, Raj (2002). *Internet and Web Technologies*. Tata McGraw-Hill Education.

Puntambekar, A.A. (2009). *Web Technologies*. Technical Publications.

Sarukkai, Ramiesh R. (2002). *Foundations of Web Technology*. Springer.

Xavier, C. (2007). *Web Technology and Design*. New Age International.



Online links

<http://books.google.co.in/books?id=Zwzc4qUZnqMC&lpg=PA33&dq=Introduction%20to%20PHP&pg=PA33#v=onepage&q=Introduction%20to%20PHP&f=false>

<http://tabs.stanford.edu/webworks/Winter%200405%20-%20Beginning%20PHP.pdf>

<http://php.net/manual/en/introduction.php>

<http://alison.com/courses/Introduction-to-PHP-and-MySQL-Programming>

<http://grid.ucy.ac.cy/~EPL425/labs/PHPIntroduction.pdf>

<http://www.oreillyschool.com/individual-courses/introphp/>

Unit 2: Language Basics

Notes

CONTENTS

Objectives

Introduction

2.1 Lexical Structure

2.2 Data Types

2.2.1 Integers

2.2.2 Floating Point Number

2.2.3 String

2.2.4 Special Characters

2.2.5 Here Documents

2.2.6 Boolean

2.2.7 Compound Data Types

2.2.8 Resources

2.2.9 Null

2.3 Variables

2.3.1 Declaring Variables

2.3.2 Variable Variables and Variable References

2.3.3 Variable Scope

2.3.4 Environment Variable

2.4 Expressions and Operators

2.4.1 Number of Operands

2.4.2 Operator Precedence

2.4.3 Operator Associativity

2.4.4 Operator Concept

2.5 Summary

2.6 Keywords

2.7 Review Questions

2.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Discuss about the Lexical Structure
- Understand the Data Types used in PHP

Notes

- Explain the Variables in PHP
- Discuss about the Expressions and Operators in PHP

Introduction

PHP has become a more popular technology for web programming in the recent years due to its open source technology. This means that it is free to use and is compatible with almost any server, say Windows, Linux/Unix and Apache servers. Though it is an open source technology, it has many advanced features included in it and is now capable of handling almost any issue related to the internet.

PHP stands for “PHP Hypertext Preprocessor”. PHP is a server-side scripting language. This means that a PHP page is processed on a Web server by a PHP script engine. When the client makes a request for any page, the request is sent to the server, the server locates the requested page and executes the PHP code if any and the result is sent to the browser (client) in the form of HTML. The browser then compiles the HTML to display the output.

Now let's start with the basics of PHP programming. PHP code is written within `<?php ... ?>`. To understand better, let's look at the following example:



Example:

```
<html>
< head>
< title>SmartWebby's PHP Tutorials</title>
< /head>
< body>
< ?php
echo "Welcome to SmartWebby's Guide to PHP Programming"
?>
< /body>
< /html>
```

Save the above code as 'sample.php'. In the above example we have used PHP to display the message “Welcome to SmartWebby’s Guide to PHP Programming”. Once the client requests this sample webpage, it just displays “Welcome to SmartWebby’s Guide to PHP Programming”. Now if you view the source code from your browser, it will contain only the HTML code as shown below:



Example:

```
<html>
< head>
< title>SmartWebby's PHP Tutorials</title>
< /head>
< body>
Welcome to SmartWebby's Guide to PHP Programming
< /body>
< /html>
```

Thus it proves that PHP scripts are processed only by the server and not by the client (browser).

2.1 Lexical Structure

Embedding:

PHP code is inserted into an HTML page between the start and end tags `<? and ?>`. The word PHP follows the start tag.

```
html code
<?php
    ... PHP code goes here
?>
more html code
```

Case Sensitivity:

- The names of user-defined classes and functions, as well as built-in constructs and keywords are case-insensitive.
- Variables are case sensitive.

Statements and Semicolons:

PHP uses semicolons to separate simple statements. A compound statement that uses curly braces to mark a block of code does not need a semicolon after the closing brace. The semicolon before the closing brace is not optional. The semicolon before a closing PHP tag is optional.

```
for($x = 1; $x <= 5; $x++){
echo "x now has the value $x\n";

echo 'twice x is ` . $x * 2 . "\n';
}
```

Whitespace and Linebreaks:

White space and linebreaks are ignored in a PHP program. A statement can be spread across any number of lines.

Comments:

Comments are ignored. There are three styles of comments in PHP:

1. *Shell-style comments:* When PHP encounters a hash mark (#) within the code, everything from the hash mark to the end of the line or the end of the section of PHP code (whichever comes first) is considered a comment.

```
<?php $d = 4 #Set $d to 4 ?> Then another <?php echo $d ?>

Then another 4
```

2. *C++ style comments:* When PHP encounters two slash characters (//) within the code, everything from the slashes to the end of the line or the end of the section of PHP code (whichever comes first) is considered a comment.

```
<?php $d = 4 //Set $d to 4 ?> Then another <?php echo $d ?>

Then another 4
```

3. *C style comments:* When PHP encounters a slash followed by an asterisk, (/*) everything after that until it encounters an asterisk followed by a slash (*/) is considered a comment. This style of comment, unlike the other two, can span multiple lines. Also, C style comments continue past end markers.

Notes

```
<?php $d = 4 /* Set $d to 4 */ ?> Then another <?php echo $d ?>  
Then another 4
```

Literals:

Literals are, of course a part of PHP just as the are in any language.

Identifiers:

Identifiers are used to name variables, functions, constants, and classes. The first character of an identifier must be either an ASCII letter, the underscore character (_) or any of the characters between ASCII 0x7F and ASCII 0xFF. After the initial character, these characters and the digits 0-9 are valid.

Variable names:

Variable names always begin with a dollar sign (\$) and are case sensitive.

Function names:

Function names are not case sensitive.

Class names:

Class names follow the standard rules for PHP identifiers and are not case sensitive. The class name stdClass is reserved.

Constants:

A constant is an identifier for a simple value; only scalar values – boolean, integer, double, and string – can be constants. Once set, the value of a constant cannot change during the execution of the PHP program. Constants are referred to by their identifiers and are set using the define() function:

```
define('LUG', 'Melug-Central');  
echo LUG;
```

Keywords:

A keyword is a word reserved by the language for its core functionality. You cannot give a variable, function, class, or constant the same name as a keyword. In addition, you cannot use an identifier that is the same as a built-in PHP function.

Self Assessment

State whether the following statements are true or false:

1. PHP code is inserted into an HTML page between the start and end tags ?> and <?.
2. PHP uses semicolons to separate simple statements.
3. Literals are not a part of PHP just as the are in any language.
4. Function names are case sensitive.

2.2 Data Types

PHP provides eight types of values. Four are scalar, two are compound types, and two are special types.

Scalar Types:

- Integers
- Floating-point numbers
- Strings
- Booleans

Compound Types:

- Arrays
- Objects

Special Types:

- Resource
- NULL

2.2.1 Integers

Integers are a subset of the real numbers. They are written without a fraction or a decimal component. Integers fall within a set $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$ Integers are infinite.

In computer languages, integers are primitive data types. Computers can practically work only with a subset of integer values, because computers have finite capacity. Integers are used to count discrete entities. We can have 3, 4, 6 humans, but we cannot have 3.33 humans. We can have 3.33 kilograms.



Notes Integers can be specified in three different notations in PHP. Decimal, hexadecimal and octal. Octal values are preceded by 0, hexadecimal by 0x.



Example:

```
<?php

$var1 = 31;
$var2 = 031;
$var3 = 0x31;

echo "$var1\n";
echo "$var2\n";
echo "$var3\n";

?>
```

We assign 31 to three variables using three notations. And we print them to the console.

```
$ phpnotation.php
31
25
49
```

Notes

The default notation is the decimal. The script shows these three numbers in decimal.

Integers in PHP have a fixed maximum size. The size of integers is platform dependent. PHP has built-in constants to show the maximum size of an integer.



Example:

```
$ uname -mo
i686 GNU/Linux

$ php -a
Interactive shell

php> echo PHP_INT_SIZE;
4

php> echo PHP_INT_MAX;
2147483647 begin_of_the_skype_highlighting 2147483647 FREE
end_of_the_skype_highlighting

php>
```

On my 32bit Ubuntu system, an integer value size is four bytes. The maximum integer value is 2147483647begin_of_the_skype_highlighting 2147483647 FREE end_of_the_skype_highlighting.

In Java and C, if an integer value is bigger than the maximum value allowed, integer overflow happens. PHP works differently. In PHP, the integer becomes a float number. Floating point numbers have greater boundaries.



Example:

```
<?php

$var = PHP_INT_MAX;

echo var_dump($var);
$var++;
echo var_dump($var);

?>
```

We assign a maximum integer value to the \$var variable. We increase the variable by one. And we compare the contents.

```
$ phpboundary.php
int(2147483647)
float(2147483648 begin_of_the_skype_highlighting 2147483648 FREE
end_of_the_skype_highlighting)
```

As we have mentioned previously, internally, the number becomes a floating point value.

In Java, the value after increasing would be -2147483648. This is where the term integer overflow comes from. The number goes over the top and becomes the smallest negative integer value assignable to a variable.

If we work with integers, we deal with discrete entities. We would use integers to count apples.



Example:

```
<?php

# number of baskets
$baskets = 16;

# number of apples in each basket
$apples_in_basket = 24;

# total number of apples
$total = $baskets * $apples_in_basket;

echo "There are total of $total apples \n";
?>
```

In our script, we count the total amount of apples. We use the multiplication operation.

```
$ phpapples.php
```

There are total of 384 apples

The output of the script.



Did u know? PHP is a loosely-typed language, so a variable does not need to be of a specific type and can freely move between types as demanded by the code it is being used.

2.2.2 Floating Point Number

Floating point numbers represent real numbers in computing. Real numbers measure continuous quantities. Like weight, height or speed. Floating point numbers in PHP can be larger than integers and they can have a decimal point. The size of a float is platform dependent.

We can use various syntax to create floating point values.



Example:

```
<?php

$a = 1.245;
$b = 1.2e3;
$c = 2E-10;
$d = 1264275425335735;

var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
?>
```

Notes

In this example, we have two cases of notations, that are used by scientists to denote floating point values. Also the \$d variable is assigned a large number, so it is automatically converted to float type.

```
$ phpfloats.php
float(1.245)
float(1200)
float(2.0E-10)
float(1264275425340000)
```

This is the output of the above script.

According to the documentation, floating point numbers should not be tested for equality. We will show an example why.

```
$ php -a
Interactive shell

php> echo 1/3;
0.333333333333
php> $var = (0.333333333333 == 1/3);
php>var_dump($var);
bool(false)
php>
```

In this example, we compare two values that seem to be identical. But they yield unexpected result.

Let's say a sprinter for 100m ran 9.87s. What is his speed in km/h?



Example:

```
<?php

# 100m is 0.1 km

$distance = 0.1;

# 9.87s is 9.87/60*60 h

$time = 9.87 / 3600;

$speed = $distance / $time;

echo "The average speed of a sprinter is $speed \n";

?>
```

In this example, it is necessary to use floating point values.

```
$speed = $distance / $time;
```

To get the speed, we divide the distance by the time.

```
$ phpsprinter.php
```

The average speed of a sprinter is 36.4741641337

This is the output of the sprinter script. 36.4741641337 is a floating point number.



Did u know? If you want to know if a variable contains a floating-point number, you can use the `is_float()` function to test the value. If it is a floating-point number, the function will return `true`.

2.2.3 String

String is a data type representing textual data in computer programs. Probably the single most important data type in programming.

Since string are very important in every programming language, we will dedicate a whole unit to them. Here we only drop a small example.



Example:

```
<?php
```

```
$a = "PHP ";
```

```
$b = 'PERL';
```

```
echo $a, $b;
```

```
echo "\n";
```

```
?>
```

We can use single quotes and double quotes to create string literals.

```
$ phpstrings.php
```

```
PHP PERL
```

The script outputs two strings to the console. The `\n` is a special sequence, a new line. The effect of this character is like if you hit the enter key when typing text.

2.2.4 Special Characters

Anything within quotes is part of a string, even numbers. Thus, `"123"` is a string, not a number. But there are special characters which cannot be included directly in a quote. For instance, you cannot have double quotes inside a double-quoted string, and you cannot have single quotes inside a single-quoted string. To include these special characters, you need to escape them so that the parser knows that they are literal characters and not string delimiters.

In PHP, you escape a character by preceding it with a backslash. There are two types of escaped characters in PHP. There are those that are a special character in the language and need to be escaped to get the parser to treat them as a literal. For instance, `\'` says that the quote after the backslash is a literal quote, and `\\` is used to include a literal backslash in a string. There are also those that are normal characters that serve as special characters when they are escaped. For instance, `\n` inserts a line break in the string at that location, and `\r` inserts a carriage return. Yes, they are the same thing to us, but they are different characters as far as the computer is concerned.

Notes

Some of the more common escape characters are:

- `\"` embeds a literal double quote in a string
- `\'` embeds a literal single quote or apostrophe in a string
- `\\` embeds a literal backslash in a string
- `\$` embeds a literal dollar sign in a string
- `\{` embeds a literal left brace in a string
- `\}` embeds a literal right brace in a string
- `\[` embeds a literal left bracket in a string
- `\]` embeds a literal right bracket in a string
- `\n` embeds a new line character in a string
- `\r` embeds a carriage return character in a string
- `\0` through `\777` represents and embeds any valid ASCII character value in octal format.
- `\x0` through `\xff` represents and embeds any valid ASCII character value in hexadecimal format.
- `\x0` through `\xffff` represents and embeds any valid Unicode character value in hexadecimal format.

Strings in single quotes are not technically parsed, but you can use `\'` and `\\` to escape single quotes and backslashes in unparsed strings. No other escaped characters will work in single quoted string literals. They will be treated as literal text. One important thing you should not have in a string is the sub-string `?>`, which ends that PHP script, even if it does occur inside a string. In fact, even `? \>` is a bad idea. It is recommended that you use the numeric value for the question mark and write it `\x3f`. `3f` is the hex value for `?`. You can use the `is_string()` function to test whether something is a string. If it is a string, it will return true. PHP includes many string related functions, since much of what it does is string processing.



Did u know? Special characters can either be a single character preceded by a backslash or a numeric value in either octal or hexadecimal preceded by a backslash.

2.2.5 Here Documents

In order to allow people to easily write large amounts of text from within PHP, but without the need to constantly escape things, heredoc syntax was developed. Heredoc might be a little tricky to understand at first, but it's actually a big help. Put simply, it allows you to define your own string limiter so that you can make it something other than a double or single quote. So, for example, we could use the string "EOT" (end of text) for our delimiter, meaning that we can use double quotes and single quotes freely within the body of the text - the string only ends when we type EOT.

It is a little more complicated than that in practice, but not much - the string delimiter needs to be by itself on a line, in the very first column. That is, you cannot add spacing or tabs around it.

Take a look at this example:

Notes



Example:

```
<?php

$mystring = <<<EOT
    This is some PHP text.
    It is completely free
    I can use "double quotes"
    and 'single quotes',
    plus $variables too, which will
    be properly converted to their values,
    you can even type EOT, as long as it
    is not alone on a line, like this:
EOT;

?>
```

There are several key things to note about heredoc, and the example above:

- You can use anything you like; "EOT" is just an example
- You need to use <<< before the delimiter to tell PHP you want to enter heredoc mode
- Variable substitution is used in PHP, which means you do need to escape dollar symbols - if you do not, PHP will attempt variable replacement.
- You can use your delimiter anywhere in the text, but not in the first column of a new line
- At the end of the string, just type the delimiter with no spaces around it, followed by a semicolon to end the statement



Notes Without heredoc syntax, complicated string assignments can quickly become very messy. Heredoc is not used all that often in the wild - very often you will wish it were used more, because too many scripts you will come across have messy code as a result of not using heredoc!

2.2.6 Boolean

There is a duality built in our world. There is a Heaven and Earth, water and fire, jing and jang, man and woman, love and hatred. In PHP the boolean data type is a primitive data type having one of two values: True or False. This is a fundamental data type. Very common in computer programs.

Happy parents are waiting a child to be born. They have chosen a name for both possibilities. If it is going to be a boy, they have chosen John. If it is going to be a girl, they have chosen Victoria.



Example:

```
<?php

$male = False;
$r = rand(0, 1);
```

Notes

```
$male = $r ? True: False;
if ($male) {
echo "We will use name John\n";
} else {
echo "We will use name Victoria\n";
}

?>
```

The script uses a random integer generator to simulate our case.

```
$r = rand(0, 1);
```

The rand() function returns a random number from the given integer boundaries. In our case 0 or 1.

```
$male = $r ? True: False;
```

We use the ternary operator to set a \$male variable. The variable is based on the random \$r value. If \$r equals to 1, the \$male variable is set to True. If \$r equals to 0, the \$male variable is set to False.

```
if ($male) {
echo "We will use name John\n";
} else {
echo "We will use name Victoria\n";
}

?>
```

We print the name. The if command works with boolean values. If the variable \$male is True, we print the "We will use name John" to the console. If it has a False value, we print the other string.

The following script shows some common values that are considered to be True or False. For example, empty string, empty array, 0 are considered to be False.



Example:

```
<?php
class Object {};

var_dump((bool) "");
var_dump((bool) 0);
var_dump((bool) -1);
var_dump((bool) "PHP");
var_dump((bool) array(32));
var_dump((bool) array());
var_dump((bool) "false");
var_dump((bool) new Object());
var_dump((bool) NULL);

?>
```

In this script, we inspect some values in a boolean context. The var_dump() function shows information about a variable. The (bool) construct is called casting. In its casual context, the 0 value is a number. In a boolean context, it is False. The boolean context is when we use (bool)

casting, when we use certain operators (negation, comparison operators) and when we use if/else, while keywords.

```
$ phpboolean.php
bool(false)
bool(false)
bool(true)
bool(true)
bool(true)
bool(true)
bool(false)
bool(true)
bool(true)
bool(false)
```

Here is the outcome of the script.

2.2.7 Compound Data Types

Array

Array is a complex data type which handles a collection of elements. Each of the elements can be accessed by an index. In PHP, arrays are more diverse. Arrays can be treated as arrays, lists or dictionaries. In other words, arrays are all what in other languages we call arrays, lists, dictionaries.

Because collections are very important in all computer languages, we dedicate two chapters to collections - arrays. Here we show only a small example.



Example:

```
<?php

$names = array("Jane", "Lucy", "Timea", "Beky", "Lenka");
print_r($names);

?>
```

The array keyword is used to create a collection of elements. In our case we have names. The `print_r` function prints a human readable information about a variable to the console.

```
$ phpinit.php
Array
(
    [0] => Jane
    [1] => Lucy
    [2] =>Timea
    [3] =>Beky
    [4] =>Lenka
)
```

Output of the script. The numbers are indices by which we can access the names.

Notes

Objects

PHP is capable of functioning as an object-oriented programming language (or OOP). As such, it must be able to handle objects. An object is a data type that allows for the storage of not only data but also information on how to process that data. The data elements stored within an object are referred to as its properties, also sometimes called the attributes of the object. The information, or code, describing how to process the data comprises what are called the methods of the object.

To create a new object, use the new statement to instantiate a class:



Example:

```
<?php

class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();

?>
```

2.2.8 Resources

A resource is a special variable, holding a reference to an external resource. Resources are created and used by special functions.

The following is a list of few functions which create, use or destroy PHP resources.

```
fbsql_db_query()-Selects a database and executes a query on it.
ftp_connect()-opens an FTP connection to the specified host .
imap_open()-Open an IMAP stream to a mailbox
dba_popen()-establishes a persistent database instance for path with mode
using handler .
imagerotate()-Rotate an image with a given angle
```

The function `is_resource()` can be used to determine if a variable is a resource and function `get_resource_type()` will return the type of resource it is.

2.2.9 Null

There's only one value of the NULL data type. That value is available through the case-insensitive keyword NULL. The NULL value represents a variable that has no value (similar to Perl's undef or Python's None):

```
$aleph ="beta";$aleph = null; //variable's value is gone
$aleph = Null; // same
$aleph =NULL; //same
```


Self Assessment

Notes

Fill in the blanks:

5. PHP providestypes of values.
6.are a subset of the real numbers.
7.is a data type representing textual data in computer programs.
8. Anis a data type that allows for the storage of not only data but also information on how to process that data.

2.3 Variables

The main way to store information in the middle of a PHP program is by using a variable.

Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign (\$).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types – a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables:

- **Integers:** are whole numbers, without a decimal point, like 4195.
- **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
- **Booleans:** have only two possible values either true or false.
- **NULL:** is a special type that only has one value: NULL.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'
- **Arrays:** are named and indexed collections of other values.
- **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

2.3.1 Declaring Variables

Rules for declaring a variable is:

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like +, -, %, (,), . &, etc.

2.3.2 Variable Variables and Variable References

PHP allows you to do some neat things with variables. It allows you to create aliases for variables, and it also allows you to have variables whose name is a variable.

A variable reference, or alias, is a variable assigned to refer to the same information as another variable. To assign an alias to a variable, you use the reference operator, which is an equals sign followed by an ampersand (=&).

```
$that = 'abc'; // $that now equals 'abc'
$this =& $that; // $this now equals 'abc';
$that = 'def'; // both now equal 'def';
$this = 'ghi'; // both now equal 'ghi';
```

Since they are both now references for the same location in memory, changing the value of one, changes the value of the other. Note however that if you unset () one alias, the other aliases for the same value will not be affected. This is because unset () removes the reference to the memory location for that variable, rather than changing the value stored in that memory location. You can also use references with functions to return values by reference instead of by value. To do this, you precede the name of the function with an ampersand (&) when you define the function, and then use the reference operator to assign its returned value.

Variable variables are variables whose name is a variable. In other words, the name of the variable refers to a memory location that stores the name of the variable that stores the information you want.

gets parsed twice.

```
$foo_bar = "abc";
$tip = "foo_bar";
echo $tip; // writes out 'foo_bar'
echo $$tip // writes out 'abc'
```

Sometimes curly brackets are used to make the code a little clearer. Curly brackets, in PHP, are grouping elements. By using them with a variable name, you can make it clear what each dollar sign is acting on in the variable name. This makes the code easier to read.

```
$$tip // harder to read
${$tip} // easier to read
```

2.3.3 Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types:

- Local variables
- Function parameters
- Global variables
- Static variables

In PHP, all variables are local in scope, even the global ones.

Local Scope

A local variable is one that is specific to a given instance of a given function. It is created when the function processing begins and is deleted as soon as the function is completed. Local scope is

specific to functions. PHP does not have a block level scope. If you want a block level scope, your best bet is to emulate it with a recursive function.

Global Scope

Global scope refers to any variable that is defined outside of any function. They can be accessed from any part of the program that is not inside a function. To access a global variable from within a function, you can call it into the function with the global keyword:

```
global $varToInclude;
```

PHP also stores all global variables in an array called `$GLOBALS[]`. Its index is the name of the variable. This array is accessible from within functions and can be used to update global variables directly.

```
global $somVar;
$someVar = 'abc'
// is the same as
```

- `$GLOBALS["someVar"] = 'abc';`

Static Scope

Normally when a function terminates, all of its variables are also cleaned up. Sometimes you want a local variable to persist between instances of a given function. To do this, you use the static keyword when you first declare the variable. Then each time you call the function, that variable will still have the information it contained from the last time the function was called. Static `$aValueToRemember;` Even though the value persists, the variable is still local to the function.

Parameters

A parameter is a local variable whose value is passed to the function by the calling code. Unlike other variables, parameters are declared in a parameter list as part of the function declaration.

- Parameters are also sometimes called arguments.

The concept of variable scope is discussed in detail in unit 4.

2.3.4 Environment Variable

PHP environment variables allow your scripts to glean certain types of data dynamically from the server. This supports script flexibility in a potentially changing server environment. For example, the `SITE_HTMLROOT` variable provided by (mt) Media Temple will automatically provide the correct path to your document root on any (gs) Grid-Service server, without necessitating any changes in your script. (mt) Media Temple provides several dozen variables like this for your convenience.

Self Assessment

State whether the following statements are true or false:

9. All variables in PHP are denoted with a leading dollar sign (\$).
10. Booleans are floating-point numbers.

Notes

11. Variable names must begin with a letter or underscore character.
12. Local scope refers to any variable that is defined outside of any function.

2.4 Expressions and Operators

The basic building block of a program is the statement. A statement is a piece of code that does something. Statements themselves are made up of expressions and operators. An expression is a piece of code that evaluates to some value. An operator is a code element that acts on an expression in some way. For instance, a minus sign can be used to tell the computer to delete the value of the expression after it from the expression before it.

Since there is not really much to understand about expressions except for the assembly of them into compound expressions and statements using operators, we are going to look at the operators used to turn expressions into more complex expressions and statements. This takes a look at operators and how to use them to form complex expressions and statements. It starts with a general overview and then proceeds to look at different types of operators and their uses.

Statements themselves are made up of expressions and operators. An *expression* is a piece of code that evaluates to some value. The simplest form of expression is a literal or a variable. A literal evaluates to itself. A variable evaluates to the value assigned to it.

For instance, any of the following are valid expressions:

```
"abc"  
123  
$a  
$x == 7  
($a + $b) / $c
```

Although a literal or variable may be a valid expression, they are not expressions that do anything. The way you get expressions to do things is by linking simple expressions together with operators. An operator combines simple expressions together into more complex expressions by creating relationships between the simple expressions that can be evaluated. For instance, if the relation you want to establish is the cumulative joining of two numeric values together, you could write $6 + 7$. The numbers 6 and 7 are each valid expressions. The equation $6 + 7$ is also a valid expression, whose value, in this case, happens to be 13.

2.4.1 Number of Operands

Most operators in PHP are binary operators; they combine two operands (or expressions) into a single, more complex expression. PHP also supports a number of unary operators, which convert a single expression into a more complex expression. Finally, PHP supports a single ternary operator that combines three expressions into a single expression.

2.4.2 Operator Precedence

Here's a list of the operators you've met so far, and the order of precedence. This can make a difference, as we saw during the mathematical operators. Don't worry about these too much, unless you're convinced that your math or logical is correct. In which case, you might have to consult the following (Figure 2.1):

Figure 2.1: Operator Precedence

*/%	Highest Precedence
+,-	
<<=>>=	
== !=	
&&	
And	
XOR	
OR	Lowest Precedence

The only operators you haven't yet met on the list above are the `===` and `!==` operators.

In recent editions of PHP, two new operators have been introduced: the triple equals sign (`===`) and an exclamation, double equals (`!==`). These are used to test if one value has the same as another AND are of the same type. An example would be:

```
$number = 3;
$text = 'three';
if ($number === $text) {
    print("Same");
}
else {
    print("Not the same");
}
```

So this asks, "Do the variables match exactly?" Since one is text and the other is a number, the answer is "no", or false. We won't be using these operators much, if at all.



Caution Operator precedence should be used based on precedence, otherwise result will affect.

2.4.3 Operator Associativity

Associativity defines the order in which operators with the same order of precedence are evaluated.



Example: Look at:

`2 / 2 * 2`

The division and multiplication operators have the same precedence, but the result of the expression depends on which operation we do first:

`2/(2*2) // 0.5 (2/2)*2 // 2`

The division and multiplication operators are left-associative; this means that in cases of ambiguity, the operators are evaluated from left to right. In this example, the correct result is 2.

2.4.4 Operator Concept

Simple answer can be given using expression `4 + 5` is equal to 9. Here 4 and 5 are called operands and `+` is called operator. PHP language supports following type of operators:

Notes

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

There are following arithmetic operators supported by PHP language:

Assume variable A holds 10 and variable B holds 20 then:

Table 2.1: Variables in Arithmetic Operators

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denominator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9



Example:

```

<html>
<head><title>Arithmetical Operators</title></head>
<body>
<?php
    $a = 42;
    $b = 20;

    $c = $a + $b;
    echo "Addtion Operation Result: $c <br/>";
    $c = $a - $b;
    echo "Substraction Operation Result: $c <br/>";
    $c = $a * $b;
    echo "Multiplication Operation Result: $c <br/>";
    $c = $a / $b;
    echo "Division Operation Result: $c <br/>";
    $c = $a % $b;
    echo "Modulus Operation Result: $c <br/>";
    $c = $a++;
    echo "Increment Operation Result: $c <br/>";
    $c = $a--;

```

```

    echo "Decrement Operation Result: $c <br/>";
?>
</body>
</html>

```

Comparison Operators

There are following comparison operators supported by PHP language:

Table 2.2: Comparison Operators

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Assume variable A holds 10 and variable B holds 20 then:



Example:

```

<html>
<head><title>Comparision Operators</title><head>
<body>
<?php
    $a = 42;
    $b = 20;

    if( $a == $b ){
        echo "TEST1 : a is equal to b<br/>";
    }else{
        echo "TEST1 : a is not equal to b<br/>";
    }

    if( $a > $b ){
        echo "TEST2 : a is greater than b<br/>";
    }else{
        echo "TEST2 : a is not greater than b<br/>";
    }

    if( $a < $b ){
        echo "TEST3 : a is less than b<br/>";
    }else{

```

Notes

```

        echo "TEST3 : a is not less than b<br/>";
    }
    if( $a != $b ){
        echo "TEST4 : a is not equal to b<br/>";
    }else{
        echo "TEST4 : a is equal to b<br/>";
    }
    if( $a >= $b ){
        echo "TEST5 : a is either grater than or equal to b<br/>";
    }else{
        echo "TEST5 : a is nieghter greater than nor equal to b<br/>";
    }
    if( $a <= $b ){
        echo "TEST6 : a is either less than or equal to b<br/>";
    }else{
        echo "TEST6 : a is nieghter less than nor equal to b<br/>";
    }
?>
</body>
</html>

```

Logical Operators

There are following logical operators supported by PHP language:

Table 2.3: Variables in Logical Operators

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Assume variable A holds 10 and variable B holds 20 then:



Example:

```

<html>
<head><title>Logical Operators</title></head>
<body>
<?php
    $a = 42;
    $b = 0;

```


Notes

```
if( $a && $b ){
    echo "TEST1 : Both a and b are true<br/>";
}else{
    echo "TEST1 : Either a or b is false<br/>";
}
if( $a and $b ){
    echo "TEST2 : Both a and b are true<br/>";
}else{
    echo "TEST2 : Either a or b is false<br/>";
}
if( $a || $b ){
    echo "TEST3 : Either a or b is true<br/>";
}else{
    echo "TEST3 : Both a and b are false<br/>";
}
if( $a or $b ){
    echo "TEST4 : Either a or b is true<br/>";
}else{
    echo "TEST4 : Both a and b are false<br/>";
}
$a = 10;
$b = 20;
if( $a ){
    echo "TEST5 : a is true <br/>";
}else{
    echo "TEST5 : a is false<br/>";
}
if( $b ){
    echo "TEST6 : b is true <br/>";
}else{
    echo "TEST6 : b is false<br/>";
}
if( !$a ){
    echo "TEST7 : a is true <br/>";
}else{
    echo "TEST7 : a is false<br/>";
}
if( !$b ){
    echo "TEST8 : b is true <br/>";
}else{
    echo "TEST8 : b is false<br/>";
}
?>
</body>
</html>
```

Notes

Assignment Operators

There are following assignment operators supported by PHP language:

Table 2.4: Assignment Operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A



Example:

```
<html>
<head><title>Assignment Operators</title></head>
<body>
<?php
    $a = 42;
    $b = 20;

    $c = $a + $b; /* Assignment operator */
    echo "Addtion Operation Result: $c <br/>";
    $c += $a; /* c value was 42 + 20 = 62 */
    echo "Add AND Assignment Operation Result: $c <br/>";
    $c -= $a; /* c value was 42 + 20 + 42 = 104 */
    echo "Subtract AND Assignment Operation Result: $c <br/>";
    $c *= $a; /* c value was 104 - 42 = 62 */
    echo "Multiply AND Assignment Operation Result: $c <br/>";
    $c /= $a; /* c value was 62 * 42 = 2604 */
    echo "Division AND Assignment Operation Result: $c <br/>";
    $c %= $a; /* c value was 2604/42 = 62*/
    echo "Modulus AND Assignment Operation Result: $c <br/>";
?>
</body>
</html>
```

Conditional Operator

Notes

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax:

Table 2.5: Conditional Operator

Operator	Description	Example
?:	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Operators Categories

All the operators we have discussed above can be categorized into following categories:

- Unary prefix operators, which precede a single operand.
- Binary operators, which take two operands and perform a variety of arithmetic and logical operations.
- The conditional operator (a ternary operator), which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
- Assignment operators, which assign a value to a variable.



Example:

```
<html>
<head><title>Assignment Operators</title></head>
<body>
<?php
    $a = 42;
    $b = 20;

    $c = $a + $b;    /* Assignment operator */
    echo "Addition Operation Result: $c <br/>";
    $c += $a;    /* c value was 42 + 20 = 62 */
    echo "Add AND Assignment Operation Result: $c <br/>";
    $c -= $a;    /* c value was 42 + 20 + 42 = 104 */
    echo "Subtract AND Assignment Operation Result: $c <br/>";
    $c *= $a;    /* c value was 104 - 42 = 62 */
    echo "Multiply AND Assignment Operation Result: $c <br/>";
    $c /= $a;    /* c value was 62 * 42 = 2604 */
    echo "Division AND Assignment Operation Result: $c <br/>";
    $c %= $a;    /* c value was 2604/42 = 62*/
    echo "Modulus AND Assignment Operation Result: $c <br/>";
?>
</body>
</html>
```



Task Critically analyse how many operators are used in PHP.

Notes

Self Assessment

Fill in the blanks:

13. The basic building block of a program is the.....
14. Anis a piece of code that evaluates to some value. The simplest form of expression is a literal or a variable.
15. PHP also supports a number ofoperators, which convert a single expression into a more complex expression.
16. Anis a code element that acts on an expression in some way.



Case Study

Apache Software Foundation

The Apache Web Server from the Apache Software Foundation. It is a legendary product in many ways. The Internet as we know it now almost certainly would not be here if it was not for Apache’s web server. It is currently, and has been for many years, the number one web server on the Internet according to most analytical reports. This is despite the extreme lengths that some rivals have gone to try and skew the figures. The widely read monthly survey by Netcraft shows that in July 2008, Apache was serving almost 50% of the world’s websites. Apache has been the most popular web server on the Internet since April 1996.

The Apache web server became so popular for several reasons:

- **Cost:** It is Open Source and free.
- **Cross Platform:** Apache runs on almost every mainstream Operating System.
- **Modular:** Plug-in modules enable wide support for building interactive and dynamic websites using a multitude of backend services. Highly popular is the combination of the MySQL database and the PHP programming language.
- **Performance:** Consistently Apache has outperformed its rivals in terms of raw page impression performance, efficiency of memory and processor cycle usage and its ability to scale to support many websites within one running server instance.

Questions

1. Explain the software foundation of Apache.
2. What do you mean by PHP programming language?

2.5 Summary

- PHP is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document.
- All variables in PHP are prefixed with a dollar sign. The dollar sign is not technically part of the variable name, but it is required as the first character for the PHP parser to recognize the variable.

- Local scope is specific to functions. PHP does not have a block level scope. If you want a block level scope, your best bet is to emulate it with a recursive function.
- An operator combines simple expressions together into more complex expressions by creating relationships between the simple expressions that can be evaluated.
- When expressions and operators are assembled in such a way as to produce to a piece of code that actually does something, you have a statement. Statements end in semicolons and are the programming equivalent of the complete sentence.
- Most operators in PHP are binary operators; they combine two operands (or expressions) into a single, more complex expression. PHP also supports a number of unary operators, which convert a single expression into a more complex expression.

2.6 Keywords

Boolean: A Boolean value assesses the truth value of something. Booleans only have two values, true and false.

Floating-point Numbers: Floating-point numbers are also sometimes called real numbers. They are numbers that have a fractional component.

Integer: An integer is a whole number. That is to say, it is a number with no fractional component.

Keywords: A keyword is a reserved word in the PHP programming language. Keywords are used to perform a specific task in the computer program.

Logical Operators: Logical operators provide ways for you to build complex logical expressions.

Null: Null is a special data type which can have only one value, which is it.

Object: An object is a data type that allows for the storage of not only data but also information on how to process that data.

Operator: An operator is a code element that acts on an expression in some way.

Parameters: A parameter is a local variable whose value is passed to the function by the calling code.

Variable: A variable is just a name assigned to reference a location in memory where some value is stored.

2.7 Review Questions

1. What do you mean by lexical structure?
2. Explain the delimiters and Literals in PHP with example.
3. How many data types are used in PHP? Explain with example.
4. What are the special character and here documents in PHP? Discuss briefly.
5. Explain the compound data types used in PHP.
6. What are the variables? How many variables are used in PHP?
7. Explain about the scope of variables. How are they defined in PHP?
8. What are the expressions? Why these are used in PHP? Give the example.

Notes

9. What are the operators?
10. What are the difference between assignment operators and comparison operators? Explain with example.

Answers: Self Assessment

- | | |
|---------------|----------------|
| 1. False | 2. True |
| 3. False | 4. False |
| 5. Eight | 6. Integers |
| 7. String | 8. Object |
| 9. True | 10. False |
| 11. True | 12. False |
| 13. Statement | 14. Expression |
| 15. Unary | 16. Operator |

2.8 Further Readings



Books

Bangia, Ramesh (2008). "Web Technology (including HTML, CSS, XML, ASP, JAVA)." Firewall Media.

Jackson (2007). "Web Technologies: A Computer Science Perspective." Pearson Education India.

Kamal, Raj (2002). "Internet and Web Technologies." Tata McGraw-Hill Education.

Puntambekar, A.A. (2009). "Web Technologies." Technical Publications.

Sarukkai, Ramiesh R. (2002). "Foundations of Web Technology." Springer.

Xavier, C. (2007). "Web Technology and Design." New Age International.



Online links

http://www.tutorialspoint.com/php/php_variable_types.htm

http://www.tutorialspoint.com/php/php_global_variables.htm

<https://kb.mediatemple.net/questions/36/Using+Environment+Variables+in+PHP#gs>

<http://www.homeandlearn.co.uk/php/php3p12.html>

http://www.tutorialspoint.com/php/php_assignment_operators_examples.htm

Unit 3: Flow Control Statements and PHP in Web Page

Notes

CONTENTS

Objectives

Introduction

3.1 Conditional Statements

3.1.1 If Statement

3.1.2 Switch Statement

3.1.3 Other Formats

3.2 Advance Conditional

3.2.1 Comparing Strings

3.2.2 Similarity

3.3 Iteration

3.3.1 While Statement

3.3.2 Do-while Statement

3.3.3 For Statement

3.4 Termination Statements

3.4.1 Break Statement

3.4.2 Continue Statement

3.4.3 Stepping Out Multiple Levels

3.5 Including Code

3.6 Embedding PHP in Web Pages

3.6.1 XML Style

3.6.2 SGML Style

3.6.3 ASP Style

3.6.4 Script Style

3.6.5 Echoing Content Directly

3.7 Summary

3.8 Keywords

3.9 Review Questions

3.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the Conditional Statements used in PHP
- Explain Advance Conditionals used in PHP

Notes

- Discuss the Iteration
- Explain the Termination Statement
- Understand the Including Code
- Explain Embedding PHP in Web Pages

Introduction

One of the main reasons for using scripting languages such as PHP is to build logic and intelligence into the creation and deployment of web based data. In order to be able to build logic into PHP based scripts, it is necessary for the script to be able to make decisions and repeat tasks based on specified criteria. For example, it may be necessary to repeat a section of a script a specified number of times, or perform a task only if one or more conditions are found to be true (i.e. only let the user log in if a valid password has been provided). In programming terms this is known as flow control and looping. In the simplest terms this involves some standard scripting structures provided by languages such as PHP to control the logic and overall behavior of a script. Each of these structures provides a simple and intuitive way to build intelligence into scripts.

3.1 Conditional Statements

Sometimes when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this. Conditional statements are the set of commands used to perform different actions based on different conditions.

3.1.1 If Statement

The If Statement is a way to make decisions based upon the result of a condition.



Example: You might have a script that checks if boolean value is true or false, if variable contains number or string value, if an object is empty or populated, etc. The condition can be anything you choose, and you can combine conditions together to make for actions that are more complicated.

Use the if statement to execute a statement if a logical condition is true. Use the optional else clause to execute a statement if the condition is false. The syntax for If statement looks as follows:

```
if (condition) {
    statements_1
} else {
    statements_2
}
```

Condition can be any expression that evaluates to true or false. If condition evaluates to true, statements_1 are executed; otherwise, statements_2 are executed. statement_1 and statement_2 can be any statement, including further nested if statements.

You may also compound the statements using elseif to have multiple conditions tested in sequence. You should use this construction if you want to select one of many sets of lines to execute.

```
if (condition_1) {
    statement_1
}
```



```
[elseif (condition_2) {
    statement_2
}]
...
[elseif (condition_n_1) {
    statement_n_1
}]
[else {
    statement_n
}]
```

Let's have a look at the examples. The first example decides whether a student has passed an exam with a pass mark of 57:



Example:

```
<?php
$result = 70;

if ($result >= 57) {
    echo "Pass <br />";
}
else {
    echo "Fail <br />";
}
?>
```



Did u know? In the programming concept the Switch statement is sometimes called the case statement.

Next example use the elseif variant on the if statement. This allows us to test for other conditions if the first one wasn't true. The program will test each condition in sequence until:

- It finds one that is true. In this case it executes the code for that condition.
- It reaches an else statement. In which case it executes the code in the else statement.
- It reaches the end of the if ... elseif ... else structure. In this case it moves to the next statement after the conditional structure.



Example:

```
<?php
$result = 70;

if ($result >= 75) {
    echo "Passed: Grade A <br />";
}
elseif ($result >= 60) {
    echo "Passed: Grade B <br />";
}
```

Notes

```
}  
elseif ($result >= 45) {  
    echo "Passed: Grade C <br />";  
}  
else {  
    echo "Failed <br />";  
}  
?>
```


3.1.2 Switch Statement

Switch statements work the same as if statements. However the difference is that they can check for multiple values. Of course you do the same with multiple if..else statements, but this is not always the best approach.

A switch statement allows a program to evaluate an expression and attempt to match the expression's value to a case label. If a match is found, the program executes the associated statement. The syntax for the switch statement as follows:


```
switch (expression) {  
    case label_1:  
        statements_1  
        [break;]  
    case label_2:  
        statements_2  
        [break;]  
    ...  
    default:  
        statements_n  
        [break;]  
}
```

The program first looks for a case clause with a label matching the value of expression and then transfers control to that clause, executing the associated statements. If no matching label is found, the program looks for the optional default clause, and if found, transfers control to that clause, executing the associated statements.



Notes If no default clause is found, the program continues execution at the statement following the end of switch. Use break to prevent the code from running into the next case automatically.

Let's consider an example:

 *Example:*

```
<?php  
$flower = "rose";  
  
switch ($flower)
```

```

{
  case "rose" :
    echo $flower." costs $2.50";
    break;
  case "daisy" :
    echo $flower." costs $1.25";
    break;
  case "orchid" :
    echo $flower." costs $1.50";
    break;
  default :
    echo "There is no such flower in our shop";
    break;
}
?>

```

However, in addition to simple equality you can also test the expression for other conditions, such as greater than and less than relationships. The expression you are testing against must be repeated in the case statement. Have a look at the example:



Example:

```

<?php
$myNumber = 5;

switch ($myNumber) {
  case 0:
    echo "Zero is not a valid value.";
    break;
  case $myNumber < 0:
    echo "Negative numbers are not allowed.";
    break;
  default:
    echo "Great! Ready to make calculations.";
    break;
}
?>

```

If an expression successfully evaluates to the values specified in more than one case statement, only the first one encountered will be executed. Once a match is made, PHP stops looking for more matches.



Caution If you forget the break statement, then the code will fall through, which is to say that the code will start executing at the label that matches the value of the expression being evaluated, and then will proceed to process all codes until either the end of the switch statement or until it find a break statement.

3.1.3 Other Formats

PHP likes to cover its bases, so it provides alternate methods for coding if statements and switch statements.

Alternate If

There are two alternate ways to code an if statement, one is a ternary operator, and another one is an alternate syntax. The ternary operator takes the form of:

```
(condition) ? value_if_true : value_if_false;  
$hours = ($hours < 13)? $hours: $hours -= 12;
```

Since the ternary operator is, in fact, an operator, it returns a value, which is the value of one of the expressions. If the condition evaluates to true, then the first value is returned (value_if_true), otherwise the second value is returned (value_if_false). This means that the expressions for the true and false values need to be things that evaluate to values, not full blown statements, just simple expressions.



Notes The alternative syntax makes use of labels for all components of the if statement.

The curly brackets are omitted. Since there are no curly brackets, there needs to be some way to specify the end of the statement block. For this we use an endif label.

The code looks like this:

```
if (condition):  
statements;  
elseif (condition):  
statements;  
else:  
statements;  
endif;  
if ($x < 0):  
echo "Cannot use a negative number."  
elseif ($x > 999999):  
echo "Cannot use that big a number."  
else:  
doSomethingWith($x);  
endif;
```

Alternate Switch

The alternate switch statement has the same syntax as the alternate if statement. The curly brackets are omitted and a closing end switch is used to denote the end of the statement block.

```
switch ($someNumber):  
case 0:  
echo "Zero is not a valid value."  
break;
```

```

case $someNumber < 0:
echo "we cannot use negative numbers.";
break;
default:
echo "Ready to compute.";
break;
endswitch;

```

Self Assessment

State whether the following statements are true or false:

1. There are four alternate ways to code an if statement.
2. Switch statements work the same as if statements.
3. The If Statement is a way to make decisions based upon the result of a condition.

3.2 Advance Conditional

One useful thing to remember about conditional statements is that they can be nested within each other. Below is an example of how the discount program from our example could be written to use nested IF:ELSE statements. There are other ways of doing this - such as using elseif () or switch () but this demonstrates how statements can be nested.



Example:

```

<?php
$age = 30;
$price = 3.00;
if ($age >65)
{
$discount =.90;
print "You have received our senior's discount, your price is $" .
$price*$discount;
}
else
{
if ($age <18)
{
$discount =.95;
print "You have received our student's discount, your price is $" .
$price*$discount;
}
else
{
print "Sorry you do not qualify for a discount, your price is $" . $price;
}
}
?>

```

Notes

This program will first check if they are eligible for the senior's discount. If they are not, it will then check if they are eligible for a student discount, before returning the non-discounted price.

3.2.1 Comparing Strings

Time-to-time you want to compare various strings and control your script execution regarding the result. You can do this in more ways. The most simple way to use the compare operator (==) as follows:



Example:

```
<?php
    $str1 = "Test";
    $str2 = "Test";

    if ($str1 == "Test")    echo "OK-1";
    if ($str1 == $str2)    echo "OK-2";
?>
```

However you can use PHP built in functions as well like strcmp and strcasecmp. These functions make a binary safe string comparison where the strcasecmp is a case-insensitive version. With this functions you can also decide which string is greater or smaller as it returns < 0 if *STR1* is less than *STR2* > 0 if *STR1* is greater than *STR2*, and 0 if they are equal. So you can use these functions like this:



Example:

```
<?php
    $str1 = "Test";
    $str2 = "Test";
    $str3 = "Apple";
    $str4 = "Zebra";

    if (strcmp($str1,$str2) == 0) echo "OK";
    if (strcmp($str1,"Test") == 0) echo "OK";
    if (strcmp($str1,$str3) > 0) echo "$str1 > $str3";
    if (strcmp($str1,$str4) < 0) echo "$str1 < $str4";
?>
```

Sometimes it can happen that the strings seems to be equal but the comparison reports that they are different. The most common problem in this case that there are some spaces before or after the relevant text. This results that the strings are different. To solve this problem it make sense to use the PHP built in trim function to remove all unwanted spaces like this:



Example:

```
<?php
    $str1 = "Test";
    $str2 = " Test ";

    if ($str1 == $str2) echo "OK-1";
```

```

if ($str1 == trim($str2)) echo "OK-2";
if (strcmp($str1,$str2) == 0) echo "OK-3";
if (strcmp($str1,trim($str2)) == 0) echo "OK-4";
?>

```

The built in trim function definitely works best when supported by PHP hosting.

3.2.2 Similarity

PHP also provides numerous ways of comparing strings to see whether they are similar. We can compare for textual similarity and for phonic similarity.

Phonic Similarity

Phonic similarity first, since it is a little easier. Phonic similarity just means that things are compared based on whether they sound alike. To compare things by how they sound you can use one of two functions. The two functions are:

- soundex()
- metaphone()

Both return strings that represent the pronunciation of a word, you can use the result values in comparisons for equality.

```

if (metaphone($a) == metaphone($b) {
}

```

The only real purpose of these functions is to look for homophones, or words that sound alike. This is something that is primarily of use if you are planning on writing spell-checkers and other advanced text processing tools. The result strings themselves are not pronounceable, but are strings to represent the sound of the word in a standardized way. Of the two, metaphone() tends to be more accurate because its algorithms for the rules of pronunciation are better.

Textual Similarity

Textual similarity is a little more useful because it allows you to compare how similar two text strings are:

```
similar_text()
```

The similar_text() function takes two strings and returns an index value that returns the number of characters that are the same, allowing for additions, subtraction, and repetition. It also takes an optional third parameter which returns a value that represents the percentage of similarity.

```

similar_text($a, $b[, $percent]);
$match = similar_text($a, $b, $mper);

```

levenshtein()

The levenshtein() function returns a weighted score representing the difference between two strings. It takes two strings and three optional arguments. The three optional arguments allow you to specify how much weight to assign to different types of difference. This allows you to specify how you want scores weighted with numeric values. Otherwise all scores are given equal weight. Other different types are:

- insertions between string one and two
- replacements between string one and two

Notes

- deletions between strings one and two
`levenshtein($a, $b[, $ins, $rep, $del])`
`$diff = levenshtein($a, $b, 100, 5, 1)`



Did u know? The result strings themselves are not pronounceable, but are strings to represent the sound of the word in a standardized way. Of the two, `metaphone()` tends to be more accurate because its algorithms for the rules of pronunciation are better.

Self Assessment

Fill in the blanks:

4. Thefunction returns a weighted score representing the difference between two strings.
5. PHP also provides numerous ways of comparingto see whether they are similar.
6.means that things are compared based on whether they sound alike.

3.3 Iteration

It is generally accepted that computers are great at performing repetitive tasks an infinite number of times, and doing so very quickly. It is also common knowledge that computers really don't do anything unless someone programs them to tell them what to do.

Loop statements are the primary mechanism for telling a computer to perform the same task over and over again until a set of criteria are met. This is where *for*, *while* and *do ... while* loops are of use.

3.3.1 While Statement

There will frequently be instances where code needs to be repeated until a certain condition is met, with no way of knowing in advance how many repetitions are going to be needed to meet that criteria. To address this PHP, provides the *while* loop.

Essentially, the *while* loop repeats a set of tasks until a specified condition is met. The *while* loop syntax is defined follows:

```
<?php
while (condition)
{
    // PHP statements go here
}
?>
```

In the above syntax, *condition* is an expression that will return either *true* or *false* and the *// PHP statements go here* comment represents the PHP to be executed while the expression is *true*. For example:



Example:

```
<?php
$myCount = 0;
```



```

$j = 10;

while ( $myCount < 100 )
{
    $myCount = $myCount + $j;
}

?>

```

In the above example the *while* expression will evaluate whether \$myCount is less than 100. If it is already greater than 100 the code in the braces is skipped and the loop exits without performing any tasks. If \$myCount is not greater than 100 the code in the braces is executed and the loop returns to the while statement and repeats the evaluation of \$myCount. This process repeats until \$myCount is greater than 100, at which point the loop exits.

3.3.2 Do-while Statement

You can think of the *do-while* loop as an inverted *while* loop. The *while* loop evaluates an expression before executing the code contained in the body of the loop. If the expression evaluates to *false* on the first check then the code is not executed. The *do-while* loop, on the other hand, is provided for situations where you know that the code contained in the body of the loop will *always* need to be executed at least once. For example, you may want to keep stepping through the items in an array until a specific item is found. You know that you have to at least check the first item in the array to have any hope of finding the entry you need. The syntax for the *do-while* loop is as follows:



Example:

```

<?php
do
{
    PHP statements
} while (conditional expression)

?>

```

In the *do-while* example below the loop will continue until \$i equals 0:

```

$i = 10;
do
{
    $i--;
} while ($i > 0)

```

3.3.3 For Statement

Suppose you wanted to add a number to itself ten times. One way to do this might be to write the following PHP script:

```

<?php
$myVar = 10;

$myVar += $myVar;

```

Notes

```
$myVar += $myVar;  
$myVar += $myVar;  
$myVar += $myVar;  
$myVar += $myVar;  
$myVar += $myVar;  
$myVar += $myVar;  
$myVar += $myVar;  
$myVar += $myVar;  
$myVar += $myVar;
```

?>

Whilst this is somewhat ungainly and time consuming to type, it does work. What would happen, however, if there was a requirement to perform this task 100 times or even 10,000 times. Writing a script to perform this as above would be prohibitively time consuming. This is exactly the situation the *for* loop is intended to handle.

The syntax of a PHP *for* loop is as follows:

```
for ( initializer; conditional expression; loop expression )  
{  
    // PHP statements to be executed go here  
}
```

The *initializer* typically initializes a counter variable. Traditionally the variable *\$i* is used for this purpose.



Example:

```
$i = 0
```

This sets the counter to be the value *\$i* and sets it to zero.

The *conditional expression* specifies the test to perform to verify whether the loop has been performed the required number of times. For example, if we want to loop 1000 times:

```
$i < 1000
```

Finally, the *loop expression* specifies the action to perform on the counter variable. For example to increment by 1:

```
$i++
```

Bringing this all together we can create a *for* loop to perform the task outlined in the earlier in this section:

```
<?php  
$j = 10;  
  
for ($i=0; $i<10; $i++)  
{  
    $j += $j;  
}  
?>
```

As with the *if* statement, the enclosing braces are optional if a single line of script is to be executed, but their use is strongly recommended.



Task Develop a PHP program to differentiate between while and do-while statement.

Self Assessment

State whether the following statements are true or false:

7. The while loop repeats a set of tasks until a specified condition is met.
8. If the expression evaluates to false on the first check then the code is executed.
9. The initializer initializes a counter variable.

3.4 Termination Statements

Sometimes you want to break out of a control statement early, perhaps because of an error or some other condition that the conditional expression itself is not testing for. We have seen an example of this already with the break statement used in a switch statement coding block. Here we will look at various ways to prematurely terminate loops and conditionals by setting multiple exit points.

3.4.1 Break Statement

The PHP break statement is placed within the code of a loop statement to cause your program to break out of the loop statement. Each of these loop statements already had a test expression that controlled when to stop the loop. So why would you want to add another “stop” code? There are numerous reasons for doing this.



Example: You would use the break statement to prevent your program from performing unwanted actions such as dividing by zero.

Below is an example of a break statement that has been added to a loop statement. You will recognize this loop statement from the previous tutorial about the PHP for statement. If it has been some time since you have read that tutorial, go back and refresh your memory.

```
for (initialize variable exp; test expression; modify variable exp)
{
    if (second test expression) { break; }
    else { do this }
}

for ($number = 1; $number < 11; $number++)
{
    if ($number == 5) { break; }
    else { echo "$number <br>"; }
}

for ($number = 1; $number < 11; $number++)
for (initialize variable expression; test expression; modify variable expression)
```

Notes

This is the same as in the previous tutorial. The for statement begins with the word **for** and has the three expressions between the parenthesis.

```
{
if ($number == 5) { break; }
if (second test expression ) { break; }
```

The first NEW thing you see here is an if statement. This is our second test expression. It controls the break statement that follows it. In our example we want to stop the loop if the value of the \$number variable is 5. When this condition is true the following is executed -> { break; }. The program will not continue through the rest of the statement -> else { do this }. It will "break out" of the loop.

When the if statement test is false, the following is ignored -> { break; }. And the else clause will be executed -> else { do this }.

```
else { echo "$number <br>"; }
else { do this }
```

This is the else clause that will be executed when the if statement is false. As in the previous tutorial, the { do this } instructions are to print the current value of the \$number variable and the HTML
 code to the web browser.

```
}
```

This right curly brace is placed at the end of the statement.

3.4.2 Continue Statement

Sometimes a situation arises where we want to take the control to the beginning of the loop (for example for, while, do-while etc.) skipping the rest statements inside the loop which have not yet been executed.

The keyword continue allow us to do this. When the keyword continue executed inside a loop the control automatically passes to the beginning of loop. Continue is usually associated with the if.

In the following example the list of odd numbers between 1 to 10 have printed. In the while loop we test the remainder (here \$x%2) of every number, if remainder is 0 then it becomes a even number and to avoid printing of even numbers continue statement is immediately used and the control passes to the beginning of the loop.



Example:

```
<?php
$x=1;
echo 'List of odd numbers between 1 to 10 <br />';
while ($x<=10)
{
if (($x % 2)==0)
{
$x++;
continue;
}
else
```

```
{
echo $x.'  


```



Caution When using break to get out of a condition, be aware that it ignores if statements. This is so you can use an if statement to test whether you need to terminate the current conditional.

3.4.3 Stepping Out Multiple Levels

PHP also gives you the option to step out of multiple levels when working with nested conditional iterative statements. Once again, the if statement is not included since it is used to test whether to make such a step. Both the break and the continue statements can be followed by a number specifying how many levels of nesting to step out.

Self Assessment

Fill in the blanks:

10. The PHPstatement is placed within the code of a loop statement to cause your program to break out of the loop statement.
11. The for statement begins with the word for and has the three expressions between the.....
12.also gives you the option to step out of multiple levels when working with nested conditional iterative statements.

3.5 Including Code

In PHP, you can insert the content of one PHP file into another PHP file before the server executes it.

The include and require statements are used to insert useful codes written in other files, in the flow of execution.

Include and require are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use include. Otherwise, in case of FrameWork, CMS or a complex PHP application coding, always use require to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

Notes

```
Syntax  
include 'filename';  
  
or  
  
require 'filename';
```



Example:

```
<html>  
<body>  
  
<?php include 'header.php'; ?>  
<h1>Welcome to my home page!</h1>  
<p>Some text.</p>  
  
</body>  
</html>
```

Self Assessment

Fill in the blanks:

- 13. The include and require statements are used touseful codes written in other files, in the flow of execution.
- 14. When the header needs to be updated, you can only update thefile.
- 15. CMS or a complex PHP application coding, always use require to include a key file to the flow of.....

3.6 Embedding PHP in Web Pages

Although it is possible to write and run stand alone PHP programs, most PHP code is embedded in HTML or XML files. This is, after all, why it was created in the first place. Processing such documents involves replacing each chunk of PHP source code with the output it produces when executed. Because a single file contains PHP and non-PHP source code, we need a way to identify the regions of PHP code to be executed. PHP provides four different ways to do this. As you will see, the first, and preferred, method looks like XML. The second method looks like SGML. The third method is based on ASP tags. The fourth method uses the standard HTML <script> tag; this makes it easy to edit pages with enabled PHP using a regular HTML editor.

3.6.1 XML Style

The way in which you normally embed PHP in a document is through what is commonly called a *PHP statement*, but is actually an *XML directive*. PHP is designed to be compatible with many standards and commonly used scripting approaches, making it easier to code. Here we are going to code to standards. After we look at the standards-defined method, we will look at the other methods.

An XML processing directive is a command embedded in an XML document for some application to process. All XML processing directives take the form of:

```
<?appname information for application ?>
```

Notice that the statement opens and closes with nested brackets and question marks, and that the name of the application being referenced must come right after the first question mark, with no space before it. The information for the application can be a short statement, or, in the case of PHP, can go on for pages and pages.

The PHP statement normally uses this form of inclusion in HTML documents. It looks like this:

```
<?php some php statements ?>
```

Line breaks are allowed within the XML processing directive, so the following is also legal.

```
<?php
    some php statements
    and some more statements
?>
```

If you want to write standards compliant code, or embed PHP in XML documents, you need to use this format. If you are not working with XML or don't need to adhere to standards, there are other approaches. Each may be appropriate within certain server situations.

Since PHP is processed by the server, it can literally go anywhere in an HTML document in question. The code is processed before the Web browser sees the code, so the Web browser never sees the PHP. If working with XML, you need to make sure that any server-side XML processing is done after the PHP processing, or code the document with the knowledge that the PHP code cannot violate the syntax rules of the application you are working with. This means that XML documents containing PHP should still be well-formed and valid.



Did u know? Notice that there is no trace of the PHP source code from the original file. The user sees only its output.

3.6.2 SGML Style

The “classic” style of embedding PHP comes from SGML instruction processing tags. To use this method, simply enclose the PHP in `<? and ?>`. Here's the “Hello world” example again:

```
<? echo "Hello, world"; ?>
```

This style, known as *short tags*, is the shortest and least intrusive, and it can be turned off so as to not clash with the XML PI (Process Instruction) tag in the *php.ini* initialization file. Consequently, if you want to write fully portable PHP code that you are going to distribute to other people (who might have short tags turned off), you should use the longer `<?php ... ?>` style, which cannot be turned off. If you have no intention of distributing your code, you don't have an issue with telling people who want to use your code to turn on short tags, and you are not planning on mixing XML in with your PHP code, then using this tag style is okay.

3.6.3 ASP Style

Realizing that some old code editors might have problems with PHP, the authors of PHP included two additional ways to include PHP code in a document. One mimicks ASP tags. The other uses standard HTML `<script>` tags.

ASP tags are not enabled by default. To turn them on you need to find the `asp_tags` flag in the `php.ini` file, or, if working from the source, build PHP with the `—enable-asp-tags` option.

It is not necessary for the server to actually process ASP tags as PHP tags, since the point of the compatibility with ASP tags is to allow users to work with editors that understand ASP tags but

Notes

not PHP tags. When you are done with the document, you can just do a global search and replace to replace all ASP tags with PHP tags before moving the documents to production.

ASP tags take the following form:

```
<% code goes here %>
```

Like PHP tags, ASP tags can go anywhere in the document.

For editors that don't understand anything that isn't HTML, you can also use `<script>` tags. This is very useful if you want to ensure that you are writing strictly formatted HTML code. It does, however, sharply delimit what you can do with PHP, since that tags can only exist in places where `<script>` tags can legally exist inside an HTML document. Thus it cannot be used to set attribute values or set the contents of tags that cannot have scripts nested inside them. Of course, you can resolve this by just using PHP to write out larger blocks of the document, including tags, but this is not necessarily an optimal solution.

```
<script language="php">
    code goes here
</script>
```

These two approaches are disabled by default in PHP. In order to use them on the server, you have to turn them on.

3.6.4 Script Style

The final method of distinguishing PHP from HTML involves a tag invented to allow client side scripting within HTML pages, the `<script>` tag. You might recognize it as the tag in which JavaScript is embedded. Since PHP is processed and removed from the file before it reaches the browser, you can use the `<script>` tag to surround PHP code. To use this method, simply specify "php" as the value of the language attribute of the tag:

```
<script language="php"> echo "Hello, world"; </script>
```

This method is most useful with HTML editors that work only on strictly legal HTML files and do not yet support XML processing commands.

3.6.5 Echoing Content Directly

Perhaps the most common use of PHP is to generate content, and the most common means of doing so is the echo command, which writes information back to the document to be sent to the client. If you are only using a given PHP statement to echo content, you can use an abbreviated tag to do this. This works for both short tags and ASP tags.

It works by appending an equal sign immediately after the opening tag.

```
<?="Rasputin" ?>
// is the same as
<?php echo "Rasputin" ?>
```

This approach does tend to reduce readability and cannot be used with the XML compliant version of the opening tag, so it is not necessarily recommended.

Self Assessment

Fill in the blanks:

16.breaks are allowed within the XML processing directive.
17. Thestyle of embedding PHP comes from SGML instruction processing tags.

18. The final method of distinguishing PHP from HTML involves a tag invented to allow client side scripting withinpages.

Notes



Case Study

Tyco Flow Control

Tycos Flow Control is an industrial manufacturer with a global presence. As the company grew through acquisitions, it also acquired a wide variety of e-mail systems across all its offices. To simplify IT management and increase communication reliability, the company investigated hosted e-mail solutions. After evaluating other vendors, Tyco Flow Control chose Microsoft Exchange Online for its features and reliability.

Business Needs

Tyco Flow Control, a major division of Tyco International, brings innovation and creativity to its line of market-leading flow control products and heat-tracing solutions in the oil and gas, power, chemical, building, and other industries. The company has 15,000 employees and annual revenues exceeding US \$4.5 billion. Tyco Flow Control has approximately 12,000 desktops in over 250 locations around the world. The company has grown through numerous acquisitions over the past ten years and, as a result, the IT department found itself managing a heterogeneous assortment of servers, networks, and e-mail systems. Offices in the Americas were consolidating on Microsoft Exchange Server for communications, but offices in Europe and Asia still had hundreds of servers running IBM Lotus Notes. Not surprisingly, having a collection of disparate systems caused problems. "We would have one or more systems down at any given time, which caused stability issues," explains Tony DeGregorio, Global Chief Information Officer for Tyco Flow Control. "We also had trouble communicating between systems and pulling together global address books and calendars." To improve stability and reliability and simplify IT management, Tyco Flow Control wanted to consolidate its communications infrastructure. The company considered on-premises solutions, but to keep its IT group focused on business value initiatives, and to save money, it chose to use an outside vendor with the skill and experience to deploy and manage such a large installation. Because of its past success with Microsoft Exchange Server, Tyco looked at Microsoft-hosted online solutions, but wanted to consider other possibilities as well.

Solution

Tyco Flow Control prepared detailed requirements for the project and spoke with representatives from Microsoft and other online vendors. Tyco quickly realized that the Microsoft solution was the only one that would meet its needs. "We did not believe other vendors were enterprise ready," says Tyco saw Microsoft as a company with deep expertise. "With Microsoft, I saw an experienced enterprise group with the clout and power to make the solution work the way I wanted it to," says DeGregorio. "I had a lot more confidence in Microsoft. The Microsoft contract promised greater uptime and availability, and the company has data centers around the world with robust failover and backup plans." Microsoft also offered greater features and flexibility. "With Microsoft, we could choose an on-premises or a hosted solution," DeGregorio notes. "We chose the hosted solution, and Microsoft gave us the option of single-tenant or multi-tenant hosting. Single-tenant was important to us for security, and that was not an option with the other vendors. Microsoft also provided off-line capabilities, stronger security, better archiving, and easy connectivity with the Microsoft Office suite, which we use internally. That way, we can

Contd...

Notes

gradually move existing Notes applications over to Microsoft Office SharePoint.” Tyco Flow Control also had confidence in the Microsoft Partner Network. “When we looked at migrating our systems, we found that other vendors were working with third-party companies that I do not think had a lot of experience,” says DeGregorio. “With Microsoft, you have got a lot of very experienced partners around the globe. We have migrated 8,000 users so far, and will migrate the rest in the coming months.”

Benefits

With its move to dedicated Microsoft Exchange Online, Tyco Flow Control has ensured that its e-mail solution is scalable, easy to manage, cost-effective, and reliable.

Robust Scalability

With its Microsoft-hosted solution, Tyco Flow Control has made it easier to adjust its technology as its business needs change. “One area we wanted to address was scalability,” explains DeGregorio. “As we acquire or divest pieces of our business, I did not want to have to buy new hardware or end up with unused resources. We do not have to worry now; there’s a good variable cost model that is very scalable for the business.”

Simplified IT Management

By choosing a hosted e-mail solution, Tyco Flow Control has been able to reduce IT management overhead and repurpose staff. “It’s been great,” says DeGregorio. “We have moved people around, and we have been able to concentrate on other areas of the business. We had two individuals who two years ago would be consumed with e-mail issues. They now make up our Network Security team.”

Cost Savings

With hosted e-mail, Tyco has reduced hardware and maintenance costs, as well as third-party licensing costs for features like e-mail filtering which are included in Microsoft Exchange Online. “By going off-premise, we saved the purchase of additional hardware,” DeGregorio says. “And it is not just that, there’s also the ongoing upkeep, maintenance, and upgrades. That’s all out of our hands now, and I do not have to worry about it. We also save money by consolidating and standardizing all the different systems we used to manage.”

Questions

1. Give a brief discussion of Tyco flow control.
2. Explain the use of Microsoft in Tyco flow control.

3.7 Summary

- Flow control means exactly what it sounds like it should, controlling the flow of something. When using flow control for programming, what you are doing is regulating the order in which the code is executed, how many times it is executed, and if it is executed at all.
- Conditionals allow us to specify whether or not to run a selected piece of code based on some prior condition.
- The switch statement, in its structure, is similar to the if statement. It takes an expression Notes to be evaluated and a statement block.
- A do-while statement is like a while statement in reverse. First it runs the block of code, and then it evaluates the conditional expression. If the conditional expression is true, then it loops back to the beginning of the statement and starts again.

- PHP gives the option to step out of multiple levels when working with nested conditional iterative statements. Once again, the if statement is not included in the since it is used to test whether to make such a step.
- Code in an included file is imported at the scope that is in effect where the include statement is found, so the included code can see and alter your code's variables.

3.8 Keywords

Default Condition: The default keyword is a catch-all case that marks the point to begin execution of none of the conditions being tested for is met. It is like the last else statement in a long string of else if's.

Else Statement: The else statement can only follow an if statement and is used to mark off a statement block to execute if the conditional expression being tested evaluates to false.

Elseif Statements: This allows us to test for another condition if the first one was not true.

Flow Control: It means exactly what it sounds like it should, controlling the flow of something.

If Statement: If statement is a simple concept. If something is true, then perform the statement block associated with it, otherwise do not.

Phonic Similarity: Phonic similarity just means that things are compared based on whether they sound alike. To compare things by how they sound you can use one of two functions.

Switch Statement: It is similar to the if statement. It takes an expression notes to be evaluated and a statement block.

Textual Similarity: Textual similarity is a little more useful because it allows you to compare how similar two text strings are.

3.9 Review Questions

1. What are the conditional statements? Explain with example.
2. Define the switch and case statements.
3. How do we compare two strings? Explain the functions.
4. What is the difference between `strcasecmp(str1, str2)`; `strnatcmp(str1, str2)`; and `strnatcasecmp(str1, str2)`?
5. What is the similarity? Explain the phonic and text similarity.
6. What are the iterative statements? Differentiate between them.
7. What are the termination statements? When are these used?
8. Differentiate between break and continue statements.
9. How is the PHP code embedded in HTML or XML files?
10. Develop a program to embedding the PHP code in HTML.

Answers: Self Assessment

1. False
2. True
3. True
4. Levenshtein()

Notes

- | | |
|-----------------|----------------------|
| 5. Strings | 6. Phonic similarity |
| 7. True | 8. False |
| 9. True | 10. Break |
| 11. Parenthesis | 12. PHP |
| 13. Insert | 14. Header Include |
| 15. Execution | 16. XML |
| 17. Line | 18. Classic |

3.10 Further Readings



Books

- Bangia, Ramesh (2008). *Web Technology (including HTML, CSS, XML, ASP, JAVA)*. Firewall Media.
- Jackson (2007). *Web Technologies: A Computer Science Perspective*. Pearson Education India.
- Kamal, Raj (2002). *Internet and Web Technologies*. Tata McGraw-Hill Education.
- Puntambekar, A.A. (2009). *Web Technologies*. Technical Publications.
- Sarukkai, Ramiesh R. (2002). *Foundations of Web Technology*. Springer.
- Xavier, C. (2007). *Web Technology and Design*. New Age International.



Online links

- http://www.techotopia.com/index.php/PHP_Flow_Control_and_Looping
- http://webcheatsheet.com/PHP/if_else_switch.php
- http://php.about.com/od/learnphp/ss/phpbasics_9.htm
- <http://www.phpf1.com/tutorial/php-string-compare.html>
- <http://www.bellaonline.com/articles/art21947.asp>
- <http://www.w3resource.com/php/statement/continue.php>
- http://www.w3schools.com/php/php_includes.asp
- <http://www.daaq.net/old/php/index.php?page=embedding+php&parent=php+basics>

Unit 4: Functions

Notes

CONTENTS

Objectives

Introduction

4.1 Defining a Function

4.2 Calling a Function

4.3 Variable Scope

4.3.1 Global Variables

4.3.2 Static Variables

4.3.3 Function Parameters

4.4 Return Values

4.5 Variable Functions

4.6 Category of PHP Functions

4.7 Summary

4.8 Keywords

4.9 Review Questions

4.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand How to Define Functions
- Understand How to Call a Function
- Define the Variable Scope
- Explain the Function Parameters
- Understand the Variable Functions
- Discuss the Categories of PHP Functions

Introduction

Function are block of code that can be used anywhere in program. All function start with the word "function()". The function name can start with a letter or underscore (not a number). Function helps set or update variables and can be rested. They enable to execute lines of code without having to retype them every time you want to use them. In function can add parameter. The parameter is like a variable. The parameters are specified inside the parentheses.

4.1 Defining a Function

A function is just a segment of code, separate from the rest of your code. You separate it because it's nice and handy, and you want to use it not once but over and over. It's a chunk of code that

Notes

you think is useful, and want to use again. Functions save you from writing the code over and over. Here's an example.

Suppose you need to check text from a textbox. You want to trim any blank spaces from the left and right of the text that the user entered. So if they entered this:

```
" Acme "
```

You want to turn it into this:

```
"Acme"
```

But you also want to check if the user entered any text at all. You don't want the textbox to be completely blank.

You can use the PHP inbuilt function called `trim()`. Like this:

```
$user_text = trim( $_POST['text1'] );
```

That will get rid of the white space in the text box. But it won't check if the text box is blank. You can add an if statement for that:



Example:

```
if ($user_text == "") {  
    error_message = "Blank textbox detected";  
}
```

But what if you have lots of textboxes on your form? You'd have to have lots of if statements, and check each single variable for a blank string. That's a lot of code to write!

Rather than do that, you can create a single function, with one if statement that can be used for each blank string you need to check. Using a function means there's less code for you to write. And it's more efficient. We'll see how to write a function for the above scenario in a moment. But first, here's the basic syntax for a function.

```
function function_name() {  
}
```

So you start by typing the word **function**. You then need to come up with a name for your function. You can call almost anything you like. It's just like a variable name. Next, you type two round brackets `()`. Finally, you need the two curly brackets as well `{ }`. Whatever your function does goes between the curly brackets. Here's a simple example that just print something out:

```
function display_error_message() {  
    print "Error Detected";  
}
```

In the example above, we've started with `function`. We've then called this particular function **display_error_message**. In between the curly brackets, there a print statement. This script:



Example:

```
<?PHP  
function display_error_message( ) {  
    print "Error Detetceted";  
}  
?>
```

To create a function, the syntax is: `function name() { }`

You can name a function anything you want, as long as the function name is not already in use as a PHP function. Function names are not case sensitive, and can include letters, numbers and/or underscores, but can only begin with letters or underscores. Let's take a look at a simple function:



Example:

```
<?php
function test() {
    echo "I am in a function!";
}
?>
```

Self Assessment

State whether the following statements are true or false:

1. A function is just a segment of code, separate from the rest of your code.
2. Functions don't save you from writing the code over and over.
3. Function names are case sensitive, and can not include letters, numbers and/or underscores.

4.2 Calling a Function

So what does the above function do? Nothing! Why? Because it has not yet been called. Until a function is called, it will sit around looking bored (or boring), waiting to be of use... Let's learn how to call our poor function and give it something to do:



Example:

```
<?php
function test() {
    echo "I am in a function!";
}

test();
?>
```

Simple! Too simple? Probably. All that the function does now that it is called is echo a single statement. Let's try something a bit more complex.


```
<?php
function mathy_stuff($a, $b) {
    echo $a * $b;
}

mathy_stuff(5, 7);
mathy_stuff(3, 100);
mathy_stuff(7000, 68);
?>
```

Now, when our new function is defined, we have two parameters, or variables, between the function's parentheses. These variables are used in the function, where they are multiplied by


Notes

one another and echoed. The variables are not actually assigned their values until the function is called, at which point the values are listed in the function’s parentheses, comma-separated in the order of the variables they need to be assigned to.



Notes A function can handle as many (or few) comma-separated parameters as you (the programmer) can keep track of, if necessary.

Returning values allows you to store the results of a user-defined function in a variable. Consider the following:




Example:

```
<?php
function example($a, $b) {
    $total = $a + $b;
    return $total;
}

$addition = example(50, 51);
echo $addition . " dalmations!";
?>
```

As you can see, instead of echoing the result of \$a + \$b, we store it in a variable and then return that variable. (A function can only return one thing, so choose carefully!) Then, the function is called as the value of the \$addition variable, and is assigned two values to use in the function. At last, the \$addition variable (containing the results of the function) is echoed along with a short string of text.

Some people consider functions hard to get a handle on, but it is not really much harder than learning how to ride a bike, and it is a lot easier on the knees!



Did u know? Function name can start with a letter or underscore “_”, but not a number!

Self Assessment

Fill in the blanks:

4. All that the function does that it is called is echo astatement.
5. A function can handle as manyseparated parameters as the programmer can keep track.
6. The \$addition variable containing the results of the function is echoed along with a shortof text.

4.3 Variable Scope

In PHP there is something called scope, which relates to variables. When you declare variables, you declare them in a particular scope. The scope is the area in which the variable is declared. In PHP, you set your scope with curly braces (the { and } characters).

This concept is already discussed in unit 2. Let us now discuss the concept of global variables, static variables, and function parameter in detail.

4.3.1 Global Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function:



Example:

```
<?
$x = 4;
function assignx () {
    $x = 0;
    print "\$x inside function is $x.
";
}
assignx();
print "\$x outside of function is $x.
";
?>
```

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword **GLOBAL** in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example:



Example:

```
<?
$somevar = 15;
function addit() {
    GLOBAL $somevar;
    $somevar++;
    print "Somevar is $somevar";
}
addit();
?>
```

4.3.2 Static Variables

The final type of variable scoping is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

You can declare a variable to be static simply by placing the keyword **STATIC** in front of the variable name.

Notes



Example:

```
<?
function keep_track() {
    STATIC $count = 0;
    $count++;
    print $count;
    print "
";
}
keep_track();
keep_track();
keep_track();
```



Task Develop a PHP program to differentiate between global and static variable declaration.

Self Assessment

State whether the following statements are true or false:

- 7. In PHP there is something called scope which do not relate to variables.
- 8. A global variable can not be accessed in any part of the program.
- 9. A static variable will not lose its value when the function exits and will still hold that value should the function be called again.

4.3.3 Function Parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters you like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.



Example:

```
<html>
<head>
<title>Writing PHP Function with Parameters</title>
</head>
<body>

<?php
function addFunction($num1, $num2)
{
    $sum = $num1 + $num2;
```

```

    echo "Sum of the two numbers is : $sum";
}
addFunction(10, 20);
?>
</body>
</html>

```

By-Value Parameters

Call by value is the default way to handle parameters in a function. To handle them the copy operator is used so modifying the parameter won't change the original variable passed to the function.

```
function by_value($a, $b){}
```



Example:

following example takes two integer parameters and add them together and then print them.

```

<html>
<head>
<title>Writing PHP Function with Parameters</title>
</head>
<body>

<?php
function addFunction($num1, $num2)
{
    $sum = $num1 + $num2;
    echo "Sum of the two numbers is : $sum";
}
addFunction(10, 20);
?>
</body>
</html>

```

By-Reference Parameters

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value. Any changes made to an argument in these cases will change the value of the original variable.



Notes You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Notes

Following example depicts both the cases.



Example:

```
<html>
<head>
<title>Passing Argument by Reference</title>
</head>
<body>
<?php
function addFive($num)
{
    $num += 5;
}

function addSix(&$num)
{
    $num += 6;
}
$orignum = 10;
addFive( &$orignum );
echo "Original Value is $orignum<br />";
addSix( $orignum );
echo "Original Value is $orignum<br />";
?>
</body>
</html>
```

Default Parameters

Default parameters like C++ are supported by PHP. Default parameters enable you to specify a default value for function parameters that are not passed to the function during the function call. The following is an example for using default parameters.



Example:

```
function increment(&$num, $increment = 1)
{
    $num += $increment;
}
$num = 4;
increment($num);
increment($num, 3);
```

This code results in \$num being incremented to 8. First, it is incremented by 1 by the first call to increment, where the default increment size of 1 is used, and second, it is incremented by 3, altogether by 4. In the condition of default parameters the default values you specify must be a constant value, such as a scalar, array with scalar values, or constant.



Caution In the condition of default parameters the default values you specify must be a constant value, such as a scalar, array with scalar values, or constant.

Variable Parameters

A function may require a variable number of arguments. For example, the `get_preferences()` example might return the preferences for any number of names, rather than for just one. To declare a function with a variable number of arguments, leave out the parameter block entirely.

```
function get_preferences( ) { // some code }
```

PHP provides three functions you can use in the function to retrieve the parameters passed to it. `func_get_args()` returns an array of all parameters provided to the function, `func_num_args()` returns the number of parameters provided to the function, and `func_get_arg()` returns a specific argument from the parameters.

```
$array = func_get_args( ); $count = func_num_args( ); $value = func_get_arg(argument_number);
```

In example the `count_list()` function takes in any number of arguments. It loops over those arguments and returns the total of all the values. If no parameters are given, it returns false.



Example: Argument counter

```
function count_list( ) { if(func_num_args( ) == 0) { return false; } else { for($i = 0; $i < func_num_args( ); $i++) { $count += func_get_arg($i); } return $count; } } echo count_list(1, 5, 9);
```

The result of any of these functions cannot directly be used as a parameter to another function.

To use the result of one of these functions as a parameter, you must first set a variable to the result of the function, then use that in the function call. The following expression will not work:

```
foo(func_num_args( ));
```

Instead, use:

```
$count = func_num_args( ); foo($count);
```



Did u know? When you call a function with default arguments, after you omit a default function argument, you must emit any following arguments. This also means that following a default argument in the function's definition, all other arguments must also be declared as default arguments.

Missing Parameters

PHP lets you be as lazy as you want – when you call a function, you can pass any number of arguments to the function. Any parameters the function expects that are not passed to it remain unset, and a warning is issued for each of them: `function takes_two($a, $b) { if (isset($a)) { echo " a is set\n"; } if (isset($b)) { echo " b is set\n"; } } echo "With two arguments:\n"; takes_two(1, 2); echo "With one argument:\n"; takes_two(1);` With two arguments: a is set b is set With one argument: Warning: Missing argument 2 for takes_two () in /path/to/script.php on line 6 a is

Notes

set. `$_GET` is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. There is no need to do global \$variable; to access it within functions or methods.

Self Assessment

Fill in the blanks:

- 10. PHP gives you option to pass your parameters inside a
- 11.is the default way to handle parameters in a function.
- 12.parameters like C++ are supported by PHP.

4.4 Return Values

Besides being able to pass functions information, you can also have them return a value. However, a function can only return one thing, although that thing can be any integer, float, array, string, etc. that you choose!

How does it return a value though? Well, when the function is used and finishes executing, it sort of changes from being a function name into being a value. To capture this value you can set a variable equal to the function. Something like:

```
$myVar = somefunction();
```


Let's demonstrate this returning of a value by using a simple function that returns the sum of two integers.



Example:

```
<?php
function mySum($numX, $numY){
    $total = $numX + $numY;
    return $total;
}
$myNumber = 0;
echo "Before the function, myNumber = ". $myNumber ."<br />";
$myNumber = mySum(3, 4); // Store the result of mySum in $myNumber
echo "After the function, myNumber = " . $myNumber ."<br />";
?>
```

When we first print out the value of `$myNumber` it is still set to the original value of 0. However, when we set `$myNumber` equal to the function `mySum`, `$myNumber` is set equal to `mySum`'s result. In this case, the result was $3 + 4 = 7$, which was successfully stored into `$myNumber` and displayed in the second echo statement!



Task Create a PHP code using `<html>`, `<head>` and `<title>` tag in the return value.

Self Assessment

Notes

State whether the following statements are true or false:

13. A function can only return one thing, although that thing can be any integer, float, array, string, etc. that you choose.
14. When the function is used and finishes executing, it sort of changes from being a function name into being a value.

4.5 Variable Functions

PHP Variable functions are used to assign a function's name to a variable.

Variable Function means that if a variable name has a format like this `$var()`, PHP will consider this as a variable function and will look for a function with a name which is assigned to it, and will try to execute it.



Caution This variable function does not work with `echo()`, `print()`, `unset()` etc. language constructs.



Example:

```
<?php
function fun()
{
echo "This is an example";
}
function disp($var){
echo "<br/>Value is:". $var;
}
$f=' fun' ;
$f();
$f=' disp' ;
$f('New Val');
?>
```

Self Assessment

Fill in the blanks:

15. PHP Variable functions are used to assign a function's name to a
16. Variable Function means that if a variable name has a format like then PHP will consider this as a variable function.

4.6 Category of PHP Functions

This is a list of functions provided by PHP's built-in extensions, grouped by category. Some functions fall under more than one header.

Notes

Arrays

array, array_count_values, array_diff, array_filter, array_flip, array_intersect, array_keys, array_map, array_merge, array_merge_recursive, array_multisort, array_pad, array_pop, array_push, array_rand, array_reduce, array_reverse, array_search, array_shift, array_slice, array_splice, array_sum, array_unique, array_unshift, array_values, array_walk, arsort, asort, compact, count, current, each, end, explode, extract, implode, in_array, key, key_exists, krsort, ksort, list,

Classes and Objects Notes

call_user_method, call_user_method_array, class_exists, get_class, get_class_methods, get_class_vars, get_declared_classes, get_object_vars, get_parent_class, is_subclass_of, method_exists

Date and Time

checkdate, date, getdate, gettimeofday, gmdate, gmmktime, gmstrftime, localtime, microtime, mktime, strftime, strtotime, time

Errors and Logging

assert, assert_options, closelog, crc32, define_syslog_variables, error_log, error_reporting, openlog, restore_error_handler, set_error_handler, syslog, trigger_error, user_error

Files, Directories, and Filesystem

basename, chdir, chgrp, chmod, chown, chroot, clearstatcache, closedir, copy, dirname, disk_free_space, disk_total_space, fclose, feof, fflush, fgetc, fgets, fgetcsv, fgets, fgetss, file, file_exists, fileatime, filectime, filegroup, fileinode, filemtime, fileowner, fileperms, filesize, filetype, flock, fopen, fpassthru, fputs, fread, fscanf, fseek, fstat, ftell, ftruncate, fwrite, getcwd, getlastmod, is_dir, is_executable, is_file, is_link, is_readable, is_uploaded_file, is_writable, is_writeable, link, linkinfo, lstat, mkdir, move_uploaded_file, opendir, pathinfo, pclose, readdir, readfile, readlink, realpath, rename, rewind, rewinddir, rmdir, set_file_buffer, stat, symlink, tempnam, tmpfile, touch, umask, unlink

Functions

call_user_func, call_user_func_array, create_function, func_get_arg, func_get_args, func_num_

args, function_exists, get_defined_functions, get_extension_funcs,
get_loaded_extensions,
register_shutdown_function, register_tick_function, unregister_tick_function

HTTP

get_browser, get_meta_tags, header, headers_sent, parse_str, parse_url,
rawurldecode,
rawurlencode, setcookie

Mail

mail

Math

abs, acos, asin, atan, atan2, base_convert, bindec, ceil, cos, decbin,
dechex, decoct, deg2rad, exp,
floor, getrandmax, hexdec, lcg_value, log, log10, max, min, mt_getrandmax,
mt_rand, mt_srand,
number_format, octdec, pi, pow, rad2deg, rand, round, sin, sqrt, srand, tan

Network

checkdnsrr, fsockopen, gethostbyaddr, gethostbyname, gethostbyname_l,
getmxrr,
getprotobyname, getservbyname, getservbyport, ip2long, long2ip, pfsockopen,
socket_get_
status, socket_set_blocking, socket_set_timeout

Output Control

flush, ob_end_clean, ob_end_flush, ob_get_contents, ob_get_length,
ob_gzhandler, ob_implicit_
flush, ob_start

PHP Options/Info

assert, assert_options, dl, extension_loaded, get_cfg_var, get_current_user,
get_extension_funcs,
get_included_files, get_loaded_extensions, get_magic_quotes_gpc,
get_required_files, getenv,
getlastmod, getmyinode, getmypid, getrusage, highlight_file, highlight_string,
ini_alter,
ini_get, ini_restore, ini_set, localeconv, parse_ini_file, php_logo_guid,
php_sapi_name, php_
uname, phpcredits, phpinfo, phpversion, putenv, set_magic_quotes_runtime,
set_time_limit,
version_compare, zend_logo_guid, zend_version

Notes

Program Execution

escapeshellarg, escapeshellcmd, exec, passthru, putenv, shell_exec, sleep, system, usleep

Strings

addslashes, addslashes, base64_decode, base64_encode, chop, chr, chunk_split, convert_cyr_string, count_chars, crypt, echo, ereg, ereg_replace, eregi, eregi_replace, explode, get_html_translation_table, get_meta_tags, hebrew, hebrevc, highlight_string, htmlentities, htmlspecialchars, implode, iptcp, join, levenshtein, localeconv, ltrim, md5, metaphone, nl2br, number_format, ord, parse_str, parse_url, print, printf, quoted_printable_decode, quotemeta, rtrim, setlocale, similar_text, soundex, split, spliti, sprintf, sql_regcase, sscanf, str_pad, str_repeat, str_replace, strcasecmp, strchr, strcmp, strcoll, strcspn, strip_tags, stripslashes, strstr, strlen, strnatcasecmp, strnatcmp, strncasecmp, strncmp, strpos, strrchr, strrev, strrpos, strspn, strstr, strtok, strtolower, strtoupper, substr, substr_count, substr_replace, trim, ucfirst, ucwords, vprintf, vsprintf, wordwrap

Type Functions

doubleval, get_resource_type, gettype, intval, is_array, is_bool, is_double, is_float, is_int, is_integer, is_long, is_null, is_numeric, is_object, is_real, is_resource, is_scalar, is_string, settype, strval

URLs

base64_decode, base64_encode, parse_url, rawurldecode, rawurlencode, urldecode, urlencode

Variable Functions

compact, empty, extract, get_defined_constants, get_defined_vars, import_request_variables, isset, list, print_r, putenv, serialize, uniqid, unserialize, unset, var_dump

Self Assessment

State whether the following statements are true or false:

- 17. The category of HTTP is get_meta_tags.
- 18. The category of Network is escapeshellarg.



Case Study

Is PHP Embarrassingly Slower than Java?

IP2C is a small library that provides IP to country resolution. It uses the free ip-to-country database. IP2C takes the database CSV file that is about 4mb and converts it into a ~600kb binary format and provides PHP and Java frontend to query the database. The library is great, easy to convert an IP to a country and when using the country flags from its side project you could spice up your statistics with the country information. This is a lot faster than using reverse DNS lookup.

The Problem: The PHP implementation is a lot slower. Embarrassingly slower. Without any caching the Java version is able to do ~6000 queries per second. The PHP counterpart can push through ~850 queries. The implementations are the same. The statistics provided by the author of the library are 8000 vs 1200. So about the same as my measurements. I like PHP, I do not use it that much anymore but I still care when I see such embarrassing numbers. I took the implementation and started profiling it. Spent the night running different tests and trying to optimize.

General outline of the algorithm is as follows. We take the dotted string IP and convert it to an IPv4 Internet network address (e.g. 69.55.232.153 becomes 1161291929). The DB holds sorted ranges of these addresses. A binary search will happen on these addresses and we have a country for the IP. Take a look at the implementation.

Incl.	Self	Called	Function	Location
101.10	33.33	(0)	{main}	allBenchmarks.php
65.15	0.59	1	runBenchmark	allBenchmarks.php
64.56	0.92	1 000	ip2country->get_country	ip2c.php
63.60	14.14	15 000	ip2country->find_country_...	ip2c.php
49.26	19.95	15 000	ip2country->getPair	ip2c.php
12.43	10.22	15 007	ip2country->readInt	ip2c.php
10.91	10.13	15 000	ip2country->readShort	ip2c.php
5.69	5.26	15 000	ip2country->seek	ip2c.php
1.73	1.73	100 000	func	allBenchmarks.php
1.61	1.61	30 008	php::fread	php:internal
0.92	0.92	30 007	php::unpack	php:internal
0.87	0.87	1	php::file	php:internal
0.31	0.31	15 000	php::fseek	php:internal
0.01	0.00	1	ip2country->ip2country	ip2c.php
0.01	0.01	1 000	php::ip2long	php:internal
0.00	0.00	2	microtime_float	allBenchmarks.php
0.00	0.00	1	require_once::/home/toom...	ip2c.php
0.00	0.00	1	php::fopen	php:internal
0.00	0.00	1	require_once::/home/toom...	ip2CRecursionRemoved.php

Let's see where the vanilla version of IP2C spends its time at. The results are based on 1000 iterations with Xdebug enabled and visualized by KCacheGrind. It processed about 210 IP addresses during this time. IO part is surprisingly low. The internal fseek, fread constitute to 2% of the execution time. On the other hand, the user level fseek which is just a wrapper alone uses 5%. readShort and readInt takes 20% of the execution time.

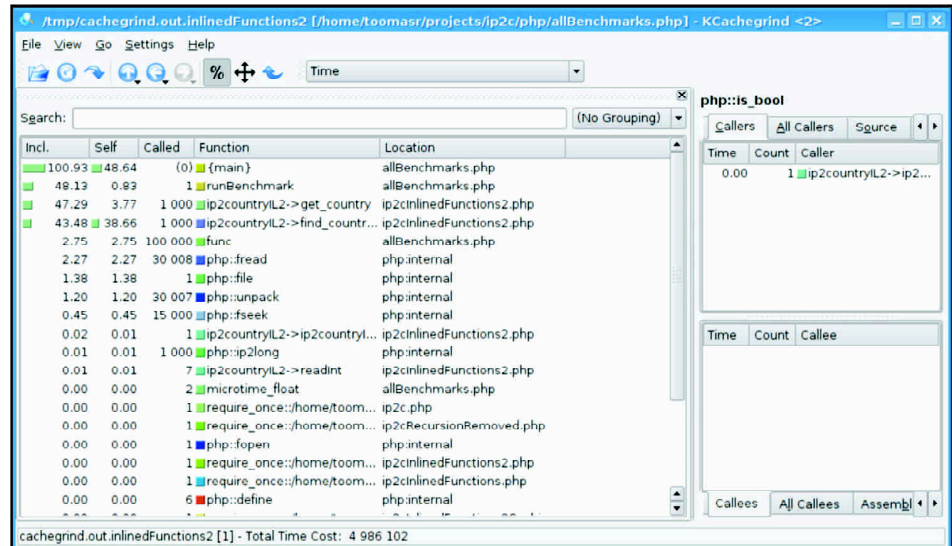
```
function readShort() {
    $a = unpack('n', fread($this->m_file, 2));
    return $a[1];}
function readInt() {
    $a =unpack('N', fread($this->m_file, 4));
```

Contd...

Notes

Notes

```
return $a[1];}
function seek($offset) {
fseek($this->m_file, $offset);}
```



Functions calls are expensive. Let’s eliminate them. readInt, readShort, fseek are now inlined. Recursion changed to iteration (e.g. 14,000 less function calls). Able to process 400 queries per second compared to the previous 210.

We see that the latest profiling results have twice the number of fread and unpacks than fseeks. It seems that fseek is used to seek out the right position, read two numbers with unpacking them. The implementation confirms that luckily we could just read once (2 bytes more) and unpack once (2 unpackings with one invocation).

```
$a =unpack('N', fread($this->m_file, 4));
$np['ip'] = $a[1];
$a =unpack('n', fread($this->m_file, 2));
$np['key'] = $a[1];
// this can be changed to
$np =unpack('Nip/nkey', fread($this->m_file, 6));
```

How does this version stack up to the Java version? Let’s disable profiling and run 100,000 iterations. Vanilla version processes ~850 IPs, when functions are inlined the number is around 1400. Java version can still do 6000.

Let’s try caching. Peeking at the Java implementation shows that Java caching version (whopping 141,242 IPs per second – yup 141k) uses just a byte[] array and makes lookups from there instead of seeking and reading from file. Easy, let’s do the same in PHP. We read everything into a string and instead of fread with access the string elements with the offset. For fseek with just set the offset. We are using 600kb more memory but can increase the throughput to ~2800. As it seems I have just wasted a night, I just should have checked the Computer Language Benchmarks. PHP in the sense of execution speed is uncomparable to Java.

Questions

1. Point out the shortcomings of PHP.
2. Explain the functions that are used in the case study.

4.7 Summary

Notes

- A function call is a piece of code that tells your program to “call in” the built-in function whenever you need it. The second type of function is the user defined function. These are the functions that you write yourself.
- Unlike PHP variable names, PHP function names are case insensitive. It is however recommended to remain rigorous when naming and calling any PHP object.
- The function takes two arguments, \$left and \$right. Using the concatenation operator, the function creates a combined string in the variable \$combined_string.
- The variables used in PHP functions can be of three types: locals, globals and statics. Any variable defined inside a function is by default limited to the local function scope, is available only in the code within that function.
- PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it.

4.8 Keywords

Default Parameters: Default parameters enable you to specify a default value for function parameters that are not passed to the function during the function call.

Function: A function call is a piece of code that tells your program to “call in” the built-in function whenever you need it.

Global Variables: A global variable can be accessed in any part of the program. If you want to use inside a function a variable defined outside it, that variable must be explicitly declared to be global in the function.

Static Variables: A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope.

4.9 Review Questions

1. What do you mean by function? How does it define in PHP?
2. How do we call a function in PHP?
3. Explain the user defined function with example.
4. What is the variable scope in PHP?
5. Differentiate between local and global scope of variables.
6. What is the function parameters used in PHP?
7. Differentiate between pass by value and pass by reference parameters.
8. Explain the return values in PHP.
9. What are the variable functions?
10. How many categories of PHP functions are there?

Notes

Answers: Self Assessment

- | | |
|-------------------|--------------|
| 1. True | 2. False |
| 3. False | 4. Single |
| 5. Comma | 6. String |
| 7. False | 8. False |
| 9. True | 10. Function |
| 11. Call by value | 12. Default |
| 13. True | 14. True |
| 15. Variable | 16. \$var() |
| 17. True | 18. False |

4.10 Further Readings



Books

Bangia, Ramesh (2008). *“Web Technology (including HTML, CSS, XML, ASP, JAVA).”* Firewall Media.

Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.

Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.

Puntambekar, A.A. (2009). *“Web Technologies.”* Technical Publications.

Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.

Xavier, C. (2007). *“Web Technology and Design.”* New Age International.



Online links

<http://www.homeandlearn.co.uk/php/php8p1.html>

<http://www.phpforkids.com/php/php-functions-user-defined.php>

<http://www.tizag.com/phpT/phpfunctions.php>

<http://www.roseindia.net/tutorial/php/phpbasics/tutorial/PHP-Variable-Functions.html>

Unit 5: Strings

Notes

CONTENTS

Objectives

Introduction

- 5.1 Quoting String Constants
 - 5.1.1 Variable Interpolation
- 5.2 Printing Strings
 - 5.2.1 echo
 - 5.2.2 print ()
 - 5.2.3 Printf()
 - 5.2.4 print_r() and var_dump()
- 5.3 Accessing Individual Characters
- 5.4 Cleaning Strings
 - 5.4.1 Removing Whitespace
 - 5.4.2 Changing Case
- 5.5 Encoding and Escaping
 - 5.5.1 HTML
 - 5.5.2 URLs
 - 5.5.3 SQL
 - 5.5.4 C-String Encoding
- 5.6 Comparing Strings
 - 5.6.1 Exact Comparisons
 - 5.6.2 Approximate Equality
- 5.7 Manipulating and Searching Strings
 - 5.7.1 Substrings
 - 5.7.2 Miscellaneous String Functions
 - 5.7.3 Decomposing a String
 - 5.7.4 String-Searching Functions
- 5.8 Regular Expressions
 - 5.8.1 Regular Expression Syntax
 - 5.8.2 Regular Expression Functions
- 5.9 Summary
- 5.10 Keywords
- 5.11 Review Questions
- 5.12 Further Readings

Notes

Objectives

After studying this unit, you will be able to:

- Understand How to Quote String Constants
- Discuss about the Printing Strings
- Understand How to Access the Character
- Explain Cleaning the Strings
- Discuss the Encoding and Escaping
- Explain Comparing Strings
- Discuss How to Manipulating and Searching Strings
- Discuss the Regular Expressions

Introduction

Strings are sequence of characters. In PHP, a character is the same as a byte, therefore there are exactly 256 different characters possible. Long string is supported in PHP, in fact there is no practical bound to the size of strings. But PHP has no native support of Unicode.

5.1 Quoting String Constants

A string literal can be specified in three different ways. These are as follows:

- Single quoted
 - Double quoted
 - Heredocs
1. **Single-Quoted Strings:** The simple way to print a string is to enclose it in a single quotes (use the character `'`). If you want to print a single quote (`'`) within a string then you must escape it with a backslash like many other language. If a backslash (`\`) needs to print before a single quote or at the end of the string then the backslash must occur twice. At the end of the every string a html break character has added for line breaking.



Example:

```
<?php
echo `One line simple string.<br />`;
echo 'Two line simple
string example<br />';
echo `Tomorrow I \\'ll learn PHP global variables.<br
/>`;
echo `This is a bad command : del c:\\*. * <br />`;
?>
```

The above echo statements display single quoted strings. In the second echo statement we have written the string in two lines though it will be displayed in a single line. In the third echo statement a single quotation character (`'`) is printed for the backslash character. In the fourth echo statement, a backslash character(`\`) has printed after c; here we have taken two

backslash characters together. - See more at: <http://www.w3resource.com/php/data-types/strings.php#sthash.NxdRMlyH.dpuf>

2. **Double-Quoted Strings:** When we want to print some special characters or the values of variables within a string then we enclose the string with double-quotes(") character.

Whenever a \$ symbol appears within a string, PHP tries to read the immediate part followed the \$ character as a variable name. If the variable type is string, that string is inserted into the string in that position, if the variable type is non-string then it is converted into string type.



Example:

```
< ?php
    $samt=2000;
    $desc="My salary amount for this month is : ";
    echo "$desc $samt <br />";
    // We can write the above example in the following way
    echo "My salary amount for this month is : $$samt";
?>
```



Did u know? String functions do not handle multibyte encodings such as UTF-8. We use the functions found in multibyte string for encodings.

3. **Heredocs:** PHP introduces a more robust string creation tool called heredoc that lets the programmer create multi-line strings without using quotations. The example given below will create a string with a heredoc syntax. The heredoc preserves the line breaks and other whitespace (including indentation) in the text. The heredoc is created with <<< followed by a delimiting identifier, followed, starting on the next line, by the text to be quoted, and then closed by the same identifier on its own line. The closing identifier must not be indented. It can only contain alphanumeric characters and underscores, and must start with a non-digit character or underscore.

```
<?php
    $str = <<<TEXT

    "That is just as I intended." Vautrin said. "You know quite well what you are about. Good,
    my little eaglet! You are born to command, you are strong, you stand firm on your feet,
    you are game! I respect you."

    TEXT;

    echo $str, "\n";
?>
```

The semicolon is optional.

```
echo "PHP " . "language"; # prints PHP language
```

PHP uses the dot (.) operator to concatenate two strings.

5.1.1 Variable Interpolation

When you define a string literal using double quotes or a heredoc, the string is subject to *variable interpolation*. In addition to the single-quote and double-quote syntaxes there is another way

Notes

to embed large pieces of text in your scripts which may include lots of double and single quotes.

Syntax

```
<<<identifier
....text here .....
....text here .....
....text here .....
identifier
```



Example:

```
<?php
    $mystring=<<<MYID
    "Tomorrow I'll not go at your house"
    "may be another day."
    'Thank you'
    MYID;
    echo $mystring;
?>
```

Self Assessment

State whether the following statements are true or false:

1. A string literal can be specified in two different ways.
2. The simple way to print a string is to enclose it in a double quotes.
3. When you define a string literal using double quotes or a heredoc, the string is subject to variable Interpolation.

5.2 Printing Strings

There are four ways to send output to the browser. The echo construct lets you print many values at once, while print () prints only one value. The printf() function builds a formatted string by inserting values into a template. The print_r() function is useful for debugging it prints the contents of arrays, objects, and other things, in a more or less human-readable form.

5.2.1 echo

The **echo ()** function is used to output the given argument. It can output all types of data and multiple outputs can be made with only one **echo ()** command.



Example:

```
<?php
    Echo "Hello";
    //Outputs a string
    Echo $variable;
    //Outputs a variable
    Echo "Multiple things " . $on . " one line";
```

```
//Outputs a string, then a variable, then a string. All are separated with
a [.] period
?>
```

5.2.2 print ()

The **print ()** function is used to output the given argument. It can output all types of data and multiple outputs can be made with only one **print ()** command.



Example:

```
<?php
Print "Hello";
//Outputs a string
Print $variable;
//Outputs a variable
Print "Multiple things " . $on . " one line";
//Outputs a string, then a variable, then a string. All are separated with
a [.] period
?>
```

5.2.3 Printf()

Printf() takes a variable number of parameters - a format string is always the first parameter, followed by zero or more other parameters of various types.



Example:

```
<?php
$foo = "lions, tigers, and bears";
printf("There were %s - oh my!", $foo);
?>
```

That will put together the string "There were lions, tigers, and bears - oh my!" And send it to output. %s is a special format string that means "string parameter to follow", which means that \$foo will be treated as text inside the string that *printf()* creates.



Task Develop a simple program to differentiate between *print()* and *printf()*.

5.2.4 print_r() and var_dump()

The *print_r ()* PHP function is used to return an array in a human readable form. It is simply written as *Print_r (\$your_array)* and also Known As: Print Array



Example:

```
<?php
$Names = array ('a' => 'Angela', 'b' => 'Bradley', 'c' => array
('Cade', 'Caleb'));
```

Notes

```
print_r ($Names);  
?>
```

The `var_dump()` function is used to display structured information (type and value) about one or more variables.

Syntax

```
var_dump(variable1, variable2, ...variablen)
```



Example:

```
<?php  
$var_name1=678;  
$var_name2="a678";  
$var_name3="678";  
$var_name4="W3resource.com";  
$var_name5=698.99;  
$var_name6=+125689.66;  
echo var_dump($var_name1)."<br>";  
echo var_dump($var_name2)."<br>";  
echo var_dump($var_name3)."<br>";  
echo var_dump($var_name4)."<br>";  
echo var_dump($var_name5)."<br>";  
echo var_dump($var_name6)."<br>";  
?>
```

Self Assessment

Fill in the blanks:

4. There areways to send output to the browser.
5. Thefunction builds a formatted string by inserting values into a template.
6. Thefunction is useful for debugging it prints the contents of arrays, objects.

5.3 Accessing Individual Characters

If you would like to process individual characters of a string in php, you can access them by using following two methods:

Method 1

By using square bracket syntax and zero based offset. So, for example, if you want to access second character of a string stored in variable `$string_var`, then use the expression `$char_value = $string_var[2]`, now `$char_value` will contain a string 1 character long containing the required character.

PHP Code:

```
$sample_string = 'This is a sample string';  
for($i = 0;$i < strlen($sample_string);$i++)  
{
```

```
    var_dump($sample_string[$i]);
}
```

Method 2

By using substr function, we can use substr function to extract 1 character substrings successively within a for loop, hence accessing individual characters in string.

PHP Code:

```
$sample_string = 'This is a sample string';
for($i = 0;$i < strlen($sample_string);$i++)
{
    var_dump(substr($sample_string, $i, 1));
}
```

Self Assessment

State whether the following statements are true or false:

7. By using square bracket syntax and zero based offset.
8. By using substr function, we can use substr function to extract 1 character substrings successively within a for loop.

5.4 Cleaning Strings

Often, the strings we get from files or users need to be cleaned up before we can use them. Two common problems with raw data are the presence of extraneous whitespace, and incorrect capitalization (uppercase versus lowercase).

5.4.1 Removing Whitespace

If you use a form on your site then you'll know that user input can often have white space before or after the text as the person filling in the form field sometimes accidentally adds spaces.



Example:

```
<?php

// The original text string with whitespace at the end
$textString = "This is some text with a space after it ";

// Remove the whitespace
$trimmedTextString = trim($textString);

// Display the new text string
echo $trimmedTextString;

?>
```

5.4.2 Changing Case

The ability to take strings of text and manipulate them is one of the essential abilities you need as a programmer. If a user enters details on your forms, then you need to check and validate this data. For the most part, this will involve doing things to text. For example, converting letters to uppercase or lowercase, checking an email address to see if all the parts are there, checking which browser the user has, trimming white space from around text entered in a text box. All of these come under the heading of string manipulation.

```
string strtoupper (string source)
string strtolower (string source)
string ucfirst (string source)
string ucwords (string source)
```

Strtoupper() is part of a small family of functions that affect the case of characters of strings. *Strtoupper()* takes one string parameter, and returns that string entirely in uppercase.



Notes Other variations include *strtolower()*, to convert the string to lowercase, *ucfirst()* to convert the first letter of every string to uppercase, and *ucwords()*, to convert the first letter of every word in the string to uppercase.

They all take one parameter and return the converted result, so once you learn one you have learnt them all:



Example:

```
<?php
    $string = "i like to program in PHP";
    $a = strtoupper($string);
    $b = strtolower($string);
    $c = ucfirst($string);
    $d = ucwords($string);
    $e = ucwords(strtolower($string));
?>
```

Each of those variables get set to a slightly different value: *\$a* becomes "I LIKE TO PROGRAM IN PHP", *\$b* becomes "i like to program in php", *\$c* becomes "I like to program in PHP", *\$d* becomes "I Like To Program In PHP", and *\$e* becomes "I Like To Program In Php".

From that, you should be able to see that in calls such as *ucwords()*, PHP will not change existing capital letters to lowercase, which is why *\$d* and *\$e* are different - for *\$e*, all the letters are lower-cased first, then passed through *ucwords()* to make PHP into Php.



Caution If you have got a mixed-case string that you want to convert to "title case," where the first letter of each word is in uppercase and the rest of the letters are in lowercase, use a combination of *strtolower()* and *ucwords()*.

Self Assessment

Notes

Fill in the blanks:

9. The ability take strings of text and manipulate them is one of the essential abilities you need as a
10.is part of a small family of functions that affect the case of characters of strings.

5.5 Encoding and Escaping

Because PHP programs often interact with HTML pages, web addresses (URLs), and databases, there are functions to help you work with those types of data. HTML, web page addresses, and database commands are all strings, but they each require different characters to be escaped in different ways. For instance, a space in a web address must be written as %20, while a literal less-than sign (<) in an HTML document must be written as <. PHP has a number of built-in functions to convert to and from these encodings.

5.5.1 HTML

Special characters in HTML are represented by *entities* such as & and <. There are two PHP functions for turning special characters in a string into their entities, one for removing HTML tags, and one for extracting only meta tags.

Entity-quoting all special characters

The `htmlspecialchars()` function changes all characters with HTML entity equivalents into those equivalents (with the exception of the space character). This includes the less-than sign (<), the greater-than sign (>), the ampersand (&), and accented characters.



Example:

```
$string = htmlentities("Einstürzende Neubauten"); echo $string; Einstürzende Neubauten
The entity-escaped version (&uuml;) correctly displays as ü in the web page. As you can see, the space has not been turned into &nbsp;.
```

The `htmlspecialchars()` function actually takes up to three arguments:

```
$output = htmlspecialchars($input, $quote_style, $charset);
```

The *charset* parameter, if given, identifies the character set. The default is "ISO-8859-1". The *quote_style* parameter controls whether single and double quotes are turned into their entity forms. `ENT_COMPAT` (the default) converts only double quotes, `ENT_QUOTES` converts both types of quotes, and `ENT_NOQUOTES` converts neither. There is no option to convert only single quotes.



Example:

```
$input = <<< End "Stop pulling my hair!" Jane's eyes flashed.<p> End; $double =
htmlspecialchars($input); // "Stop pulling my hair!" Jane's eyes flashed.&lt;p&gt; $both
= htmlentities($input, ENT_QUOTES); // "Stop pulling my hair!" Jane's eyes
flashed.&lt;p&gt; $neither = htmlentities($input, ENT_NOQUOTES); // "Stop pulling my hair!"
Jane's eyes flashed.&lt;p&gt;
```

Notes

Entity-Quoting only HTML Syntax Characters

The `htmlspecialchars()` function converts the smallest set of entities possible to generate valid HTML. The following entities are converted:

- Ampersands (&) are converted to `&`;
- Double quotes (") are converted to `"`;
- Single quotes (') are converted to `'`; (if `ENT_QUOTES` is on, as described for `htmlspecialchars()`)
- Less-than signs (<) are converted to `<`;
- Greater-than signs (>) are converted to `>`;

If you have an application that displays data that a user has entered in a form, you need to run that data through `htmlspecialchars()` before displaying or saving it. If you do not, and the user enters a string-like "angle <30" or "sturm & drang", the browser will think the special characters are HTML, and you will have a garbled page.

Like `htmlspecialchars()`, `htmlspecialchars()` can take up to three arguments:

```
$output = htmlspecialchars(input, [quote_style, [charset]]);
```

The `quote_style` and `charset` arguments have the same meaning that they do for `htmlspecialchars()`. There are no functions specifically for converting back from the entities to the original text, because this is rarely needed. There is a relatively simple way to do this, though. Use the `get_html_translation_table()` function to fetch the translation table used by either of these functions in a given quote style. For example, to get the translation table that `htmlspecialchars()` uses, do this:

```
$table = get_html_translation_table(HTML_ENTITIES);
```

To get the table for `htmlspecialchars()` in `ENT_NOQUOTES` mode, use:

```
$table = get_html_translation_table(HTML_SPECIALCHARS, ENT_NOQUOTES);
```

A nice trick is to use this translation table, flip it using `array_flip()`, and feed it to `strtr()` to apply it to a string, thereby effectively doing the reverse of `htmlspecialchars()`:

```
$str = htmlspecialchars("Einstürzende Neubauten"); // now it is encoded
$table = get_html_translation_table(HTML_ENTITIES); $rev_trans =
array_flip($table); echo strtr($str, $rev_trans);
// back to normal Einstürzende Neubauten
```

You can, of course, also fetch the translation table, add whatever other translations you want to it, and then do the `strtr()`. For example, if you wanted `htmlspecialchars()` to also encode spaces to ` `, you would do:

```
$table = get_html_translation_table(HTML_ENTITIES); $table[' '] = '&nbsp;';
$encoded =
strtr($original, $table);
```

Removing HTML Tags

The `strip_tags()` function removes HTML tags from a string:

```
$input = '<p>Howdy, &quot;Cowboy&quot;</p>'; $output = strip_tags($input);
// $output is
'Howdy, &quot;Cowboy&quot;'
```


The function may take a second argument that specifies a string of tags to leave in the string. List only the opening forms of the tags. The closing forms of tags listed in the second parameter are also preserved:

```
$input = 'The <b>bold</b> tags will <i>stay</i><p>'; $output =
strip_tags($input, '<b>'); //
$output is 'The <b>bold</b> tags will stay'
```

Attributes in preserved tags are not changed by `strip_tags()`. Because attributes such as `style` and `onmouseover` can affect the look and behaviour of web pages, preserving some tags with `strip_tags()` would not necessarily remove the potential for abuse.

Extracting Meta Tags

If you have the HTML for a web page in a string, the `get_meta_tags()` function returns an array of the meta tags in that page. The name of the meta tag (`keywords`, `author`, `description`, etc.) becomes the key in the array, and the content of the meta tag becomes the corresponding value:

```
$meta_tags = get_meta_tags('http://www.example.com/'); echo "Web page made
by {$meta_tags[author]}"; Web page made by Pradip
```

The general form of the function is:

```
$array = get_meta_tags(filename [, use_include_path]);
```

Pass a true value for `use_include_path` to let PHP attempt to open the file using the standard include path.

5.5.2 URLs

PHP provides functions to convert to and from URL encoding, which allows you to build and decode URLs. There are actually two types of URL encoding, which differ in how they treat spaces. The first (specified by RFC 1738) treats a space as just another illegal character in a URL and encodes it as `%20`. The second (implementing the `application/x-www-form-urlencoded` system) encodes a space as a `+` and is used in building query strings.

RFC 1738 Encoding and Decoding

To encode a string according to the URL conventions, use `rawurlencode()`:

```
$output = rawurlencode(input);
```

This function takes a string and returns a copy with illegal URL characters encoded in the `%dd` convention.

If you are dynamically generating hypertext references for links in a page, you need to convert them with `rawurlencode()`:

```
$name = "Programming PHP"; $output = rawurlencode($name); echo "http://
localhost/$output"; http://localhost/Programming%20PHP
```

The `rawurldecode()` function decodes URL-encoded strings:

```
$encoded = 'Programming%20PHP'; echo rawurldecode($encoded); Programming
PHP
```

Query-string Encoding

The `urlencode()` and `urldecode()` functions differ from their raw counterparts only in that they encode spaces as plus signs (`+`) instead of as the sequence `%20`. This is the format for building

Notes

query strings and cookie values, but because these values are automatically decoded when they are passed through a form or cookie, you do not need to use these functions to process the current page's query string or cookies. The functions are useful for generating query strings:

```
$base_url = 'http://www.google.com/q='; $query = 'PHP sessions -cookies';  
$url = $base_url .  
urlencode($query); echo $url;http://www.google.com/q=PHP+sessions+-cookies
```

5.5.3 SQL

Most database systems require that string literals in your SQL queries be escaped. SQL's encoding scheme is pretty simple—single quotes, double quotes, NUL-bytes, and backslashes need to be preceded by a backslash. The `addslashes()` function adds these slashes, and the `stripslashes()` function removes them:

```
$string = <<< The_End "It is never going to work," she cried, as she hit the backslash (\\) key.  
The_End; echo addslashes($string); \"It\'s never going to work,\" she cried, as she hit the  
backslash (\\) key. echo stripslashes($string); "It is never going to work," she cried, as she hit  
the backslash (\\) key.
```

Some databases escape single quotes with another single quote instead of a backslash. For those databases, enable `magic_quotes_sybase` in your `php.ini` file.

5.5.4 C-String Encoding

The `addslashes()` function escapes arbitrary characters by placing backslashes before them. The `addslashes()` and `stripslashes()` functions are used with nonstandard database systems that have their own ideas of which characters need to be escaped.

Call `addslashes()` with two arguments—the string to encode and the characters to escape:

```
$escaped = addslashes(string, charset); Specify a range of characters to escape with the "..."  
construct:
```

```
echo addslashes("hello\tworld\n", "\x00..\x1fz..\xff"); hello\tworld\n
```

Beware of specifying `'0'`, `'a'`, `'b'`, `'f'`, `'n'`, `'r'`, `'t'`, or `'v'` in the character set, as they will be turned into `'\0'`, `'\a'`, etc. These escapes are recognized by C and PHP and may cause confusion.

`stripslashes()` takes a string and returns a copy with the escapes expanded:

```
$string = stripslashes(escaped);
```



Example:

```
$string = stripslashes('hello\tworld\n'); // $string is "hello\tworld\n"
```

Self Assessment

State whether the following statements are true or false:

11. The `quote_style` and `charset` arguments have the same meaning that they do for `htmlentities()`.
12. The `get_meta_tags()` function converts the smallest set of entities possible to generate valid HTML.

5.6 Comparing Strings

Comparing two string values in a PHP-based Web page helps you decide what course of action to take.



Example: If a visitor attempts to log in to your site, you should compare the username and passwords the person submits to those saved on your server to ensure the person submitted the correct information. When comparing strings, you can directly use strings in quotation marks or provide variables with string value.

5.6.1 Exact Comparisons

PHP has two functions – `strcmp` and `strncmp` – that compare two strings and returns a numerical value based on the result. The functions compare the values of the characters and return a positive value when the first string is greater than the second, a negative value if it is less, and zero when the two strings are equal.



Example: “A” and “a” return zero, while “a” and “b” return a negative value, because the numerical representation of “a” is less than “b.” With the `strncmp` function, you also provide an integer to indicate how many characters to compare.

5.6.2 Approximate Equality

Like the case-sensitive functions, `strcasecmp` and `strncasecmp` compare two strings and return less than, greater than or equal to zero, but these functions do not take case into consideration.



Example: Either function returns “My String” and “my string” as equal strings. You provide two strings as parameters for the `strcasecmp` function and it compares the entirety of both strings. With `strncasecmp`, you provide two strings as well as an integer number to indicate the number of characters to compare starting from the beginning of each string.



Example: Comparing only the first three characters of “My String” and “My String here” returns equal.

Self Assessment

Fill in the blanks:

13. PHP hasfunctions that compare two strings and returns a numerical value based on the result.
14. The functions compare the values of the characters and return avalue when the first string is greater than the second.

5.7 Manipulating and Searching Strings

PHP has many functions to work with strings. The most commonly used functions for searching and modifying strings are those that use regular expressions to describe the string in question. The functions described in this do not use regular expressions. They are faster than regular expressions, but they work only when you are looking for a fixed string.

5.7.1 Substrings

The PHP substring function, *substr*, allows a programmer to retrieve a specified portion of a string known as a substring. String manipulation is a very important and sometimes tricky task so care must be taken to ensure that the functions are properly used and understood.



Notes We usually need to use the PHP substring function anytime we want to break a larger string into a smaller string.

Syntax:

The syntax of the PHP substring function, *substr*, is shown below:

```
$substring = substr($original, $start, $length)
```

5.7.2 Miscellaneous String Functions

The *strrev()* function takes a string and returns a reversed copy of it:

```
$string = strrev(string);
```



Example:

```
echo strrev("There is no cabal"); labac on si erehT
```

The *str_repeat()* function takes a string and a count and returns a new string consisting of the argument *string* repeated *count* times:

```
$repeated = str_repeat(string, count);
```

For example, to build a crude horizontal rule:

```
echo str_repeat('-', 40);
```

The *str_pad()* function pads one string with another. Optionally, you can say what string to pad with, and whether to pad on the left, right, or both:

```
$padded = str_pad(to_pad, length [, with [, pad_type ]]);
```

The default is to pad on the right with spaces:

```
$string = str_pad('Pradip Kumar', 30); echo "$string:35:Riya"; ABC :35:Riya
```

The optional third argument is the string to pad with:

```
$string = str_pad('Pradip Kumar', 30, '. '); echo "{$string}35"; ABC. . .  
. . . . .35
```

The optional fourth argument can be either *STR_PAD_RIGHT* (the default), *STR_PAD_LEFT*, or *STR_PAD_BOTH* (to centre). For example:

```
echo '[' . str_pad('Pradip Kumar', 30, ' ', STR_PAD_LEFT) . "]\n"; echo '[' .  
. str_pad('ABC', 30, ' ', STR_PAD_BOTH) . "]\n"; [ABC] [ ABC ]
```

5.7.3 Decomposing a String

PHP provides several functions to let you break a string into smaller components. In increasing order of complexity, they are *explode()*, *strtok()*, and *sscanf()*.

The PHP function *explode()* lets you take a string and blow it up into smaller pieces. For example, if you had a sentence you could ask *explode* to use the sentence's spaces " " as dynamite

and it would blow up the sentence into separate words, which would be stored in an array. The sentence "Hello, I would like to lose weight." Would look like this after explode got done with it:

The first argument that explode takes is the delimiter (our dynamite) which is used to blow up the second argument, the original string. Explode returns an array of string pieces from the original and they are numbered in order, starting from 0. Lets take a phone number in the form ###-###-#### and use a hyphen "-" as our dynamite to split the string into three separate chunks.



Example:

```
$rawPhoneNumber = "800-555-5555";

$phoneChunks = explode("-", $rawPhoneNumber);
echo "Raw Phone Number = $rawPhoneNumber <br />";
echo "First chunk = $phoneChunks[0]<br />";
echo "Second chunk = $phoneChunks[1]<br />";
echo "Third Chunk chunk = $phoneChunks[2]";
```

The function `sscanf()` is the input analog of `printf()`. `Sscanf()` reads from the string `str` and interprets it according to the specified format, which is described in the documentation for `sprintf()`.

Any whitespace in the format string matches any whitespace in the input string. This means that even a tab `\t` in the format string can match a single space character in the input string.

Syntax

```
sscanf ( string $str , string $format [, mixed &$... ] )
```



Example:

```
<?php
// getting the serial number
list($serial) = sscanf("SN/2350001", "SN/%d");
// and the date of manufacturing
$mandate = "January 01 2000";
list($month, $day, $year) = sscanf($mandate, "%s %d %d");
echo "Item $serial was manufactured on: $year-" . substr($month, 0, 3) . "-" .
$day.\n";
?>
```

5.7.4 String-Searching Functions

Using PHP, we can easily find out if a bigger string contains another string without using complicated methods such as regular expressions. To do this, we make use of the PHP `strpos` function. If the string we are looking for (the needle) is contained within the other string (the haystack), this function returns the (numeric) character location of the beginning of the string that we are searching for. On the other hand, if the string we are looking for is not found, this function returns false.



Caution By using a simple PHP `if` statement to check the result of the function we can determine if the string was found inside of the bigger string.

Notes

PHP strpos Syntax

The syntax for the *strpos* function is below:

```
$position = strpos($haystack, $needle);
```

Where *\$needle* is the string that we are searching for, *\$haystack* is the string that we are searching in, and *\$position* is the position in the haystack where the first match is found.



Example:

```
<?php
```

```
// the haystack as it were
$string = "I love Tutorial Arena so much";

// the needle
$find = 'Arena';

// perform the search
$position = strpos($string, $find);

// We use === below because the needle we are looking for may
// start the haystack. In that case, the function would
// return 0 as the position of the first occurrence, but the if
// statement would treat that 0 as false if we used only ==

if ($position === false)
    echo "Not found";
else
    echo "Match found at location $position";

?>
```



Did u know? The possible keys of the hash are scheme, host, port, user, pass, path, query, and fragment.

Self Assessment

State whether the following statements are true or false:

- 15. The PHP function `explode()` lets you take a string and blow it up into smaller pieces.
- 16. The function `printf()` is the input analog of `scanf()`.

5.8 Regular Expressions

In this, we show how regular expressions can achieve more sophisticated pattern matching to find, extract, and replace complex substrings within a string. While regular expressions provide capabilities beyond those described in the last, complex pattern matching is not as efficient as

simple string comparisons. This begins with a brief description of the POSIX regular expression syntax. This is not a complete description of all of the capabilities, but we do provide enough details to create quite powerful regular expressions. The second half of it describes the functions that use POSIX regular expressions.

5.8.1 Regular Expression Syntax

To demonstrate the syntax of regular expressions, we introduce the function `ereg()` :

```
boolean ereg(string pattern, string subject [, array var])
```

`ereg()` returns true if the regular expression pattern is found in the subject string. We discuss how the `ereg()` function can extract values into the optional array variable `var`.



Caution If you try to escape a character that does not need to be, such as an apostrophe, then the backslash will show up when you output the string.

5.8.2 Regular Expression Functions

PHP is an open source language for producing dynamic web pages.

Regular expressions are nothing more than a sequence or pattern of characters itself. They provide the foundation for pattern-matching functionality.

Using regular expression you can search a particular string inside another string, you can replace one string by another string and you can split a string into many chunks.

PHP has three sets of functions that allow you to work with regular expressions.

The most important set of regex functions start with `preg`. These functions are a PHP wrapper around the PCRE library (Perl-Compatible Regular Expressions). You should use the `preg` functions for all new PHP code that uses regular expressions. PHP includes PCRE by default as of PHP 4.2.0 (April 2002).

The oldest set of regex functions are those that start with `ereg`. They implement POSIX Extended Regular Expressions, like the traditional UNIX `egrep` command. These functions are mainly for backward compatibility with PHP 3, and officially deprecated as of PHP 5.3.0. Many of the more modern regex features such as lazy quantifiers, look around and Unicode are not supported by the `ereg` functions. Don't let the "extended" moniker fool you. The POSIX standard was defined in 1986, and regular expressions have come a long way since then.

The last set is a variant of the `ereg` set, prefixing `mb_` for "multibyte" to the function names. While `ereg` treats the regex and subject string as a series of 8-bit characters, `mb_ereg` can work with multi-byte characters from various code pages. If you want your regex to treat Far East characters as individual characters, you'll either need to use the `mb_ereg` functions, or the `preg` functions with the `/u` modifier. `mb_ereg` is available in PHP 4.2.0 and later. It uses the same POSIX ERE flavor.

Self Assessment

Fill in the blanks:

17.returns true if the regular expression pattern is found in the subject string.
18. PHP is an open source language for producing dynamic

Notes



Case Study

String Cheese Incident (Click-and-Mortar)

The music industry's traditional model is falling apart. Revenues have been sliding downward for years now and many bands (along with the industry powerhouses) have chosen to fight the trend to "on demand" or "music by track" choice rather than embrace it. So with all this going on, and most bands crying the blues (no pun intended :-)) – how did a jazzy improv/Grateful Dead sounding band from Boulder Colorado pull in \$14.5 million last year? What you learn from their success can help you blend on line and off line channels in your small business for incredible growth.

They did it through a solid Click-and-Mortar approach. Driving off-line consumers online to make purchases, and allowing online consumers to become fans of the band without ever seeing them in person.

The band has been touring nonstop for 11 years. The devoted following that they have developed have definitely contributed to this outstanding annual cash flow. But the money only really started rolling in during the last four years when they implemented their Click-and-Mortar growth strategy.

From 1999 to 2004, when the rest of the music industry suffered steep declines in revenues, String Cheese Incident watched their annual revenue rise from just about \$2 million per year – to 14.5 million! Not bad for five guys who love what they do and play relatively small venues.

Client-Centric Service with Click-and-Mortar Touch

String Cheese Incident recently battled Ticket master in a lawsuit and gained the right to sell tickets to their own shows on their website. This further reinforced or credibility with fans and gained them more loyalty with their followers by offering tickets for their shows at 10% below Ticket master rates. With 50% of their revenue coming from their tour dates, this has had a huge impact on their bottom line.

They also borrowed a page from the playbook of the Grateful Dead in the arena of recordings. Where most bands jealously guard every single note that emanates from their amplifiers, String Cheese Incident took the Grateful Dead policy of permitted taping of live performances to a whole new level. Just recently the band started selling downloads of its live concerts through a dedicated website (sciontheroad.com) for about \$10 per show.

Click-and-Mortar Marketing at its Best

Anyone it is ever been to a concert before knows that the big bucks lie in the merchandise and CDs for sale at the venue. But how many concert-goers do not have the cash in hand, or desire to fight the crowd to wait in line? By driving their off-line fans to their web site they have racked up some pretty impressive numbers. CD sales rack up \$2.9 million per year. Merchandising and ticket sales another \$2.9 million per year. That's another \$5.8 million that smaller bands never see. And String Cheese Incident is not content with the traditional profit centres associated with bands either. They have taken Click-and-Mortar growth to a whole new level. Recently they saw an unmet need in the marketplace and set up a travel agency with a partner that helps fans plan trips for SCI and 20 other bands on the road. That little "side business" is good for another \$1.45 million per year – all of which is done online! If a "Grateful Dead style" band whose fans thrive on live experiential

Contd...

concerts can use the Click-and-Mortar approach to create a 725% increase in gross revenues, what could you do with your small business?

Coaching Corner:

- How could you expand, extend, or multiply the benefits that your off-line clients experience by bringing them online?
- How could you make your clients' lives easier (thereby increasing your goodwill with them and creating raving fans) through online ordering, scheduling, or organization?
- *Primarily off-line right now ...* what products/services could you offer online that would complement your off-line products/services?
- *Primarily online right now ...* what physical product/service could you offer your clients (either yourself or through a strategic partner) that would add incredible value and lock your clients in for life?

When you take the time to develop a solid Click-and-Mortar growth strategy for your small business, you will be able to dominate your market niche no matter how badly your competition is floundering around.

Questions

1. Explain Client-Centric Service with Click-and-Mortar touch.
2. What is the Client-Centric Service within Click-and-Mortar touch?

5.9 Summary

- The string in PHP is implemented as an array of bytes and an integer indicating the length of the buffer. It has no information about how those bytes translate to characters, leaving that task to the programmer.
- When you define a string literal using double quotes or a heredoc, the string is subject to variable interpolation. Interpolation is the process of replacing variable names in the string with the values of those variables. There are two ways to interpolate variables into strings—the simple way and the complex way.
- Single-quoted strings do not interpolate variables. Double-quoted strings interpolate variables and expand the many PHP escape sequences.
- The `print()` function sends one value (its argument) to the browser. It returns true if the string was successfully displayed and false otherwise. The `printf()` function outputs a string built by substituting values into a template (the format string). It is derived from the function of the same name in the standard C library.
- PHP has several functions for changing the case of strings: `strtolower()` and `strtoupper()` operate on entire strings, `ucfirst()` operates only on the first character of the string, and `ucwords()` operates on the first character of each word in the string.
- Most database systems require that string literals in your SQL queries be escaped. SQL's encoding scheme is pretty simple—single quotes, double quotes, NUL-bytes, and backslashes need to be preceded by a backslash.
- Data often arrives as strings, which must be broken down into an array of values. For instance, you might want to separate out the comma-separated fields from a string such as "Ank, 25, Amr".

5.10 Keywords

Interpolation: Interpolation is the process of replacing variable names in the string with the values of those variables.

print (): The print () function sends one value (its argument) to the browser. It returns true if the string was successfully displayed and false otherwise.

printf(): The printf() function outputs a string built by substituting values into a template (the format string).

Regular Expression: A regular expression follows a strict syntax to describe patterns of characters.

strpos(): The strpos() function finds the first occurrence of a small string in a larger string.

5.11 Review Questions

1. What is the meaning of string? How does it use in PHP?
2. How do we quote the string constants in PHP? Explain with example.
3. What are the heredocuments? Why does it used?
4. What we do to print strings? Explain the functions.
5. What are the format modifiers?
6. What are the type specifiers? Discuss the printf () type specifiers.
7. How do we access the individual character?
8. How do we clean the strings? Which functions are used?
9. How do we compare the strings? Explain with example.
10. How do we perform manipulation and searching on strings?
11. Explain the regular expressions used in PHP.

Answers: Self Assessment

- | | |
|---------------|------------------|
| 1. False | 2. False |
| 3. True | 4. Four |
| 5. Printf() | 6. Print_r() |
| 7. True | 8. True |
| 9. Programmer | 10. Strtoupper() |
| 11. True | 12. False |
| 13. Two | 14. Positive |
| 15. True | 16. False |
| 17. Ereg() | 18. Web pages |

5.12 Further Readings

Notes



Books

- Bangia, Ramesh (2008). *“Web Technology (including HTML, CSS, XML, ASP, JAVA).”* Firewall Media.
- Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.
- Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.
- Puntambekar, A.A. (2009). *“Web Technologies.”* Technical Publications.
- Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.
- Xavier, C. (2007). *“Web Technology and Design.”* New Age International.



Online links

- <http://www.w3resource.com/php/data-types/strings.php>
- http://php.about.com/od/phpfunctions/g/print_r_php.htm
- <http://www.tuxradar.com/practicalphp/4/7/17>
- <http://www.totallyphp.co.uk/remove-whitespace-from-a-text-string-using-the-php-trim-function>
- http://www.brainbell.com/tutorials/php/Comparing_Strings.htm
- http://www.ehow.com/info_8697467_php-string-comparison.html
- <http://php.virginmedia.com/manual/en/function.sscanf.php>
- <http://www.tizag.com/phpT/php-string-explode.php>
- <http://www.regular-expressions.info/php.html>

Unit 6: Arrays

CONTENTS

Objectives

Introduction

6.1 Meaning of Array

6.2 Indexed versus Associative Arrays

6.2.1 Associative Arrays

6.2.2 Indexed Arrays

6.3 Identifying Elements of an Array

6.4 Storing Data in Arrays

6.4.1 Adding Values to the End of an Array

6.4.2 Assigning a Range of Values

6.4.3 Getting the Size of an Array

6.4.4 Padding an Array

6.5 Array Operations in PHP

6.5.1 Split an Array into Chunks

6.5.2 Combine an Array with Data Elements and the Other with its Keys

6.5.3 Merging Two or More Arrays

6.5.4 Searching for a Value in an Array

6.5.5 Sorting Arrays

6.6 Working with Array

6.7 Summary

6.8 Keywords

6.9 Review Questions

6.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain an Array
- Discuss Indexed & Associative Arrays
- Know How to Storing Data in Arrays
- Understand Array Operations in PHP
- Work with Array

Introduction

Notes

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier. That means that, for example, we can store 5 values of type `int` in an array without having to declare 5 different variables, each one with a different identifier. Instead of that, using an array we can store 5 different values of the same type, `int` for example, with a unique identifier. When declaring a regular array of local scope (within a function, for example), if we do not specify otherwise, its elements will not be initialized to any value by default, so their content will be undetermined until we store some value in them. The elements of global and static arrays, on the other hand, are automatically initialized with their default values, which for all fundamental types this means they are filled with zeros. In both cases, local and global, when we declare an array, we have the possibility to assign initial values to each one of its elements by enclosing the values in braces `{ }`.

6.1 Meaning of Array

An array in PHP is actually an ordered map. A map is a type that associates values to keys. This type is optimized for several different uses; it can be treated as an array, list (vector), hash table (an implementation of a map), dictionary, collection, stack, queue, and probably more. As array values can be other arrays, trees and multidimensional arrays are also possible.

An array is a collection of variables indexed and bundled into a single, easily referenced super variable that offers an easy way to pass multiple values between lines of code, functions, and even pages. Throughout much of this chapter, we will be looking at the inner workings of arrays and exploring all the built-in PHP functions that manipulate them. Before we get too deep into that, however, it's worth listing the common ways that arrays are used in real PHP code.

Many built-in PHP environment variables are in the form of arrays (for example, `$_SESSION`, which contains all the variable names and values being propagated from page to page via PHP's session mechanism). If you want access to them, you need to understand, at a minimum, how to reference arrays.

Almost any situation that calls for a number of pieces of data to be packaged and handled as one is appropriate for a PHP array.

PHP arrays are associative arrays with a little extra machinery thrown in. The associative part means that arrays store element values in association with key values rather than in a strict linear index order. (If you have seen arrays in other programming languages, they are likely to have been vector arrays rather than associative arrays — see the related sidebar for an explanation of the difference.) If you store an element in an array, in association with a key, all you need to retrieve it later from that array is the key value. For example, storage is as simple as this:

```
$state_location['San Mateo'] = 'California';
```

which stores the element 'California' in the array variable `$state_location`, in association with the lookup key 'San Mateo'. After this has been stored, you can look up the stored value by using the key, like so:

```
$state = $state_location['San Mateo']; // equals 'California'
```

Simple, no?

If all you want arrays for is to store key/value pairs, the preceding information is all you need to know. Similarly, if you want to associate a numerical ordering with a bunch of values, all you have to do is use integers as your key values, as in:

Notes



Example:

```
$my_array[1] = "The first thing";
$my_array[2] = "The second thing"; // and so on ...
```



Notes For Perl programmers: Arrays in PHP are much like hashes in Perl, with some syntactic differences. For one thing, all variables in PHP are denoted with a leading \$, not just scalar variables. Second, even though the array is associative, the indices are grouped by square brackets ([]) rather than curly braces ({}). Finally, there is no array or list type indexed only by integers. The convention is to use integers as associative indices, and the array itself maintains an internal ordering for iteration purposes.

In addition to the machinery that makes this kind of key/value association possible, arrays track some other things behind the scenes. Because of this, we sometimes treat them as other kinds of data structures. As you will see, arrays can be multidimensional. They can store values in association with a sequence of key values rather than a single key. Also, arrays automatically maintain an ordered list of the elements that have been inserted in them, independent of what the key values happen to be. This makes it possible to treat arrays as linked lists. In general, we will reveal the workings of this extra machinery as we explore the functions that use it.



Notes A note for C++ programmers: You should be aware that arrays can handle some of the same tasks that require the use of template libraries in C++. Much of the reason for having templates in the first place is to get around restrictions having to do with strict typing of data. PHP's looser typing system makes it possible, for example, to write general algorithms that iterate over the contents of arrays without committing to the type of the array elements themselves.

In PHP, there are three types of arrays:

- Indexed arrays - Arrays with numeric index
- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

Self Assessment

State whether the following statements are true or false:

1. A map is a type that associates values to keys.
2. PHP arrays are not an associative arrays.
3. An array in PHP is actually an ordered map.

6.2 Indexed versus Associative Arrays

The keys of an indexed array are integers, beginning at 0. Indexed arrays are used when you identify things by their position. Associative arrays have strings as keys and behave more like two-column tables. The first column is the key, which is used to access the value.

PHP internally stores all arrays as associative arrays, so the only difference between associative and indexed arrays is what the keys happen to be. Some array features are provided mainly for use with indexed arrays; because they assume that you have or want keys that are consecutive integers beginning at 0. In both cases, the keys are unique, that is you cannot have two elements with the same key, regardless of whether the key is a string or an integer.

PHP arrays have an internal order to their elements that is independent of the keys and values, and there are functions that you can use to traverse the arrays based on this internal order. The order is normally that in which values were inserted into the array.



Did u know? PHP arrays have an internal order to their elements that is independent of the keys and values, and there are functions that you can use to traverse the arrays based on this internal order. The order is normally that in which values were inserted into the array.

6.2.1 Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
```

or:

```
$age['Peter']="35";
$age['Ben']="37";
$age['Joe']="43";
```

The named keys can then be used in a script:



Example:

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

6.2.2 Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0):

```
$cars=array("Volvo","BMW","Toyota");
```

or the index can be assigned manually:

```
$cars[0]="Volvo";
$cars[1]="BMW";
$cars[2]="Toyota";
```

The following example creates an indexed array named \$cars, assigns three elements to it, and then prints a text containing the array values:



Example:

```
<?php
$cars=array("Volvo","BMW","Toyota");
```

Notes

```
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```



Did u know? PHP’s string parsing is pretty clever, you can confuse it. If you are having trouble with arrays or other variables not being interpreted correctly when embedded in a double-quoted string, you can put them outside quotes.

Self Assessment

Fill in the blanks:

4.arrays have strings as keys and behave more like two-column tables.
5. The keys of an indexed array are....., beginning at 0.
6. There areways to create indexed arrays.

6.3 Identifying Elements of an Array

You can access specific values from an array using the array variable’s name, followed by the element’s key (sometimes called the index) within square brackets:

```
$age['Fred']
$shows[2]
```

The key can be either a string or an integer. String values that are equivalent to integer numbers (without leading zeros) are treated as integers. Thus, \$array[3] and \$array['3'] reference the same element, but \$array['03'] references a different element. Negative numbers are valid keys, and they don’t specify positions from the end of the array as they do in Perl.

You don’t have to quote single-word strings. For instance, \$age['Fred'] is the same as \$age[Fred]. However, it’s considered good PHP style to always use quotes, because quoteless keys are indistinguishable from constants. When you use a constant as an unquoted index, PHP uses the value of the constant as the index:

```
define('index', 5);
echo $array[index]; // retrieves $array[5], not $array['index'];
```

You must use quotes if you’re using interpolation to build the array index:

```
$age["Clone$number"]
```

However, don’t quote the key if you’re interpolating an array lookup:



Example:

```
// these are wrong
print "Hello, $person['name']";
print "Hello, $person["name"]";
// this is right
print "Hello, $person[name]";
```



Task Develop a program for finding element of array.

Self Assessment

Notes

State whether the following statements are true or false:

7. String values that are equivalent to integer numbers are not treated as integers.
8. Positive numbers are valid keys.
9. You can access specific values from an array using the array variable's name.

6.4 Storing Data in Arrays

When we storing a value in an array it will create the array if it didn't already exist, but trying to retrieve a value from an array that hasn't been defined yet won't create the array. For example:

```
// $postcode not defined before this point
echo $postcode[0];                // prints nothing
echo $postcode;                   // prints nothing
$postcode[0] = 'spam@fresherscloud.net';
echo $postcode;                   // prints "Array"
```

Using simple assignment to initialize an array in your program leads to code like this:



Example:

```
$postcode[0] = 'spam@fresherscloud.net';
$postcode[1] = 'abuse@example.com';
$postcode[2] = 'root@example.com';
// ...
```

That's an indexed array, with integer indexes beginning at 0. Here's an associative array:



Example:

```
$price['Gasket'] = 15.29;
$price['Wheel']  = 75.25;
$price['Tire']   = 50.00;
// ...
```

An easier way to initialize an array is to use the `array()` construct, which builds an array from its arguments:

```
$postcode = array('spam@fresherscloud.net', 'abuse@example.com',
                 'root@example.com');
```

To create an associative array with `array()`, use the `=>` symbol to separate indexes from values:

```
$price = array('Gasket' => 15.29,
              'Wheel'   => 75.25,
              'Tire'    => 50.00);
```

Notice the use of whitespace and alignment. We could have bunched up the code, but it wouldn't have been as easy to read:

```
$price = array('Gasket'=>15.29,'Wheel'=>75.25,'Tire'=>50.00);
```

To construct an empty array, pass no arguments to `array()`:

```
$postcode = array( );
```

Notes

You can specify an initial key with => and then a list of values. The values are inserted into the array starting with that key, with subsequent values having sequential keys:

```
$days = array(1 => 'Monday', 'Tuesday', 'Wednesday',  
              'Thursday', 'Friday', 'Saturday', 'Sunday');  
  
// 2 is Tuesday, 3 is Wednesday, etc.
```

If the initial index is a non-numeric string, subsequent indexes are integers beginning at 0. Thus, the following code is probably a mistake:

```
$whoops = array('Friday' => 'Black', 'Brown', 'Green');  
  
// same as  
$whoops = array('Friday' => 'Black', 0 => 'Brown', 1 => 'Green');
```

6.4.1 Adding Values to the End of an Array

The [] syntax, is used to insert more values into the end of an existing indexed array.



Example:

```
$friend = array('Harish', 'Pinky');  
$friend[] = 'Tushar'; // $friend[2] is 'Tushar'
```

This construct assumes the array's indexes are numbers and assigns elements into the next available numeric index, starting from 0. Attempting to append to an associative array is almost always a programmer mistake, but PHP will give the new elements numeric indexes without issuing a warning:



Example:

```
$person = array('name' => 'lokendra');  
$person[] = 'Wimla'; // $person[0] is now 'Wimla'
```

6.4.2 Assigning a Range of Values

The range() function creates an array of consecutive integer or character values between the two values you pass to it as arguments. For example:



Example:

```
$numbers = range(5, 8); // $numbers = array(5, 6, 7, 8);  
$letters = range('A', 'Z'); // $numbers holds the alphabet  
$reversed_numbers = range(8, 5); // $numbers = array(8, 7, 6, 5);
```

Only the first letter of a string argument is used to build the range:

```
range('AAA', 'ZZZ') // same as range('A','Z')
```

6.4.3 Getting the Size of an Array

The count() and sizeof() functions are identical in use and effect. They return the number of elements in the array. There is no stylistic preference about which function you use. Here's an example:



Example:

```
$family = array('Fred', 'Wilma', 'Pebbles');
$size = count($family);           // $size is 3
```

These functions do not consult any numeric indexes that might be present:

```
$confusion = array( 10 => 'ten', 11 => 'eleven', 12 => 'twelve');
$size = count($confusion);       // $size is 3
```

6.4.4 Padding an Array

To create an array initialized to the same value, use `array_pad()`. The first argument to `array_pad()` is the array, the second argument is the minimum number of elements you want the array to have, and the third argument is the value to give any elements that are created. The `array_pad()` function returns a new padded array, leaving its argument array alone.

Here's `array_pad()` in action:



Example:

```
$scores = array(5, 10);
$padding = array_pad($scores, 5, 0); // $padding is now array(5, 10, 0, 0, 0)
```

Notice how the new values are appended to the end of the array. If you want the new values added to the start of the array, use a negative second argument:

```
$padding = array_pad($scores, -5, 0);
```

Assign the results of `array_pad()` back to the original array to get the effect of an in situ change:

```
$scores = array_pad($scores, 5, 0);
```

If you pad an associative array, existing keys will be preserved. New elements will have numeric keys starting at 0.

Self Assessment

Fill in the blanks:

10. The, is used to insert more values into the end of an existing indexed array.
11. The function creates an array of consecutive integer or character values between the two values you pass to it as arguments.
12. The `count()` and `sizeof()` functions arein use and effect.

6.5 Array Operations in PHP

Here we will discuss some common function of arrays with the help of examples.

6.5.1 Split an Array into Chunks

The `array_chunk()` PHP array function will chunk an array at a specified size. It takes up to three parameters. The first parameter we feed it is the target array. The second parameter is the size you wish your chunks to be. And the third parameter is optional. Parameter three uses values of either "true" or "false", when used you can choose to preserve existing keys in the array. If you

Notes

do not use the third parameter this function defaults to re-indexing the array just as if you specify “false” for the third parameter. Specify a value of “true” as the third parameter if you wish to preserve the keys in the array, and use “false” (or omit param 3) to re-index the array.

Each chunk created will be its own array since the `array_chunk()` function will output a multidimensional array. We will use code below to process a multidimensional array to show the result of using this function.



Example:

```
<?php
$myArray = array("abc","def","ghi","jkl","mno","pqr","stu","vwx","yz");
$newArray = array_chunk($myArray, 2, false);
//Now process the multidimensional array made from array chunk()
$i = 0;
foreach ($newArray as $inner_array) {
    $i++;
    echo "<h2>Chunk $i</h2>";
    while (list($key, $value) = each($inner_array)) {
        echo "$key: $value <br />";
    }
}
?>
```

Output:

Chunk 1

0: abc
1: def

Chunk 2

0: ghi
1: jkl

Chunk 3

0: mno
1: pqr

Chunk 4

0: stu
1: vwx

Chunk 5

0: yz

6.5.2 Combine an Array with Data Elements and the Other with its Keys

An array can create by combining one array with keys and a second array with corresponding data elements. Here it can be note that the number of keys and data elements in both the arrays has to be equal for this operation to be successful. We will make use of built-in function `array_combine()`.

Its syntax is:

```
array_combine( array keys, array values )
```

and the example using array_combine() is:



Example:

```
<?php
$a = array('green', 'red', 'yellow');
$b = array('avocado', 'apple', 'banana');
$c = array_combine($a, $b);
print_r($c);
?>
```

Output:

```
Array
(
    [green] =>avocado
    [red]   =>apple
    [yellow] =>banana
)
```

6.5.3 Merging Two or More Arrays

```
array array_merge ( array $array1 [, array $array2 [, array $... ]] )
```

Merges the elements of one or more arrays together so that the values of one are appended to the end of the previous one. It returns the resulting array.

If the input arrays have the same string keys, then the later value for that key will overwrite the previous one. If, however, the arrays contain numeric keys, the later value will not overwrite the original value, but will be appended.

If all of the arrays contain only numeric keys, the resulting array is given incrementing keys starting from zero.



Example:

```
array_merge() example
<?php
$array1 = array("color" => "red", 2, 4);
$array2 = array("a", "b", "color" => "green", "shape" => "trapezoid",
4);
$result = array_merge($array1, $array2);
print_r($result);
?>
```

The above example will output:

```
Array
(
    [color] =>green
    [0] => 2
    [1] => 4
    [2] => a
)
```

Notes

```
[3] => b
[shape] =>trapezoid
[4] => 4
)
```



Example:

Simple array_merge() example

```
<?php
$array1 = array();
$array2 = array(1 => "data");
$result = array_merge($array1, $array2);
?>
```

Don't forget that numeric keys will be renumbered!

```
Array
(
    [0] => data
)
```

If you want to append array elements from the second array to the first array while not overwriting the elements from the first array and not re-indexing, use the + array union operator:

```
<?php
$array1 = array(0 => 'zero_a', 2 => 'two_a', 3 => 'three_a');
$array2 = array(1 => 'one_b', 3 => 'three_b', 4 => 'four_b');
$result = $array1 + $array2;
var_dump($result);
?>
```

The keys from the first array will be preserved. If an array key exists in both arrays, then the element from the first array will be used and the matching key's element from the second array will be ignored.

```
array(5) {
    [0]=>
    string(6) "zero_a"
    [2]=>
    string(5) "two_a"
    [3]=>
    string(7) "three_a"
    [1]=>
    string(5) "one_b"
    [4]=>
    string(6) "four_b"
}
```

6.5.4 Searching for a Value in an Array

Searching for a value in an array is made simple using the function array_search(). If the keyword is found in the array, then the corresponding key of that value is returned for further operations.

The syntax and example are:

```
array_search( keyword, array name )
```



Example:

```
<?php
$days = array(1=>"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun");
$key = array_search("Wed", $days); //Searching for Wed in days array
echo $key;
// We can also print the value that key points at
Echo $days[$key];
?>
```



Caution Must be understood that the sort function is case-sensitive. All capital letters come before all lowercase letters. So "A" is less than "Z", but "Z" is less than "a".

6.5.5 Sorting Arrays

Sorting changes the internal order of elements in an array and optionally rewrites the keys to reflect this new order. For example, you might use sorting to arrange a list of scores from biggest to smallest, to alphabetize a list of names, or to order a set of users based on how many messages they posted.

PHP provides three ways to sort arrays—sorting by keys, sorting by values without changing the keys, or sorting by values and then changing the keys. Each kind of sort can be done in ascending order, descending order, or an order defined by a user-defined function.

PHP Functions for Sorting an Array

Effect	Ascending	Descending	User-defined Order
Sort array by values, then reassign indexes starting with 0	sort()	rsort()	usort()
Sort array by values	asort()	arsort()	uasort()
Sort array by keys	ksort()	krsort()	uksort()

The sort(), rsort(), and usort() functions are designed to work on indexed arrays, because they assign new numeric keys to represent the ordering. They're useful when you need to answer questions like "what are the top 10 scores?" and "who's the third person in alphabetical order?" The other sort functions can be used on indexed arrays, but you'll only be able to access the sorted ordering by using traversal functions such as foreach and next.

To sort names into ascending alphabetical order, you'd use this:

```
$names = array('cath', 'angela', 'brad', 'dave');
sort($names); // $names is now 'angela', 'brad', 'cath', 'dave'
```

To get them in reverse alphabetic order, simply call rsort() instead of sort().

If you have an associative array mapping usernames to minutes of login time, you can use arsort() to display a table of the top three, as shown here:

Notes*Example:*

```
$logins = array('njt' => 415,
               'kt'  => 492,
               'rl'  => 652,
               'jht' => 441,
               'jj'  => 441,
               'wt'  => 402);

arsort($logins);
$num_printed = 0;
echo("<table>\n");
foreach ($logins as $user => $time ) {
echo("<tr><td>$user</td><td>$time</td></tr>\n");
if (++$num_printed == 3) {
break;                // stop after three
}
}
echo("</table>\n");
<table>
<tr><td>rl</td><td>652</td></tr>
<tr><td>kt</td><td>492</td></tr>
<tr><td>jht</td><td>441</td></tr>
</table>
```

Self Assessment

State whether the following statements are true or false:

13. The array_chunk() PHP array function will takes up to two parameters.
14. Each chunk created will be its own array since the array_chunk() function will input a multidimensional array.
15. If all of the arrays contain only numeric keys, the resulting array is given incrementing keys starting from zero.

6.6 Working with Array

Arrays crop up in almost every PHP program. PHP has several useful functions for modifying or applying an operation to all elements of an array. You can merge arrays, find the difference, calculate the total, and more, all using built-in functions. In addition to their obvious use for storing collections of values, they're also used to implement various abstract data types. In this section, we show how to work with the arrays with the help of examples.

Following example is defining and print working of array:

*Example:*

```
<?php
    $Top3Sites = array
("fastcreators.com", "howtoforge.com", "scriptsbible.com");
```



```
print_r($Top3Sites);
?>
```

Output:

```
Array
(
    [0] => fastcreators.com
    [1] => howtoforge.com
    [2] => scriptsbible.com
)
```

In the examples above we have defined all the values within the script one by one, but this assignment may be done in a different way, as it is described in the example bellow:

array3a.php	Resulting page
<pre><pre> <? \$Thearray= array ("zero","one","two","three","four","five","six","seven", "eight","nine") ?> Thearray[0]: <? print \$Thearray[0]; ?> Thearray[1]: <? print \$Thearray[1]; ?> Thearray[2]: <? print \$Thearray[2]; ?> Thearray[3]: <? print \$Thearray[3]; ?> Thearray[4]: <? print \$Thearray[4]; ?> Thearray[5]: <? print \$Thearray[5]; ?> Thearray[6]: <? print \$Thearray[6]; ?> Thearray[7]: <? print \$Thearray[7]; ?> Thearray[8]: <? print \$Thearray[8]; ?> Thearray[9]: <? print \$Thearray[9]; ?> </pre></pre>	<pre>Thearray(0): Zero Thearray(1): pne Thearray(2): two Thearray(3): three Thearray(4): four Thearray(5): five Thearray(6): six Thearray(7): seven Thearray(8): eight Thearray(9): nine</pre>

In this example the array has been defined as comma separated element (the first element in the array is element 0 !). The array command has been used to define the array.

We may want to generate an array from a data in a text. In the example bellow, each word in a text will be stored in an array (one word in each element of the array), and then the array will be printed out (with command `print_r`), and finally word number 5 will be shown:

array3b.php	Resulting page
<pre><pre> <? \$TheText="zero one two three four five six seven eight nine"; \$Thearray=split (" ",\$TheText); print_r (\$Thearray); ?> <hr> The element number 5 is : <? print \$Thearray[5]; ?> </pre></pre>	<pre>Array ([0] => zero [1] => one [2] => two [3] => three [4] => four [5] => five [6] => six [7] => seven [8] => eight [9] => nine)</pre>
	The element number 5 is : five

In this example we have defined the variable `$TheText`, and within this variable we have include several words separated by spaces.

In the next line, we have split the variable `$TheText` into an array named `$Thearray`. Split command have been used to brake `$TheText` and " " (space) has been used as a delimiter to separate the substrings.

In the response page we have printed the array by using command `print_r`, and element number 5 has been printed out.

It may happened to have a string we want to split, but we do not know how many substrings we may get. In that case we may use command `sizeof` to discover how many elements are in our

Notes


array, and then we may use that value to write them by using a foreach control structure (see example below).

array4.php	Resulting page
<pre> <pre> <? \$TheText="my dog is very nice and my cat is barking"; \$Thearray=split (" ",\$TheText) ; ?> How many words do I have in \$TheArray? <? print sizeof (\$Thearray); ?> <? Foreach (\$Thearray as \$key =>\$val){ print "
Word number \$key is \$val"; } ?> </pre> </pre>	<p>How many words do I have in \$TheArray? 10</p> <p>Word number 0 is my Word number 1 is dog Word number 2 is is Word number 3 is very Word number 4 is nice Word number 5 is and Word number 6 is my Word number 7 is cat Word number 8 is is Word number 9 is barking</p>

Self Assessment

Fill in the blanks:

- Arrays crop up in almost everyprogram.
- PHP has several useful functions for modifying or applying anto all elements of an array



Case Study

Binary Tree Branches out Big with Microsoft

New York-based Binary Tree started in 1993 helping businesses migrate to Lotus Notes. Ten years ago, its customers started requesting solutions for migrating to Microsoft. Driven by market demand and supported by a solid Microsoft product offering and partner ecosystem, Binary Tree evolved into a platform agnostic software company and developed new solutions for providing migrations to Microsoft. Today, with thousands of customers and millions of users migrated, Binary Tree’s decision has yielded impressive returns: More than half of its annual revenue comes from Microsoft platform migrations.

Situation

A Turning Tide of Customers When Binary Tree opened its doors in 1993, it helped companies migrate to Lotus Notes. By 1999, Binary Tree was noticing a new trend in the migration market—an increase in demand for migrations to Microsoft. Some of Binary Tree’s former customers who originally migrated to Lotus Notes were now pounding on Binary Tree’s doors to demand a complete overhaul to the Microsoft solution. “Those requests kept escalating until they reached a point where it was obvious that this was a big opportunity for us,” said Steven Pivnik, CEO of Binary Tree. “We’ve learned to listen to our customers and there are a number of reasons why they want to migrate to the Microsoft platform. Many of them cite that the Microsoft platform is maturing substantially and that they are looking for reduced costs, better productivity, and lower costs of ownership.”

Questions

- Give the application of Binary Tree.
- Explain the tree concept in term of simple accessing data.

6.7 Summary

Notes

- Using a built-in function `array_merge()` we can merge two or more arrays to form a single array. The values from the second array are appended at the end of first array and so on.
- Arrays are objects. Therefore, an array must be instantiated before it can be used. An array can be instantiated using the `new` operator or using an initializer.
- An index can be a numerical expression, but the result of the expression must be of integer type. Every array knows its own length. `arrayname.length` can be used to determine the size of the array.
- An array can create by combining one array with keys and a second array with corresponding data elements.
- Merges the elements of one or more arrays together so that the values of one are appended to the end of the previous one.
- Searching for a value in an array is made simple using the function `array_search()`.
- Sorting changes the internal order of elements in an array and optionally rewrites the keys to reflect this new order.

6.8 Keywords

Arrays: An array is a collection of variables indexed and bundled into a single, easily referenced super variable that offers an easy way to pass multiple values between lines of code, functions, and even pages.

Associative Arrays: An associative array, map, or dictionary is an abstract data type composed of a collection of (key, value) pairs, such that each possible key appears at most once in the collection.

Chunks: A chunk is a textual unit of adjacent word tokens which can be mutually linked through unambiguously identified dependency chains with no recourse to idiosyncratic lexical information.

Indexed Arrays: Indexed arrays store a series of one or more values organized such that each value can be accessed using an unsigned integer value.

Merging: Merging in revision control, is a fundamental operation that reconciles multiple changes made to a revision-controlled collection of files.

Sorting: Sorting refers to ordering data in an increasing or decreasing fashion according to some linear relationship among the data items.

6.9 Review Questions

1. What are the indexed and associative arrays? Explain with example.
2. What are the ways to identifying elements of an array?
3. Write a PHP program to search a special element in an array?
4. How do we store the data in arrays?
5. Write a PHP program to adding values at the end of an array.
6. What is the padding of an array? Explain with example.

Notes

7. How do we perform array operations in PHP?
8. Write a PHP program to sort an array.
9. Write a PHP program to merge two arrays.
10. Write the concepts working with array.

Answers: Self Assessment

- | | |
|---------------|----------------|
| 1. True | 2. False |
| 3. True | 4. Associative |
| 5. Integers | 6. Two |
| 7. False | 8. False |
| 9. True | 10. [] syntax |
| 11. Range() | 12. Identical |
| 13. False | 14. False |
| 15. True | 16. PHP |
| 17. Operation | |

6.10 Further Readings



Books

- Bangia, Ramesh (2008). *Web Technology (including HTML, CSS, XML, ASP, JAVA)*. Firewall Media.
- Jackson (2007). *Web Technologies: A Computer Science Perspective*. Pearson Education India.
- Kamal, Raj (2002). *Internet and Web Technologies*. Tata McGraw-Hill Education.
- Puntambekar, A.A. (2009). *Web Technologies*. Technical Publications.
- Sarukkai, Ramiesh R. (2002). *Foundations of Web Technology*. Springer.
- Xavier, C. (2007). *Web Technology and Design*. New Age International.



Online links

- http://www.howtoforge.com/php_arrays
- <http://php.net/manual/en/language.types.array.php>
- <http://www.cplusplus.com/doc/tutorial/arrays/>
- http://www.w3schools.com/php/php_arrays.asp
- <http://docs.oracle.com/javase/6/docs/api/java/util/Arrays.html>
- http://www.w3schools.com/js/js_obj_array.asp

Unit 7: Multidimensional Arrays

Notes

CONTENTS

Objectives

Introduction

- 7.1 Concept of Multidimensional Array
 - 7.1.1 Two-dimensional Arrays
 - 7.1.2 Three-dimensional Arrays
- 7.2 Extracting Multiple Values
 - 7.2.1 Slicing an Array
 - 7.2.2 Splitting an Array into Chunks
 - 7.2.3 Keys and Values
 - 7.2.4 Checking whether an Element Exists
 - 7.2.5 Removing and Inserting Elements in an Array
- 7.3 Converting between Arrays and Variables
 - 7.3.1 Creating Variables from an Array
 - 7.3.2 Creating an Array from Variables
- 7.4 Traversing Arrays
 - 7.4.1 The foreach Construct
 - 7.4.2 The Iterator Functions
 - 7.4.3 Using a for loop
 - 7.4.4 Calling a Function foreach Array Element
 - 7.4.5 Reducing an Array
 - 7.4.6 Searching for Values
- 7.5 Sorting of Multidimensional Arrays
 - 7.5.1 User Defined Sorts
 - 7.5.2 Reverse User Sorts
- 7.6 Acting on Entire Arrays
 - 7.6.1 Merging Two Arrays
 - 7.6.2 Calculating the Sum of an Array
 - 7.6.3 Filtering Elements from an Array
 - 7.6.4 Calculating the difference between Two Arrays
- 7.7 Using Arrays
 - 7.7.1 Stacks
 - 7.7.2 Sets
- 7.8 Summary
- 7.9 Keywords
- 7.10 Review Questions
- 7.11 Further Readings

Notes

Objectives

After studying this unit, you will be able to:

- Understand the Concepts of Multidimensional Array
- Discuss How to Extract the Arrays
- Discuss How to Perform Conversion between Arrays and Variables
- Explain Array Traversal
- Understand the Multidimensional Array Sorting
- Explain How to Perform different Operations on Array
- Discuss the Use of Arrays

Introduction

The value of an array element can be another array. This is useful when you want to store data that has a more complicated structure than just a key and a single value. A standard key/value pair is fine for matching up a meal name (such as breakfast or lunch) with a single dish but what about when each meal consists of more than one dish? Then, element values should be arrays, not strings.

7.1 Concept of Multidimensional Array

A multidimensional array is an array containing one or more arrays.

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.



Example:

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
(
    "Griffin"=>array
    (
        "Peter",
        "Lois",
        "Megan"
    ),
    "Quagmire"=>array
    (
        "Glenn"
    ),
    "Brown"=>array
    (
        "Cleveland",
        "Loretta",
        "Junior"
    )
);
```

7.1.1 Two-dimensional Arrays

Notes

Let us suppose that you have to maintain the record of a flower shop. One-dimensional array is enough to keep titles and prices. But if you want to keep more than one item of each type you need to use something different – one of the ways to do it is using multidimensional arrays. The table below depicts two-dimensional array. Each row represents a type of flower and each column – a certain attribute.

Table 7.1

Title	Price	Number
Rose	1.25	15
Daisy	0.75	25
Orchid	1.15	7

To store data in form of array represented by preceding example using PHP, let's prepare the following code:



Example:

```
<?php
$shop = array( array("rose", 1.25 , 15),
              array("daisy", 0.75 , 25),
              array("orchid", 1.15 , 7)
            );
?>
```

This example shows that now \$shop array, in fact, contains three arrays. As you remember, to access data in one-dimensional array you have to point to array name and index. The same is true in regards to a two-dimensional array, with one exception: each element has two indexes – row and column.

To display elements of this array we could have organized manual access to each element or make it by putting For loop inside another For loop:



Example:

```
<?php
echo "<h1>Manual access to each element</h1>";

echo $shop[0][0]." costs ".$shop[0][1]." and you get ".$shop[0][2]."<br />";
echo $shop[1][0]." costs ".$shop[1][1]." and you get ".$shop[1][2]."<br />";
echo $shop[2][0]." costs ".$shop[2][1]." and you get ".$shop[2][2]."<br />";

echo "<h1>Using loops to display array elements</h1>";

echo "<ol>";
for ($row = 0; $row < 3; $row++)
{
    echo "<li><b>The row number $row</b>";
}
```

Notes

```
echo "<ul>";

for ($col = 0; $col < 3; $col++)
{
    echo "<li>".$shop[$row][$col]."</li>";
}

echo "</ul>";
echo "</li>";
}
echo "</ol>";
?>
```

Perhaps, instead of the column numbers you prefer to create their names. For this purpose, you can use associative arrays. The following code will store the same set of flowers using column names:



Example:

```
<?php
$shop = array( array( Title => "rose",
                    Price => 1.25,
                    Number => 15
                ),
              array( Title => "daisy",
                    Price => 0.75,
                    Number => 25,
                ),
              array( Title => "orchid",
                    Price => 1.15,
                    Number => 7
                )
            );
?>
```

It is easier to work with this array, in case you need to get a single value out of it. Necessary data can be easily found, if you turn to a proper cell using meaningful row and column names that bear logical content. However, we are losing the possibility to use simple for loop to view all columns consecutively.

You can view outer numerically indexed \$shop array using the for loop. Each row of the \$shop array is an associative array. Hence, inside the for loop you need for each loop. Also you can get each element from associative array manually:



Example:

```
<?php
echo "<h1>Manual access to each element from associative array</h1>";

for ($row = 0; $row < 3; $row++)
{
    echo $shop[$row]["Title"]." costs ".$shop[$row]["Price"]." and you
get ".$shop[$row]["Number"];
}
```



```

    echo "<br />";
}

echo "<h1>Using foreach loop to display elements</h1>";

echo "<ol>";
for ($row = 0; $row < 3; $row++)
{
    echo "<li><b>The row number $row</b>";
    echo "<ul>";

    foreach($shop[$row] as $key => $value)
    {
        echo "<li>".$value."</li>";
    }

    echo "</ul>";
    echo "</li>";
}
echo "</ol>";
?>

```

7.1.2 Three-dimensional Arrays

Three-dimensional array is characterized by height, width and depth. If you feel comfortable to imagine two-dimensional array as a table, then imagine a pile of such tables. Each element can be referenced by its layer, row and column.

If we classify flowers in our shop into categories, then we can keep data on them using three-dimensional array. We can see from the code below, that three-dimensional array is an array containing array of arrays.



Example:

```

<?php
$shop = array(array(array("rose", 1.25, 15),
                    array("daisy", 0.75, 25),
                    array("orchid", 1.15, 7)
                ),
              array(array("rose", 1.25, 15),
                    array("daisy", 0.75, 25),
                    array("orchid", 1.15, 7)
                ),
              array(array("rose", 1.25, 15),
                    array("daisy", 0.75, 25),
                    array("orchid", 1.15, 7)
                )
            );
?>

```

Notes

As this array has only numeric indexes, we can use nested for loops to display it:



Example:

```
<?php
echo "<ul>";
for ( $layer = 0; $layer < 3; $layer++ )
{
    echo "<li>The layer number $layer";
    echo "<ul>";

    for ( $row = 0; $row < 3; $row++ )
    {
        echo "<li>The row number $row";
        echo "<ul>";

        for ( $col = 0; $col < 3; $col++ )
        {
            echo "<li>".$shop[$layer][$row][$col]."</li>";
        }
        echo "</ul>";
        echo "</li>";
    }
    echo "</ul>";
    echo "</li>";
}
echo "</ul>";
?>
```

This way of creating multidimensional arrays allows to create four- and five-dimensional arrays. Syntax rules do not limit the number of dimensions, but the majority of practical tasks logically correspond to the constructions of three or less dimensions.

Self Assessment

State whether the following statements are true or false:

1. Two-dimensional array is enough to keep titles and prices.
2. Each element in the main array can also be an array in a multidimensional array.
3. Three-dimensional array is characterized by height, width, and depth.

7.2 Extracting Multiple Values

To copy all of an array's values into variables, use the `list()` construct:

```
list($variable, ...) = $array;
```

The array's values are copied into the listed variables, in the array's internal order. By default that's the order in which they were inserted, but the sort functions described later let you change that. Here's an example:

```
$person = array('name' => 'Fred', 'age' => 35, 'wife' => 'Betty');
list($n, $a, $w) = $person;           // $n is 'Fred', $a is 35, $w is 'Betty'
```

Notes

If you have more values in the array than in the list(), the extra values are ignored:

```
$person = array('name' => 'Fred', 'age' => 35, 'wife' => 'Betty');
list($n, $a) = $person;           // $n is 'Fred', $a is 35
```

If you have more values in the list() than in the array, the extra values are set to NULL:

```
$values = array('hello', 'world');
list($a, $b, $c) = $values;      // $a is 'hello', $b is 'world', $c
is NULL
```

Two or more consecutive commas in the list() skip values in the array:

```
$values = range('a', 'e');
list($m,, $n,, $o) = $values;    // $m is 'a', $n is 'c', $o is 'e'
```

7.2.1 Slicing an Array

The array_slice() function returns selected parts of an array.

Syntax

```
array_slice(array, start, length, preserve)
```

Table 7.2

Parameter	Description
Array	Required. Specifies an array
Start	Required. Numeric value. Specifies where the function will start the slice. 0 = the first element. If this value is set to a negative number, the function will start slicing that far from the last element. -2 means start at the second last element of the array.
Length	Optional. Numeric value. Specifies the length of the returned array. If this value is set to a negative number, the function will stop slicing that far from the last element. If this value is not set, the function will return all elements, starting from the position set by the start-parameter.
Preserve	Optional. Possible values: <ul style="list-style-type: none"> • true - Preserve keys • false - Default - Reset keys

7.2.2 Splitting an Array into Chunks

To divide an array into smaller, evenly sized arrays, use the array_chunk() function:

```
array array_chunk ( array $input , int $size [, bool $preserve_keys = false
] )
```

The function returns an array of the smaller arrays. The third argument, preserve_keys, is a Boolean value that determines whether the elements of the new arrays have the same keys as in the original (useful for associative arrays) or new numeric keys starting from 0 (useful for indexed arrays). The default is to assign new keys, as shown here:



Example:

```
$nums = range(1, 7);
$rows = array_chunk($nums, 3);
print_r($rows);
Array
```

Notes

```
(
    [0] => Array
        (
            [0] => 1
            [1] => 2
            [2] => 3
        )
    [1] => Array
        (
            [0] => 4
            [1] => 5
            [2] => 6
        )
    [2] => Array
        (
            [0] => 7
        )
)
```

7.2.3 Keys and Values

The `array_keys()` function returns an array consisting of only the keys in the array, in internal order:

```
$array_of_keys = array_keys(array);
```



Example:

```
$person = array('name' => 'amitabh', 'age' => 65, 'wife' => 'jaya');
$keys = array_keys($person); // $keys is array('name', 'age', 'wife')
```

PHP also provides a (less generally useful) function to retrieve an array of just the values in an array, `array_values()`:

```
$array_of_values = array_values(array);
```

As with `array_keys()`, the values are returned in the array's internal order:

```
$values = array_values($person); // $values is array('amitabh', 65, 'jaya');
```

7.2.4 Checking whether an Element Exists

To see if an element exists in the array, use the `array_key_exists()` function:

```
if (array_key_exists(key, array)) { ... }
```

The function returns a Boolean value that indicates whether the second argument is a valid key in the array given as the first argument.

It's not sufficient to simply say:

```
if ($person['name']) { ... } // this can be misleading
```

Even if there is an element in the array with the key name, its corresponding value might be false (i.e., 0, NULL, or the empty string). Instead, use `array_key_exists()` as follows:



Example:

```
$person['age'] = 0; // unborn?
if ($person['age']) {
    echo "true!\n";
}
if (array_key_exists('age', $person)) {
    echo "exists!\n";
}
exists!
```

In PHP 4.0.6 and earlier versions, the `array_key_exists()` function was called `key_exists()`. The original name is still retained as an alias for the new name.

Many people use the `isset()` function instead, which returns true if the element exists and is not NULL:

```
$a = array(0,NULL,'');
function tf($v) { return $v ? "T" : "F"; }
for ($i=0; $i < 4; $i++) {
    printf("%d: %s %s\n", $i, tf(isset($a[$i])), tf(array_key_exists($i,
$a)));
}
0: T T
1: F T
2: T T
3: F F
```

7.2.5 Removing and Inserting Elements in an Array

The `array_splice()` function can remove or insert elements in an array:

```
$removed = array_splice(array, start [, length [, replacement ] ]);
```

We'll look at `array_splice()` using this array:

```
$subjects = array('physics', 'chem', 'math', 'bio', 'cs', 'drama',
'classics');
```

We can remove the math, bio, and cs elements by telling `array_splice()` to start at position 2 and remove 3 elements:

```
$removed = array_splice($subjects, 2, 3); // $removed is array('math',
'bio', 'cs') // $subjects is array('physics', 'chem');
```

If you omit the length, `array_splice()` removes to the end of the array:

```
$removed = array_splice($subjects, 2); // $removed is array('math', 'bio',
'cs', 'drama', 'classics') // $subjects is array('physics', 'chem');
```

If you simply want to delete the elements and you don't care about their values, you don't need to assign the results of `array_splice()`:

```
array_splice($subjects, 2); // $subjects is array('physics', 'chem');
```

Notes

To insert elements where others were removed, use the fourth argument:

```
$new = array('law', 'business', 'IS'); array_splice($subjects, 4, 3, $new);
// $subjects is array('physics', 'chem', 'math', 'bio', 'law', 'business', 'IS')
```

The size of the replacement array doesn't have to be the same as the number of elements you delete. The array grows or shrinks as needed:

```
$new = array('law', 'business', 'IS'); array_splice($subjects, 2, 4, $new);
// $subjects is array('physics', 'chem', 'math', 'law', 'business', 'IS')
```

To get the effect of inserting new elements into the array, delete zero elements:

```
$subjects = array('physics', 'chem', 'math');
$new = array('law', 'business');
array_splice($subjects, 2, 0, $new); // $subjects is array('physics', 'chem', 'law', 'business', 'math')
```

Although the examples so far have used an indexed array, `array_splice()` also works on associative arrays:

```
$capitals = array('India' => 'New Delhi', 'Great Britain' => 'London', 'New Zealand' => 'Wellington', 'Australia' => 'Canberra', 'Italy' => 'Rome');
$down_under = array_splice($capitals, 2, 2); // remove New Zealand and Australia
$france = array('France' => 'Paris');
```

`array_splice($capitals, 1, 0, $france);` // insert France between USA and G.B. Before PHP 4.3.0, appending to an array in which the current maximum key was negative would create a new key. Since PHP 4.3.0, the new key will be 0.

Self Assessment

Fill in the blanks:

4. The array's values are copied into the listed variables, in the array'sorder.
5. In PHP 4.0.6 and earlier versions, the `array_key_exists()` function was called.....

7.3 Converting between Arrays and Variables

PHP provides two functions, `extract()` and `compact()`, that convert between arrays and variables.

```
int extract ( array source [, int extract_type [, string prefix]])
```

`Extract()` is a very popular function that converts elements in an array into variables in their own right. `Extract` takes a minimum of one parameter, an array, and returns the number of elements extracted. This is best explained using code, so here goes:



Example:

```
<?php
    $Wales = 'Swansea';
    $capitalcities['England'] = 'London';
    $capitalcities['Scotland'] = 'Edinburgh';
    $capitalcities['Wales'] = 'Cardiff';
    extract($capitalcities);
    print $Wales;
?>
```

After calling `extract`, the “England”, “Scotland”, and “Wales” keys become variables in their own right (`$England`, `$Scotland`, and `$Wales`), with their values set to “London”, “Edinburgh”, and “Cardiff” respectively. By default, `extract()` will overwrite any existing variables, meaning that `$Wales`’s original value of “Swansea” will be overwritten with “Cardiff”.

This behaviour can be altered using the second parameter, and averted using the third parameter. Parameter two takes a special constant value that allows you to decide how values will be treated if there is an existing variable, and parameter three allows you to prefix each extract variable with a special string. Here are the possible values of the second parameter:

Table 7.3

EXTR_OVERWRITE	On collision, overwrite the existing variable
EXTR_SKIP	On collision, do not overwrite the existing variable
EXTR_PREFIX_SAME	On collision, prefix the variable name with the prefix specified by parameter three
EXTR_IF_EXISTS	Only set variables if they already exist
EXTR_PREFIX_IF_EXISTS	Only create prefixed variables if non-prefixed version already exists
EXTR_REFS	Extract variables as references

The last option, `EXTR_REFS`, can either be used on its own or in combination with others using the bitwise OR operator `|`.

Here are some examples based upon the `$capitalcities` array from the previous example:



Example:

```
<?php
    $Wales = 'Swansea';
    extract($capitalcities, EXTR_SKIP);
    print $Wales;
    print $Scotland;
    extract($capitalcities, EXTR_PREFIX_SAME, "country");
    print $Wales;
    print $country_England;
    extract($capitalcities, EXTR_PREFIX_ALL, "country");
?>
```

On line one we pre-set `$Wales` to be a value so that you can clearly see how the second parameter works. On line two we call `extract()` using two parameters, with `EXTR_SKIP` as parameter two so that our `$Wales` will not be overwritten. However, as you will be able to see if you run the script, `$England` and `$Scotland` were set.

On line five we use `EXTR_PREFIX_SAME`, which will extract variables and use the third parameter as a prefix if it finds any collisions. As we set `$Wales` ourselves and our previous call to `extract()` created `$England` and `$Scotland`, all our variables will need to be prefixed. As you can see on line six, `$Wales` is still set to “Swansea”, and on line seven we have our prefixed variable `$country_England`. Note that PHP places an underscore `_` after your prefix to make the variable easy to read.

Finally we call `extract()` with `EXTR_PREFIX_ALL`, which will unconditionally create variables with prefixes, overwriting `$country_England`, etc. Note that `EXTR_OVERWRITE` is rarely if ever used, because it is the same as using `extract()` without second or third parameters.

7.3.1 Creating Variables from an Array

The `extract()` function automatically creates local variables from an array. The indexes of the array elements are the variable names:

```
extract($person); // $name, $age, and $wife are now set
```

If a variable created by the extraction has the same name as an existing one, the extracted variable overwrites the existing variable.

You can modify `extract()`'s behavior by passing a second argument. The most useful value is `EXTR_PREFIX_SAME`, which says that the third argument to `extract()` is a prefix for the variable names that are created. This helps ensure that you create unique variable names when you use `extract()`. It is good PHP style to always use `EXTR_PREFIX_SAME`, as shown here:



Example:

```
$shape = "round";
$array = array("cover" => "bird", "shape" => "rectangular");
extract($array, EXTR_PREFIX_SAME, "book");
echo "Cover: $book_cover, Book Shape: $book_shape, Shape: $shape";
Cover: bird, Book Shape: rectangular, Shape: round
```

7.3.2 Creating an Array from Variables

The `compact()` function is the complement of `extract()`. Pass it the variable names to `compact()` either as separate parameters or in an array. The `compact()` function creates an associative array whose keys are the variable names and whose values are the variable's values. Any names in the array that do not correspond to actual variables are skipped. Here's an example of `compact()` in action:

```
$color = 'indigo';
$shape = 'curvy';
$floppy = 'none';
$a = compact('color', 'shape', 'floppy');
// or
$names = array('color', 'shape', 'floppy');
$a = compact($names);
```

Self Assessment

State whether the following statements are true or false:

6. `Extract()` is a very popular function that converts elements in an array into variables in their own right.
7. The `compact()` function is the complement of `extract()`.

7.4 Traversing Arrays

The most common task with arrays is to do something with every element—for instance, sending mail to each element of an array of addresses, updating each file in an array of filenames, or adding up each element of an array of prices. There are several ways to traverse arrays in PHP, and the one you choose will depend on your data and the task you're performing.

7.4.1 The foreach Construct

Notes

The foreach construct provides an easy way to iterate over arrays. Foreach works only on arrays and objects, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable. There are two syntaxes:

```
foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement
```

The first form loops over the array given by `array_expression`. On each iteration, the value of the current element is assigned to `$value` and the internal array pointer is advanced by one (so on the next iteration, you'll be looking at the next element).

The second form will additionally assign the current element's key to the `$key` variable on each iteration.

When foreach first starts executing, the internal array pointer is automatically reset to the first element of the array. This means that you do not need to call `reset()` before a foreach loop.

As foreach relies on the internal array pointer changing it within the loop may lead to unexpected behavior.

In order to be able to directly modify array elements within the loop precede `$value` with `&`. In that case the value will be assigned by reference.



Example:

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$amp;value) {
    $value = $value * 2;
}
// $arr is now array(2, 4, 6, 8)
unset($value); // break the reference with the last element
?>
```

Referencing `$value` is only possible if the iterated array can be referenced (i.e. if it is a variable). The following code won't work:

```
<?php
foreach (array(1, 2, 3, 4) as &$amp;value) {
    $value = $value * 2;
}
?>
```

7.4.2 The Iterator Functions

Every PHP array keeps track of the current element you're working with; the pointer to the current element is known as the *iterator*. PHP has functions to set, move, and reset this iterator. The iterator functions are:

`current()`

Returns the element currently pointed at by the iterator

Notes

reset()

Moves the iterator to the first element in the array and returns it

next()

Moves the iterator to the next element in the array and returns it

prev()

Moves the iterator to the previous element in the array and returns it

end()

Moves the iterator to the last element in the array and returns it

each()

Returns the key and value of the current element as an array and moves the iterator to the next element in the array

key()

Returns the key of the current element

The each() function is used to loop over the elements of an array. It processes elements according to their internal order:

```
reset($addresses);
while (list($key, $value) = each($addresses)) {
    echo "$key is $value<BR>\n";
}
0 is spam@cyberpromo.net
1 is abuse@example.com
```

This approach does not make a copy of the array, as foreach does. This is useful for very large arrays when you want to conserve memory.

The iterator functions are useful when you need to consider some parts of the array separately from others



Example:

Building a table with the iterator functions

```
$ages = array('Person' => 'Age',
             'Fred'    => 35,
             'Barney'  => 30,
             'Tigger'  => 8,
             'Pooh'    => 40);

// start table and print heading
reset($ages);
list($c1, $c2) = each($ages);
echo("<table><tr><th>$c1</th><th>$c2</th></tr>\n");
// print the rest of the values
while (list($c1,$c2) = each($ages)) {
    echo("<tr><td>$c1</td><td>$c2</td></tr>\n");
}
```

```
// end the table
echo("</table>");
<table><tr><th>Person</th><th>Age</th></tr>
<tr><td>Fred</td><td>35</td></tr>
<tr><td>Barney</td><td>30</td></tr>
<tr><td>Tigger</td><td>8</td></tr>
<tr><td>Pooh</td><td>40</td></tr>
</table>
```

7.4.3 Using a for loop

A For loop is very similar to a WHILE loop in that it continues to process a block of code until a statement becomes false, however everything is defined in a single line. The basic structure for a For loop is:

```
for ( start; conditional; increment) { code to execute; }
```

Let's go back to our first example using the While loop, where we printed out the numbers 1 through 10 and do the same thing using a For loop.

```
<?php
for ($num=1; $num <= 10; $num++ )
{
    print $num . " ";
}
?>
```

The For loop can also be used in conjunction with a conditional, just like we did with the While loop:



Example:

```
<?php
for ($num=1; $num <= 10; $num++ )
{
    if ($num < 5)
    {
        print $num . " is less than 5 <br>";
    }
    else
    {
        print $num . " is not less than 5 <br>";
    }
}
?>
```

7.4.4 Calling a Function foreach Array Element

PHP provides a mechanism, `array_walk()`, for calling a user-defined function once per element in an array:

```
array_walk(array, function_name);
```

The function you define takes in two or, optionally, three arguments: the first is the element's value, the second is the element's key, and the third is a value supplied to `array_walk()` when it

Notes

is called. For instance, here's another way to print table columns made of the values from an array:

```
function print_row($value, $key) {
    print("<tr><td>$value</td><td>$key</td></tr>\n");
}
$person = array('name' => 'Fred', 'age' => 35, 'wife' => 'Wilma');
array_walk($person, 'print_row');
```

A variation of this example specifies a background color using the optional third argument to array_walk(). This parameter gives us the flexibility we need to print many tables, with many background colors:

```
function print_row($value, $key, $color) {
    print("<tr><td bgcolor=$color>$value</td><td bgcolor=$color>$key</td></tr>\n");
}
$person = array('name' => 'Fred', 'age' => 35, 'wife' => 'Wilma');
array_walk($person, 'print_row', 'blue');
```

The array_walk() function processes elements in their internal order.

7.4.5 Reducing an Array

The below function applies iteratively the **function** to the elements of the **array**, so as to reduce the array to a single value. If the optional **initial** is available, it will be used at the beginning of the process, or as a final result in case the array is empty. If the array is empty and initial is not passed.

```
array_reduce ( $array, callback $function [, int $initial] );
```

Table 7.4

Parameter	Description
array	Required. Specifies an array.
function	Required. Callback function.
initial	Optional. Specifies the initial value to send to the function.

Return Values

Returns a reduced array.



Example:

```
<?php
function call_back_function($v1,$v2)
{
return $v1 . "-" . $v2;
}
$array=array("a"=>"banana","b"=>"apple","c"=>"orange");

print_r(array_reduce($array));
print_r("<br />");
```

```
print_r(array_reduce($array, 10));
?>
```

7.4.6 Searching for Values

The `in_array()` function searches an array for a specific value. This function returns `TRUE` if the value is found in the array, or `FALSE` otherwise.

Syntax

```
in_array(search, array, type)
```

Parameter	Description
search	Required. Specifies the what to search for
array	Required. Specifies the array to search
type	Optional. If this parameter is set, the <code>in_array()</code> function searches for the search-string and specific type in the array



Notes If the search parameter is a string and the type parameter is set to `TRUE`, the search is case-sensitive.



Example:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");

if (in_array("Glenn", $people))
{
    echo "Match found";
}
else
{
    echo "Match not found";
}
?>
```

Self Assessment

Fill in the blanks:

- Every PHP array keeps track of the current element you're working with; the pointer to the current element is known as the
- A For loop is very similar to aloop in that it continues to process a block of code until a statement becomes false.

7.5 Sorting of Multidimensional Arrays

Sorting arrays with more than one dimension, or by something other than alphabetical or numerical order, is more complicated. PHP knows how to compare two numbers or two text strings, but in a multidimensional array, each element is an array. PHP does not know how to compare two arrays, so you need to create a method to compare them.

Notes



Notes The order of the words or numbers is fairly obvious but for complicated objects, it becomes more problematic.

7.5.1 User Defined Sorts

Here is the definition of a two-dimensional array we used earlier. This array stores three products with a code, a description, and a price for each.

```
$products = array( array( 'TIR', 'Tires', 100 ),  
array( 'OIL', 'Oil', 10 ),  
array( 'SPK', 'Spark Plugs', 4 ) );
```

If we sort this array, what order will the values end up in? Because we know what the contents represent, there are at least two useful orders. We might want the products sorted into alphabetical order using the description or by numeric order by the price. Either result is possible, but we need to use the function `usort()` and tell PHP how to compare the items. To do this, we need to write our own comparison function.

The following code sorts this array into alphabetical order using the second column in the array the description.

```
function compare($x, $y)  
{  
if ( $x[1] == $y[1] )  
return 0;  
else if ( $x[1] < $y[1] )  
return -1;  
else  
return 1;  
}  
usort($products, 'compare');
```

So far in this book, we have called a number of the built-in PHP functions. To sort this array, we have defined a function of our own.

We define a function using the keyword `function`. We need to give the function a name. Names should be meaningful, so we'll call it `compare()`. Many functions take parameters or arguments. Our `compare()` function takes two, one called `x` and one called `y`. The purpose of this function is to take two values and determine their order. For this example, the `x` and `y` parameters will be two of the arrays within the main array, each representing one product. To access the Description of the array `x`, we type `$x[1]` because the Description is the second element in these arrays, and numbering starts at zero. We use `$x[1]` and `$y[1]` to compare the Descriptions from the arrays passed into the function.

When a function ends, it can give a reply to the code that called it. This is called returning a value. To return a value, we use the keyword `return` in our function. For example, the line `return 1;` sends the value 1 back to the code that called the function.

To be used by `usort()`, the `compare()` function must compare `x` and `y`. The function must return 0 if `x` equals `y`, a negative number if it is less, and a positive number if it is greater. Our function will return 0, 1, or -1, depending on the values of `x` and `y`. The final line of code calls the built-in function `usort()` with the array we want sorted (`$products`) and the name of our comparison

function (compare()). If we want the array sorted into another order, we can simply write a different comparison function. To sort by price, we need to look at the third column in the array, and create this comparison function:



Example:

```
function compare($x, $y)
{
    if ( $x[2] == $y[2] )
        return 0;
    else if ( $x[2] < $y[2] )
        return -1;
    else
        return 1;
}
```

When `usort($products, compare)` is called, the array will be placed in ascending order by price. The “u” in `usort()` stands for “user” because this function requires a user-defined comparison function. The `uasort()` and `uksort()` versions of `asort` and `ksort` also require a user-defined comparison function.

Similar to `asort()`, `uasort()` should be used when sorting an associative array by value. Use `asort` if your values are simple numbers or text. Define a comparison function and use `uasort()` if your values are more complicated objects such as arrays. Similar to `ksort()`, `uksort()` should be used when sorting an associative array by key. Use `ksort` if your keys are simple numbers or text.



Caution Define a comparison function and use `uksort()` if your keys are more complicated objects such as arrays.

7.5.2 Reverse User Sorts

The `rsort()` array function in PHP will sort an array in reverse or descending order. This function behaves opposite the `sort()` function.

```
<?php
$myArray = array(500,200,700,400,100);
rsort($myArray); // Sort the array in reverse
foreach($myArray as $key => $value) {
    echo "$value <br />";
}
?>
```

Self Assessment

State whether the following statements are true or false:

10. PHP know how to compare two arrays.
11. The `rsort()` array function in PHP will sort an array in reverse or ascending order.

7.6 Acting on Entire Arrays

PHP has several useful functions for modifying or applying an operation to all elements of an array. You can merge arrays, find the difference, calculate the total, and more, all using built-in functions.

Notes

7.6.1 Merging Two Arrays

Let’s say you have two arrays of arrays with the same structure but different count of arrays in them:

```
$arr1 = array(array(1,"b"), array(2,"a"), array(5,"c"));
$arr2 = array(array(3,"e"));
```

Now, the data in the \$arr1 and \$arr2 is sorted, and now what I would like it to merge these two arrays, so I did this:

```
$res = array_merge($arr1, $arr2);
[code]....
```

7.6.2 Calculating the Sum of an Array

The array_sum() function returns the sum of all the values in the array.

Calculate the sum of values in an array and returns the sum of values in an array as an integer or float.

Table 7.5

Parameter	Description
array	Required. Specifies an array.

It returns the sum of values in an array as an integer or float.



Example:

Try out following example:

```
<?php
$a = array(2, 4, 6, 8);
echo "sum(a) = " . array_sum($a) . "<br />";
$b = array("a" => 1.2, "b" => 2.3, "c" => 3.4);
echo "sum(b) = " . array_sum($b) . "<br />";
?>
```

7.6.3 Filtering Elements from an Array

The array_filter() function filters the values of an array using a callback function.

This function passes each value of the input array to the callback function. If the callback function returns true, the current value from input is returned into the result array. Array keys are preserved. **Syntax**

```
array_filter(array, callbackfunction);
```

Table 7.6

Parameter	Description
array	Required. Specifies the array to filter
callbackfunction	Required. Specifies the callback function to use

Technical Details

Notes

Return Value:	Returns the filtered array
PHP Version:	4.0.6+



Example:

Fill an array with values, specifying keys:

```
<?php
function test_odd($var)
{
return($var & 1);
}
$a1=array("a","b",2,3,4);
print_r(array_filter($a1,"test_odd"));
?>
```

7.6.4 Calculating the difference between Two Arrays

The `array_diff()` function compares the values of two (or more) arrays, and returns the differences.



Did u know? This function compares the values of two (or more) arrays, and return an array that contains the entries from array1 that are not present in array2.

Syntax

```
array_diff(array1,array2,array3...);
```

Parameter	Description
array1	Required. The array to compare from
array2	Required. An array to compare against
array3,...	Optional. More arrays to compare against

Technical Details

Return Value:	Returns an associative array containing the entries from <i>array1</i> that are not present in any of the other arrays
PHP Version:	4.0.1+



Example:

Compare the values of two associative arrays, and return the differences:

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"green","g"=>"blue");

$result=array_diff($a1,$a2);
print_r($result);
?>
```

Notes

Self Assessment

Fill in the blanks:

12. Thefunction returns the sum of all the values in the array.
13. Thefunction filters the values of an array using a callback function.

7.7 Using Arrays

Arrays crop up in almost every PHP program. In addition to their obvious use for storing collections of values, they're also used to implement various abstract data types. In this, we show how to use arrays to implement sets and stacks.

7.7.1 Stacks

Arrays are often used as stacks (Last In, First Out, or LIFO) and queue (First In, First Out, or FIFO) structures. PHP simplifies this approach by providing a set of functions can be used to push and pull (for stacks) elements from an array.



Example:

```
<?php
$stack = array();
array_push($stack, 'bar', 'baz');
var_dump($stack);
$last_in = array_pop($stack);
var_dump($last_in, $stack);
?>
```

In this example, we first, create an array, and we then add two elements to it using `array_push()`. Next, using `array_pop()`, we extract the last element added to the array, resulting in this output:

```
array(2) {
  [0]=>
  string(3) "bar"
  [1]=>
  string(3) "baz"
}
string(3) "baz"
array(1) {
  [0]=>
  string(3) "bar"
}
```

7.7.2 Sets

Some PHP functions are designed to perform set operations on arrays. For example, `array_diff()` is used to compute the difference between two arrays:

```
<?php
$a = array(1, 2, 3);
```

```
$b = array(1, 3, 4);
var_dump(array_diff($a, $b));
?>
```

The call to `array_diff()` in the code above will cause all the values of `$a` that do not also appear in `$b` to be retained, while everything else is discarded:

```
array(1) {
  [1]=>
  int(2)
}
```

If you intend to use an array as a queue, you can add elements to the beginning and extract them from the end by using the `array_unshift()` and `array_shift()` functions:



Example:

```
<?php
$stack = array('qux', 'bar', 'baz');
$first_element = array_shift($stack);
var_dump($stack);
array_unshift($stack, 'foo');
var_dump($stack);
?>
```

In this example, we use `array_shift()` to push the first element out of the array. Next, using `array_unshift()`, we do the reverse and add a value to the beginning of the array. This example results in:

```
array(2) {
  [0]=>
  string(3) "bar"
  [1]=>
  string(3) "baz"
}
array(3) {
  [0]=>
  string(3) "foo"
  [1]=>
  string(3) "bar"
  [2]=>
  string(3) "baz"
}
```



Task Add the following to the end of your for loop

```
for ($i = 0; $i < $totalRows; ++$i) {
  $postData[$i] = mysql_fetch_array($result);
}
print $postData[0]['threadTopic'] . "<BR>";
```

Notes

Refresh your page and see what happens. Now change the 0 to 1, save your work, and refresh the page. Now change 'threadTopic' to 'datePosted'. Again, reload the page.

Self Assessment

State whether the following statements are true or false:

- 14. Arrays are often used as stacks and queue structures.
- 15. All PHP functions are designed to perform set operations on arrays.



Case Study

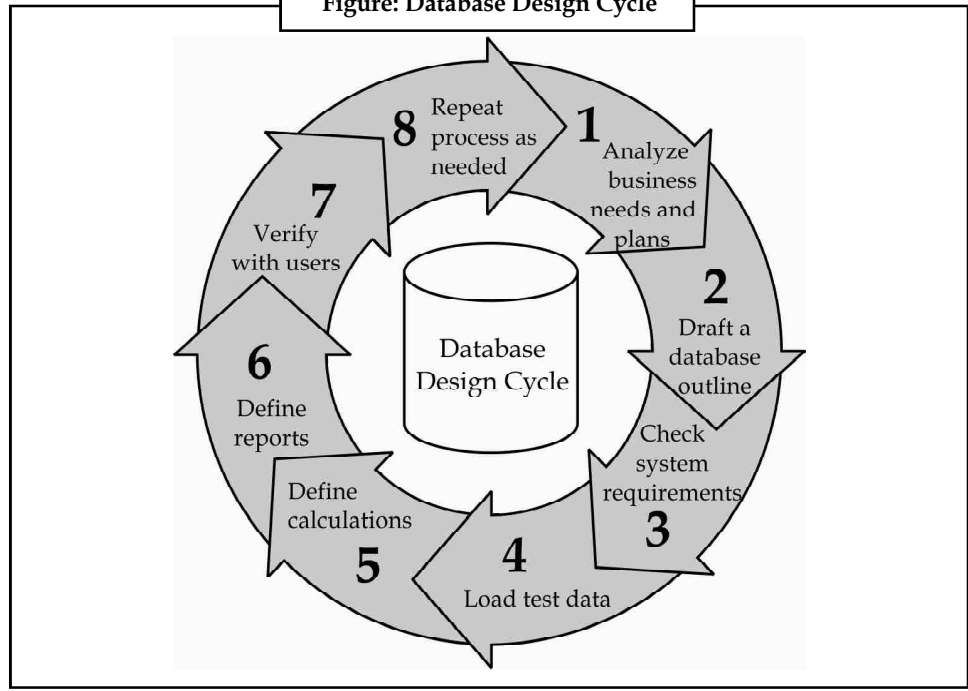
Designing a Single-Server, Multidimensional Database

To implement a multidimensional database, first you install Analytic Services, and then you design and create an application and databases. You analyze data sources and define requirements very carefully and then decide whether a single-server approach or a partitioned, distributed approach best serves your needs. Using a case study, this provides an overview of the database planning process and discusses working rules that you can follow to design a single-server, multidimensional database solution for your organization.

Process for Designing a Database

As illustrated in the Database Design Cycle, designing an application is a cyclic process that moves from a planning stage to a verification stage.

Figure: Database Design Cycle



Contd...

The database design process includes the following basic steps:

1. **Analyze business needs and design a plan.** The application and database that you create must satisfy the information needs of your users and your organization. Therefore, you identify source data, define user information access needs, review security considerations, and design a database model.
2. **Draft a database outline.** The *outline* determines the structure of the database-what information is stored and how different pieces of information relate to one another.
3. **Check system requirements.** How you meet system requirements and define system parameters affects the efficiency and performance of the database.
4. **Load test data into the database.** After an outline and a security plan are in place, you load the database with test data to enable the later steps of the process.
5. **Define calculations.** You test outline consolidations and write and test formulas and calculation scripts for specialized calculations.
6. **Define reports.** Users access data through print and online reports and spreadsheets or on the World Wide Web. If you plan to provide predefined reports to users, you design report layouts and run reports.
7. **Verify with users.** You want to ensure that the database satisfies your user goals. You must solicit and carefully consider the opinions of users.
8. **Repeat the process.** To fine-tune the design, you repeat steps 1 through 7.

Questions

1. Explain the Process for Designing a Database.
2. Discuss the Multidimensional Database.

Notes

7.8 Summary

- Array does not have to be a simple list of keys and values; each array element can contain another array as a value, which in turn can hold other arrays as well.
- One-dimensional array is enough to keep the two types elements. But if you need to keep more than one item of each type you need to use something different one of the ways to do it is using multidimensional arrays.
- PHP provides two functions, `extract ()` and `compact ()`, that convert between arrays and variables. The names of the variables correspond to keys in the array, and the values of the variables become the values in the array.
- Every PHP array keeps track of the current element you're working with; the pointer to the current element is known as the iterator. PHP has functions to set, move, and reset this iterator.
- Sorting arrays with more than one dimension, or by something other than alphabetical or numerical order, is more complicated. PHP knows how to compare two numbers or two text strings, but in a multidimensional array, each element is an array.
- The functions `sort ()`, `asort()`, and `ksort()` all have a matching reverse sort with an "r" in the function name. The user-defined sorts do not have reverse variants, but you can sort a multidimensional array into reverse order.

Notes

7.9 Keywords

array_keys(): The array_keys() function returns an array consisting of only the keys in the array, in internal order.

array_slice(): The array_slice() function returns a new array consisting of a consecutive series of values from the original array.

array_splice(): The array_splice() function can remove or insert elements in an array.

each(): Returns the key and value of the current element as an array and moves the iterator to the next element in the array.

extract(): The extract() function automatically creates local variables from an array. The indexes of the array elements are the variable names.

Multidimensional Array: A multidimensional array is an array that contains at least one other array as the value of one of the indexes.

Variables: A variable is a value that can change, depending on conditions or on information passed to the program.

Values: A value is an expression which cannot be evaluated any further (a normal form).

7.10 Review Questions

1. What are the concepts of multidimensional array? Define their types with example.
2. What is the slicing of array? Explain with example.
3. Write a PHP program to split an array in chunks.
4. How do we remove and insert elements in array? Explain.
5. Which functions are used to perform a conversion between array and variables? Give the suitable example.
6. What are the different methods for traversing an array?
7. What is the different between for and foreach? Discuss with example.
8. How do we sort a multidimensional array?
9. Define the terms:
 - (a) Merging Two Arrays
 - (b) Calculating the Sum of an Array
 - (c) Filtering Elements from an Array
 - (d) Calculating the Difference between Two Arrays
10. How does arrays are used in different data structures? Explain with example.

Answers: Self Assessment

- | | |
|------------------|-------------|
| 1. False | 2. True |
| 3. True | 4. Internal |
| 5. Key_exists() | 6. True |

- | | | |
|--------------------|-----------------|-------|
| 7. True | 8. Iterator | Notes |
| 9. WHILE | 10. False | |
| 11. False | 12. Array_sum() | |
| 13. Array_filter() | 14. True | |
| 15. False | | |

7.11 Further Readings



Books

- Bangia, Ramesh (2008). *“Web Technology (including HTML, CSS, XML, ASP, JAVA).”* Firewall Media.
- Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.
- Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.
- Puntambekar, A.A. (2009). *“Web Technologies.”* Technical Publications.
- Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.
- Xavier, C. (2007). *“Web Technology and Design.”* New Age International.



Online links

- <http://my.safaribooksonline.com/book/programming/php/0596005601/working-with-arrays/learnphp5-chp-4-sect-5>
- http://www.w3schools.com/php/php_arrays_multi.asp
- http://webcheatsheet.com/PHP/multidimensional_arrays.php
- <http://oreilly.com/catalog/progphp/chapter/ch05.html>
- <http://www.tuxradar.com/practicalphp/5/6/4>
- <http://oreilly.com/catalog/progphp/chapter/ch05.html>
- <http://php.net/manual/en/control-structures.foreach.php>
- http://php.about.com/od/learnphp/ss/php_loops_2.htm
- http://www.tutorialspoint.com/php/php_function_array_reduce.htm
- http://www.w3schools.com/php/func_array_in_array.asp
- http://www.developphp.com/view_lesson.php?v=506
- <http://php.bigresource.com/Merge-two-sorted-arrays-and-the-resulting-array-should-also-be-sorted-SQj8427kE.html>
- http://www.tutorialspoint.com/php/php_function_array_sum.htm

Unit 8: Objects

CONTENTS

Objectives

Introduction

8.1 Basics of Objects

8.2 Terminology

8.3 Creating an Object

8.4 Accessing Properties and Methods

8.4.1 Characteristics of Static Keyword

8.5 Classes

8.5.1 Constructors and Destructors

8.5.2 Visibility

8.5.3 Inheritance

8.5.4 Abstract Classes

8.5.5 Static Classes

8.5.6 Class Constants

8.5.7 The "final" Keywords

8.6 Introspection

8.6.1 Examining Classes

8.6.2 Examining an Object

8.6.3 Sample Introspection Program

8.7 Serialization

8.8 Summary

8.9 Keywords

8.10 Review Questions

8.11 Further Readings

Objectives

After studying this unit, you will be able to:

- Discuss the Basics of Objects
- Explain the Terminologies of Objects
- Understand How to Create an Object in PHP
- Understand the Use of Properties and Methods
- Explain the Concepts of Classes

- Discuss Object Introspection
- Understand the Serialization

Introduction

Starting with PHP 5, the object model was rewritten to allow for better performance and more features. This was a major change from PHP 4. PHP 5 has a full object model. Among the features in PHP 5 are the inclusions of visibility, abstract and final classes and methods, additional magic methods, interfaces, cloning and type hinting. PHP treats objects in the same way as references or handles, meaning that each variable contains an object reference rather than a copy of the entire object. We can imagine our universe made of different objects like sun, earth, moon, etc. Similarly we can imagine our car made of different objects like wheel, steering, gear, etc. Same way there is object-oriented programming concepts which assume everything as an object and implement a software using different objects.

8.1 Basics of Objects

In PHP we can define objects in similar ways; we can assign properties to them, as well as make them do things programmatically. Obviously these objects will only exist in the program itself, but through user interfaces such as web pages, we can interact with them and make them perform tasks and procedures, as well as calculate, retrieve and modify properties. PHP gives us a very simple way to define an object programmatically, and this is called a class.

A class is a wrapper that defines and encapsulates the object along with all of its methods and properties. Within the class wrapper, we can either define values of certain properties of the object, or tell the object to do things. Therefore it is important at this point to realise that a class has two main components: methods and properties. A method is essentially a function within a class, and a class may contain no methods, or thousands of methods. Properties are exactly what they seem to be – they are the properties of the object, and we usually use methods to set, modify and get properties from an object.



Did u know? Object handling was completely rewritten for PHP 5, expanding the feature set and enhancing performance of PHP programming.

Self Assessment

State whether the following statements are true or false:

1. PHP gives us most difficult way to define an object programmatically.
2. A class is a wrapper that defines and encapsulates the object along with all of its methods and properties.
3. A method is essentially a function within a class and a class may contain no methods.

8.2 Terminology

Before we go in detail, lets define important terms related to Object-oriented Programming.

- **Class:** This is a programmer-defined datatype, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.

Notes

- **Object:** An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable:** These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.
- **Member Function:** These are the function defined inside a class and are used to access object data.
- **Inheritance:** When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Parent Class:** A class that is inherited from by another class. This is also called a base class or super class.
- **Child Class:** A class that inherits from another class. This is also called a subclass or derived class.
- **Polymorphism:** This is an object oriented concept where same function can be used for different purposes.



Example: Function name will remain same but it make take different number of arguments and can do different task.

- **Overloading:** A type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.
- **Data Abstraction:** Any representation of data in which the implementation details are hidden (abstracted).
- **Encapsulation:** It refers to a concept where we encapsulate all the data and member functions together to form an object.
- **Constructor:** It refers to a special type of function which will be called automatically whenever there is an object formation from a class.
- **Destructors:** It refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.

Self Assessment

Fill in the blanks:

4.function are the function defined inside a class and are used to access object data.
5.refers to a concept where we encapsulate all the data and member functions together to form an object.

8.3 Creating an Object

Once you defined your class, then you can create as many objects as you like of that class type. Following is an example of how to create object using **new** operator.



Example:

```
$physics = new Books;
$maths = new Books;
$chemistry = new Books;
```

Here we have created three objects and these objects are independent of each other and they will have their existence separately. Next we will see how to access member function and process member variables.

Self Assessment

Fill in the blanks:

6. Once the class is defined, now it become essential to create an.....
7. The objects must beof each other.

8.4 Accessing Properties and Methods

After creating your objects, you will be able to call member functions related to that object. One member function will be able to process member variable of related object only.

Following example shows how to set title and prices for the three books by calling member functions.



Example:

```
$physics->setTitle( "Physics for High School" );
$chemistry->setTitle( "Advanced Chemistry" );
$maths->setTitle( "Algebra" );

$physics->setPrice( 10 );
$chemistry->setPrice( 15 );
$maths->setPrice( 7 );
```

Now you call another member functions to get the values set by in above example:



Example:

```
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();
$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```

This will produce following result:

```
Physics for High School
Advanced Chemistry
Algebra
10
15
7
```

Notes



Caution You should be able to see that from within the context of the class, to access member variables or properties you use the self keyword.

8.4.1 Characteristics of Static Keyword

Declaring class members or methods as static makes them accessible without needing an instantiation of the class. A member declared as static can not be accessed with an instantiated class object (though a static method can).



Example:

```
<?php
class Foo
{
    public static $my_static = 'foo';

    public function staticValue() {
        return self::$my_static;
    }
}

print Foo::$my_static . "\n";
$foo = new Foo();
print $foo->staticValue() . "\n";
```

Self Assessment

State whether the following statements are true or false:

8. After creating your objects, you will be able to call member functions related to that object.
9. A member declared as static can be accessed with an instantiated class object.

8.5 Classes

The general form for defining a new class in PHP is as follows:



Example:

```
<?php
class phpClass{
    var $var1;
    var $var2 = "constant string";
    function myfunc ($arg1, $arg2) {
        [..]
    }
    [..]
}
?>
```

Here is the description of each line:

Notes

- The special form class, followed by the name of the class that you want to define.
- A set of braces enclosing any number of variable declarations and function definitions.
- Variable declarations start with the special form var, which is followed by a conventional \$ variable name; they may also have an initial assignment to a constant value.
- Function definitions look much like standalone PHP functions but are local to the class and will be used to set and access object data.



Example:

Here is an example which defines a class of Books type:

```
<?php
class Books{
    /* Member variables */
    var $price;
    var $title;
    /* Member functions */
    function setPrice($par){
        $this->price = $var;
    }
    function getPrice(){
        echo $this->price ."<br/>";
    }
    function setTitle($par){
        $this->title = $par;
    }
    function getTitle(){
        echo $this->title . " <br/>";
    }
}
?>
```

The variable **\$this** is a special variable and it refers to the same object i.e. itself.

8.5.1 Constructors and Destructors

Constructor Functions are special type of functions which are called automatically whenever an object is created. So we take full advantage of this behaviour, by initializing many things through constructor functions.

PHP provides a special function called `__construct()` to define a constructor. You can pass as many as arguments you like into the constructor function.

Following example will create one constructor for Books class and it will initialize price and title for the book at the time of object creation.

Notes



Example:

```
function __construct( $par1, $par2 ){
    $this->price = $par1;
    $this->title = $par2;
}
```

Now we don't need to call set function separately to set price and title. We can initialize these two member variables at the time of object creation only. Check following example below:



Example:

```
$physics = new Books( "Physics for High School", 10 );
$maths = new Books ( "Advanced Chemistry", 15 );
$chemistry = new Books ("Algebra", 7 );

/* Get those set values */
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();

$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```

This will produce following result:

```
Physics for High School
Advanced Chemistry
Algebra
10
15
7
```

Like a constructor function you can define a destructor function using function **__destruct()**. You can release all the resources within a destructor.



Did u know? The OOPs functionality was introduced in PHP version 3.

8.5.2 Visibility

There are three type of visibility available in PHP for controlling your property or method.

Public: Public method or variable can be accessible from anywhere. I mean from inside the class, out side the class and in child class also.

Public visibility is least restricted visibility available in PHP. If you will not define the visibility factor with your method or property then public will be by default applied. Public methods or variables can be accessible from anywhere. For example, it can be accessible from using object

(outside the class), or inside the class, or in child class. Following is the example of the public visibility in PHP classes:



Example:

```
class test
{
public $abc;
public $xyz;
public function xyz()
{
}
}
```

```
$objA = new test();
```

```
echo $objA->abc; //accessible from outside
```

```
$objA->xyz(); //public method of the class test
```

So in above example class test is the very basic class. In this class every thing is open. Minimum restriction in the class is to access its property and methods using object outside the class.

Private: Method or property with private visibility can only be accessible inside the class. You can not access private method or variable from outside of your class.

Private method or properties can only be accessible within the class. You can not access private variable or function of the class by making object out side the class. But you can use private function and property within the class using \$this object. Private visibility in php classes is used when you do not want your property or function to be exposed outside the class. Following example of Private visibility in PHP classes.



Example:

```
Class test
```

```
{
public $abc;
private $xyz;
public function pubDo($a)
{
echo $a;
}
private function privDo($b)
{
echo $b;
}
public function pubPrivDo()
{
$this->xyz = 1;
$this->privDo(1);
}
}
```

```
$objT = new test();
```

```
$objT->abc = 3; //Works fine
```

```
$objT->xyz = 1; //Throw fatal error of visibility
```

```
$objT->pubDo("test"); //Print "test"
```

Notes

```
$objT->privDo(1);//Fatal error of visibility
$objT->pubPrivDo();//Within this method private function privDo and
variable xyz is called using $this variable.
```

Protected: Method or variable with protected visibility can only be access in the derived class or in other word in child class. Protected will be used in the process of inheritance.

Protected visibility in PHP classes are only useful in case of inheritance and interface. Protected method or variable can be accessible either within class or child class. Here we will take very basic example:



Example:

```
class parent
{
protected $pr;
public $a
protected function testParent()
{
echo this is test;
}
}
class child extends parent
{
public function testChild()
{
$this->testParent(); //will work because it
}
}
$objParent = new parent();
$objParent->testParent();//Throw error
$objChild = new Child();
$objChild->setChild();//work because test child will call test parent.
If you will take anaylze above section you can found that method
testParent() is not accessible from object of class. But it is
accessible in child class.
```

8.5.3 Inheritance

PHP class definitions can optionally inherit from a parent class definition by using the extends clause. The syntax is as follows:



Example:

```
class Child extends Parent {
    <definition body>
}
```

The effect of inheritance is that the child class (or subclass or derived class) has the following characteristics:

- Automatically has all the member variable declarations of the parent class.
- Automatically has all the same member functions as the parent, which (by default) will work the same way as those functions do in the parent.

Following example inherit Books class and adds more functionality based on the requirement.

Notes



Example:

```
class Novel extends Books{
    var publisher;
    function setPublisher($par){
        $this->publisher = $par;
    }
    function getPublisher(){
        echo $this->publisher. "<br />";
    }
}
```

Now apart from inherited functions, class Novel keeps two additional member functions.



Task Develop a PHP program for inheritance.

8.5.4 Abstract Classes

An abstract class is one that cannot be instantiated, only inherited. You declare an abstract class with the keyword `abstract`, like this:

When inheriting from an abstract class, all methods marked `abstract` in the parent's class declaration must be defined by the child; additionally, these methods must be defined with the same visibility.



Example:

```
abstract class MyAbstractClass {
    abstract function myAbstractFunction() {
    }
}
```

Note that function definitions inside an abstract class must also be preceded by the keyword `abstract`. It is not legal to have abstract function definitions inside a non-abstract class.

8.5.5 Static Classes

Static classes are used for utility scripts that (in the past) would have been in common functions or held global variables. The advantage of using a static class is more to increase code readability (and ease of debugging) than for any other reason.



Notes A static class may have properties and methods but there will always be only one 'instance' (hence static) and any changes will be made 'globally'.

Static classes are often used for algorithms that may be used by multiple classes. They may also be used to construct objects of a specific type and to abstract logic from classes (to encourage

Notes

loose coupling of objects – i.e. the ability of one class to operate without dependency on another). The example that follows allows the passing in of a username and password and will decide which class of user to instantiate and return. This is important in the case where we want to create a User object, but we have now defined the User class as abstract – how do we know what class of User to create?



Example:

```
class UserHandling
{
    public static $numUsers = 0;

    //a static class has no constructor

    /*
     * This static method accepts a username and password
     * and returns an object of type User
     */
    static public function makeUser($username,$password)
    {
        /*
         * Logic in here to look in a database
         * and get a userTypeID based on username and
         * password
         */
        if($isValidUser)
        {
            if($isAdmin)
            {
                $user = new AdminUser();
            }
            else
            {
                $user = new RegisteredUser();
            }
        }
        else
        {
            $user = new UnregisteredUser();
        }
        UserHandling::$numUsers++;
        //Changing a static variable
        //inside the class still
        //needs the full reference.
        //There is no "$this" inside a
        //static class.
    }
}
```

```

    return $user;
}
}

```

```

//To make a user object, from anywhere in the code
$user = UserHandling::makeUser($username,$password);

```

Properties accessed from outside a static class (if they're public) are also handled in the same way:

```

echo UserHandling::$numUsers;

```



Did u know? A static method cannot access non-static variables and methods, since these require an instance of the class.

8.5.6 Class Constants

It is possible to define constant values on a per-class basis remaining the same and unchangeable. Constants differ from normal variables in that you don't use the \$ symbol to declare or use them.

The value must be a constant expression, not (for example) a variable, a property, a result of a mathematical operation, or a function call.

It's also possible for interfaces to have constants as of PHP 5.3.0, it's possible to reference the class using a variable. The variable's value cannot be a keyword (e.g. self, parent and static).



Example:

Defining and using a Constant

```

<?php
class MyClass
{
    const constant = 'constant value';

    function showConstant() {
        echo self::constant . "\n";
    }
}

echo MyClass::constant . "\n";

$classname = "MyClass";
echo $classname::constant . "\n"; // As of PHP 5.3.0

$class = new MyClass();
$class->showConstant();

echo $class::constant."\n"; // As of PHP 5.3.0
?>

```

8.5.7 The “final” Keywords

PHP 5 introduces the final keyword, which prevents child classes from overriding a method by prefixing the definition with final. If the class itself is being defined final then it cannot be extended.

Following example results in Fatal error: Cannot override final method BaseClass::moreTesting()



Example:

```
<?php
class BaseClass {
    public function test() {
        echo "BaseClass::test() called<br>";
    }

    final public function moreTesting() {
        echo "BaseClass::moreTesting() called<br>";
    }
}

class ChildClass extends BaseClass {
    public function moreTesting() {
        echo "ChildClass::moreTesting() called<br>";
    }
}
?>
```

Self Assessment

Fill in the blanks:

10. Method or property withvisibility can only be accessible inside the class.
11. Anclass is one that cannot be instantiated, only inherited.

8.6 Introspection

Introspection is a common feature in any programming language which allows object classes to be manipulated by the programmer. You'll find introspection particularly useful when you don't know which class or method you need to execute at design time.

Introspection in PHP offers the useful ability to examine classes, interfaces, properties, and methods. PHP offers a large number functions that you can use to accomplish the task. In order to help you understand introspection, I'll provide a brief overview of some of PHP's classes, methods, and functions using examples in PHP to highlight how they are used.

8.6.1 Examining Classes

There are some introspection methods used to determine the capability of the class. You can use these functions to extract basic information about classes such as their name, the name of their parent class, and so on.

- `class_exists()` - checks whether a class has been defined
- `get_class()` - returns the class name of an object
- `get_parent_class()` - returns the class name of an object's parent class
- `is_subclass_of()` - checks whether an object has a given parent class

8.6.2 Examining an Object

Object introspection refers to the ability of a class object to reveal information about itself, such as methods and variables. This might seem a little useless at first. After all, if the developer can view the code he or she can easily browse the source to determine the components of a given class. However, there is a common use for object introspection. It is most often used in conjunction with factory classes. A factory class is a special class that creates other objects. This becomes useful if you have a script that can accept variable input and produce different objects depending on the inputs.



Notes Using this model a factory class can intercept the input, determine the type of class to produce, then create a new instance of this class. If the factory class code is truly abstract then it may be impossible to tell by looking at the code what sort of class will be produced by the factory.

This is where object introspection comes into play. Using the PHP functions `get_object_vars()`, `get_parent_class()`, and `get_class_methods()` it is very easy to determine the composition of a given class or object. The purpose of polling this information is usually to determine variables that exist, perhaps to reset them or manipulate them, or to determine if the object initialized properly. With PHP 5 it becomes possible to simplify the task of the developer and enforce certain object constructs that may aid in determining unknown objects' composition. Using interface and abstract classes certain objects can be pre-defined. For instance, using an interface forces any class extending the interface to override the methods declared in the interface. The following code:

```
interface Foo {
    public function get_results();
    public function set_results($id);
}
```

Will force any object that extends `Foo` to declare the functions `get_results()` and `set_results()` where `set_results()` will accept one input variable named `$id`. In this way you can construct your class libraries so that they must conform to certain conventions.

8.6.3 Sample Introspection Program

Here is the example PHP code that contains the definition for `Introspection` and `Child` classes:



Example:

```
<?php
```

```
class Introspection
```

Notes

```
{
    public function description() {

        echo "I am a super class for the Child class.\n";
    }
}

class Child extends Introspection
{

    public function description() {
        echo "I'm " . get_class($this) , " class.\n";

        echo "I'm " . get_parent_class($this) , "'s child.\n";
    }
}

if (class_exists("Introspection")) {
    $introspection = new Introspection();

    echo "The class name is: " . get_class($introspection) . "\n";
    $introspection->description();
}

if (class_exists("Child")) {
    $child = new Child();

    $child->description();

    if (is_subclass_of($child, "Introspection")) {
        echo "Yes, " . get_class($child) . " is a subclass of
Introspection.\n";
    }
    else {

        echo "No, " . get_class($child) . " is not a subclass of
Introspection.\n";
    }
}
}
```

Self Assessment

Notes

State whether the following statements are true or false:

12. PHP offer a single function that you can use to accomplish the task.
13. Object introspection refers to the ability of a class object to reveal information about itself.

8.7 Serialization

Serialize() returns a string containing a byte-stream representation of any value that can be stored in PHP, unserialize() can use this string to recreate the original variable values. Using serialize to save an object will save all variables in an object. The functions in an object will not be saved, only the name of the class.

In order to be able to unserialize() an object, the class of that object needs to be defined. That is, if you have an object \$a of class A on page1.php and serialize this, you will get a string that refers to class A and contains all values of variables contained in \$a. If you want to be able to unserialize this on page2.php, recreating \$a of class A, the definition of class A must be present in page2.php. This can be done for example by storing the class definition of class A in an include file and including this file in both page1.php and page2.php.

Self Assessment

Fill in the blanks:

14. Usingto save an object will save all variables in an object.
15. In order to be able toan object, the class of that object needs to be defined.



Case Study

Business Objects (Real Time Reporting)

Challenges:

- Real time operational reporting
- Significant effort and extensive delays to access information for better decision-making
- Sub-optimal Customer Service and relationship management
- Improve efficiency without adding headcount

Client uses its system that is hosted by the Parent in France. Client deals with selling, Leasing and servicing the EFT machines to super markets, retail banks, retail shops and all merchandise shops. Reporting was very inefficient and was very time consuming and due to the dependency on the global team based out of France, Australian operation's needs and challenges were not attended to as a priority.

Objectives: Notes

Perisoft provided SAP BusinessObjects EDGE BI software licences and implemented their reporting solution with a road map of enabling Client's Australian team to the ad hoc reporting features as well to make them self-reliant. The project was very cost-effective

Contd...

Notes

compared to the Parent company internal cost and lead times for developing various ad hoc reports. The presentation features of BusinessObjects was an added bonus of the project where the customer were presented with excellent interactive graphical format reports by the sales rep on the run.

Key functionality delivered include:

DIFOT Analysis (Delivery in Full On time)

KPI reporting to analyze the SLA's with the customers for the field services

Implementation Highlights:

- Integrated Solution with SAP ERP and avoiding Excel sheet based solutions, etc.
- Zero dependency on Client France IT team
- Excellent through put of Reports and dashboards.
- Increased customer satisfaction and vital statistics to demonstrate the SLA's, etc.

Questions

1. Explain the basic objectives of real time reporting.
2. Discuss the challenges and its solution of real time reporting.

8.8 Summary

- A class is a wrapper that defines and encapsulates the object along with all of its methods and properties.
- A method is essentially a function within a class, and a class may contain no methods, or thousands of methods. Properties are exactly what they seem to be, they are the properties of the object, and we usually use methods to set, modify and get properties from an object.
- The data associated with an object are called its properties. The functions associated with an object are called its methods. When you define a class, you define the names of its properties and give the code for its methods.
- PHP is not an object-oriented language, it has extensive support for objects. Also, since PHP 5, many core aspects of the language use objects rather than ordinary (procedural) functions.
- Classes can be considered as a collection of methods, variables and constants. They often reflect a real-world thing, like an animal class or a bird class.
- Visibility is a big part of OOP. It allows you to control where your class members can be accessed from, for instance to prevent a certain variable to be modified from outside the class.
- Abstract classes are special because they can never be instantiated. Instead, you typically inherit a set of base functionality from them in a new class. For that reason, they are commonly used as the base classes in a larger class hierarchy.

8.9 Keywords

Abstract Classes: These classes are special because they can never be instantiated.

Class: A class is a template definition of the methods and variables in a particular kind of object.

Constant: A constant is, just like the name implies, a variable that can never be changed.

Constructor: Constructor is a default member function available with a class. It can be explicitly defined in a class to initialize the variables. Constructor is called when an object is created for a class.

Destructor: Destructor is special functions which are automatically called when an object is created and destroyed.

Inheritance: Inheritance is one of the most important aspects of OOP. It allows a class to inherit members from another class.

Introspection: Object introspection refers to the ability of a class object to reveal information about itself, such as methods and variables.

Serialize(): This function returns a string containing a byte-stream representation of any value that can be stored in PHP.

Unserialize(): This function can use this string to recreate the original variable values.

8.10 Review Questions

1. What are the basics of objects? Explain in detail.
2. What are the terminologies used in PHP? How the PHP terms are different from other languages?
3. How do we create an object in PHP? Explain with example.
4. What do you mean by methods and properties? How does it access in PHP?
5. What is the class in PHP? What is its use in programming?
6. What is the difference between constructor and destructor? Explain with example.
7. Explain the difference between visibility and inheritance.
8. What is the difference between abstract classes and static classes?
9. Why we use final keyword? Explain with example.
10. What do you mean by introspection? Explain.
11. What do you mean by serialization? Explain with example.
12. What is the difference between serialization and deserialization?

Answers: Self Assessment

- | | |
|-------------------|---------------|
| 1. False | 2. True |
| 3. True | 4. Member |
| 5. Encapsulation | 6. Object |
| 7. Independent | 8. True |
| 9. False | 10. Private |
| 11. Abstract | 12. False |
| 13. True | 14. Serialize |
| 15. Unserialize() | |

Notes

8.11 Further Readings



Books

Bangia, Ramesh (2008). *“Web Technology (including HTML, CSS, XML, ASP, JAVA).”* Firewall Media.

Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.

Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.

Puntambekar, A. A. (2009). *“Web Technologies.”* Technical Publications.

Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.

Xavier, C. (2007). *“Web Technology and Design.”* New Age International.



Online links

<http://www.php.net/manual/en/oop5.intro.php>

http://www.tutorialspoint.com/php/php_object_oriented.htm

<http://blog.karlbunyan.com/2004/12/14/php-5-static-classes/>

<http://php.net/manual/en/language.oop5.constants.php>

<http://www.madirish.net/127>

<http://phpmaster.com/introspection-and-reflection-in-php/>

Unit 9: Web Techniques

Notes

CONTENTS

Objectives

Introduction

9.1 HTTP Basics

9.1.1 Request and Response

9.1.2 The HTTP Header

9.2 Web Variables

9.3 Server Information

9.4 Processing Forms

9.4.1 Create the Form

9.4.2 Getting the Form Data in the PHP Script

9.4.3 Validating the Form Data

9.4.4 Saving the Form Data to a MySQL Database

9.4.5 Program for Processing Form

9.5 Setting Response Headers

9.5.1 Different Content Types

9.5.2 Redirections

9.5.3 Expiration

9.5.4 Authentication

9.6 Maintaining State

9.6.1 Cookies

9.6.2 Sessions

9.6.3 Combining Cookies and Sessions

9.7 Security Socket Layer (SSL)

9.7.1 Enabling and Disabling SSL Support

9.7.2 Licensing Information

9.8 Summary

9.9 Keywords

9.10 Review Questions

9.11 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the Concepts of HTTP
- Discuss the Web Variables

Notes

- Understand the Server Information
- Explain the Processing Forms
- Discuss the Setting of Response Headers
- Discuss How to Maintaining State in Web Applications
- Explain about Security Socket Layer

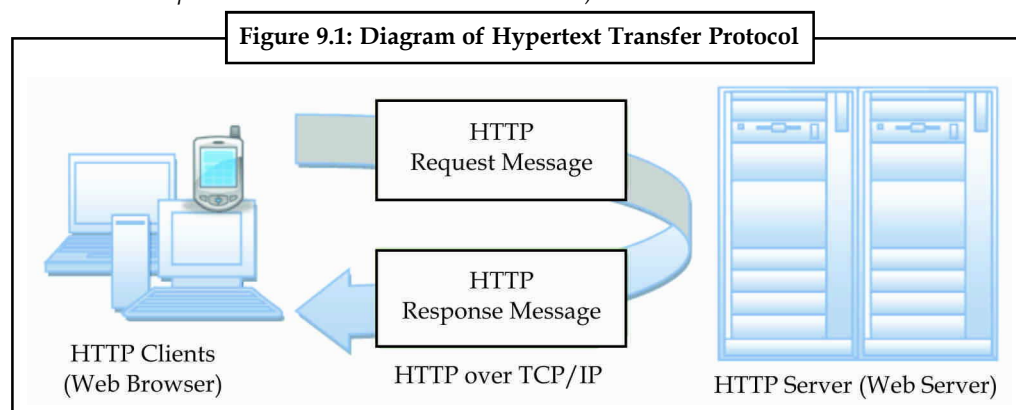
Introduction

PHP can provide dynamic content according to browser type, randomly generated numbers or User Input. It also demonstrated how the client browser can be redirected. PHP was designed as a web scripting language and, although it is possible to use it in purely command-line and GUI scripts, the Web accounts for the vast majority of PHP uses. A dynamic website may have forms, sessions, and sometimes redirection, and this unit explains how to implement those things in PHP. You will also learn how PHP provides access to form parameters and uploaded files, how to send cookies and redirect the browser, how to use PHP sessions, and more.

9.1 HTTP Basics

HTTP (Hypertext Transfer Protocol) is perhaps the most popular application protocol used in the Internet (or the WEB).

- HTTP is an *asymmetric request-response client-server* protocol as illustrated. An HTTP client sends a request message to an HTTP server. The server, in turn, returns a response message. In other words, HTTP is a *pull protocol*, the client *pulls* information from the server (instead of server *pushes* information down to the client).



- HTTP is a stateless protocol. In other words, the current request does not know what has been done in the previous requests.
- HTTP permits negotiating of data type and representation, so as to allow systems to be built independently of the data being transferred.
- Quoting from the RFC2616: “The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers.”



Did u know? The Hypertext Transfer Protocol (HTTP) is a networking protocol for distributed, collaborative, hypermedia information systems; HTTP is the foundation of data communication for the World Wide Web.

9.1.1 Request and Response

HTTP defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

The HTTP/1.0 specification defined the GET, POST and HEAD methods and the HTTP/1.1 specification added 5 new methods: OPTIONS, PUT, DELETE, TRACE and CONNECT. By being specified in these documents their semantics are well known and can be depended upon. Any client can use any method and the server can be configured to support any combination of methods. If a method is unknown to an intermediate it will be treated as an unsafe and non-idempotent method. There is no limit to the number of methods that can be defined and this allows for future methods to be specified without breaking existing infrastructure. For example WebDAV defined 7 new methods and RFC5789 specified the PATCH method.

GET

- Requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. (This is also true of some other HTTP methods.)
- The W3C has published guidance principles on this distinction, saying, “Web application design should be informed by the above principles, but also by the relevant limitations.

HEAD

- Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

POST

- Requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI. The data Posted might be, as examples, an annotation for existing resources; a message for a bulletin board, newsgroup, mailing list, or comment thread; a block of data that is the result of submitting a web form to a data-handling process; or an item to add to a database.

PUT

- Requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.

DELETE

- Deletes the specified resource.

Notes

TRACE

- Echoes back the received request so that a client can see what (if any) changes or additions have been made by intermediate servers.

OPTIONS

- Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

CONNECT

- Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

PATCH

- Is used to apply partial modifications to a resource.

HTTP servers are required to implement at least the GET and HEAD methods and, whenever possible, also the OPTIONS method.

The response message consists of the following:

- A Status-Line (for example, HTTP/1.1 200 OK, which indicates that the client's request succeeded)
- Response Headers, such as Content-Type: text/html
- An empty line
- An optional message body

The Status-Line and headers must all end with <CR><LF> (a carriage return followed by a line feed). The empty line must consist of only <CR><LF> and no other whitespace.

9.1.2 The HTTP Header

HTTP header fields are components of the message header of requests and responses in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction.

The header fields are transmitted after the request or response line, the first line of a message. Header fields are colon-separated name-value pairs in clear-text string format, terminated by a carriage return (CR) and line feed (LF) character sequence. The end of the header fields is indicated by an empty field, resulting in the transmission of two consecutive CR-LF pairs. Long lines can be folded into multiple lines; continuation lines are indicated by presence of space (SP) or horizontal tab (HT) as first character on next line. Few fields can also contain comments (i.e. in. User-Agent, Server, Via fields), which can be ignored by software.

A core set of fields is standardized by the Internet Engineering Task Force (IETF) in RFC 2616 and other updates and extension documents (e.g., RFC 4229), and must be implemented by all HTTP-compliant protocol implementations. Additional field names and permissible values may be defined by each application.

The permanent registry of headers and repository of provisional registrations are maintained by the IANA.



Caution Many field values may contain a quality (q) key-value pair, specifying a weight to use in content negotiation.

There are no limits to the size of each header field name or value, or the number of headers, in the standard itself. However, most servers, clients and proxy software impose some limits for practical and security reasons. For example, the Apache 2.3 server by default limits each header size to 8190 bytes, and there can be at most 100 headers in single request.

Self Assessment

State whether the following statements are true or false:

1. PATCH returns the HTTP methods that the server supports for specified URL.
2. The permanent registry of headers and repository of provisional registrations are maintained by the IANA.
3. HTTP is a stateless protocol.

9.2 Web Variables

Once upon a time, most PHP programmers simply grabbed variables and values by using them in their code. Let's say you had two fields in a form, named "user_name" and "user_email". You could simply use \$user_name and \$user_email in your code (the receiving, or target, page), and they would have the values filled in by the user in the form. Life was simple, wasn't it? \$user_name and \$user_email were examples of a special kind of PHP variable, called a register global variable. You simply used it, and it was magically there to pull in data transferred from a form or a link on another page.

Self Assessment

Fill in the blanks:

4. Most PHP programmers simply grabbed variables and values by using them in their
5. \$user_name and \$user_email were examples of a special kind of PHP variable, called avariable.

9.3 Server Information

In php there is one predefined variable called \$_SERVER, with that only we can get the complete information about server. We can get all server information with the below code. And easily access the information for your coding part.

Once check the below script:



Example:

```
<?php
echo "<table border = '2' style='color:#ffffff;'>";
$i = 0;
foreach($_SERVER as $key=>$value){
```

Notes

```

if($i%2 == 0) {
echo "<tr bgcolor='#993300'>";
} else {
echo "<tr bgcolor='#0099FF'>";
}
echo "<td>";
echo $key;
echo "</td>";
echo "<td>";
echo $value;
echo "</td>";
echo "</tr>";
$i++;
}
echo "</table>";
?>
    
```

Table 9.1

HTTP_ACCEPT	*/*
HTTP_ACCEPT_LANGUAGE	en-us
HTTP_USER_AGENT	Mozilla/4.0(compatible;MSIE 8.0; Window NT 5.1; Trident/4.0)
HTTP_ACCEPT_ENCODING	gzip,deflate
HTTP_HOST	localhost
HTTP_CONNECTION	Keep-Alive
PATH	C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\Intel\
SystemRoot	C:\WINDOWS
COMSPEC	C:\WINDOWS\system32\cmd.exe
PATHEXT	COM;EXE;BAT;CMD;VBS;VBE;JS;JSE;WSF;WSH
WINDIR	C:\WINDOWS
SERVER_SIGNATURE	Apache/2.2.2 (Win32) PHP/5.1.4 Server at localhost Port 80

Self Assessment

Fill in the blanks:

6. In php there is one predefined variable called.....
7. We can get all server information with the below.....

9.4 Processing Forms

A very common application of PHP is to have an HTML form gather information from a website’s visitor and then use PHP to do process that information.

Imagine we are an art supply store that sells brushes, paint, and erasers. To gather order information from our prospective customers we will have to make a page with an HTML form to gather the customer’s order.

9.4.1 Create the Form

Notes

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

The example below contains an HTML form with two input fields and a submit button:



Example:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```

When a user fills out the form above and clicks on the submit button, the form data is sent to a PHP file, called "welcome.php":

"welcome.php" looks like this:



Example:

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br>
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

Output could be something like this:

```
Welcome John!
You are 28 years old.
```



Did u know? Using Simfatic Forms you can create feature-rich web forms without coding.

9.4.2 Getting the Form Data in the PHP Script

Instead of being limited to getting data from files, PHP allows you to receive data from an HTML form as well. This capability is one of the reasons that PHP is so popular for web programming.

This programming technique requires the use of two files. The first is an HTML page which is either a static page or a dynamic one generated by a PHP script.

We will return to the area-of-a-triangle problem for this demonstration.

Notes

N.B. For the following code to work your web server must be running.

PHP permits the processing of data passed from an HTML form back to a script on a web server using the GET or POST methods. Below are given two pairs of files.

GET & POST Methods with Register Global Variables

Using register global variables is the least preferred technique for two reasons:

- Sloppy programming can result in security problems.
- It requires that the register_globals directive be set to ON in the php.ini file, a file which you may not have permissions for on the server.

GET & POST Methods with \$_REQUEST Superglobal Variables

Using request superglobal variables is the preferred technique for two reasons:

- This technique circumvents the default OFF setting of the register_globals directive.
- Its use results in almost “auto-documentation” which differentiates between variables whose values are passed from another script and those which are used only locally



Example:

```
<html>
<head>
<title>Calculating the area of a triangle</title>
</head>
<body>
<p><a href=index.html>Back to DFS's PHP Page</a></p>
<hr>
<h1 align="center">Calculating the Area of a Triangle</h1>
<p>Enter the dimensions of a triangle and click on "Calculate!" to have a
PHP script calculate its area.</p>
<form method="GET" action="triprocessregister.php">
<table border>
<tr>
<td>Base</td><td><input type="text" name="b" value="" size="5"></td>
</tr>
<tr>
<td>Height</td><td><input type="text" name="h" value="" size="5"></td>
</tr>
<tr>
<td colspan="2" align=center><input type="submit" value="Calculate!"><input
type="reset"></td>
</tr>
</table>
</form>
</body>
</html>
```

9.4.3 Validating the Form Data

Notes

User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and reduces the server load.

You should consider server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page.



Notes The user will then get the error messages on the same page as the form. This makes it easier to discover the error.

It is always a good idea to have a “blank” option as the first option in your select box. It forces the user to make a conscious selection from the box and avoids a situation where the user might skip over the box without meaning to. Of course, this requires validation.



Example:

```
<?php
if(empty($varMovie)) {
$errorMessage .= "<li>You forgot to enter a movie!</li>";
}
if(empty($varName)) {
$errorMessage .= "<li>You forgot to enter a name!</li>";
}
if(empty($varGender)) {
$errorMessage .= "<li>You forgot to select your Gender!</li>";
}
?>
```

(For a generic, easy to use form validation script, see PHP Form Validation Script)

It is also a good idea to put your validation checks in the same order as the inputs appear on the form. This way, if there are multiple errors, correcting them will be easier for the user. One other missing piece is that, as before, we want to preserve the user’s choice in the select box, just in case there’s a validation error in one of the other fields. Here is how to do that:




Example:

```
<p>
Please choose your gender
<select name="formGender">
<option value="">Select...</option>
<option value="M"
<? if($varGender=="M") echo(" selected=\"selected\");?> >Male
</option>
<option value="F"
<? if($varGender=="F") echo(" selected=\"selected\");?> >Female
</option>
```

Notes

```
</select>
</p>
```

This code is not the easiest to look at! Basically what is happening here is that for whatever option the user has already selected, we want to put a selected="selected" property in that option box. Now the select box choice will be preserved when the form is submitted. If this code seems ugly, do not worry.



Notes Many select boxes will be populated from a database table, and would not require you to write a bunch of embedded "if" statements. Also, using a select box for 'Gender' probably is not the best choice: radio buttons might make more sense.

9.4.4 Saving the Form Data to a MySQL Database

```
<form method=POST action=formdata.php>

    <table width="640" border=0 align="center">
    <tr>
        <td align=right><b>First Name</b></td>
        <td><input type=text name=FName size=25</td>
        <td><div align="right"><b>Telephone</b></div></td>
        <td><input type=text name=Tel size=25</td>
    </tr>
    <tr>
        <td align=right><b>Last Name</b></td>
        <td><input type=text name=LName size=25</td>
        <td><div align="right"><b>Fax</b></div></td>
        <td><input type=text name=Fax size=25</td>
    </tr>
    <tr>
        <td align=right><b>Title</b></td>
        <td><input type=text name=Title size=25</td>
        <td><div align="right"><b>Email</b></div></td>
        <td><input type=text name=Email size=50</td>
    </tr>
    <tr>
        <td align=right><b>Company</b></td>
        <td><input type=text name=Comp size=25</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td align=right><b>Address</b></td>
        <td><input type=text name=Addr size=25</td>
```

```

        <td><div align="right"><b>Estimated Annual Volume</b></div></td>
        <td><input type="text" name="EAV" size="25"></td>
</tr>
<tr>
        <td align="right"><b>City</b></td>
        <td><input type="text" name="City" size="25"></td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
</tr>
<tr>
        <td align="right"><b>State/Province</b></td>
        <td><input type="text" name="SProv" size="25"></td>
        <td><div align="right"><b>Application</b></div></td>
        <td><input type="text" name="Appl" size="25"></td>
</tr>
<tr>
        <td align="right"><b>Country</b></td>
        <td><input type="text" name="Ctry" size="25"></td>
        <td><div align="right"><b>Type of System</b></div></td>
        <td><input type="text" name="Syst" size="25"></td>
</tr>
<tr>
        <td align="right"><b>Zip/Postal Code</b></td>
        <td><input type="text" name="ZPC" size="25"></td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
</tr>
<tr>
        <td align="right">&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
</tr>
<tr>
        <td align="right">&nbsp;</td>
        <td><div align="right"><strong><font color="#FFFF00" face="Arial,
Helvetica, sans-serif">COIL
        DESIGN</font></strong></div></td>
        <td><font color="#FFFF00" face="Arial, Helvetica, sans-
serif"><strong>PARAMETERS</strong></font></td>
        <td>&nbsp;</td>
</tr>
<tr>
        <td align="right">&nbsp;</td>
        <td>&nbsp;</td>

```

Notes

```

        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td align=right><b>Primary Resistance (ohms)</b></td>
        <td><input type=text name=Pres size=25></td>
        <td><div align="right"><b>Primary Inductance (mH)</b></div></td>
        <td><input type=text name=Pind size=25></td>
    </tr>
    <tr>
        <td align=right><b>Secondary Resistance (ohms)</b></td>
        <td><input type=text name=Sres size=25></td>
        <td><div align="right"><b>Secondary Inductance (H)</b></div></td>
        <td><input type=text name=Sind size=25></td>
    </tr>
    <tr>
        <td align=right>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td align=right><b>Peak Operating Current (Amps)</b></td>
        <td><input type=text name=POC size=25></td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td align=right>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td align=right><b>Output Energy (mJ)</b></td>
        <td><input type=text name=Egy size=25></td>
        <td><div align="right"><b>Output Voltage (kV)</b></div></td>
        <td><input type=text name=Volt size=25></td>
    </tr>
    <tr>
        <td align=right>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>

```

Notes

```

<tr>
  <td align=right><b># HV Towers per Coil</b></td>
  <td><input type=text name=TPC size=25</td>
  <td><div align="right"><b># of Coils per Package</b></div></td>
  <td><input type=text name=CPP size=25</td>
</tr>
<tr>
  <td align=right>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
<tr>
  <td align=right>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
<tr>
  <th colspan=4><b>Please enter any additional information here:</b></th>
</tr>
<tr>
  <th colspan=4><textarea name=Mess cols=50 rows=10 id="Message"></textarea></th>
</tr>
</table>
</dl>
<div align="center">
  <p>
    <input type=hidden name=BodyTag value="&lt;body
    bgcolor=&quot;#484589&quot;;
    text=&quot;#FFFFFF&quot;;
    link=&quot;#FFFF00&quot;;
    alink=&quot;#FFFFFF&quot;;
    vlink=&quot;#FF7F00&quot;;&gt;">
    <input type=hidden name=FA value=SendMail>
  </p>
  <p><font color="#FFFF00" face="Arial, Helvetica, sans-
  serif"><strong>PLEASE MAKE SURE ALL INFORMATION<br>
  IS CORRECT BEFORE SUBMITTING</strong></font></p>
  <p>
    <input type=submit value="Submit Form">
  </p>
</div>
</form>
THE FILE THAT PROCESSES THE FORM DATA (formdata.php)
*****

```

Notes

```
<?php
$con = mysql_connect("localhost","XXX","XXX");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("customform", $con);

$sql="INSERT INTO formdata (Fname, Lname, Title, Comp, Addr, City, SProv,
Ctry, ZPC, Tel, Fax, Email, EAV, Appl, Syst, Pres, Pind, Sres, Sind, POC,
Egy, Volt, TPC, CPP, Mess)
VALUES
('$ _POST[Fname]','$ _POST[Lname]','$ _POST[Title]','$ _POST[Comp]','$ _POST[Addr]','$ _POS
T[City]','$ _POST[SProv]','$ _POST[Ctry]','$ _POST[ZPC]','$ _POST[Tel]','$ _POST[Fax]','$ _POST[Em
ail]','$ _POST[EAV]','$ _POST[Appl]','$ _POST[Syst]','$ _POST[Pres]','$ _POST[Pind]','$ _POST[Sres]','$ _PO
ST[Sind]','$ _POST[POC]','$ _POST[Egy]','$ _POST[Volt]','$ _POST[TPC]','$ _POST[CPP]','$ _POST[Mess]')";

if (!mysql_query($sql,$con))
{
    die('Error: ' . mysql_error());
}
echo "Your Information Was Successfully Posted";

mysql_close($con);

$to = "recipient email address here";
$subject = "Custom Form";
$email = $_POST['Email'] ;
$message = $_POST['Comp'] ;
$headers = "From: $Email";
$sent = mail($to, $subject, $message, $headers) ;
if($sent)
{print "Your mail was sent successfully"; }
else
{print "We encountered an error sending your mail"; }
?>
```

9.4.5 Program for Processing Form

Create the Form

Let's look at the form we used orial and making a few updates to it.



Example:

```
<form action="php-form-processor.php" method="post">
    Which is your favorite movie?
    <input type="text" name="formMovie" maxlength="50" value="<?=$varMovie;?>"
```



```

/>

    What is your name?
    <input type="text" name="formName" maxlength="50" value="<?=$varName;?>"
/>

    Please choose your gender?
    <select name="formGender">
        <option value="">Select...</option>
        <option value="M">Male</option>
        <option value="F">Female</option>
    </select>

<input type="submit" name="formSubmit" value="Submit" />
</form>

```

We're still using the post method. The action is now "php-form-processor.php", since this is a new example, and we've added a new input: a "select" box, also known as a "drop-down" or "pull-down" box. A select box contains one or more "options". Each option has a "value", just like other inputs, and also a string of text between the option tags. This means when a user selects "Male", the "formGender" value when accessed by PHP will be "M".

Getting the Form Data in the PHP Script

Let's look at some PHP code to process this form.



Example:

```

<?php
if($_POST['formSubmit'] == "Submit")
{
    $varMovie = $_POST['formMovie'];
    $varName = $_POST['formName'];
    $varGender = $_POST['formGender'];
    $errorMessage = "";

    // - - - snip - - -
}

?>

```

Select box input is accessed just like a text box.

To set headers, access the \$headers property, and use its set() method.



Example:

```

<?php
$response->headers->set('Header-Label', 'header value');

```

Notes

You can also set all the headers at once by passing an array of key-value pairs where the key is the header label and the value is one or more header values.



Example:

```
<?php
$response->headers->setAll([
    'Header-One' => 'header one value',
    'Header-Two' => [
        'header two value A',
        'header two value B',
        'header two value C',
    ],
]);
```

N.b.: Header labels are sanitized and normalized, so if you enter a label `header_foo` it will be converted to `Header-Foo`.

To get the headers, use `getHeaders()` or access the `$headers` property and use the `get()` method.



Example:

```
<?php
// set a header
$response->headers->set('Content-Type', 'text/plain');

// get a header
$header = $response->headers->get('Content-Type');

// $header->label is 'Content-Type'
// $header->value is 'text/plain'
```

9.5 Setting Response Headers

The HTTP response that a server sends back to a client contains headers that identify the type of content in the body of the response, the server that sent the response, how many bytes are in the body, when the response was sent, etc. PHP and Apache normally take care of the headers for you, identifying the document as HTML, calculating the length of the HTML page, and so on. Most web applications never need to set headers themselves. However, if you want to send back something that's not HTML, set the expiration time for a page, redirect the client's browser, or generate a specific HTTP error, you'll need to use the `header()` function.

The only catch to setting headers is that you must do so before any of the body is generated. This means that all calls to `header()` (or `setcookie()`, if you're setting cookies) must happen at the very top of your file, even before the `<html>` tag. For example:

```
<?php
    header('Content-Type: text/plain');
?>
Date: today
From: fred
```

To: barney
 Subject: hands off!
 My lunchbox is mine and mine alone. Get your own, you filthy scrounger!

9.5.1 Different Content Types

The Content-Type header identifies the type of document being returned. Ordinarily this is “text/html”, indicating an HTML document, but there are other useful document types. For example, “text/plain” forces the browser to treat the page as plain text. This type is like an automatic “view source,” and it is useful when debugging.

9.5.2 Redirections

You can use a simple PHP script to redirect a user from the page they entered to a different web page. One reason you may want to do this is that the page they are trying to access no longer exists. Using this method, they can be seamlessly transferred to the new page without having to click a link to continue.

- Users are redirected quickly and seamlessly
- When using the ‘Back’ button, the user is taken to the last viewed page, not the redirect page
- Works on all browsers



Example:

```
<?php
    header( 'Location: http://www.yoursite.com/new_page.html' ) ;

?>
```

Change the code on the redirect page to be simply this. You need to replace the URL above with the URL you wish to direct to.

Be sure that you do not have any text sent to the browser before this, or it will not work. Your safest bet is to simply remove all content from the page but the redirect code.



Example:

```
<html>

<?php

    //this will NOT work, the browser received the HTML tag before the
    script

    header( 'Location: http://www.yoursite.com/new_page.html' ) ;

?>
```

9.5.3 Expiration

A server can explicitly inform the browser, and any proxy caches that might be between the server and browser of a specific date and time for the document to expire. Proxy and browser caches can hold the document until that time or expire it earlier. Repeated reloads of a cached document do not contact the server. However, an attempt to fetch an expired document does contact the server.

To set the expiration time of a document, use the Expires header:

```
header('Expires: Fri, 18 Jan 2011 05:30:00 GMT');
```

To expire a document three hours from the time the page was generated, use `time()` and `gmstrftime()` to generate the expiration date string:

```
$now = time(); $then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 60*60*3);
header("Expires: $then");
```

To indicate that a document “never” expires, use the time a year from now:

```
$now = time(); $then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 365*86440);
header("Expires: $then");
```

To mark a document as already expired, use the current time or a time in the past:

```
$then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT"); header("Expires: $then");
```

This is the best way to prevent a browser or proxy cache from storing your document:

```
header("Expires: Mon, 26 Jul 1910 05:00:00 GMT"); header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT"); header("Cache-Control: no-store, no-cache, must-revalidate"); header("Cache-Control: post-check=0, pre-check=0", false); header("Pragma: no-cache");
```

9.5.4 Authentication

It is possible to use the `header()` function to send an “Authentication Required” message to the client browser causing it to pop up a Username/Password input window. Once the user has filled in a username and a password, the URL containing the PHP script will be called again with the predefined variables `PHP_AUTH_USER`, `PHP_AUTH_PW`, and `AUTH_TYPE` set to the user name, password and authentication type respectively. These predefined variables are found in the `$_SERVER` and `$HTTP_SERVER_VARS` arrays. Both “Basic” and “Digest” (since PHP 5.1.0) authentication methods are supported.



Example:

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="My Realm"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Text to send if user hits Cancel button';
    exit;
} else {
    echo "<p>Hello {$_SERVER['PHP_AUTH_USER']}</p>";
    echo "<p>You entered {$_SERVER['PHP_AUTH_PW']} as your password.</p>";
}
```

```
}
? >
```

Self Assessment

Fill in the blanks:

8. A very common application of PHP is to have anform gather information from a website's visitor.
9. Theheader identifies the type of document being returned.

9.6 Maintaining State

Due to the fast evolution of Web programming, the stateless nature of the HTTP protocol brought many problems to certain Web applications that required maintaining their state across several HTTP requests. This demanded a rapid development of several mechanisms aimed at tackling this issue through diverse methods.

Particularly, in the universe of PHP programming, session management emerged as a direct response to the above mentioned problem, and currently this mechanism is being used by PHP developers worldwide, in cases where a Web application needs to keep track of its "state" during the occurrence of different HTTP requests.

Whether you're a beginner developer or an experienced PHP programmer, it's quite probable you've already used some of the built-in functions included within the PHP session management module, when constructing user authentication systems, shopping carts or web-based email applications, to name a few illustrative cases.

Indeed, PHP sessions are very simple to use. They hide all the complexities inherent to where and how to store session data, and provide developers with a transparent mechanism for managing information that must persist or keep state even if different HTTP requests are triggered across the same Web application.

9.6.1 Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser. For example name, age, or identification number, etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.
- Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser). A PHP script that sets a cookie might send headers that look something like this:



Example:

```
HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
```

Notes

```
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;
           path=/; domain=tutorialspoint.com
Connection: close
Content-Type: text/html
```

- As you can see, the Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded. The expires field is an instruction to the browser to “forget” the cookie after the given time and date.
- If the browser is configured to store cookies, it will then keep this information until the expiry date. If the user points the browser at any page that matches the path and domain of the cookie, it will resend the cookie to the server. The browser’s headers might look something like this:



Example:

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)
Host: zink.demon.co.uk:1126
Accept: image/gif, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=xyz
```

- A PHP script will then have access to the cookie in the environmental variables `$_COOKIE` or `$HTTP_COOKIE_VARS[]` which holds all cookie names and values. Above cookie can be accessed using `$HTTP_COOKIE_VARS["name"]`.

PHP provided `setcookie()` function to set a cookie. This function requires upto six arguments and should be called before `<html>` tag. For each cookie this function has to be called separately.

```
setcookie(name, value, expire, path, domain, security);
```

Here is the detail of all the arguments:

- **Name** - This sets the name of the cookie and is stored in an environment variable called `HTTP_COOKIE_VARS`. This variable is used while accessing cookies.
- **Value** - This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** - This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path** - This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain** - This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.

- **Security** - This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Following example will create two cookies **name** and **age** these cookies will be expired after one hour.



Example:

```
<?php
    setcookie("name", "John Watkin", time()+3600, "/", "", 0);
    setcookie("age", "36", time()+3600, "/", "", 0);
?>
<html>
<head>
<title>Setting Cookies with PHP</title>
</head>
<body>
<?php echo "Set Cookies"?>
</body>
</html>
```

Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either `$_COOKIE` or `$HTTP_COOKIE_VARS` variables. Following example will access all the cookies set in above example.



Example:

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
echo $_COOKIE["name"]. "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["name"]. "<br />";

echo $_COOKIE["age"] . "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["name"] . "<br />";
?>
</body>
</html>
```

Notes

You can use `isset()` function to check if a cookie is set or not.



Example:

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
    if( isset($_COOKIE["name"]))
        echo "Welcome " . $_COOKIE["name"] . "<br />";
    else
        echo "Sorry... Not recognized" . "<br />";
?>
</body>
</html>
```

Deleting Cookie with PHP

Officially, to delete a cookie you should call `setcookie()` with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired:



Example:

```
<?php
    setcookie( "name", "", time()- 60, "/", "", 0);
    setcookie( "age", "", time()- 60, "/", "", 0);
?>
<html>
<head>
<title>Deleting Cookies with PHP</title>
</head>
<body>
<?php echo "Deleted Cookies" ?>
</body>
</html>
```



Caution Do not use whitespace or semicolons in a cookie name otherwise it will be invalid.

9.6.2 Sessions

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the php.ini file called session.save_path. Before using any session variable make sure you have setup this path.

When a session is started following things happen:

PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.

A cookie called PHPSESSID is automatically sent to the user's computer to store unique session identification string.

A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Starting a PHP Session:

A PHP session is easily started by making a call to the session_start() function. This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to session_start() at the beginning of the page.

Session variables are stored in associative array called \$_SESSION[]. These variables can be accessed during lifetime of a session.

The following example starts a session then register a variable called counter that is incremented each time the page is visited during the session.

Make use of isset() function to check if session variable is already set or not.

Put this code in a test.php file and load this file many times to see the result:



Example:

```
<?php
    session_start();
    if( isset( $_SESSION['counter'] ) )
    {
        $_SESSION['counter'] += 1;
    }
    else
    {
        $_SESSION['counter'] = 1;
    }
    $msg = "You have visited this page ". $_SESSION['counter'];
    $msg .= "in this session.";
?>
<html>
<head>
<title>Setting up a PHP session</title>
</head>
```

Notes

```
<body>
<?php echo ( $msg ); ?>
</body>
</html>
```

Destroying a PHP Session:

A PHP session can be destroyed by session_destroy() function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use unset() function to unset a session variable.

Here is the example to unset a single variable:



Example:

```
<?php
    unset($_SESSION['counter']);
?>
```

Here is the call which will destroy all the session variables:



Example:

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



Did u know? If the user's browser does not support cookies or has cookies turned off, the session ID is propagated in URLs within the website.

9.6.3 Combining Cookies and Sessions

Using a combination of cookies and your own session handler, you can preserve state across visits. Any state that should be forgotten when a user leaves the site, such as which page the user is on, can be left up to PHP's built-in sessions. Any state that should persist between user visits, such as a unique user ID, can be stored in a cookie. With the user's ID, you can retrieve the user's more permanent state, such as display preferences, mailing address, and so on, from a permanent store, such as a database.

Example allows the user to select text and background colours and stores those values in a cookie. Any visits to the page within the next week send the colour values in the cookie. Saving state across visits.



Example:

```
<?php
if($_POST['bgcolor'])
{
setcookie ('bgcolor', $_POST['bgcolor'], time( ) + (60 * 60 * 24 * 7));
}
$bgcolor = empty($bgcolor) ? 'gray' : $bgcolor;
?>
<body bgcolor="<? = $bgcolor?>">
```

```

<form action="<? = $PHP_SELF?>" method="POST">
<select name="bgcolor">
<option value="gray">Gray</option>
<option value="white">White</option>
<option value="black">Black</option>
<option value="blue">Blue</option>
<option value="green">Green</option>
<option value="red">Red</option>
</select>
<input type="submit" />
</form>
</body>

```



Task Critically examine the situation when a session is started.

Self Assessment

Fill in the blanks:

10.can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.
11.are text files stored on the client computer and they are kept of use tracking purpose.

9.7 Security Socket Layer (SSL)

SSL (Secure Socket Layer) is a network layer security protocol which is responsible for ensuring security of data or messages in transit through http, ldap, smtp, imap or pop application layers and practically ensures a reliable end-to-end secure and authenticated connection between the client and the server over the open Internet.

The Secure Socket Layer, developed by Netscape, is a common protocol for providing secure transmission of messages on the Internet. It uses the program level between the HTTP and TCP layers. It is being replaced by the Transport Layer Security protocol, which ensures that a third party does not monitor communications between the two sources.

Most HTTP servers such as Apache and IIS, are capable of supporting an SSL session. Also, most web browsers like Internet explorer, Mozilla, Firefox, Maxthon, Chrome, Opera and others, are equipped with an SSL-enabled software component at the client end. SSL protocol uses a standard key cryptographic method such as public key encryption to authenticate the client and the server through certificate examination. An SSL session is initiated by a web browser which contacts a secure web server, usually on TCP port 443 and using HTTPS protocol.

SSL stands for Secure Sockets Layer, a protocol developed by Netscape Communications and RSA Data Security. SSL provides data encryption, certificate-based authentication, and security negotiations between two communicating applications using TCP/IP over an established network connection. An SSL connection is initiated by the client using a URL starting with https://.

Notes

9.7.1 Enabling and Disabling SSL Support

When building Qt from source, the configuration system checks for the presence of the openssl/opensslv.h header provided by source or developer packages of OpenSSL. By default, an SSL-enabled Qt library dynamically loads any installed OpenSSL library at runtime. However, it is possible to link against the library at compile-time by configuring Qt with the -openssl-linked option. When building a version of Qt linked against OpenSSL, the build system will attempt to link with libssl and libcrypto libraries located in the default location on the developer’s system. This location is configurable: set the OPENSSL_LIBS environment variable to contain the linker options required to link Qt against the installed library. For example, on a Unix/Linux system: ./configure -openssl-linked OPENSSL_LIBS='-L/opt/ssl/lib -lssl -lcrypto' To disable SSL support in a Qt build, configure Qt with the -no-openssl option.

9.7.2 Licensing Information


Due to import and export restrictions in some parts of the world, we are unable to supply the OpenSSL Toolkit Qt packages. Developers wishing to use SSL communication in their deployed applications should either ensure that their users have the appropriate libraries installed, or they should consult a suitably qualified legal professional to ensure that applications using code from the OpenSSL project are correctly certified for import and export in relevant regions of the world.

When the Qt Network module is built with SSL support, the library is linked against OpenSSL in a way that requires OpenSSL license compliance.

Self Assessment

Fill in the blanks:

12. The Secure Socket Layer, developed by, is a common protocol for providing secure transmission of messages on the Internet. It uses the program level between the HTTP and TCP layers.
13. When themodule is built with SSL support, the library is linked against OpenSSL in a way that requires OpenSSL license compliance.
14. An SSL connection is initiated by the client using a.....
15. An SSL session is initiated by a web browser which contacts a secure.....

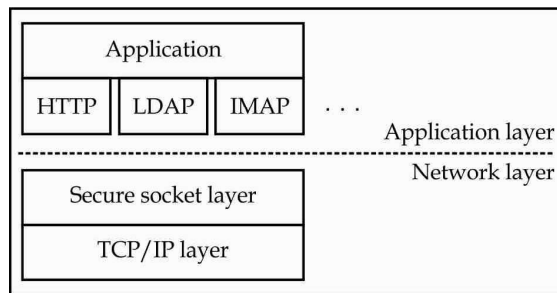


Case Study

SSL Record Protocol

The Transmission Control Protocol/Internet Protocol (TCP/IP) governs the transport and routing of data over the Internet. Other protocols, such as the HyperText Transport Protocol (HTTP), Lightweight Directory Access Protocol (LDAP), or Internet Messaging Access Protocol (IMAP), run “on top of” TCP/IP in the sense that they all use TCP/IP to support typical application tasks such as displaying web pages or running email servers.

Contd...



SSL runs above TCP/IP and below high-level application protocols.

The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP or IMAP. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.

These capabilities address fundamental concerns about communication over the Internet and other TCP/IP networks:

- SSL server authentication allows a user to confirm a server's identity. SSL-enabled client software can use standard techniques of public-key cryptography to check that a server's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the client's list of trusted CAs. This confirmation might be important if the user, for example, is sending a credit card number over the network and wants to check the receiving server's identity.
- SSL client authentication allows a server to confirm a user's identity. Using the same techniques as those used for server authentication, SSL-enabled server software can check that a client's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the server's list of trusted CAs. This confirmation might be important if the server, for example, is a bank sending confidential financial information to a customer and wants to check the recipient's identity.
- An encrypted SSL connection requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality. Confidentiality is important for both parties to any private transaction. In addition, all data sent over an encrypted SSL connection is protected with a mechanism for detecting tampering—that is, for automatically determining whether the data has been altered in transit.

The SSL protocol includes two sub-protocols: the SSL record protocol and the SSL handshake protocol. The SSL record protocol defines the format used to transmit data. The SSL handshake protocol involves using the SSL record protocol to exchange a series of messages between an SSL-enabled server and an SSL-enabled client when they first establish an SSL connection. This exchange of messages is designed to facilitate the following actions:

- Authenticate the server to the client.
- Allow the client and server to select the cryptographic algorithms, or ciphers, that they both support.
- Optionally authenticate the client to the server.
- Use public-key encryption techniques to generate shared secrets.
- Establish an encrypted SSL connection.

Contd...

Notes

SSL technology is used to establish a secure and encrypted communication channel between two Internet connected devices.

Questions

1. What are the different protocols used in SSL?
2. Explain the all phases of process of negotiation in SSL.

9.8 Summary

- PHP can provide dynamic content according to browser type, randomly generated numbers or User Input.
- A server can explicitly inform the browser, and any proxy caches that might be between the server and browser of a specific date and time for the document to expire.
- Using a combination of cookies and your own session handler, you can preserve state across Visits.
- A PHP session is easily started by making a call to the `session_start()` function.
A server can explicitly inform the browser, and any proxy caches that might be between the server and browser of a specific date and time for the document to expire.
- A very common application of PHP is to have an HTML form gather information from a website's visitor and then use PHP to do process that information.

9.9 Keywords

Cookies: A cookie is basically a string that contains several fields. A server can send one or more cookies to a browser in the headers of a response.

GET: This is a simple request for a document or resource residing at a specific URI (Uniform Resource Indicator). It is the most common type of Web request.

HEAD: This is similar to a GET request, except that it is only looking for HTTP header information on the resource, not the resource itself.

HTTP Header: The HTTP header contains details about the transaction between the client and server with slight variations, depending on whether it is a request or a response.

Hypertext Transfer Protocol (HTTP): It is the network protocol used to transmit Web content over the Internet.

POST: It indicates that information is being sent the server inside the HTTP body. The URI Notes should point to a resource capable of handling the data being posted.

Secure Sockets Layer (SSL): It is the standard security technology for establishing an encrypted link between a web server and a browser.

Sessions: It allow you to easily create multi-page forms (such as shopping carts), save user authentication information from page to page, and store persistent user preferences on a site.

Stateless: It means that once a web server completes a client's request for a web page, the connection between the two goes away.

9.10 Review Questions

1. What are the basic concepts of HTTP? Discuss the GET and POST methods.
2. How many Web variables are used in a Web application?

3. What are the information that a server stores?
4. What is the processing form and how does it create? How do we get the form data in the PHP script?
5. Why do we validate a form data and how it stores in MySQL database?
6. What do we do to set the HTTP response headers?
7. What do we do to maintain the state of a Web page?
8. What do you understand by stateless? Why HTTP is called a stateless protocol?
9. Define the cookies and sessions of a Web application. How do we combine them?
10. What is the Secure Sockets Layer? Why do we use it?

Notes

Answers: Self Assessment

- | | |
|--------------------|--------------|
| 1. False | 2. True |
| 3. True | 4. Code |
| 5. Register Global | 6. \$_SERVER |
| 7. Information | 8. HTML |
| 9. Content-Type | 10. Security |
| 11. Cookies | 12. Netscape |
| 13. Qt Network | 14. URL |
| 15. Web Server | |

9.11 Further Readings



Books

Bangia, Ramesh (2008). *“Web Technology (including HTML, CSS, XML, ASP, JAVA).”* Firewall Media.

Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.

Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.

Puntambekar, A. A. (2009). *“Web Technologies.”* Technical Publications.

Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.

Xavier, C. (2007). *“Web Technology and Design.”* New Age International.



Online links

http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

http://en.wikipedia.org/wiki/List_of_HTTP_header_fields

http://wiki.lunarpages.com/PHP_and_Register_Global_Variables

Notes

<http://www.anil2u.info/2010/03/how-to-get-complete-information-about-server-using-php-code/>

http://www.w3schools.com/php/php_forms.asp

<http://www.daniweb.com/web-development/php/threads/308496/store-form-data-into-mysql-database-then-email-same-form-data>

<http://auraphp.com/Aura.Http/version/1.0.0/>

<http://www.devshed.com/c/a/PHP/Introduction-to-Maintaining-the-State-of-Applications-with-PHP-Sessions/>

http://www.tutorialspoint.com/php/php_cookies.htm

http://it.toolbox.com/wiki/index.php/Secure_Socket_Layer

<http://qt-project.org/doc/qt-5.0/qtnetwork/ssl.html>

Unit 10: Database

Notes

CONTENTS

Objectives

Introduction

10.1 Using PHP to Access a Database

10.1.1 Create a Connection to a MySQL Database

10.1.2 Closing a Connection

10.2 Relational Databases and SQL

10.2.1 How Relational Database works?

10.2.2 Security

10.2.3 MySQL Tools

10.3 PEARDB Basics

10.3.1 Advantages and Disadvantages of PEAR DB

10.3.2 Why use a Database Abstraction Layer?

10.3.3 When not to Use a Database Abstraction Layer?

10.3.4 Using PEAR DB

10.3.5 Code Sample: PEAR-DB/Demos/EmployeeReport.php

10.3.6 Code Explanation

10.4 Advanced Database Techniques

10.4.1 Placeholders

10.4.2 Prepare/Execute

10.4.3 Shortcuts

10.4.4 Details about a Query Response

10.4.5 Sequences

10.4.6 Metadata

10.4.7 Transactions

10.5 Summary

10.6 Keywords

10.7 Review Questions

10.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand How to Access a Database using PHP
- Explain Relational Databases and SQL used with PHP

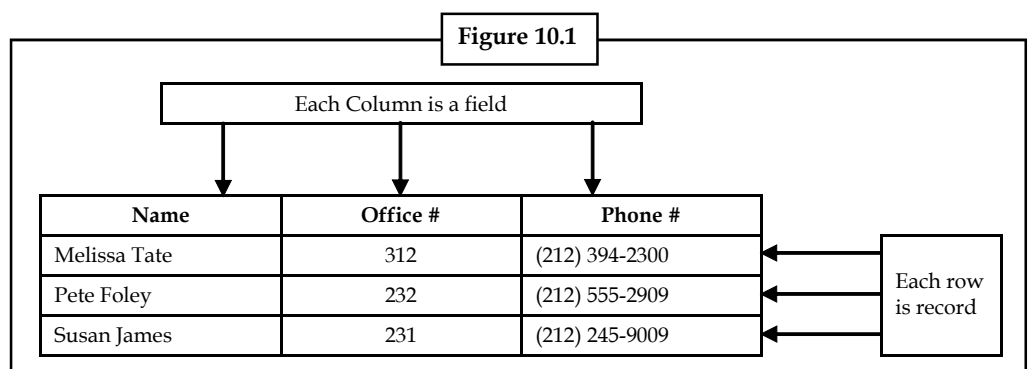
Notes

- Discuss the Basic Concepts of PEAR DB
- Understand the Advanced Database Techniques

Introduction

A database is a place to store information. That information could be sales figures, an inventory of computers you own, a list of key customers, timesheet information; the possibilities are endless. By having the database on your site, you can easily share the information with people who need access to it.

The best way to think of a database is as a table (see Figure 10.1). The columns in the table are called fields. There are many types of fields you can create: text, numeric, currency, names, dates, to name a few. The rows in the table are called records. The records contain all the database information.



A database is much more powerful than a simple table. With a table, you can only look at the information in the one way that the table is designed. A database allows you to create many views of your information. Each view allows you to:

- Select the fields displayed in the view.
- Pick the order in which the fields appear.
- Set up a filter to control which records appear in the view.
- Sort the order in which the records appear.
- Total up fields in the view.
- View fields from more than one database.

A database also allows you to control how views are printed. You can create and print reports using a number of professionally designed templates.

The real power of your database is that it is located in your web office. This allows you to easily share the information in the database with any member of your site, or even with guest users. It also lets other members add information to the database. Of course, you control access to all of this with database permissions.

Your web office database can be as simple as a list of names and phone numbers:

When you create your own database, you determine how the information appears, what the names of the fields are, what the views look like, and who has access to the database. If you already have information that you would like in your database, you can import that information

into the database. You can also export information from your database for use in another program, such as Microsoft Excel®.

10.1 Using PHP to Access a Database

There are two ways to access databases from PHP. One is to use a database-specific extension; and the other is to use the database-independent PEAR DB library. There are advantages and disadvantages to each approach.

The MySQL extension's function names, parameters, error handling, and so on are completely different from those of the other database extensions. If you want to move your database from MySQL to PostgreSQL, it will involve significant changes to your code. The PEAR DB, on the other hand, hides the database-specific functions from you; moving between database systems.

It can be as simple as changing one line of your program. The portability of an abstraction layer like PEAR's DB library comes at a price. Features that are specific to a particular database (for example, finding the value of an automatically assigned unique row identifier) are unavailable. Code that uses the PEAR DB is also typically a little slower than code that uses a database-specific extension.



Caution Keep in mind that an abstraction layer like PEAR DB does absolutely nothing when it comes to making sure your actual SQL queries are portable. If your application uses any sort of non-generic SQL, you will have to do significant work to convert your queries from one database to another.

For large applications, you should consider writing a functional abstraction layer; that is, for each database your application needs to support, write a set of functions that perform various database actions, such as `get_user_record()`, `insert_user_record()`, and whatever else you need, then have a configuration option that sets the type of database to which your application is connected. This approach lets you use all the intricacies of each database you choose to support without the performance penalty and limitations of an abstraction layer.

For simple applications, we prefer the PEAR DB to the database-specific extensions, not just for portability but also for ease of use. The speed and feature costs are rarely significant enough to force us into using the database-specific extensions. For most databases, you will need to recompile PHP with the appropriate database drivers built into it. This is necessary whether or not you use the PEAR DB library. The help information for the `configure` command in the PHP source distribution gives information on how to build PHP with support for various databases.



Example:

`--with-mysql[=DIR]` include MySQL support. DIR is the MySQL base directory. If unspecified, the bundled MySQL library will be used. `--with-oci8 [=DIR]` include Oracle-oci8 support. Default DIR is `ORACLE_HOME`. `--with-ibm-db2[=DIR]` include IBM DB2 support. DIR is the DB2 base install directory, defaults to `/home/db2inst1/sqllib` `--with-pgsql[=DIR]` Include PostgreSQL support. DIR is the PostgreSQL base install directory, defaults to `/usr/local/pgsql`. You cannot build PHP with support for a database whose client libraries you do not have on your system. For example, if you do not have the Oracle client libraries, you cannot build PHP with support for Oracle databases. Use the `phpinfo()` function to check for database support in your installation of PHP. The MySQL database is very often used with PHP.

10.1.1 Create a Connection to a MySQL Database

Before you can get content out of your MySQL database, you must know how to establish a connection to MySQL from inside a PHP script. To perform basic queries from within MySQL is very easy.

Let's get started. The first thing to do is connect to the database. The function to connect to MySQL is called `mysql_connect`. This function returns a resource which is a pointer to the database connection. It's also called a database handle, and we'll use it in later functions. Don't forget to replace your connection details.



Example:

```
<?php
$username = "your_name";
$password = "your_password";
$hostname = "localhost";

//connection to the database
$dbhandle = mysql_connect($hostname, $username, $password)
    or die("Unable to connect to MySQL");
echo "Connected to MySQL<br>";
?>
```

All going well, you should see "Connected to MySQL" when you run this script. If you can't connect to the server, make sure your password, username and hostname are correct.

Once you've connected, you're going to want to select a database to work with. Let's assume the database is called 'examples'. To start working in this database, you'll need the `mysql_select_db()` function:



Example:

```
<?php
//select a database to work with
$selectd = mysql_select_db("examples", $dbhandle)
    or die("Could not select examples");
?>
```

Now that you're connected, let's try and run some queries. The function used to perform queries is named - `mysql_query()`. The function returns a resource that contains the results of the query, called the result set. To examine the result we're going to use the `mysql_fetch_array` function, which returns the results row by row. In the case of a query that doesn't return results, the resource that the function returns is simply a value true or false.

A convenient way to access all the rows is with a while loop. Let's add the code to our script:



Example:

```
<?php
//execute the SQL query and return records
$result = mysql_query("SELECT id, model, year FROM cars");
//fetch the data from the database
while ($row = mysql_fetch_array($result)) {
    echo "ID:". $row{'id'} . " Name:". $row{'model'} . "
```

```

        ".$row{'year'}."<br>";
    }
?>

```

Finally, we close the connection. Although this isn't strictly speaking necessary, PHP will automatically close the connection when the script ends, you should get into the habit of closing what you open.



Example:

```

<?php
//close the connection
mysql_close($dbhandle);
?>

```

Here is a code in full:



Example:

```

<?php
$username = "your_name";
$password = "your_password";
$hostname = "localhost";

//connection to the database
$dbhandle = mysql_connect($hostname, $username, $password)
    or die("Unable to connect to MySQL");
echo "Connected to MySQL<br>";

//select a database to work with
$dbselected = mysql_select_db("examples",$dbhandle)
    or die("Could not select examples");

//execute the SQL query and return records
$result = mysql_query("SELECT id, model, year FROM cars");

//fetch the data from the database
while ($row = mysql_fetch_array($result)) {
    echo "ID:".$row{'id'}." Name:".$row{'model'}."Year: ". //display the
    results
    $row{'year'}."<br>";
}
//close the connection
mysql_close($dbhandle);
?>

```

To create 'examples' database on your MySQL server you should run the following script:

```

CREATE DATABASE `examples`;
USE `examples`;
CREATE TABLE `cars` (
    `id` int UNIQUE NOT NULL,

```

Notes

```
`name` varchar(40),
`year` varchar(50),
PRIMARY KEY(id)
);
INSERT INTO cars VALUES(1,'Mercedes','2000');
INSERT INTO cars VALUES(2,'BMW','2004');
INSERT INTO cars VALUES(3,'Audi','2001');
```

10.1.2 Closing a Connection

The connection will be closed automatically when the script ends. To close the connection before, use the `mysql_close()` function:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con) Notes
{
die('Could not connect: ' . mysql_error());
}
// some code
mysql_close($con);
?>
```

If you use a database-specific extension, your code is intimately tied to that database.

Self Assessment

State whether the following statements are true or false:

1. There are four ways to access databases from PHP.
2. The MySQL extension's function names, parameters, error handling, and so on are same as other database extensions.
3. To perform basic queries from within MySQL is very easy.
4. A convenient way to access all the rows is with FOR loop.
5. The connection will be closed automatically when the script ends.

10.2 Relational Databases and SQL

Databases have become more and more indispensable in our daily life. We deal with data everyday and everywhere – cellular phone contacts, medical records, logistical data and transaction records, etc. They are all stored in databases. It is hard to imagine what the world would be like without databases. Perhaps there would be no ATM, no credit card, no GIS and no airline reservation.

A relational database is a database that has a collection of tables of data items, all of which is formally described and organized according to the relational model. The term is in contrast to only one table as the database, and in contrast to other models which also have many tables in one database.



Notes In the relational model, each table schema must identify a primary column used for identifying a row called the primary key.

Notes

Tables can relate by using a foreign key that points to the primary key. The relational model offers various levels of refinement of the table relations called database normalization. The database management system (DBMS) of a relational database is called an RDBMS, and is the software of a relational database.

10.2.1 How Relational Database Works?

Relational database stores data as collections of tables. Each table contributes a set of columns, which are the properties of the table that are worthwhile and need to make persist. Relationships, critical elements in relational database, can be added between tables, to indicate that two sets of data are interrelated.

Relational databases use a hierarchical system of tables to store information as opposed to a flat file.

- Data is organized in a structured manner using rows and columns.
- In relational databases, data is stored as “objects”.
- There are many database objects and they can be identified from views such as these:
 - ❖ Dba_objects (Oracle)
 - ❖ Sysobjects (MS SQL)

10.2.2 Security

Relational databases also have excellent security. In most database programs there is a special database that contains access permissions for users and databases. This allows a database administrator the ability to tune permissions to needs. The basic set of permissions in MySQL includes the following:

- Select Privilege - Ability to search data in tables
- Insert Privilege - Ability to add data in tables
- Update Privilege - Ability to modify data in tables
- Delete Privilege - Ability to delete data in tables
- Index Privilege - Ability to index tables
- Alter Privilege - Ability to alter tables
- Create Privilege - Ability to create databases
- Drop Privilege - Ability to delete databases
- Grant Privilege - Ability to delete databases
- Reload Privilege - Ability to reload database privileges
- Shutdown Privilege - Ability to shutdown the database program

Notes

- Process Priviledge - Ability to change the individual threads running in the database
- program
- File Priviledge - Ability to import/export data from/to files

The administrator has a special account that has all the privileges to start with and can be used to create custom accounts. If you intend to run a database yourself, you will have to create databases and assign permissions yourself.


10.2.3 MySQL Tools

MySQL is a freely available yet full featured relational database. It has a number of tools to manage the databases it operates, but there are only three which you can use regularly and only one of those is necessary if you are not managing the database yourself. The command is “mysql” and it is a shell that allows an operator to enter SQL commands directly to the database.

This command requires a few command line arguments to get it connected properly. A typical Notes command line may be use is as follows:

```
mysql -uuser -ppassword -hhost.domain.com database
```

- -u specifies the database user account to use
- -p specifies the database user password to use
- -h specifies the database server to connect to
- database specifies the database to act on



Task How will you use MySQL tools in relational database.

Self Assessment

Fill in the blanks:

6.have become more and more indispensable in our daily life.
7.can relate by using a foreign key that points to the primary key.
8. Theof a relational database is called an RDBMS, and is the software of a relational database.
9. Relational databases use asystem of tables to store information as opposed to a flat file.
10. The command isand it is a shell that allows an operator to enter SQL commands directly to the database.

10.3 PEAR DB Basics

PEAR DB smooths over a lot of the rough edges of database access in a PHP program, but there are two reasons why it’s not always the right choice: PEAR DB might not be available on some systems, and a program that uses the built-in PHP functions tailored to a particular database is faster than one that uses PEAR DB. Programmers who don’t anticipate switching or using more than one database program often pick those built-in functions.

The basic model of database access with the built-in functions is the same as with PEAR DB. You call a function that connects to the database. It returns a variable that represents the connection. You use that connection variable with other functions to send queries to the database program and retrieve the results.

The differences are in the details. The applicable functions and how they work differ from database to database. In general, you have to retrieve results one row at a time instead of the convenience that `getAll()` offers, and there is no unified error handling.

10.3.1 Advantages and Disadvantages of PEAR DB

Whether or not you decide to use PEAR DB or a similar database abstraction layer depends on your needs. If you need to be able to work on many applications and get your work done quickly, then PEAR DB is certainly helpful. If performance is key, then you may find the extra weight of PEAR DB to be prohibitive.

10.3.2 Why use a Database Abstraction Layer?

A database abstraction layer is an application programming interface which unifies the communication between a computer application and databases such as SQL Server, DB2, MySQL, PostgreSQL, Oracle or SQLite. Traditionally, all database vendors provide their own interface tailored to their products which leaves it to the application programmer to implement code for all database interfaces he or she would like to support.



Caution Database abstraction layers reduce the amount of work by providing a consistent API to the developer and hide the database specifics behind this interface as much as possible. There exist many abstraction layers with different interfaces in numerous programming languages.

10.3.3 When not to Use a Database Abstraction Layer?

The biggest downside of using a database abstraction layer is that the benefits come at a performance cost. Imagine you were planning to travel around Europe and had the choice of bringing an interpreter who could speak all European languages and learning the languages yourself. It would certainly be easier to bring the interpreter, but this would make each conversation you had somewhat slower. The abstraction layer is the interpreter.

10.3.4 Using PEAR DB

PEAR DB works by abstracting not only the calls necessary to work with the databases (such as `mysql_connect()`, `pgsql_query()`, etc.), but also clashes in SQL syntax, such as the oft-argued “LIMIT” clause. In PHP 5.1 there’s a new extension called PHP Data Objects (PDO) that abstracts only the functions, which gives a half-way house between PEAR DB and just using normal DB calls.

The connection string for connecting to the database with PEAR DB is:

```
Syntax
driver://username:password@host/database
```

Some of the drivers supported by PEAR DB are

- `mysqli`

Notes

- mysql
- mssql
- oci8
- odbc
- pgsql
- sybase
- dbase
- sqlite

10.3.5 Code Sample: PEAR-DB/Demos/EmployeeReport.php



Example:

```
<html>
<head>
<title>Employee Report</title>
</head>
<body>
<?php
require_once 'DB.php';
@$DB = DB::connect('mysqli://root:pwdpwd@localhost/Northwind');
if (DB::isError($DB))
{
    echo 'Cannot connect to database: ` . $DB->getMessage();
}
else
{
    $Query = 'SELECT * FROM Employees';
    $Result = $DB->query($Query);
    $NumResults = $Result->numRows();
    echo "<b>$NumResults Employees</b>";
?>
<table border="1">
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Title</th>
<th>Email</th>
<th>Extension</th>
</tr>
<?php
while ($Row = $Result->fetchRow(DB_FETCHMODE_ASSOC))
{
```

```

echo '<tr>';
echo '<td>' . $Row['FirstName'] . '</td>';
echo '<td>' . $Row['LastName'] . '</td>';
echo '<td>' . $Row['Title'] . '</td>';
echo '<td>' . $Row['Email'] . '</td>';
echo '<td align="right">x' . $Row['Extension'] . '</td>';
echo '</tr>';
}
?>
</table>
<?php
    $Result->free();
    $DB->disconnect();
}
?>
</body>
</html>

```

10.3.6 Code Explanation

As you can see, the PEAR DB API is very similar to the mysqli object-oriented API. Let's walk through the code.

1. First, we include the PEAR DB library. Notice that we simply use DB.php for the path: `require_once 'DB.php'`; This will only work if:

- ❖ DB.php is in the same directory as EmployeeReport.php. This isn't likely as DB.php itself includes files, which would also have to be in the same directory.
- ❖ The `include_path` directive in `php.ini` includes a path to the pear folder containing DB.php.

2. Next, we connect to the database:

```
@$DB = DB::connect('mysqli://root:pwdpwd@localhost/Northwind');
```

This line of code will create a connection object if the connection is successful or an error object if it is not. The `::` syntax will be covered when we discuss object-oriented PHP programming, but the crux of it is that the `connect()` method is a class-level method rather than an object-level method, so it can be called without first instantiating an object.

We then use the class-level `isError()` method to check if `$DB` is an error object, which would mean that the connection failed. If it did fail, we output an error.

```

if (DB::isError($DB))
{
    echo 'Cannot connect to database: ' . $DB->getMessage();
}

```

3. If the connection succeeded, we run our query:

```

$query = 'SELECT * FROM Employees';
$result = $DB->query($query);
$numResults = $result->numRows();

```

Notes

- And, after writing out our header row, we loop through the query results outputting a row for each record returned:

```
while ($Row = $Result->fetchRow(DB_FETCHMODE_ASSOC))
{
    echo '<tr>';
    echo '<td>' . $Row['FirstName'] . '</td>';
    echo '<td>' . $Row['LastName'] . '</td>';
    echo '<td>' . $Row['Title'] . '</td>';
    echo '<td>' . $Row['Email'] . '</td>';
    echo '<td align="right">x' . $Row['Extension'] . '</td>';
}
```

The fetchRow() method can take one of several constants to specify how a row is returned. In this example, we use DB_FETCHMODE_ASSOC to get the row as an associative array. Other options are DB_FETCHMODE_ORDERED (the default) and DB_FETCHMODE_OBJECT, which get the row as an indexed array and an object, respectively.

Self Assessment

State whether the following statements are true or false:

- The basic model of database access with the built-in functions is the same as with PEAR DB.
- A database abstraction layer is an application programming interface which unifies the communication between a computer application and databases.
- The biggest upside of using a database abstraction layer is that the benefits come at a performance cost.
- PEAR DB API is different from the mysqli object-oriented API.

10.4 Advanced Database Techniques

PEAR DB goes beyond the database primitives shown earlier; it provides several shortcut functions for fetching result rows, as well as a unique row ID system and separate prepare/execute steps that can improve the performance of repeated queries.

10.4.1 Placeholders

Just as printf() builds a string by inserting values into a template, the PEAR DB can build a query by inserting values into a template. Pass the query() function SQL with ? In place of specific values, and add a second parameter consisting of the array of values to insert into the SQL:

```
$result = $db->query(SQL, values);
```

For example, this code inserts three entries into the movies table:

```
$movies = array(array('Dr No', 1962),
                array('Goldfinger', 1965),
                array('Thunderball', 1965));
foreach ($movies as $movie) {
    $db->query('INSERT INTO movies (title,year) VALUES (?,?)', $movie);
}
```

There are three characters that you can use as placeholder values in an SQL query:

Notes

- ? A string or number, which will be quoted if necessary (recommended)
- | A string or number, which will never be quoted
- & A filename, the contents of which will be included in the statement (e.g., for storing an image file in a BLOB field)

10.4.2 Prepare/Execute

If a query should be issued many times, it is more efficient to split it into a prepare and an execute phase:

```
$comp = $db->prepare($sql)
```

- returns a compiled query object

```
$res = $db->execute($comp, $arr)
```

- issues the query, filling in elements in

```
$arr for placeholders in the sql string
```

The example uses a two step process to first prepare the query with the SQL string and then execute it by passing the data. This is useful if the same SQL query needs to be run multiple times.

```
$resource = $db->prepare($query);
$result = $db->execute($resource, $data);

while($row = $result->fetchRow()) {
    print_r($row);
}
```

The example above prepares the query and the resulting resource is then passed to the execute function.



Did u know? The data is retrieved with the fetchRow function and then echoed with print_r.

10.4.3 Shortcuts

These shortcuts perform a query and get the result in one go:

- \$db->getOne(\$sql) ... fetches first column of first row of an sql query
- \$db->getRow(\$sql) ... fetches first row of an sql query
- \$db->getCol(\$sql, \$col) ... fetches column \$col from an sql query
- \$db->getAll(\$sql) ... fetches an array of all rows of an sql query

10.4.4 Details about a Query Response

The following functions provide information on the response of a query:

- `$resp->numRows()` ... the number of rows in the response
- `$resp->numCols()` ... the number of columns in the response
- `$resp->affectedRows()` ... the number of rows affected by an INSERT etc.
- `$resp->tableInfo()` ... information on types and flags returned by a SELECT query



Example:

Book Store Query

```
<html>
<head>
<title>Books</title>
</head>
<body>
<table border=1>
<tr><th>Book</th><th>Year</th><th>Author</th></tr>
>
<?php
// connect
require_once 'DB.php';
$db =
DB::connect("mysql://user0:passw0rd@localhost/lib
rary0");
if (DB::iserror($db)) {
die($db->getMessage());
}
// issue a query
$sql = "SELECT books.title, books.year,books.author
FROM books, authors
WHERE books.authorid=authors.authorid
ORDER BY books.year ASC";
$q = $db->query($sql);
if (DB::iserror($q)) {
die ($q->getMessage());
}
// generate table
while ($q->fetchInto($row) :
?>
<tr>
<td><?php echo $row[0] ?> </td>
<td><?php echo $row[1] ?> </td>
<td><?php echo $row[2] ?> </td>
</tr>
```

```
<?php endwhile; ?>
</table>
<?php $q->free(); $db->disconnect(); ?>
</body>
</html>
```

10.4.5 Sequences

Sequences are a way of offering unique IDs for data rows. If you do most of your work with e.g. MySQL, think of sequences as another way of doing AUTO_INCREMENT.

It's quite simple, first you request an ID, and then you insert that value in the ID field of the new row you're creating. You can have more than one sequence for all your tables, just be sure that you always use the same sequence for any particular table. To get the value of this unique ID use nextID(), if a sequence doesn't exist, it will be created automatically. The sequence is automatically incremented each time nextID().

Using a Sequence



Example:

```
<?php
// Once you have a valid MDB2 object named $mdb2...
$id = $mdb2->nextID('mySequence');
if (PEAR::isError($id)) {
    die($id->getMessage());
}

// Use the ID in your INSERT query
$res =& $mdb2->query("INSERT INTO myTable (id, text) VALUES ($id,
'foo')");
?>
```

10.4.6 Metadata

The getListOf() method lets you query the database for information on available databases, users, views, and functions:

```
$data = $db->getListOf(what);
```

The what parameter is a string identifying the database feature to list. Most databases support "databases;" some support "users," "views," and "functions."

For example, this stores a list of available databases in \$dbs:

```
$dbs = $db->getListOf("databases");
```

10.4.7 Transactions

PEAR DB defaults to auto-committing all queries. However using the beginTransaction() method one can open a new transaction and with the commit() and rollback() methods, a transaction is finished. These three methods optionally accept a string name of a save point to set, release or rollback to respectively. The method inTransaction() may be used to check if a transaction is currently open.

Notes



Example:

Doing a Transaction

```
<?php
// Create a valid MDB2 object named $mdb2
// at the beginning of your program...
require_once 'MDB2.php';

$mdb2 =& MDB2::connect('pgsql://usr:pw@localhost/dbnam');
if (PEAR::isError($mdb2)) {
    die($mdb2->getMessage());
}

// check if transaction are supported by this driver
if (!$mdb2->supports('transactions')) {
    exit();
}

// Open a transaction
$res = $mdb2->beginTransaction();

..

// check if we are inside a transaction and if savepoints are supported
if ($mdb2->inTransaction() && $mdb2->supports('savepoints')) {
    // Set a savepoint
    $savepoint = 'MYSAVEPOINT';
    $res = $mdb2->beginTransaction($savepoint);

    ..

    // determine if the savepoint should be released or to rollback to
    the savepoint
    if ($error_condition) {
        $res = $mdb2->rollback($savepoint);
    } else {
        $res = $mdb2->commit($savepoint);
    }
}

..

// determine if the commit or rollback
if ($error_condition) {
    $res = $mdb2->rollback();
} else {
    $res = $mdb2->commit();
}

?>
```


Self Assessment

Notes

Fill in the blanks:

15. The PEAR DB can build a query by inserting values into a.....
16.are a way of offering unique IDs for data rows.
17. Themethod lets you query the database for information on available databases, users, views.

*Case Study***Success Story on Big Fish Games Triples Database**

Big Fish Games is a global leader in the online games industry and distributes more games worldwide than any other online site. Within three years of its debut, BigFishGames.com rocketed into the Top 10 game portals on the Web and now serves millions of downloads everyday.

Their Business Challenge

BigFishGames.com is a fast-growing website with over 25 million unique customer accounts and over 2.5 million visitors per month. In addition to the English site, Big Fish Games also offers international game portals in Japanese, German, French and Spanish. Their ever-growing user base is a huge boost to their business, but it also raises big challenges around IT capacity planning. To ensure the highest quality game experience, Big Fish Games has to accurately predict demand and increase bandwidth at the right time to keep a balance between overutilizing the system, introducing delays and a bad user experience, and underutilizing the system, resulting in a waste of capacity and money.

Their MySQL Solution

Big Fish Games started using MySQL as a small start-up. MySQL allowed Big Fish Games to quickly grow their business with lower cost and hardware requirements, and has scaled with the company as it has grown into an industry leader. Today, Big Fish Games deploys 40 MySQL servers to power its popular gaming website which offers thousands of games, with new games introduced everyday. To achieve the scalability and reliability required by this high-trafficked website, Big Fish Games relies on MySQL Replication plus, DRBD is used to improve high availability. In addition to customer-facing material such as the dynamic website content, e-commerce store, game coupons and discussion forums, the MySQL database is also used for internal operations, tracking game downloads, account authentication, game activations and server logs.

MySQL Query Analyzer

In order to accommodate the growth in website traffic, the DBA team at Big Fish Games has been looking into opportunities to improve application performance. Tuning and optimizing the database is one of the options, but it would not help if the performance problem is caused by poorly-written SQL code. To gain insights into the quality of the SQL code and execution statistics, Big Fish Games has been using the command line tools to identify target areas for potential performance improvement. However, for every problem resolution, extra effort was required to combine information from multiple sources because each command only provided a limited perspective. Now, the MySQL Query Analyzer

Contd...

Notes

provides a consolidated view of query activities and execution details, and has enabled Big Fish Games to quickly identify poorly running queries and tackle the root causes directly in the SQL code. With the help of the MySQL Query Analyzer, the DBA team caught a “bad” query running 400,000 times overnight which never showed up in query logs. Furthermore, the MySQL Query Analyzer is very easy to use and does not require the user to be a world-class MySQL expert to fully leverage its benefits. Since the Query Analyzer uses a Service Agent listening to application queries and performances metrics, the MySQL servers can always be live and operational when being analyzed. There is no need to switch the servers back and forth between on-line and off-line, which eliminates unnecessary risks to server availability and reliability. After deploying the MySQL Query Analyzer, Big Fish Games tripled its database performance within three days, rather than weeks.

MySQL Enterprise Monitor

Big Fish Games also relies on the MySQL Enterprise Monitor and the Dashboard graphs, which show the number of queries per second, CPU load and replication status, to ensure that the website is performing well. Big Fish Games finds the MySQL Enterprise Monitor valuable because it is built for MySQL and offers more relevant and useful information than generic monitoring tools. The MySQL Enterprise Monitor provides critical data points for Big Fish Games to analyze and determine the optimal number of slaves to serve its current website traffic and to plan for the future capacity requirements. This tool also helps the DBA team to gain insight into the system status, usage patterns and potential problems, without having to wait to be notified by the operations group.

Big Fish Games chooses to deploy MySQL Enterprise for the following reasons:

- **High Performance:** MySQL provides fast transaction speed to serve over 300,000 simultaneous users on BigFishGames.com.
- **Ease of use:** MySQL is very easy to use which allows DBAs to manage MySQL servers without a steep learning curve.
- **Low Maintenance:** Using MySQL Enterprise Monitor, Big Fish Games employs just two DBAs to monitor over 70 MySQL servers, 40 in active production and 30 in the testing environment.
- **Low TCO:** MySQL enabled Big Fish Games to launch their business, grow fast and establish themselves as the industry leader at a fraction of the cost compared to using a proprietary database.
- **Unlimited Deployment:** MySQL Enterprise Unlimited gives Big Fish Games the fixed-Notes cost predictability to deploy additional servers without additional costs. This is especially beneficial for companies with rapidly growing data.
- **24x7 Support:** MySQL offers top quality support, with longtime MySQL developers providing guaranteed 30 minutes response time for MySQL Enterprise Platinum customers. It is invaluable for Big Fish Games to receive problem solving advice from MySQL support engineers when business-critical applications go down at midnight.
- **Support for Popular Operating Systems:** MySQL is well-integrated with all major Linux, Solaris and Unix distributions, saving time for DBAs and improving administrative experience.

Contd...

- **Support for C, C++, C#, PHP, Python, Ruby and Java:** MySQL supports drivers for a wide range of programming languages, including PHP, used by Big Fish Games for the front-end presentation layer, and Java, used with the Tomcat application server in the middleware layer.

Memcached

In addition to MySQL Replication, Big Fish Games further increases scalability by using Memcached, a distributed caching layer. All the web content is stored in Memcached, and most of the website queries are processed by this in-memory cache, which significantly improves response time as well as scalability.

Sun Fire x64 Servers

- Big Fish Games utilizes a 3-tier server deployment strategy, and Sun's x64 servers have been chosen because of their excellent reputation for performance and reliability.
- Sun Fire X2100 server is best for applications which require lots of local disk space but less I/O or CPU speed.
- Sun Fire X4100 server works well for applications which demand fast processors but do not need speedy local disk I/O.

Sun Fire X4140 server is optimal with 8 drive bays for applications where faster local disk I/O via RAID 10 and battery backed up write cache is essential. By identifying the requirements for each application and the right server for each condition, Big Fish Games has gained 20x in performance by merely replacing an X4100 server with an X4140 machine.

Questions

1. What do you mean by MySQL Query Analyzer?
2. Explain MySQL Replication.

10.5 Summary

- A database is a place to store information. That information could be sales figures, an inventory of computers you own, a list of key customers, timesheet information.
- There are two ways to access databases from PHP. One is to use a database-specific extension; and the other is to use the database-independent PEAR DB library.
- Whether or not you decide to use PEAR DB or a similar database abstraction layer depends on your needs.
- A relational database is a database that has a collection of tables of data items, all of which is formally described and organized according to the relational model.
- Sequences are a way of offering unique IDs for data rows.
- PEAR DB works by abstracting not only the calls necessary to work with the databases (such as `mysql_connect()`, `pgsql_query()`, etc), but also clashes in SQL syntax, such as the oft-argued "LIMIT" clause.

10.6 Keywords

Database: A database is an organized collection of data for one or more purposes, usually in digital form.

Notes

MySQL: It is a freely available yet full featured relational database.

PEAR DB: It works by abstracting not only the calls necessary to work with the databases (such as `mysql_connect()`, `pgsql_query()`, etc), but also clashes in SQL syntax, such as the oft-argued “LIMIT” clause.

`phpinfo()`: This function is check the database support in your installation process of PHP.

Relational Database: A relational database can be seen as the data handling part of another application.

Sequence: A sequence is really a table in the database that keeps track of the last-assigned ID.

10.7 Review Questions

1. How do we access a database using PHP? Write the steps of database connectivity with PHP.
2. How the relational database and SQL is used with PHP? How it provide the security of database?
3. What are the MySQL tools? How does it work with PHP?
4. What are the basic concepts of PEAR DB? Write its advantages and disadvantages.
5. Why we use a database abstraction layer? When we can avoid it?
6. Discuss the advance database techniques.
7. What is the Query Response? Why it is used?
8. What are the sequences? Why these are used in databases?
9. What is metadata? How it is useful?
10. How the transactions are performed in the databases?

Answers: Self Assessment

- | | |
|------------------|--------------------------------------|
| 1. False | 2. False |
| 3. True | 4. False |
| 5. True | 6. Databases |
| 7. Tables | 8. Database Management System (DBMS) |
| 9. Hierarchical | 10. MySQL |
| 11. True | 12. True |
| 13. False | 14. False |
| 15. Template | 16. Sequences |
| 17. Getlistof() | |

10.8 Further Readings



Books

Bangia, Ramesh (2008). “Web Technology (including HTML, CSS, XML, ASP, JAVA).” Firewall Media.

Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.

Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.

Puntambekar, A.A. (2009). *“Web Technologies.”* Technical Publications.

Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.

Xavier, C. (2007). *“Web Technology and Design.”* New Age International.

Notes



Online links

http://hss.webexone.com/help/us/WebOfcHelp/html/introduction_to_databases.htm

http://webcheatsheet.com/PHP/connect_mysql_database.php

http://en.wikibooks.org/wiki/Structured_Query_Language/Relational_Databases

<http://www.visual-paradigm.com/product/vpuml/tutorials/databasedesign.jsp>

http://librairie.immateriel.fr/fr/read_book/9780596005603/ch07s12

<http://www.contactuspro.com/PEAR-DB.html>

http://en.wikipedia.org/wiki/Database_abstraction_layer

<http://www.tuxradar.com/practicalphp/9/6/0>

http://www.contactuspro.com/PEAR-DB.html#Using_PEAR_DB

<http://www.electrictoolbox.com/php-pdo-bound-placeholders/>

<http://www.macs.hw.ac.uk/~hwloidl/Courses/F21SC/Lecture19DatabaseSupport.pdf>

<http://pear.php.net/manual/en/package.database.mdb2.intro-sequences.php>

Unit 11: Graphic

CONTENTS

Objectives

Introduction

11.1 Embedding an Image in a Page

11.1.1 Here's What's Happening: The Image Element Parameters

11.1.2 Image Formats for the Web

11.1.3 Activating an Image: Turning an Image into a Link

11.2 Graphic Design (GD) Extension

11.3 Basic Graphics Concept

11.3.1 Discrete Grids - Graphing Pixels

11.3.2 Where is the Origin?

11.3.3 Measuring Angles

11.4 Creating and Drawing Images

11.4.1 Structure of a Graphics Program

11.4.2 Changing the Output Format

11.4.3 Testing for Supported Image Formats

11.4.4 Reading an Existing File

11.4.5 Basic Drawing Functions

11.5 Image with Text

11.6 Dynamically Generated Buttons

11.7 Scaling Images

11.8 Colour Handling

11.8.1 Choosing a Colour

11.8.2 Using the Alpha Channel

11.8.3 Identifying Colours

11.8.4 True Colour Indexes

11.8.5 Text Representation of an Image

11.9 Summary

11.10 Keywords

11.11 Review Questions

11.12 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand How to Embedding an Image in a Page
- Discuss about the Graphic Design(GD) Extensions

- Understand the Basic Concepts of Basic Graphics
- Explain How to Creating and Drawing Images in Graphics Notes
- Discuss Dynamic Generation of Buttons
- Understand How to Scaling the Images
- Explain the Colour Handling

Introduction

PHP makes it very easy to do many things needed on a website, among which is to create an image. The ability to generate an image in PHP can be useful if you want to do things like create CAPTCHA images, or even design a banner or logo on the fly the way some free blogging software do.

There are two types of computer graphics, vector and raster. Raster graphics are the representation of images as an array of pixels. Vector graphics use geometrical primitives such as points, lines, curves or polygons to represent images. The primitives are created using mathematical equations. The Cairo Graphics Library takes a vector approach to graphics, allowing smaller size, infinite zooming, and moving, scaling and rotating without degrading image quality.

11.1 Embedding an Image in a Page

The command to place an image is constant. You will use the same format every time. Now might be a good time to talk about where to store everything on your web server because you are starting to call for additional items to fill up your home page. Until now, all you did was put text on the page.

At this point in your HTML, it is a good idea for you to place whatever images you are going to use in a subdirectory called "images". That means place the image in a directory (to be called "images") under the directory where your web pages are located (which would be the "root" directory for your site).

Here's the format for placing an image:

```
<IMG SRC="image.gif" ALT="some text" WIDTH=32 HEIGHT=32>
```

11.1.1 Here's What's Happening: The Image Element Parameters

The tag defines an image in an HTML page.

The tag has two required attributes: src and alt.

Images are not technically inserted into an HTML page, images are linked to HTML pages. The tag creates a holding space for the referenced image.

To link an image to another document, simply nest the tag inside <a> tags.

Table 11.1

Attribute	Value	Description
align	top bottom middle left right	Specifies the alignment of an image according to surrounding elements

Contd...

Notes

alt	Text	Specifies an alternate text for an image
height	pixels	Specifies the height of an image
src	URL	Specifies the URL of an image
width	pixels	Specifies the width of an image
hspace	pixels	Specifies the whitespace on left and right side of an image
vspace	pixels	Specifies the whitespace on top and bottom of an image

11.1.2 Image Formats for the Web

Image file formats are standardized means of organizing and storing digital images. Image files are composed of digital data in one of these formats that can be rasterized for use on a computer display or printer. An image file format may store data in uncompressed, compressed, or vector formats.



Caution Once rasterized, an image becomes a grid of pixels, each of which has a number of bits to designate its color equal to the color depth of the device displaying it.

Best file types for these general purposes:

Table 11.2

	Photographic Images	Graphics, including Logos or Line Art
Properties	Photos are continuous tones, 24-bit color or 8-bit Gray, no text, few lines and edges	Graphics are often solid colors, with few colors, up to 256 colors, with text or lines and sharp edges
For Unquestionable Best Quality	TIF or PNG (lossless compression and no JPG artifacts)	PNG or TIF (lossless compression, and no JPG artifacts)
Smallest File Size	JPG with a higher Quality factor can be decent.	TIF LZW or GIF or PNG (graphics/logos without gradients normally permit indexed color of 2 to 16 colors for smallest file size)
Maximum Compatibility (PC, Mac, Unix)	TIF or JPG	TIF or GIF
Worst Choice	256 color GIF is very limited color, and is a larger file than 24-bit JPG	JPG compression adds artifacts, smears text and lines and edges

Major considerations to choose the necessary file type include:

- Compression quality - Lossy for smallest files (JPG), or Lossless for best quality images (TIF, PNG).
- Full RGB color for photos (TIF, PNG, JPG), or Indexed Color for graphics (PNG, GIF, TIF).
- 16-bit color (48-bit RGB data) is sometimes desired (TIF and PNG).
- Transparency or Animation is used in graphics (GIF and PNG).
- Documents - line art, multi-page, text, fax, etc. - this will be TIF.
- CMYK color is certainly important for commercial pre-press (TIF).

11.1.3 Activating an Image: Turning an Image into a Link

Notes

To make an image into a click-able hyperlink, simply replace the hyperlink text with some HTML image code. Images can have a relative path (/images/sample-image.gif), or they can have an absolute path (http://hyperlinkcode.com/images/sample-image.gif)



Example:

```
<a href="http://www.hyperlinkcode.com"></a>
```

Self Assessment

State whether the following statements are true or false:

1. Image file formats are standardized means of organizing and storing digital images.
2. The command to place an image is not constant.

11.2 Graphic Design (GD) Extension

The GD library is used for dynamic image creation. From PHP we use with the GD library to create GIF, PNG or JPG images instantly from our code. This allows us to do things such as create charts on the fly, create an anti-robot security image, create thumbnail images, or even build images from other images.

PHP is not limited to create just HTML output. It can also be used to create and manipulate image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP, and XPM. Even more convenient, PHP can output image streams directly to a browser. You will need to compile PHP with the GD library of image functions for this to work. GD and PHP may also require other libraries, depending on which image formats you want to work with. You can use the image functions in PHP to get the size of JPEG, GIF, PNG, SWF, TIFF and JPEG2000 images.



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

PHP 4.3 is a bundled version of the GD lib. This bundled version has some additional features like alpha blending, and should be used in preference to the external library since it is codebase is better maintained and more stable. With the assistance of the GD library, you can use PHP to create applications that use dynamic images to display stock quotes, reveal poll results, monitor system performance, and even create games. However it is not like using Photoshop or GIMP; you cannot draw a line by moving your mouse. Instead, you need to precisely specify a shape's type, size, and position. GD has an existing API, and PHP tries to follow its syntax and function-naming conventions. So, if you are familiar with GD from other languages, such as C or Perl, you can easily use GD with PHP. If GD is new to you, it may take a few minutes to figure it out, but soon you will be drawing like Picasso.

The feature set of GD varies greatly depending on which version GD you are running and which features were enabled during configuration. Versions of GD up to 1.6 supported reading and writing GIFs, but this code was removed due to patent problems. Instead, newer versions of GD support JPEGs, PNGs, and WBMPs. Because PNGs are generally smaller than GIFs, allow you to

Notes

use many more colours, have built-in gamma correction, and are supported by all major web browsers, the lack of GIF support is classified as a feature, not a bug. Besides supporting multiple file formats, GD lets you draw pixels, lines, rectangles, polygons, arcs, ellipses, and circles in any colour you want.

You can also draw text using a variety of font types, including built-in, TrueType, and PostScript Type 1 fonts. The ins and outs of the three main text-drawing functions. These two recipes form the basis combines an image template with real-time data to create dynamic images. GD also lets you make transparent GIFs and PNGs. Setting a colour as transparent and using transparencies in patterns. All these features work with GD 1.8.4, which is the latest stable version of the library. If you have an earlier version, you should not have a problem. However, if a particular recipe needs a specific version of GD, we note it in the recipe.

PHP also supports GD 2.x, which, as of this writing, is still in beta. Despite its beta status, the new version is relatively stable and has many new features. In particular, Version 2.x allows true colour images, which lets GD read in PNGs and JPEGs with almost no loss in quality. Also, GD 2.x supports PNG alpha channels, which allow you to specify a transparency level for each pixel. Both versions of GD are available for download from the official GD site at <http://www.boutell.com/gd/>. The GD section of the online PHP Manual at <http://www.php.net/image> also lists the location of the additional libraries necessary to provide support for JPEGs and Type 1 fonts.

There are two easy ways to see which version, if any, of GD is installed on your server and how it is configured. One way is to call `phpinfo()`. You should see `with-gd` at the top under “Configure Command”; further down the page there is also a section titled “gd” that has more information about which version of GD is installed and what features are enabled. The other option is to check the return value of function `imagecreate()`. If it returns true, GD is installed. The `imagetypes()` function returns a bit field indicating which graphics formats are available. The basic image generation process has three steps: creating the image, adding graphics and text to the canvas, and displaying or saving the image.

```
$image = ImageCreate(200, 50);  
$background_color = ImageColorAllocate($image, 255, 255, 255); // white  
$gray = ImageColorAllocate($image, 204, 204, 204); // gray  
ImageFilledRectangle($image, 50, 10, 150, 40, $gray);  
header('Content-type: image/png');  
ImagePNG($image);
```



Did u know? Both versions of GD are available for download from the official GD site at <http://www.boutell.com/gd/>. The GD section of the online PHP Manual at <http://www.php.net/image> also lists the location of the additional libraries necessary to provide support for JPEGs and Type 1 fonts.



Caution If you want to use a feature that is not enabled, you should rebuild PHP yourself or get your ISP to do so.

Self Assessment

Notes

Fill in the blanks:

3. The is used for dynamic image creation.
4. The feature set of GD varies greatly depending on which version GD you are running and which features were enabled during

11.3 Basic Graphics Concept

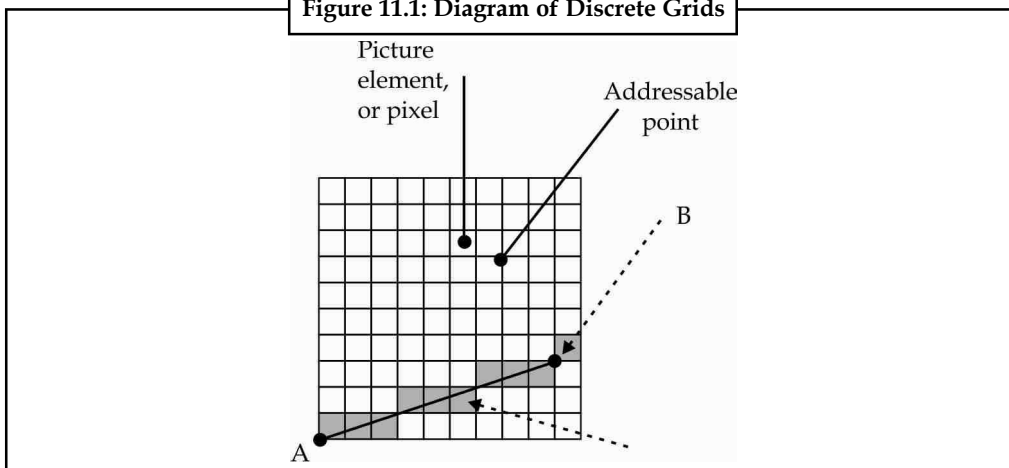
Graphics programming requires specialized knowledge of page flipping, animation, pixel addressing, color representation, video-related concepts, coordinate systems, matrices, and the way physical objects behave in the real world (physics). You also need to know how to use Windows Embedded CE and DirectDraw, and how to apply this specialized knowledge to your graphics development. A complete discussion of graphics concepts is beyond the scope of this documentation. The following topics provide a basic introduction to the complex graphics concepts you need to know to create the graphics applications using Windows Embedded CE and DirectDraw.

11.3.1 Discrete Grids - Graphing Pixels

A pixel (short for picture element, using the common abbreviation “pix” for “picture”) is one of the many tiny dots that make up the representation of a picture in a computer’s memory. Each such information element is not really a dot, nor a square, but an abstract sample.

With care, pixels in an image can be reproduced at any size without the appearance of visible dots or squares; but in many contexts, they are reproduced as dots or squares and can be visibly distinct when not fine enough. The intensity of each pixel is variable; in color systems, each pixel has typically three or four dimensions of variability such as red, green and blue, or cyan, magenta, yellow and black.

Figure 11.1: Diagram of Discrete Grids

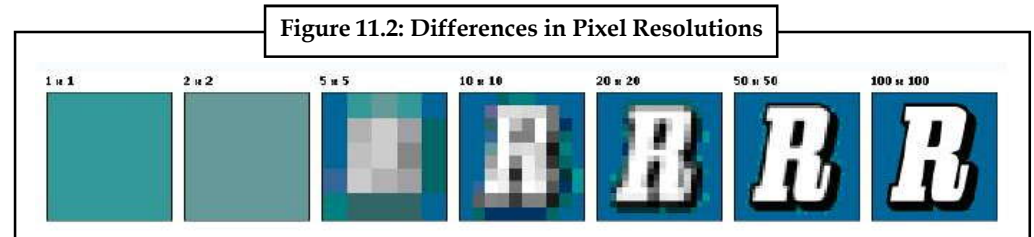


The term resolution is often used as a pixel count in digital imaging, even though American, Japanese, and international standards specify that it should not be so used, at least in the digital camera field. An image of N pixels high by M pixels wide can have any resolution less than N lines per picture height, or N TV lines. But when the pixel counts are referred to as resolution, the convention is to describe the pixel resolution with the set of two positive integer numbers, where the first number is the number of pixel columns (width) and the second is the number of pixel rows (height), for example as 640 by 480.

Notes

Another popular convention is to cite resolution as the total number of pixels in the image, typically given as number of mega pixels, which can be calculated by multiplying pixel columns by pixel rows and dividing by one million. Other conventions include describing pixels per length unit or pixels per area unit, such as pixels per inch or per square inch. None of these pixel resolutions are true resolutions, but they are widely referred to as such; they serve as upper bounds on image resolution.

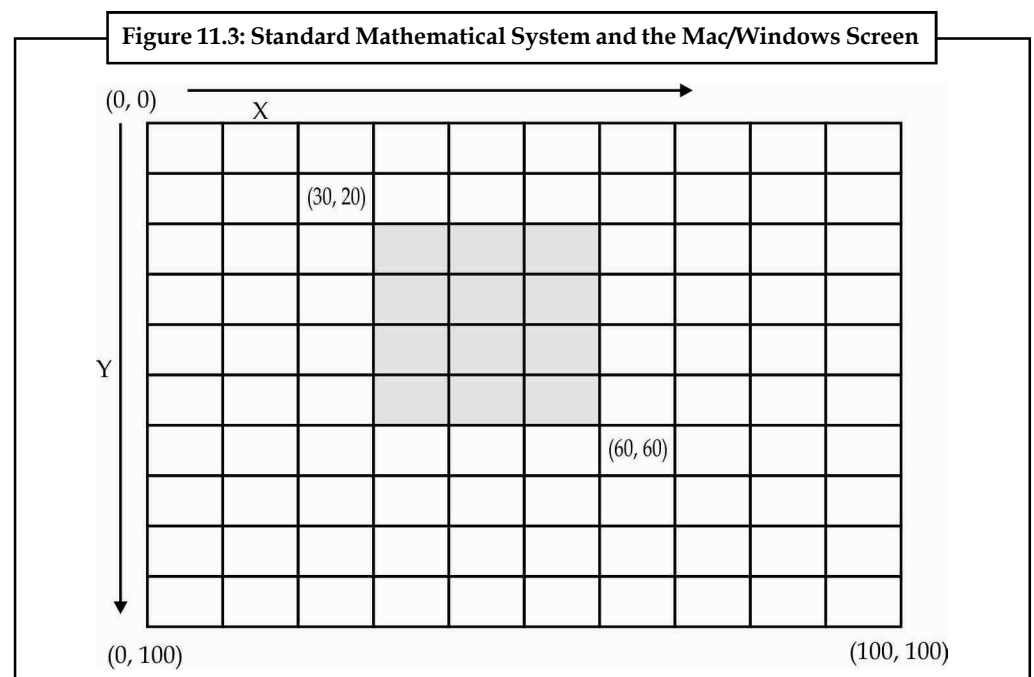
Below is an illustration of how the same image might appear at different pixel resolutions, if the pixels were poorly rendered as sharp squares (normally, a smooth image reconstruction from pixels would be preferred, but for illustration of pixels, the sharp squares make the point better).



11.3.2 Where is the Origin?

When the Macintosh, the first popular graphics-oriented computer was developed but at the top left corner. The positive horizontal direction was still to the right, but the positive vertical direction goes down, rather than up. There are still four quadrants, as points can have negative coordinates, but three of the quadrants are off the screen. As with many other parts of the interface, this arrangement was subsequently copied into other graphics user systems, including the various versions of Windows. The standard mathematical system and the Mac/Windows screen are illustrated in Figure 11.3.

A graphing module used on either of these systems will need to have the option of using the native system directly, or passing conventional mathematical coordinates and then translating to the native screen.



In pixel coordinate system the origin (0, 0) is in upper left corner. Positive x values to the right and positive y values are to the bottom. The coordinates x and y are in pixels.

11.3.3 Measuring Angles

Yet another option needed for graphing is created by the fact that when the focus is on polar coordinates that is, the angle and distance from the origin are given rather than on rectangular coordinates, there are two common systems for measuring angles. In the standard system used mathematics and physics, the angle zero is the positive horizontal axis (East; also called the polar axis) and angles are measured counterclockwise (either in degrees or radians.) However, in the bearing system used in navigation, the positive vertical axis (North) is taken as the zero angles, and bearings are measured clockwise. For the Macintosh/Windows raster, it makes sense to use the positive horizontal axis for zero and measure counterclockwise, as only one quadrant is on the screen. In addition, because some people prefer to graph in degrees and others in radians, a graphing module perhaps ought to provide the option to do either.

Self Assessment

State whether the following statements are true or false:

5. A pixel is one of the many tiny dots that make up the representation of a picture in a computer's memory.
6. An image of N pixels high by M pixels wide can have any resolution more than N lines per picture height, or N TV lines.

11.4 Creating and Drawing Images

```

```

The first thing the above line does is to call the `imagecreate()` function with the dimensions of the image, namely its width and height in that order. This function returns a resource identifier for the image which we save in `$my_img`. The identifier is needed for all our operations on the image.

If the function fails for some reason, it will return `FALSE`. If you want your code to be robust, you should test for this.

11.4.1 Structure of a Graphics Program

The GD library is used for dynamic image creation. From PHP we use with the GD library to create GIF, PNG or JPG images instantly from our code. This allows us to do things such as create charts on the fly, create an anti-robot security image, create thumbnail images, or even build images from other images.

If you are unsure if you have GD library, you can run `phpinfo()` to check that GD Support is enabled. If you don't have it, you can download it for free.

11.4.2 Changing the Output Format

There are many image formats out there for many different uses. A format stores an image in a lossless or lossy format; with lossy formats you suffer some image degradation but save disk space because the image is saved using fewer bytes. A lossless format preserves the image

Notes

exactly, pixel for pixel. You can break formats down into static images and movie clips. Within either category there are standards (static formats and clip codecs) which may be proprietary standards (developed and controlled by one company), or open standards (which are community or consortium-controlled). Open standards generally outlive any one particular company and will always be royalty-free and freely obtained by the viewer. Proprietary formats may only work with a specific video card, or while the codec may be free, the viewer may cost.

As you may have deduced, generating an image stream of a different type requires only two changes to the script: send a different Content-Type and use a different image-generating function.



Example:

```
<?php
    $im = ImageCreate(200,200);
    $white = ImageColorAllocate($im,0xFF,0xFF,0xFF);
    $black = ImageColorAllocate($im,0x00,0x00,0x00);
    ImageFilledRectangle($im,50,50,150,150,$black);
    header('Content-Type: image/jpeg');
    ImageJPEG($im);
?>
```

11.4.3 Testing for Supported Image Formats

If you are writing code that must be portable across systems that may support different image formats, use the `ImageTypes()` function to check which image types are supported. This function returns a bitfield; you can use the bitwise AND operator (`&`) to check if a given bit is set. The constants `IMG_GIF`, `IMG_JPG`, `IMG_PNG`, and `IMG_WBMP` correspond to the bits for those image formats.

11.4.4 Reading an Existing File

There are several way to do it. We will start with reading text files. Most of the time we need to read our file line by line. For this operation we use this function - `file()`; It reads the entire file into array.`file()` - Reads text file into an array.



Example:

```
<?php
// file example
$lines = file("/tmp/files/InputTextFile.txt");
foreach ($lines as $line_num => $line) {
    echo "Line #{$line_num} : " . $line . "n";
}
?>
```

As you can see it is very simple to output text file content. What more can we add here. Just that `file()` supports a few optional parameters:

`FILE_USE_INCLUDE_PATH` - Search for the file in the `include_path`.

`FILE_IGNORE_NEW_LINES` - Do not add newline at the end of each array element (in our case `$line`).

`FILE_SKIP_EMPTY_LINES` - Skip empty lines.

A similar function that reads the entire content of a file, but this time to a string is `file_get_contents()`. It supports two additional parameters, which sometimes can be very helpful: `offset` and `maxlen` bytes (they were added in PHP 5.1). `Offset` specifies where to start reading from, and `maxlen` specifies how many bytes to read from the source. This function is binary safe.

`file_get_contents()` - Reads entire file into a string. It also shows how to read 1KB starting at the 128th byte.



Example:

```
<?php
// file_get_contents example
$file = file_get_contents("/tmp/files/InputTextFile.txt",0,null,128,1024);
echo $file;
?>
```

More direct way to print the file to the buffer is `readfile()` function. What it does is just that. You do not even have to echo it. It returns the bytes that have been read. `readfile` - Outputs the entire file to the output buffer



Example:

```
<?php
// readfile() example
$file = "/tmp/files/InputTextFile.txt";
$bytesRead = readfile($file);
echo $bytesRead;
?>
```

The next function that can help us reading a file is `fgets()`. It reads a line starting from the file pointer position and returns it as a string. You can also specify the length of bytes you want it to read. If you tend to use it, have in mind, that reading file ends when `length - 1` bytes have been read, a newline is reached, or EOF is reached (whichever comes first). This function requires file handle to operate with. In our example we read our file in 8KB chunks at most.

`fgets()` - Gets a line from file pointer



Example:

```
<?php
// fgets example
// Windows OS
$file = "c:/tmp/files/InputTextFile.txt";
$handle = fopen($file, "rt");
if ($handle) {
    while (!feof($handle)) {
        $buffer = fgets($handle, 8192);
        echo $buffer;
    }
    fclose($handle);
}
```

Notes

The last function that we will take a look at and show an example of it is `fread()`. It is used to read binary files. It takes a file handle and reads length bytes from the file pointer. Reading file ends when length bytes or 8192 bytes (8KB) have been read, end of file (EOF) is reached or when a packet becomes available (for network streams), whichever comes first.

`fread()` - Binary-safe file read



Example:

```
<?php
// fread example
$file = "/tmp/files/picture.gif";
// REMEMBER: If this is Windows OS you have to use "rb"
$handle = fopen($file, "r");
$contents = fread($handle, filesize($file));
fclose($handle);
?>
```

11.4.5 Basic Drawing Functions

GD has functions for drawing basic points, lines, arcs, rectangles, and polygons. This describes the base functions supported by GD 2.x.

In order to define a colour in GD, we need to use the `ImageCreatTrueColor()` function (note that function names are not case-sensitive for GD, just like any other PHP function). We need to provide this function with a reference pointer to the image where we want to use this colour (i.e. the current image we are working on). We also need to provide the function with the RGB (Red Green Blue) values for the new colour: remember RGB values for each colour range from 0-255, so (255,255,255) is white, and (0,0,0) is black, with millions of colours in between.

In the example below, we will create a new image, set its background colour to red, and output this graphic to the browser as a PNG image:



Example:

```
<?php

// example1.php

// set the HTTP header type to PNG
header("Content-type: image/png");

// set the width and height of the new image in pixels
$width = 350;
$height = 360;

// create a pointer to a new true colour image
$im = ImageCreateTrueColor($width, $height);

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



```

ImageFillToBorder($im, 0, 0, $red, $red);

// send the new PNG image to the browser
ImagePNG($im);

// destroy the reference pointer to the image in memory to free up resources
ImageDestroy($im);

?>

```

The `ImageLine` function is used to draw a new line in the image. Apart from requiring the image resource and colour in its parameter list, it also requires the coordinates of the start and end points of the line in the following order: start x, start y, end x, end y. The `ImageDashedLine` works the same way, except that as the name implies it draws a dashed line.

`ImageRectangle` that draws an empty rectangle in the colour specified; and `ImageFilledRectangle` which draws a rectangle filled with the colour specified. For both functions, the parameters are identical: `ImageRectangle (image, start x, start y, end x, end y, colour)`.

To draw a circle or an ellipse, we use the same `ImageEllipse` function. The parameters provided to this function are: image resource, center x, center y, width, height, colour. To draw a circle rather than an ellipse is to set the width and height to the same value.

It is quite trivial to define your own shapes to be created using GD, even complex polygon and composite shapes. The trick is to encapsulate all of the code for your custom shape inside a PHP function, like the `drawDiamond` function above.

Self Assessment

State whether the following statements are true or false:

7. The GD library is used for dynamic image creation.
8. A lossless format preserves the image exactly, pixel for pixel.

11.5 Image with Text

In order to add text to an image, you need two things: the image and the font.

For our example, we will use an existing image, rather than creating our own. We will also need a ".ttf" font file (of any font type/style that you want) uploaded to the server that your script will be running from.

Three functions are used to create an image from an existing image, so that it can be used/edited. These functions are `imagecreatefromgif()`, `imagecreatefromjpeg()` and `imagecreatefrompng()`. You simply choose which function to use based on your image file type.

The `imagefttext()` function will be used to actually write the text to the image. The syntax is: `imagefttext(resource $image, float $size, float $angle, int $x, int $y, int $color, string $fontfile, string $text)`.

Most of the `imagefttext()` function's parameters are self-explanatory, but let's review each on briefly.

- Image = Identifies the Image to Add Text To
- Size = Specifies Font Size, Usually Measured in Pixels

Notes

- Angle = Degree/ Angle Text Should be Printed, With 0 Being Left-to-Right “Normal” Text
- X = Coordinate (From Left) Indicating Beginning Location of Text
- Y = Coordinate (From Top) Indicating Beginning Location of Text
- Color = Color of Text, Usually Already Stored In Variable Using `imagecolorallocate()` Function
- Font File = Path to the TrueType “.ttf” Font File
- Text = String of Text to Be Printed On Image (UTF-8 Encoding)

Now let’s go over the basic outline of the steps that we need to perform:

- Set the Content Type
- Create Image from Existing File
- Allocate a Color For the Text
- Set Path to Font File
- Set Text to be Printed on Image
- Print Text on Image
- Send Image to Browser
- Clear Memory

From here, the code should be relatively simple.



Example:

```
<?php
//Set the Content Type
header('Content-type: image/jpeg');

// Create Image From Existing File
$img_image = imagecreatefromjpeg('sunset.jpg');

// Allocate A Color For The Text
$white = imagecolorallocate($img_image, 255, 255, 255);

// Set Path to Font File
$font_path = 'font.TTF';

// Set Text to Be Printed On Image
$text = "This is a sunset!";

// Print Text On Image
imagefttext($img_image, 25, 0, 75, 300, $white, $font_path, $text);

// Send Image to Browser
imagejpeg($img_image);

// Clear Memory
imagedestroy($img_image);
?>
```



Task Critically analyse the things that you need to insert in order to add text to an image.

Self Assessment

Fill in the blanks:

9. Thefunction will be used to actually write the text to the image.
10.functions are used to create an image from an existing image so that it can be used/edited.

11.6 Dynamically Generated Buttons

You want to create an image based on a existing image template and dynamic data (typically text). For instance, you want to create a hit counter. Load the template image, find the correct position to properly center your text, add the text to the canvas, and send the image to the browser:

```
// Configuration settings
$image = ImageCreateFromPNG('button.png');
$text = $_GET['text'];
$font = ImagePSLoadFont('Times');
$size = 24;
$color = ImageColorAllocate($image, 0, 0, 0); // black
$bg_color = ImageColorAllocate($image, 255, 255, 255); // white
// Print centered text
list($x, $y) = pc_ImagePSCenter($image, $text, $font, $size);
ImagePSText($image, $text, $font, $size, $color, $bg_color, $x, $y);
// Send image header('Content-type: image/png');
ImagePNG($image); // Clean up Image
PSFreeFont($font);
ImageDestroy($image);
```

Building dynamic images with GD is easy; all you need to do is to combine a few recipes together. At the top of the code in the Solution, we load in an image from a stock template button; it acts as the background on which we overlay the text. We define the text to come directly from the query string. Alternatively, we can pull the string from a database (in the case of access counters) or a remote server (stock quotes or weather report icons). After that, we continue with the other settings: loading a font and specifying its size, colour, and background colour. Before printing the text, however, we need to compute its position; `pc_ImagePSCenter()`. Last, we serve the image, and de-allocate the font and image from memory. For example, the following code generates a page of HTML and image tags using dynamic buttons:



Example:

```
<?php
if (isset($_GET['button']))
{
```

Notes

```
// Configuration settings
$image = ImageCreateFromPNG('button.png');
$text = $_GET['button'];
// dynamically generated text
$font = ImagePSLoadFont('Times'); $size = 24; $color =
ImageColorAllocate($image, 0, 0, 0);
// black
$bg_color = ImageColorAllocate($image, 255, 255, 255); // white
// Print centered text list($x, $y) = pc_ImagePSCenter($image, $text,
$font, $size);
ImagePSText($image, $text, $font, $size, $color, $bg_color, $x, $y);
// Send image
header('Content-type: image/png');
ImagePNG($image); // Clean up Image
PSFreeFont($font);
ImageDestroy($image);
}
else {
?>
<html>
<head>
<title>Sample Button Page</title>
</head>
<body>


</body>
</html>
<?php } ?>
```

In this script, if a value is passed in for `$_GET['button']`, we generate a button and send out the PNG. If `$_GET['button']` is not set, we print a basic HTML page with two embedded calls back to the script with requests for button images – one for a Previous button and one for a Next button. A more general solution is to create a separate `button.php` page that returns only graphics and set the image source to point at that page.

Self Assessment

State whether the following statements are true or false:

11. Building dynamic images with GD is difficult.
12. We define the text to come directly from the query string.

11.7 Scaling Images

One of the most basic features of computers today is the ability to edit graphics. Many times, you need to build web applications that take image data from users and scale it down to a format that can easily be displayed on your website. Fortunately, PHP ships with the very powerful GD Image Processing Library to provide all the powerful tools you need to accomplish this task. The most basic tasks in image processing: scaling.

Scaling images in PHP is quite easy, but there are some things to consider.

Read the Original Image with `imagecreatefromjpeg()`

First of all you'll need to read the original image. If it's a JPEG file the `imagecreatefromjpeg()` function is the right choice:

```
$source_image = imagecreatefromjpeg("osaka.jpg");
```

If it's a GIF file you'll take `imagecreatefromgif()`, and if it's a PNG you will prefer `imagecreatefrompng()`.

Get the Size of the Original Image: `getimagesize()` vs. `imagesx()` and `imagesy()`

Reading the image is quite easy, but the first pitfall you'll encounter if you prepare to scale an image, and need to find out the dimensions of the original image.

```
$source_image = imagecreatefromjpeg("osaka.jpg");
$source_image_size = getimagesize("osaka.jpg");
```

The problem with the `getimagesize()` function is that it needs to reopen the file to get the actual image size. This is usually not a big issue if you're reading a local file, but it will get critical if you're reading a file from the network like in:

```
$source_image = imagecreatefromjpeg("http://someurl/osaka.jpg");
$source_image_size = getimagesize("http://someurl/osaka.jpg");
```

Every time you call `getimagesize()` the whole image file get transferred over the network, and that quickly became mission-critical. Since you already have the image loaded with `imagecreatefromjpeg()` there is no need to load it again:

```
$source_image = imagecreatefromjpeg("osaka.jpg");
$source_imagex = imagesx($source_image);
$source_imagey = imagesy($source_image);
```

Create the Destination Image: `imagecreate()` vs. `imagecreatetruecolor()`

Now we need to prepare the (scaled) destination image. There are two PHP functions which can be used to create an image: `imagecreate()` and `imagecreatetruecolor()`. The first creates a palette based image (with a maximum of 256 different colors), and the second one creates a true color image (with a maximum - as far as I know - of $256 \times 256 \times 256 = 16$ million colors).

It's obvious: As long as you work with photographic images you'll need more than 256 colors.

So let's decide to use `imagecreatetruecolor()` and define a target size of 300x200 pixels:

```
$dest_imagex = 300;
$dest_imagey = 200;
$dest_image = imagecreatetruecolor($dest_imagex, $dest_imagey);
```

Notes

Scale the Image: imagecopyresized() vs. imagecopyresampled()

Now it's time to do the actual scaling of the image. Again PHP offers to function for this purpose: imagecopyresized() and imagecopyresampled(). The first one uses a very simple algorithm to scale the image, it's fast but the quality is really poor. The second one uses a better, but slower algorithm, resulting in a very high quality image.

Poor quality, but fast:

```
imagecopyresized($dest_image, $source_image, 0, 0, 0, 0,
                $dest_imagex, $dest_imagey, $source_imagex, $source_imagey);
```

Best quality, but slow:

```
imagecopyresampled($dest_image, $source_image, 0, 0, 0, 0,
                  $dest_imagex, $dest_imagey, $source_imagex, $source_imagey)
```

And Finally Push the Image to the Browser: imagejpeg() vs. imagepng()

After scaling the image, we now need to push the image to the user's browser. Probably the most popular image formats in the Internet are currently PNG and JPEG. Both will work great with photographic images but true-colored and loss-less PNG images usually results in larger file sizes than JPEG images.

To send a PNG image (with best compression rate 9) to the browser:

```
header("Content-Type: image/png");
imagepng($dest_image, NULL, 9);
```

Or a JPEG image (with best quality 100):

```
header("Content-Type: image/jpeg");
imagejpeg($dest_image, NULL, 100);
```

And in comparison, imagepng() on the left vs. imagejpeg() on the right

Self Assessment

Fill in the blanks:

- 13. One of the most basic features of computers today is the ability to edit.....
- 14. The problem with thefunction is that it needs to reopen the file to get the actual image size.

11.8 Colour Handling

HP (recursive acronym for 'PHP: Hypertext Preprocessor') is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

Firstly, the colours to be chosen should be defined. It would be possible to create a script that would generate any HTML colour at random, but a set of colours that are complementary to the site is a better choice. The colours should be assigned to variables with an arbitrary letter and a number, for example:

```
$c1 = "#ffffff";
$c2 = "#ffff99";
$c3 = "#00ccff";
```

```
$c4 = "#66ff99";
$c5 = "#ffcc00";
$c6 = "#d7ebff";
$c7 = "#ccffcc";
```

The numbers should start at 1, and increase by 1 each time.

11.8.1 Choosing a Colour

This will be based on the `rand()` function which chooses a random number between, and including, two specified numbers. The minimum number will be 1, but the maximum number may not always be the same; colours may be added or removed.

The number of the last colour variable must be manually entered each time the number of colours changes, so the maximum would be set as 7 using the above colours, as shown:

```
$number = rand(1,7);
```

The first number is the minimum, and the second number is the maximum.

Setting the Background Colour

The colour variable corresponding to the number chosen by `rand()` needs to be assigned to a variable, for example:

```
$bgcolour = "{$c$number}";
```

This takes the value of `$number`, puts it after a `c`, and makes it a variable. This means that `$bgcolour` will have the value of one of `$c1`, `$c2`, `$c3`, etc. depending on the chosen number.

The chosen colour must then be incorporated into the `<BODY>` tag. The easiest way to do this is to `echo()` the whole `<BODY>` tag at the end of the script, as shown:

```
echo ("<BODY BGCOLOR=\"{$bgcolour}\">");
```

The backslashes (`\`) before and after `$bgcolour` mean that the speech marks are output, rather than ending the printed text.

Finally,

The finished script should look something like:



Example:

```
<?php

$c1 = "#ffffff";
$c2 = "#ffff99";
$c3 = "#00ccff";
$c4 = "#66ff99";
$c5 = "#ffcc00";
$c6 = "#d7ebff";
$c7 = "#ccffcc";

$number = rand(1,7);
$bgcolour = "{$c$number}";
echo ("<BODY BGCOLOR=\"{$bgcolour}\">");
?>
```

11.8.2 Using the Alpha Channel

Alpha blending is a toggle that determines whether the alpha channel, if present, should be applied when drawing. If alpha blending is off, the old pixel is replaced with the new pixel. If an alpha channel exists for the new pixel, it is maintained, but all pixel information for the original pixel being overwritten is lost.



Example: Illustrates alpha blending by drawing a gray rectangle with a 50% alpha channel over an orange ellipse.



Example:

```
<?php
$im = ImageCreateTrueColor(150,150);
$white = ImageColorAllocate($im,255,255,255);
ImageAlphaBlending($im, false);
ImageFilledRectangle($im,0,0,150,150,$white);
$red = ImageColorResolveAlpha($im,255,50,0,63);
ImageFilledEllipse($im,75,75,80,50,$red);
$gray = ImageColorResolveAlpha($im,70,70,70,63);
ImageAlphaBlending($im, false);
ImageFilledRectangle($im,60,60,120,120,$gray);
header('Content-Type: image/png');
ImagePNG($im);
?>
```

11.8.3 Identifying Colours

To check the colour index for a specific pixel in an image, use `ImageColorAt()`:

```
$color = ImageColorAt(image, x, y);
```

For images with an 8-bit colour palette, the function returns a colour index that you then pass to `ImageColorsForIndex()` to get the actual RGB values:

```
$values = ImageColorsForIndex(image, index);
```

The array returned by `ImageColorsForIndex()` has keys “red”, “green”, and “blue”. If you call `ImageColorsForIndex()` on a colour from a true colour image, the returned array has an extra key, “alpha”.

11.8.4 True Colour Indexes

The color index returned by `ImageColorResolveAlpha()` is really a 32-bit signed long, with the first three 8-bit bytes holding the red, green, and blue values, respectively. The next bit indicates whether initializing is enabled for this color, and the remaining seven bits hold the transparency value.



Example:

```
$green = ImageColorResolveAlpha($im,0,0,255,127);
```

This code sets `$green` to 2130771712, which in hex is 0x7F00FF00 and in binary is 01111111000000001111111100000000.

This is equivalent to the following ImageColorResolveAlpha() call:

```
$green = 127<<24 | 0<<16 | 255<<8 | 0;
```

You can also drop the two 0 entries in this example and just make it:

```
$green = 127<<24 | 255<<8;
```

To deconstruct this value, you can use something like this:

```
$a = ($col & 0x7F000000) >> 24;
$r = ($col & 0x00FF0000) >> 16;
$g = ($col & 0x0000FF00) >> 8;
$b = ($col & 0x000000FF);
```

Direct manipulation of true color values like this is rarely necessary. One application is to generate a color-testing image that shows the pure shades of red, green, and blue.



Example:

```
$im = ImageCreateTrueColor(256,60);
for($x=0; $x<256; $x++) {
    ImageLine($im, $x, 0, $x, 19, $x);
    ImageLine($im, 255-$x, 20, 255-$x, 39, $x<<8);
    ImageLine($im, $x, 40, $x, 59, $x<<16);
}
ImagePNG($im);
```

11.8.5 Text Representation of an Image

An interesting use of the ImageColorAt() function is to loop through each pixel in an image and check the color, and then do something with that color data.



Example:

Converting an image to text

```
<html><body bgcolor=#000000><tt>
<?php
    $im = imagecreatefromjpeg('php-tiny.jpg');
    $dx = imagesx($im);
    $dy = imagesy($im);
    for($y = 0; $y < $dy; $y++) {
        for($x=0; $x < $dx; $x++) {
            $col = imagecolorat($im, $x, $y);
            $rgb = imagecolorsforindex($im,$col);
            printf('<font color=%02x%02x%02x>#</font>',
                $rgb['red'],$rgb['green'],$rgb['blue']);
        }
        echo "<br>\n";
    }
    imagedestroy($im);
?>
</tt></body></html>
```

Self Assessment

State whether the following statements are true or false:

15. Hypertext Preprocessor is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.
16. Alpha blending is a toggle that determines whether the alpha channel should be applied when drawing.



Case Study

History of Computer Graphics

The advance in computer graphics was to come from one MIT student, Ivan Sutherland. In 1961 Sutherland created another computer drawing program called Sketchpad. Using a light pen, Sketchpad allowed one to draw simple shapes on the computer screen, saved them and even recalled them later. The light pen itself had a small photoelectric cell in its tip. This cell emitted an electronic pulse whenever it was placed in front of a computer screen and the screen's electron gun fired directly at it. By simply timing the electronic pulse with the current location of the electron gun, it was easy to pinpoint exactly where the pen was on the screen at any given moment. Once that was determined, the computer could then draw a cursor at that location.

Sutherland seemed to find the perfect solution for many of the graphics problems he faced. Even today, many standards of computer graphics interfaces got their start with this early Sketchpad program. One example of this is in drawing constraints. If one wants to draw a square for example, she/he does not have to worry about drawing four lines perfectly to form the edges of the box. One can simply specify that she/he wants to draw a box, and then specify the location and size of the box. The software will then construct a perfect box, with the right dimensions and at the right location. Another example is that Sutherland's software modeled objects – not just a picture of objects. In other words, with a model of a car, one could change the size of the tires without affecting the rest of the car. It could stretch the body of the car without deforming the tires.

These early computer graphics were Vector graphics, composed of thin lines whereas modern day graphics are Raster based using pixels. The difference between vector graphics and raster graphics can be illustrated with a shipwrecked sailor. He creates an SOS sign in the sand by arranging rocks in the shape of the letters "SOS." He also has some brightly coloured rope, with which he makes a second "SOS" sign by arranging the rope in the shapes of the letters. The ropk SOS sign is similar to raster graphics. Every pixel has to be individually accounted for. The rope SOS sign is equivalent to vector graphics. The computer simply sets the starting point and ending point for the line and perhaps bends it a little between the two end points. The disadvantages to vector files are that they cannot represent continuous tone images and they are limited in the number of colours available. Raster formats on the other hand work well for continuous tone images and can reproduce as many colours as needed. Also in 1961 another student at MIT, Steve Russell, created the first video game, Spacewar. Written for the DEC PDP-1, Spacewar was an instant success and copies started flowing to other PDP-1 owners and eventually even DEC got a copy. The engineers at DEC used it as a diagnostic program on every new PDP-1 before shipping it. The sales force picked up on this quickly enough and when installing new units, would run the world's first video game for their new customers.

Contd...

Notes

E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called "Simulation of a two-giro gravity attitude control system" in 1963. In this computer generated film, Zajac showed how the attitude of a satellite could be altered as it orbits the Earth. He created the animation on an IBM 7090 mainframe computer. Also at BTL, Ken Knowlton, Frank Sinton and Michael Noll started working in the computer graphics field. Sinton created a film called Force, Mass and Motion illustrating Newton's laws of motion in operation. Around the same time, other scientists were creating computer graphics to illustrate their research. At Lawrence Radiation Laboratory, Nelson Max created the films, "Flow of a Viscous Fluid" and "Propagation of Shock Waves in a Solid Form." Boeing Aircraft created a film called "Vibration of an Aircraft."

It was not long before major corporations started taking an interest in computer graphics. TRW, Lockheed-Georgia, General Electric and Sperry Rand are among the many companies that were getting started in computer graphics by the mid 1960's. IBM was quick to respond to this interest by releasing the IBM 2250 graphics terminal, the first commercially available graphics computer. Ralph Baer, a supervising engineer at Sanders Associates, came up with a home video game in 1966 that was later licensed to Magnavox and called the Odyssey. While very simplistic, and requiring fairly inexpensive electronic parts, it allowed the player to move points of light around on a screen. It was the first consumer computer graphics product.

Also in 1966, Sutherland at MIT invented the first computer controlled head-mounted display (HMD). Called the Sword of Damocles because of the hardware required for support, it displayed two separate wireframe images, one for each eye. This allowed the viewer to see the computer scene in stereoscopic 3D. After receiving his Ph.D. from MIT, Sutherland became Director of Information Processing at ARPA (Advanced Research Projects Agency), and later became a professor at Harvard. Dave Evans was director of engineering at Bendix Corporation's computer division from 1953 to 1962. After which he worked for the next five years as a visiting professor at Berkeley. There he continued his interest in computers and how they interfaced with people. In 1968 the University of Utah recruited Evans to form a computer science program, and computer graphics quickly became his primary interest. This new department would become the world's primary research center for computer graphics.

In 1967 Sutherland was recruited by Evans to join the computer science program at the University of Utah. There he perfected his HMD. Twenty years later, NASA would rediscover his techniques in their virtual reality research. At Utah, Sutherland and Evans were highly sought after consultants by large companies but they were frustrated at the lack of graphics hardware available at the time so they started formulating a plan to start their own company. A student by the name of Ed Catmull got started at the University of Utah in 1970 and signed up for Sutherland's computer graphics class. Catmull had just come from The Boeing Company and had been working on his degree in physics. Growing up on Disney, Catmull loved animation yet quickly discovered that he did not have the talent for drawing. Now Catmull (along with many others) saw computers as the natural progression of animation and they wanted to be part of the revolution. The first animation that Catmull saw was his own. He created an animation of his hand opening and closing. It became one of his goals to produce a feature length motion picture using computer graphics. In the same class, Fred Parkes created an animation of his wife's face. Because of Evan's and Sutherland's presence, UU was gaining quite a reputation as the place to be for computer graphics research so Catmull went there to learn 3D animation.

As the UU computer graphics laboratory was attracting people from all over, John Warnock was one of those early pioneers; he would later found Adobe Systems and create a

Contd...

Notes

revolution in the publishing world with his PostScript page description language. Tom Stockham led the image processing group at UU which worked closely with the computer graphics lab. Jim Clark was also there; he would later found Silicon Graphics, Inc. The first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are “behind” the object from the viewer’s perspective, and thus should be “hidden” when the computer creates (or renders) the image.

Questions

1. Why the graphics was introduced? How it was developed?
2. Write the time-to-time advancement in graphics.

11.9 Summary

- GD has an existing API, and PHP tries to follow its syntax and function-naming conventions. So, if you are familiar with GD from other languages, such as C or Perl, you can easily use GD with PHP.
- PHP cleans up the image when the script ends, but, if you wish to manually deallocate the memory used by the image, calling `ImageDestroy($image)` forces PHP to get rid of the image immediately.
- The standard rectangular or Cartesian (named after René Descartes) has two perpendicular real number lines (axes) which divide the plane into four quadrants. The place where the number lines cross is the origin (0, 0) and the numbering is positive to the right and up; negative to the left and down.
- One of the most basic features of computers today is the ability to edit graphics. Many times, you need to build web applications that take image data from users and scale it down to a format that can easily be displayed on your website.
- Colour support improved markedly between GD 1.x and GD 2.x. In GD 1.x there was no notion of the alpha channel, colour handling was rather simple, and the library supported only 8-bit palette images (256 colours).
- An interesting use of the `ImageColorAt()` function is to loop through each pixel in an image and check the colour, and then do something with that colour data.

11.10 Keywords

.bmp: (pronounced “bimp”) This is a “bitmap.” You will probably never place a bitmap as an image, although some browsers do allow it.

.jpeg or .jpg: (pronounced “j-peg”) There are two names to denote this format because of the PC and MAC formats allowing 3 and 4 letters after the dot. JPEG is an acronym for Joint Photographic Experts Group, the organization that invented the format.

.png: Pronounced as ‘ping’, this stands for Portable Network Graphic and is ultimately the replacement for .gif, with partial transparency options, but browser support is sketchy – some browsers still do not like to display .png files.

Alpha Blending: Alpha blending is a toggle that determines whether the alpha channel, if present, should be applied when drawing.

ALT: It stands for “alternate text” and tells the browser that if it cannot find the image, then just displays this text. It also tells anyone who cannot view your image what the image is about.

Antialiasing: Antialiasing is where pixels at the edge of a shape are moved or recolored to make a gradual transition between the shape and its background.

Gif: This is pronounced “jif” or “gif” (hard “G”) depending on whom you speak to. “jif”, like the peanut butter. This is an acronym for Graphics Interchange Format.

HEIGHT: It stands for, as you might guess, the height of the image in pixels. Again, the height can be just about anything, but generally will be less than the height of the web browser.

image.gif: It is the name of the image. Notice it is following the same type of format as your HTML documents. There is a name (image) then a dot and then there is a suffix (gif).

Image: An image is a rectangle of pixels that have various colours. Colours are identified by their position in the palette, an array of colours.

IMG: It stands for “image.” It announces to the browser that an image will go here on the page.

SRC: It stands for “source.” This again is an attribute, a command inside a command. It is telling the browser where to go to find the image. Again, it is best for you to place the images you want to use in a subdirectory called “images”.

WIDTH: It stands for just that, the width of the image in pixels. It can range from 1 pixel to, well, just about any number, but generally will be less than the width of the web browser.

11.11 Review Questions

1. What is the meaning of Graphics? How it is used with PHP? Explain with example.
2. How an image embedding in a page with PHP?
3. How an Image converts into a Link? Explain with example.
4. Why we use GD library? Explain the GD extensions with example.
5. What are the basic concepts of graphics? Explain briefly.
6. What is the process to create and draw the images with graphics?
7. How the images embedded with text? Explain with example.
8. How we create dynamic buttons in Graphics? Explain with example.
9. What is the meaning of scaling of image? How does it perform in Graphics? Explain with example.
10. How the colour handlings proceed in graphics? Write a program to create a red rectangle on white background.

Answers: Self Assessment

- | | |
|-------------------|------------------|
| 1. True | 2. False |
| 3. GD library | 4. Configuration |
| 5. True | 6. False |
| 7. True | 8. True |
| 9. Imagettftext() | 10. Three |

Notes

- | | |
|--------------|--------------------|
| 11. False | 12. True |
| 13. Graphics | 14. Getimagesize() |
| 15. True | 16. True |

11.12 Further Readings



Books

- Bangia, Ramesh (2008). *“Web Technology (including HTML, CSS, XML, ASP, JAVA).”* Firewall Media.
- Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.
- Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.
- Puntambekar, A.A. (2009). *“Web Technologies.”* Technical Publications.
- Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.
- Xavier, C. (2007). *“Web Technology and Design.”* New Age International.



Online links

- <http://php.net/manual/en/intro.cairo.php>
- http://www.w3schools.com/tags/tag_img.asp
- <http://www.scantips.com/basics09.html>
- <http://www.hyperlinkcode.com/html-image-hyperlink.php>
- <http://ecomputernotes.com/computer-graphics/basic-of-computer-graphics/what-is-a-pixel-in-computer-graphics>
- <http://www.thesitewizard.com/php/create-image.shtml>
- <http://www.alphadevx.com/a/56-Basic-Shapes-in-PHP-GD>
- <http://www.phpforkids.com/php/php-gd-library-adding-text-writing.php>
- https://blogs.oracle.com/oswald/entry/scaling_images_in_php_done
- <http://www.bbc.co.uk/dna/place-lancashire/plain/A1021645>

Unit 12: Portable Document Format

Notes

CONTENTS

Objectives

Introduction

12.1 PDF Extensions

12.2 Documents and Pages

12.2.1 Initializing the Document

12.2.2 Setting Metadata

12.2.3 Creating a Page

12.2.4 Outputting Basic Text

12.2.5 Terminating and Streaming a PDF Document

12.3 Text

12.3.1 Coordinates

12.3.2 Text Functions

12.3.3 Text Attributes

12.3.4 Fonts

12.3.5 Embedding Fonts

12.4 Images and Graphics

12.4.1 Images

12.4.2 Graphics

12.4.3 Patterns

12.4.4 Templates

12.5 Navigation

12.5.1 Bookmarks and Thumbnails

12.5.2 Links

12.6 Summary

12.7 Keywords

12.8 Review Questions

12.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Discuss about the Portable Document Format Extensions
- Understand the Uses of Documents and Pages

Notes

- Understand the Role of Text in Portable Document Format
- Explain the Uses of Images and Graphics
- Understand the Navigation

Introduction

PDF stands for Portable Document Format. As the name implies, it is a data format that can be used to describe documents. Adobe, the developers of PDF, market software to create, edit and visualize PDF files. Because the specifications of the file format are publicly available and meanwhile even became an official ISO-standard, a lot of other companies develop PDF-related software as well. In pre-press, PDF is commonly used as a format to exchange data, either complete pages that need to be printed or advertisements that needs to be included in a publication. The file format is also popular for soft proofing and reviewing content, because there are applications that allow you to make annotations on the PDF pages.

12.1 PDF Extensions

A common problem when developing a web application is having producing a high-quality PDF out of an existing layout/view/template. Perhaps for a reporting engine, an invoice, a receipt, or any number of other situations.

Often this involves using somewhat cryptic output primitives and creating the PDF by hand. Wouldn't it be nice if there were a way to re-use all that beautiful HTML, CSS, and maybe even Javascript that you already wrote?

Well, there is. It's called wkhtmltopdf. Normally a command line utility, with the release of 0.10.0_beta5 antialize included a simple C API to be able to build bindings in other popular languages.



Example:

```
<?php
    wkhtmltox_convert('pdf',
        array('out' => 'test.pdf', 'imageQuality' => '95'), // global settings
        array(
            array('page' => 'http://www.visionaryrenesis.com/'),
            array('page' => 'http://www.google.com/', 'web.printMediaType' =>
true)
        )
    );

?>
```

PDF has a unique number of advantages:

- It is a cross platform standard. This means that somebody can create a PDF file on a Unix workstation and you can open it on a Mac or PC and still see the document just like it was intended to be viewed.
- PDF files can be device independent. They can be printed on a cheap ink jet printer as well as on an expensive imagesetter. This does not necessarily mean that the output will be optimized for each device. A lot depends on the way the document is created.

- PDF files are compact. PDF supports a number of sophisticated compression algorithms as well as a clever file structure to keep the file size of PDF files down to an absolute minimum.
- PDF files can contain multimedia elements like movies or sound as well as hypertext elements like bookmarks, links to e-mail addresses or web pages and thumbnail views of pages.
- PDF supports security. The creator of a PDF file can set various security options. It is possible to lock a PDF so it can only be opened with a password. It is also possible to forbid changing the content of a PDF or disable the option to print a PDF file.



Did u know? The PDF format is so powerful and has been adopted around the world because it acts as a container for various types of content.

Self Assessment

State whether the following statements are true or false:

1. A common problem when developing a web application is having producing a low-quality PDF out of an existing layout.
2. PDF is a cross platform standard.
3. PDF files can be device dependent.

12.2 Documents and Pages

PDFs have become widespread enough among businesses and institutions to make automatically and dynamically producing them in large quantity is a valuable skill to learn. Portable Document Files from Adobe are easily produced in PHP, making it an even better glue language than before. Since anyone can install PHP, they can use the simple PHP 5 class of PDFlib that's provided to make as many PDFs in whatever format you need.

The irony is, of course, PDFlib is now commercial and offers a PDFlib Lite. Either way, the PDFlib library has an unwieldy learning curve, and instead. This is a freeware PDF library written completely in PHP, so no PECL or PEAR installation is required, which does slow it down. So, a large business shouldn't use this to process hundreds of PDFs, but for smaller jobs it works.



Example:

```
<?php
require('classes/fpdf/fpdf.php');
class PDF extends FPDF {

function Header() {
    $this->SetFont('Times','',12);
    $this->SetY(0.25);
    $this->Cell(0, .25, "John Doe ".$this->PageNo(), 'T', 2, "R");
    //reset Y
    $this->SetY(1);
}
}
```

Notes

```
function Footer() {
//This is the footer; it's repeated on each page.
//enter filename: phpjabber logo, x position: (page width/2)-half the
picture size,
//y position: rough estimate, width, height, filetype, link: click it!
    $this->Image("logo.jpg", (8.5/2)-1.5, 9.8, 3, 1, "JPG", "http://
www.phpjabbers.com");
}

}

//class instantiation
$pdf=new PDF("P","in","Letter");

$pdf->SetMargins(1,1,1);

$pdf->AddPage();
$pdf->SetFont('Times','',12);

$lipsum1="Lorem ipsum dolor sit amet, nam aliquam dolore est, est in
eget.";

$lipsum2="Nibh lectus, pede fusce ullamcorper vel porttitor.";

$lipsum3 ="Duis maecenas et curabitur, felis dolor.";

$pdf->SetFillColor(240, 100, 100);
$pdf->SetFont('Times','BU',12);

//Cell(float w[,float h[,string txt[,mixed border[,
//int ln[,string align[,boolean fill[,mixed link]]]]]])
$pdf->Cell(0, .25, "lipsum", 1, 2, "C", 1);

$pdf->SetFont('Times','',12);
//MultiCell(float w, float h, string txt [, mixed border [, string align [,
boolean fill]])
$pdf->MultiCell(0, 0.5, $lipsum1, 'LR', "L");
$pdf->MultiCell(0, 0.25, $lipsum2, 1, "R");
$pdf->MultiCell(0, 0.15, $lipsum3, 'B', "J");

$pdf->AddPage();
$pdf->Write(0.5, $lipsum1.$lipsum2.$lipsum3);

$pdf->Output();
?>
```

On above create a PDF in Portrait mode with a Letter size and measured in inches. Then, set the margins to 1 inch, add our first page, and then set the Font to Times, Size 12. Then, my 3 dummy texts, and after that set the fill color that will be the space behind my title's text with a bold/underlined modifier. After that, print the title, then reset the font and specify some different types of MultiCells. The page is made up of cells on an x, y plane in FPDF, which makes it pretty convenient to work with for basic PDF editing.

The headers and footers are functions automatically called in the parent class, but they do nothing by default. That means we have to redefine them to our needs. In this case, a name and page number as the header and a logo with a link as the footer. These are repeated on each page.



Notes In order to place the text in the proper location, we have to use the SetX and SetY functions. Don't forget to reset them to their normal flow when you're done in the header or footer, though!

Cells are individual units on the page. It is convenient to define a line with a single cell, but at the same time this can be achieved much more quickly by use of the MultiCell function which will automatically create a new cell as text is passed through. Write is just streamed text. One thing to absolutely keep in mind is that you can't output any text to the browser when trying to send this PDF to the browser, if you're making it on the fly. If you're making it for storage or later use, then there's no problem outputting text to the user as well. This is accomplished by setting the parameters to Output appropriately.

The spacing is done with the line height. You have to do some math to figure out what you want. The line height is the entire height of the line, which means the lines are up against each other. That means whatever you set for the line height is spacing between the letters, as well. With these skills in mind, you should be able to do whatever you want to a PDF. For even more functionality, there is a Scripts section, but of course there is always the PDFlib library which provides all kinds of specialization. FPDF is a highly documented library though, and it's really simple to use.

12.2.1 Initializing the Document

As mentioned earlier, there are different settings for the measurement units on the PDF pages within FPDF. You can control them by sending parameters to the constructor when you instantiate a new copy of the class. Previously, you saw the `$pdf = new FPDF();` statement of instantiation, which creates an object with the default attributes. You can send the following parameters into the constructor:

- The page orientation has the options of portrait (P) or landscape (L), portrait being the default.
- The units of measurement have the options of point (pt), millimeter (mm), centimeter (cm), and inches (in), with millimeter as the default.
- The page size has the options of A3, A4, A5, Letter, and Legal, with A4 as the default.

Here is a constructor call that defines portrait, inches, and letter layout:

```
$pdf = new FPDF('P', 'in', 'Letter');
```

12.2.2 Setting Metadata

The `pdf_set_info()` function inserts information fields into the PDF file:

```
pdf_set_info(pdf, fieldname, value);
```

Notes

There are five standard field names: Subject, Author, Title, Creator, and Keywords. You can also add arbitrary information fields.

In addition to informational fields, the pdflib library has various parameters that you can change with pdf_get_parameter() and pdf_set_parameter():

```
$value = pdf_get_parameter(pdf, name); pdf_set_parameter(pdf, name, value);
```

A useful parameter to set is openaction, which lets you specify the zoom (magnification) of the file when it's opened. The values "fitpage", "fitwidth", and "fitheight" fit the file to the complete page, the width of the page, and the height of the page, respectively. If you don't set open action, your document is displayed at whatever zoom the viewer had set at the time the document was opened.

12.2.3 Creating a Page

To create a new page in the document. The function used for it is PDF_begin_page();

The basic syntax is given below is

```
PDF_begin_page($objPDF, $width, $height);
```

Where \$width and \$height are width and height of the page. The table below lists common standard page sizes.

Table 12.1: Standard Page Sizes

format	width	height	format	width	height	format	width	height
A0	2380	3368	A4	595	842	letter	612	792
A1	1684	2380	A5	421	595	legal	612	1008
A2	1190	1684	A6	297	421	ledger	1224	792
A3	842	1190	B5	501	709	11x17	792	1224

12.2.4 Outputting Basic Text

To put text on a page, you must select the font you want to use, set the default font to be that font at a particular size, and then add the text.



Example:

```
$font = pdf_findfont($pdf, "Times-Roman", "host", 0);
pdf_setfont($pdf, $font, 48);
pdf_show_xy($pdf, "Hello, World", 200, 200);
```

With PDF documents, the (0, 0) coordinate indicates the bottom-left corner of the page.

12.2.5 Terminating and Streaming a PDF Document

Call pdf_close() to complete the PDF document. If no filename was provided in the pdf_open_file() call, you can now use the pdf_get_buffer() function to fetch the PDF buffer from memory. To send the file to the browser, you must send Content-Type, Content-Disposition, and Content-Length HTTP headers, as shown in Example Finally, call pdf_delete() to free the PDF file once it's sent to the browser.



Example:

```
<?php
$pdf = pdf_new( );
pdf_open_file($pdf);
pdf_set_info($pdf,'Creator','hello.php');
pdf_set_info($pdf,'Author','Rasmus Lerdorf');
pdf_set_info($pdf,'Title','Hello world (PHP)');
pdf_begin_page($pdf,612,792);

$pdf = pdf_findfont($pdf,'Helvetica-Bold','host',0);
pdf_setfont($pdf,$font,38.0);
pdf_show_xy($pdf,'Hello world!',50,700);

pdf_end_page($pdf);
pdf_set_parameter($pdf, "openaction", "fitpage");
pdf_close($pdf);

$buf = pdf_get_buffer($pdf);
$len = strlen($buf);
header('Content-Type: application/pdf');
header("Content-Length: $len");
header('Content-Disposition: inline; filename=hello.pdf');
echo $buf;
pdf_delete($pdf);
?>
```

Self Assessment

Fill in the blanks:

4.are individual units on the page.
5. The headers and footers are functions automatically called in the.....
6. The PDFlib library has an unwieldycurve.

12.3 Text

Text is the heart of a PDF file. As such, there are many options for changing the appearance and layout of text. We will discuss over here coordinate system used in PDF documents, functions for inserting text and changing text attributes, and font usage.

12.3.1 Coordinates

In dynamic generation of PDF files from within PHP, It is often refer to the unit points when discussing sizes or positions within a PDF document. A point in PDF documents is a unit of measure: 1 printed inch = 72 points. Thus, there are 72 points to every printed inch in a PDF document. All PDF documents use points to represent distances and dimensions, and unless

Notes

specifically instructed otherwise, points should be used in any PDF-related function where a location or distance is required.

By the same token, the coordinate system used by PDFLib documents is often a further point of confusion. In PDFLib PDF documents, the X-axis coordinates increase from left to right. However, Y-axis coordinates begin from the bottom and increase toward the top of a given page. Thus, the location (0,0) does not refer to the upper-left corner of the page; it refers to the lower-left corner.



Task Develop a PHP program to show the use of text functions in PDF.

12.3.2 Text Functions

When you are working with PDFLib, a great number of global settings affect the way a PDF document will be rendered. Some of these settings affect the document on a global scale and some affect specific parts, such as text rendering.

To determine the value of these settings or to change them, you should consider four functions. These functions are `pdf_get_parameter()`, `pdf_set_parameter()`, `pdf_get_value()`, and `pdf_set_value()`.

The syntax for these function is as follows:

```
pdf_set_parameter($pdf_r, $parameter, $value)
pdf_set_value($pdf_r, $parameter, $value)
pdf_get_parameter($pdf_r, $parameter);
pdf_get_value($pdf_r, $parameter);
```

For each of the preceding functions, the `$pdf_r` parameter represents the PDF Object resource. `$parameter` is a string representing the name of the parameter. For the `pdf_set_*` functions, `$value` represents the new value to set the setting to. Finally, both `pdf_get_*` functions return the value of the specified parameter.



Notes As you have noticed, two sets of functions seemingly provide the same functionality to PHP (`pdf_*_parameter()` and `pdf_*_value()`). Although PHP is an untyped language (meaning that there is no difference between the string “1” and the integer value 1), the underlying PDFLib library does distinguish between the two. Hence, when setting parameters that are expected to be strings by PDFLib, the `pdf_*_parameter()` family is expected to be used, whereas the `pdf_*_value()` family is used for integer values. It is important to use the appropriate function or the results will not be as expected.

12.3.3 Text Attributes

There are three common ways to alter the appearance of text. One is to underline, overline, or strike out the text using parameters. Another is to change the stroking and filling. The third is to change the text’s color.

Each of the underline, overline, and strikethrough parameters may be set to “true” or “false” independently of the others.



Example:

```
pdf_set_parameter($pdf, "underline", "true"); // enable underlining
```

Stroking text means drawing a line around the path defined by the text. The effect is an outline of the text.

Filling text means to fill the shape defined by the text.

You can set whether text should be stroked or filled with the `textrendering` parameter. The valid values are shown in Table 12.2.

Table 12.2: Values for the Text Rendering Parameter

Value	Effect
0	Normal
1	Stroke (outline)
2	Fill and stroke
3	Invisible
4	Normal, add to clipping path
5	Fill and stroke, add to clipping path
6	Invisible, add to clipping path

The type parameter is either "stroke", "fill", or "both", indicating whether you're specifying the color to be used for outlining the letters, filling the letters, or both. The colorspace parameter is one of "gray", "rgb", "cmyk", "spot", or "pattern". The "gray", "spot", and "pattern" colorspaces take only one color parameter, whereas "rgb" takes three and "cmyk" takes all four. Example shows colors, underlines, overlines, strikeouts, stroking, and filling at work. Changing text attributes.



Example:

```
<?php
$p = pdf_new( );
pdf_open_file($p);
pdf_begin_page($p, 612, 792);
$font = pdf_findfont($p, "Helvetica-Bold", "host", 0);
pdf_setfont($p, $font, 38.0);
pdf_set_parameter($p, "overline", "true");
pdf_show_xy($p, "Overlined Text", 50, 720);
pdf_set_parameter($p, "overline", "false");
pdf_set_parameter($p, "underline", "true");
pdf_continue_text($p, "Underlined Text");
pdf_set_parameter($p, "strikeout", "true");
pdf_continue_text($p, "Underlined strikeout Text");
pdf_set_parameter($p, "underline", "false");
pdf_set_parameter($p, "strikeout", "false");
pdf_setcolor($p, "fill", "rgb", 1.0, 0.1, 0.1);
pdf_continue_text($p, "Red Text");
```

Notes

```
pdf_setcolor($p,"fill","rgb", 0, 0, 0);
pdf_set_value($p,"textrendering",1);
pdf_setcolor($p,"stroke","rgb", 0, 0.5, 0);
pdf_continue_text($p, "Green Outlined Text");
pdf_set_value($p,"textrendering",2);
pdf_setcolor($p,"fill","rgb", 0, .2, 0.8);
pdf_setlinewidth($p,2);
pdf_continue_text($p, "Green Outlined Blue Text");
pdf_end_page($p);
pdf_close($p);
$buf = pdf_get_buffer($p);
$len = strlen($buf);
header("Content-Type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=coord.pdf");
echo $buf; pdf_delete($p);
?>
```



Caution You need to be careful of not violating any font license terms, because some fonts are not supposed to be embedded.

12.3.4 Fonts

The Base14 fonts are built-in into PDF and all viewers can display them. Using these fonts may decrease the size of the result file and make the processing faster, avoiding loading external fonts. However the fonts support only Latin1 character set and you have to load external fonts if you need to use another character set.

The Base 14 fonts:

- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Symbol
- ZapfDingbats

12.3.5 Embedding Fonts

To make a PDF portable across different systems, it may be necessary to embed the fonts used in the document in the PDF itself (sometimes this is required when submitting documents for publication, etc.).

1. Open the original PDF in the Preview application.
2. Select File: Print
3. Click on the Preview button – a new document should be generated.
4. Select File: Print again (you should be in the newly-created document when you do this).
5. Click the PDF button, and then select Save as PDF from the pop-up menu.
6. Choose a filename and save the file.

The file you just saved should have all of the fonts embedded (at least, that has been my experience, but your mileage may vary).

Self Assessment

State whether the following statements are true or false:

7. Text is the heart of a PDF file.
8. In PDFLib PDF documents, the Y-axis coordinates increase from left to right.
9. Using the Base 14 fonts may decrease the size of the result file and make the processing faster, avoiding loading external fonts.

12.4 Images and Graphics

PHP employs the Boutell GD library to provides a wealth of functions for generating and manipulating images. This works in conjunction with a host of other libraries for handling image formats and fonts.

- JPEG, PNG, BMP, WBMP and TIFF formats
- TrueType, FreeType and other fonts

Image support in PHP is a bit of a potpourri assembled at build time. As well as the GD functions there is support for PDF (Acrobat), Shockwave (Flash) and Ming, an open source library that allows creation of SWF formats. Expect some variations across different PHP installations. Windoze tends to install from binary and arrive fully loaded, which can be rather nice. Windows also tends to be less painful than Unix when it comes to fonts.

12.4.1 Images



Example:

```
<?php
$my_img = imagecreate( 200, 80 );
$background = imagecolorallocate( $my_img, 0, 0, 255 );
$text_colour = imagecolorallocate( $my_img, 255, 255, 0 );
$line_colour = imagecolorallocate( $my_img, 128, 255, 0 );
```

Notes

```
imagestring( $my_img, 4, 30, 25, "thesitewizard.com",
    $text_colour );
imagesetthickness ( $my_img, 5 );
imageline( $my_img, 30, 45, 165, 45, $line_colour );

header( "Content-type: image/png" );
imagepng( $my_img );
imagecolordeallocate( $line_color );
imagecolordeallocate( $text_color );
imagecolordeallocate( $background );
imagedestroy( $my_img );
?>
```

The above code creates a 200x80 PNG image with a blue background and yellow text. It can be called from within your web page simply by referencing the php file. For example, if the PHP file that contains the above code is called myimage.php, then the HTML code to invoke it can simply be:

```

```

Creating the Image

The first thing the code does is to call the `imagecreate()` function with the dimensions of the image, namely its width and height in that order. This function returns a resource identifier for the image which we save in `$my_img`. The identifier is needed for all our operations on the image.

If the function fails for some reason, it will return `FALSE`. If you want your code to be robust, you should test for this.

Using Colours in PHP

Before you can use any sort of colours in your image at all, you will need to allocate the colour. Colours are represented by three digits, known as the RGB value. The first digit denotes the red component, the second the green and the third blue, hence RGB, for Red-Green-Blue. These are the same colour values that you use for your web page as well as numerous other computer applications.

Colours are allocated using the `imagecolorallocate()` function. This function will automatically fill the background of the image with the colour the first time you call it, as well as return an identifier for that particular colour. Subsequent calls to `imagecolorallocate()` will simply create a colour identifier for your colour, without affecting your image background.

As you can see from the above code, my script allocates a blue identifier for the image, and in so doing, causes `imagecolorallocate()` to set the background to blue automatically. It also allocates a colour identifier for yellow and one for a shade of green. The latter two identifiers will be used later to write text and draw a line.

`imagecolorallocate()` returns `FALSE` if the function fails for any reason.

Writing Text to the Image

To write text to your image, you will need to use the `imagestring()` function. This function uses a set of built-in fonts to do the writing. The fonts have various sizes, ranging from 1 to 5, where

1 is the smallest font size and 5 the largest. The size of the font is specified in the second parameter to the function. In my example, I used font size 4.

The third and fourth parameters to `imagestring()` specify the x, y coordinate for the top left hand corner of the text. In the case of the example above, my text will begin 25 pixels from the top edge of the image, and 30 pixels from the left.

The fifth parameter is for the text to print, and the final parameter the colour of the text. This is the same colour that was allocated earlier using `imagecolorallocate()`.

Drawing a Line and Setting the Thickness of the Brush

The `imageline()` function can be used to draw a line to the image. To set the thickness of the brush used to draw the line, you may want to call the `imagesetthickness()` function in my example. The numeric parameter to `imagesetthickness()` is the thickness of the brush in pixels. In my code, I set it to 5 pixels. If you don't call `imagesetthickness()`, the line will be 1 pixel thick.

The `imageline()` function is called with the start and end coordinates of the line, in x, y format. In my code, the line starts from 30, 45 and ends on 165, 45. That is, it will be a horizontal line 45 pixels from the top, starting 30 pixels from the left edge and ending 165 pixels from that same edge. Since `$line_colour` was set to a shade of green earlier, the line will be green.



Did u know? The scale parameter indicates the proportional scaling factor to be used when placing the image in the document.

12.4.2 Graphics

To draw a graphical shape, first specify a path and then fill and/or stroke the path with appropriately configured fill and/or stroke colors. The functions that define these paths are straightforward. For example, to draw a line, you position the cursor at the starting point of the line using a call to `pdf_moveto()`, then specify the path for this line with a call to `pdf_lineto()`. The starting points of other functions, such as `pdf_circle()` and `pdf_rect()`, are defined directly in the calls.

The `pdf_moveto()` function starts the path at a particular point:

```
pdf_moveto(pdf, x, y);
```

With `pdf_lineto()`, you can draw a line from the current point to another point:

```
pdf_lineto(pdf, x, y);
```

Use `pdf_circle()` to draw a circle of radius r at a particular point:

```
pdf_circle(pdf, x, y, r);
```

The `pdf_arc()` function draws an arc of a circle:

```
pdf_arc(pdf, x, y, r, alpha, beta);
```

The circle is centered at (x, y) and has radius r . The starting point of the arc is α degrees (measured counterclockwise from the horizontal axis), and the endpoint is β degrees.

Use `pdf_curveto()` to draw a Bézier curve from the current point:

```
pdf_curveto(pdf, x1, y1, x2, y2, x3, y3);
```

The points (x_1, y_1) , (x_2, y_2) , and (x_3, y_3) are control points through which the curve must pass.

Notes

You can draw a rectangle with `pdf_rect()`:

```
pdf_rect(pdf, x, y, width, height);
```

To draw a line from the current point back to the point that started the path, use `pdf_closepath()`:

```
pdf_closepath(pdf);
```

12.4.3 Patterns

A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts.

“Each pattern is a three part rule, which expresses a relation between a certain context, a problem, and a solution.”

This “pattern as a rule” definition has been highly influential. In addition, a pattern is also a rule that explains how to create the particular configuration which resolves the forces within that context.

The `pdf_begin_pattern()` call returns a pattern handle:

```
$pattern = pdf_begin_pattern(pdf, width, height, xstep, ystep, painttype);
```

The width and height parameters specify the size of the pattern. If you are creating a pattern from an image, these are the dimensions of the image. The `xstep` and `ystep` parameters specify the horizontal and vertical tiling spacing (i.e., the distance between repetitions of the image). To tile the image without a gap between repetitions, set the `xstep` and `ystep` arguments to the same values as width and height. The final argument, `painttype`, can be either 1 or 2. 1 means that the pattern supplies its own color information. 2 means that the current fill and stroke colors are used instead. Patterns based on images only use a `painttype` of 1.

12.4.4 Templates

It is common to have parts of a document, such as header/footer sections or background watermarks, repeated on multiple pages. It would be trivial to write a little PHP function to generate such things on each page, but if you did this the final PDF file would end up containing the same sequence of PDF calls on every page. PDF has built-in functionality known as “Form XObjects” (renamed “Templates” in `pdflib`) to more efficiently handle repeating elements.

To create a template, simply call `pdf_begin_template()`, perform the various operations to create the PDF components you want this template to contain, then call `pdf_end_template()`. It is a good idea to do a `pdf_save()` right after beginning the template and a `pdf_restore()` just before ending it to make sure that any context changes you perform in your template don’t leak out of this template into the rest of the document.

The `pdf_begin_template()` function takes the dimensions of the template and returns a handle for the template:

```
$template = pdf_begin_template(pdf, width, height);
```

The `pdf_end_template()`, `pdf_save()`, and `pdf_restore()` functions take no arguments beyond the pdf handle:

```
pdf_end_template(pdf); pdf_save(pdf); pdf_restore(pdf);
```

The below Example uses templates to create a two-page document with the PHP logo in the top-left and top-right corners and the title “pdf Template Example” and a line at the top of each page. If you wanted to add something like a page number to your header, you would need to do that on each page. There is no way to put variable content in a template.



Example:

```
<?php
$p = pdf_new( );
pdf_open_file($p); // define template
$im = pdf_open_jpeg($p, "php-big.jpg");
$template = pdf_begin_template($p,612,792);
pdf_save($p);
pdf_place_image($p, $im, 14, 758, 0.25);
pdf_place_image($p, $im, 562, 758, 0.25);
pdf_moveto($p,0,750); pdf_lineto($p,612,750);
pdf_stroke($p);
$font = pdf_findfont($p,"Times-Bold","host",0);
pdf_setfont($p,$font,38.0);
pdf_show_xy($p,"pdf Template Example",120,757);
pdf_restore($p);
pdf_end_template($p);
pdf_close_image ($p,$im);// build pages
pdf_begin_page($p,595,842);
pdf_place_image($p, $template, 0, 0, 1.0);
pdf_end_page($p);
pdf_begin_page($p,595,842);
pdf_place_image($p, $template, 0, 0, 1.0);
pdf_end_page($p);
pdf_close($p);
$buf = pdf_get_buffer($p);
$len = strlen($buf);
header("Content-Type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=templ.pdf");
echo $buf; pdf_delete($p);
?>
```

Self Assessment

Fill in the blanks:

10. Ais the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts.
11. Windows also tends to be less painful thanwhen it comes to fonts.
12.are allocated using the imagecolorallocate() function.

12.5 Navigation

PDF provides several navigation features for PDF files. Bookmarks function as a table of contents for the document, and you can provide viewers with thumbnail images indicating what's at the

Notes

other end of each bookmark. In addition, any part of a PDF page can be linked to another part of the current PDF file, another PDF file, or a completely different file.

12.5.1 Bookmarks and Thumbnails

Bookmarks make it easy to quickly navigate through long PDF documents. You can create a bookmark with the `pdf_add_bookmark()` function, which returns a bookmark handle:

```
$bookmark = pdf_add_bookmark(pdf, text, parent, open);
```

The text parameter is the label that the user sees. To create a nested menu of bookmarks, pass a bookmark handle as the parent option. The current location in the PDF file (as it is being created) is the destination of the bookmark. Bookmarks can have thumbnails associated with them. To make a thumbnail, load an image and call `pdf_add_thumbnail()`:

```
pdf_add_thumbnail(pdf, image);
```

The Example creates a top-level bookmark named “Countries” and nests two bookmarks, “France” and “India”, under the “Countries” bookmark. It also creates a representative thumbnail image for each page. These thumbnails can be viewed in Acrobat Reader’s thumbnail panel.



Example:

```
<?php
$p = pdf_new( );
pdf_open_file($p);
pdf_begin_page($p,595,842);
$top = pdf_add_bookmark($p, "Countries");
$im = pdf_open_png($p, "fr-flag.png");
pdf_add_thumbnail($p, $im);
pdf_close_image($p,$im);
$font = pdf_findfont($p,"Helvetica-Bold","host",0);
pdf_setfont($p, $font, 20);
pdf_add_bookmark($p, "France", $top);
pdf_show_xy($p, "This is a page about France", 50, 800);
pdf_end_page($p);
pdf_begin_page($p,595,842);
$im = pdf_open_png($p, "nz-flag.png");
pdf_add_thumbnail($p, $im);
pdf_close_image($p,$im);
pdf_setfont($p, $font, 20);
pdf_add_bookmark($p, "India", $top);
pdf_show_xy($p, "This is a page about India", 50, 800);
pdf_end_page($p); pdf_close($p);
$buf = pdf_get_buffer($p);
$len = strlen($buf); header("Content-Type:application / pdf");
header("Content-Length:$len");
header("Content-Disposition: inline; filename=bm.pdf");
echo $buf; pdf_delete($p);
?>
```

12.5.2 Links

Notes

pdflib supports functions that specify a region on a page that, when clicked on, takes the reader somewhere else. The destination can be either another part of the same document, another PDF document, some other application, or a web site.

The `pdf_add_locallink()` function adds a local link to another place within the current PDF file:

```
pdf_add_locallink(pdf, llx, lly, urx, ury, page, zoom);
```

All links in PDF files are rectangular. The lower-left coordinate is (llx,lly) and the upper-right coordinate is (urx,ury). Valid zoom values are "retain", "fitpage", "fitwidth", "fitheight", and "fitbbox".

The following call defines a 50 x 50 area that, if clicked, takes the reader to page 3 and retains the current zoom level:

```
pdf_add_locallink($p, 50, 700, 100, 750, 3, "retain");
```

The `pdf_add_pdflink()` function adds a link to another PDF file. It takes the same parameters as the `pdf_add_locallink()` function, with the addition of a new parameter containing the filename to link to:

```
pdf_add_pdflink(pdf, llx, lly, urx, ury, filename, page, zoom);
```



Example:

```
pdf_add_pdflink($p, 50, 700, 100, 750, "another.pdf", 3, "retain");
```

The `pdf_add_launchlink()` function adds a link to another file, whose MIME type causes the appropriate program to be launched to view the file:

```
pdf_add_launchlink($p, 50, 700, 100, 750, "/path/document.doc");
```

The `pdf_add_weblink()` function creates a link whose destination is a URL:

```
pdf_add_weblink(pdf, llx, lly, urx, ury, url);
```

Self Assessment

State whether the following statements are true or false:

13. PDF provides only one navigation feature for PDF files.
14. Bookmarks make it complex to quickly navigate through long PDF documents.
15. The text parameter is the label that the user sees.



Case Study

The History of PDF

The paperless office. Remember that buzz word that never seems to vanish completely even though history has proven that the use of computers has until now only lead to an increase in the use of paper? PDF started off on the dream of a paperless office, as the pet project of one of Adobe's founders, John Warnock. Initially it was an internal project at Adobe to create a file format so documents could be spread throughout the company and displayed on any computer using any operating system. In his paper which led to the development of PDF, John Warnock wrote: 'Imagine being able to send full text

Contd...

Notes

and graphics documents (newspapers, magazine articles, technical manuals etc.) over electronic mail distribution networks. These documents could be viewed on any machine and any selected document could be printed locally. This capability would truly change the way information is managed.'

PDF 1.0

The first time Adobe actually talked about this technology was at a Seybold conference in San Jose in 1991. At that time, it was referred to as 'IPS' which stood for 'Interchange PostScript.' Version 1.0 of PDF was announced at Comdex Fall in 1992 where the technology won a 'best of Comdex' award. The tools to create and view PDF-files, Acrobat, were released in on 15 June 1993. This first version was of no use for the pre-press community. It already featured internal links and bookmarks and fonts could be embedded but the only color space supported was RGB.

PDF 1.1

Acrobat 2 became available in November 1994. It supported the new PDF 1.1 file format which added support for:

- external links
- article threads
- security features
- device independent color
- notes

Acrobat 2.0 itself also got some nice enhancements, including a new architecture of Acrobat Exchange to support plug-ins in and the possibility to search PDF files. Acrobat 2.1 added multimedia support with the possibility of adding audio or video data to a PDF document.

PDF 1.2 *The pre-press world wakes up*

In November 1996, Adobe launched Acrobat 3.0 (code name: Amber) and the matching PDF 1.2 specifications. PDF 1.2 was the first version of PDF that was really usable in a pre-press environment. Besides forms, the following pre-press related options were included:

- support for OPI 1.3 specifications
- support for the CMYK colour space
- spot colors could be maintained in a PDF
- halftone functions could be included as well as overprint instructions.

The release of a plug-in to view PDF files in the Netscape browser increased the popularity of PDF file on the booming Internet. Adobe also added the possibility to link PDF files to HTML pages and vice versa. PDF also slowly began to get accepted by the graphic arts industry. Initially the black-and-white digital printing market began using PDF for output on fast Xerox digital presses.

PDF 1.3 *Listening to pre-press needs*

Acrobat 4, internally known as 'Stout' within Adobe, was launched in April 1999. It brought us PDF 1.3. The new PDF specs included support for:

- 2-byte CID fonts
- OPI 2.0 specifications

Contd...

- a new color space called DeviceN to improve support for spot colors
- smooth shading, a technology that allows for efficient and very smooth blends (transitions from one color or tint to another).
- annotations Acrobat itself also had its fair share of novelties, including:
- support for page sizes up to 5080 x 5080 mm, up from 1143 x 1143 mm

Illustrator 9 and PDF 1.4 Acrobat will have to wait

Mid 2000, Adobe did something weird: they released Illustrator 9. Although launching a new version of a drawing application is not that bizarre, Illustrator 9 did have one amazing feature: it was the first application to support PDF 1.4 and its transparency feature. This was the first time Adobe did not accompany a new version of PDF with a new version of Acrobat. They also did not release the full specs of PDF 1.4, although technote 5407 documented the transparency support in PDF 1.4.

PDF 1.5 and Acrobat 6 More choice for already confused users

In April 2003, Adobe announced Acrobat 6 which started shipping late May. The internal codename for Acrobat 6 was 'Newport'. As usual, the new version of Acrobat also brought along a new version of PDF, version 1.5. PDF 1.5 brings along a number of new features to will probably take a pretty long time before they get implemented or supported in applications. The new stuff includes.

- Improved compression techniques including object streams and JPEG 2000 compression
- Support for layers
- Improved support for tagged PDF

2005: another year, another PDF revision

In January 2005 Adobe started shipping Acrobat 7 (original code name: Vegas). Of course it offered support for a new PDF flavor. PDF 1.6 offers the following improvements: NChannel is an extension of the DeviceN mechanism for defining spot colors in a PDF document. It is backwards compatible with DeviceN and enables more accurate handling of color blending by including additional dot gain and color mixing information. The major new feature is the ability to embed 3D data.

PDF 1.7 Adobe goes ISO

Probably the most 'unexciting' PDF-version to ever be released, PDF 1.7 contained improved support for commenting and security. Support for 3D also got improved, with the possibility to add comments to 3D-objects and more elaborate control over 3D animations. A PDF 1.7 file can include default printer settings such as paper selection, number of copies, scaling. One interesting development with PDF 1.7 is the fact that it became an official ISO-standard. (ISO 32000-1:2008) in January 2008. The official specs were released on 1 July. James King from Adobe posted some interesting background information about this on his blog.

Questions

1. What was the main aim to create a PDF document?
2. Write about the versions of PDF and its features.

12.6 Summary

- pdflib supports functions that specify a region on a page that, when clicked on, takes the reader somewhere else.
- PDF provides several navigation features for PDF files. Bookmarks function as a table of contents for the document, and you can provide viewers with thumbnail images indicating what's at the other end of each bookmark.
- To draw a graphical shape, first specify a path and then fill and/or stroke the path with appropriately configured fill and/or stroke colors.
- Before you can use any sort of colours in your image at all, you will need to allocate the colour.

12.7 Keywords

CCVS: CCVS is a library for providing a conduit between your server and credit-card processing centers via a modem.

Cybercash: Cybercash is a provider of credit-card processing services.

Document: A PHP document is made up of a number of pages. Each page contains text and/ or images.

EXIF: The Exchangeable Image File Format (EXIF) extension provides a function to read the information stored on a device; many digital cameras store their information in EXIF format.

FDF: The Forms Data Format (FDF) is a library for creating forms in PDF documents and extracting data from or populating those forms.

Pattern: A pattern is a reusable component, defined outside of a page context that is used in place of a color for filling or stroking a path.

PDF: Stands for "Portable Document Format" which is a multi-platform file format developed by Adobe Systems.

12.8 Review Questions

1. What is the mean of PDF? What are the PDF files and why these are used?
2. What are the PDF extensions? Why these are used?
3. What are the uses of PDF documents and pages? How do these create?
4. What is the role of text in PDF?
5. Discuss about the text attributes and text functions.
6. What is the importance of font in PDF? How does it embedded in it?
7. How do we use images and graphics in PDF?
8. Write a PHP program to place an image in several places on a page.
9. How the templates and patterns are used in PDF?
10. What is the meaning of navigation? How PDF provide navigation?

Answers: Self Assessment**Notes**

- | | |
|-----------------|-------------|
| 1. False | 2. True |
| 3. False | 4. Cells |
| 5. Parent Class | 6. Learning |
| 7. True | 8. False |
| 9. True | 10. Pattern |
| 11. Unix | 12. Colours |
| 13. False | 14. False |
| 15. True | |

12.9 Further Readings**Books**

Bangia, Ramesh (2008). *“Web Technology (including HTML, CSS, XML, ASP, JAVA).”* Firewall Media.

Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.

Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.

Puntambekar, A.A. (2009). *“Web Technologies.”* Technical Publications.

Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.

Xavier, C. (2007). *“Web Technology and Design.”* New Age International.

**Online links**

<http://www.prepressure.com/pdf/basics/introduction>

<http://blog.perplexedlabs.com/2010/09/15/convert-html-to-pdf-in-php-libwkhtmltox-extension/>

<http://www.phpjammers.com/create-pdf-file-with-php-php36.html>

http://prasonagrawal.com/library/oreilly/webprog/php/ch10_02.htm#progphp-CHP-10-EX-1

<http://82.157.70.109/mirrorbooks/php5/067232511X/ch28lev1sec2.html>

<http://php.net/manual/en/haru.builtin.fonts.php>

<http://hints.macworld.com/article.php?story=20060203175741232>

<http://staffweb.cms.gre.ac.uk/~mk05/web/PHP/graphics.html>

<http://www.thesitewizard.com/php/create-image.shtml>

http://docstore.mik.ua/oreilly/webprog/php/ch10_05.htm

Unit 13: Extensible Markup Language

CONTENTS

Objectives

Introduction

13.1 Basics of XML

13.1.1 Document

13.1.2 Is XML just like HTML?

13.1.3 Is XML just Like SGML?

13.1.4 Why XML?

13.1.5 XML Development Goals

13.1.6 How is XML Defined?

13.2 Lightning Guide to XML

13.3 Generating XML

13.4 Parsing XML

13.4.1 Element Handlers

13.4.2 Character Encoding

13.4.3 Case Folding

13.4.4 Using the Parser

13.4.5 Errors

13.4.6 Methods as Handlers

13.4.7 Sample Parsing Application

13.5 Transforming XML with XSLT

13.6 Web Services

13.6.1 Servers

13.6.2 Clients

13.7 Summary

13.8 Keywords

13.9 Review Questions

13.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain Basics of XML
- Discuss the Features of Extensible Markup Language (XML)

- Explain How to Generating XML
- Discuss How to Parsing XML
- Understand How to Transform XML with XSLT
- Explain the Web Services

Introduction

XML is a markup language for structured documentation. Structured documents are documents that contain both content (words, pictures, etc.) and some indication of what role that content plays (for example, content in a section heading has a different meaning from content in a footnote, which means something different than content in a figure caption, etc.). Almost all documents have some structure. A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way of adding markup to documents.

13.1 Basics of XML

Extensible Markup Language (XML) is an abbreviated version of Standard Generalized Markup Language (SGML), for the exchange of structured documents over the Internet. Unlike HTML, XML readily enables the definition, transmission, validation, and interpretation of data between differing computing platforms and applications. XML permits people in a specialized field, such as chemistry, finance, or environmental data collection, to develop XML schema that define the markup language for the exchange of specialized data unique to their fields. XML schema is the primary data format supported for data exchange by the State/EPA Environmental Information Exchange Network (Exchange Network).

XML is extensible, meaning a developer can extend the language by devising new tags to describe and share data in any specialized way desired as long as the new tags follow the XML syntax defined by the W3C XML specification. XML is very useful for organizations that do not share but need to develop a common data exchange format.

13.1.1 Document

The number of applications currently being developed that are based on, or make use of, XML documents is truly amazing (particularly when you consider that XML is not yet a year old)! For our purposes, the word “document” refers not only to traditional documents, like this one, but also to the myriad of other XML “data formats”. These include vector graphics, e-commerce transactions, mathematical equations, object metadata, server APIs, and a thousand other kinds of structured information.

13.1.2 Is XML just like HTML?

No, HTML is an abbreviation for HyperText Markup Language while XML stands for eXtensible Markup Language. The differences are as follows:

1. HTML was designed to display data with focus on how data looks while XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.
2. HTML is a markup language itself while XML provides a framework for defining markup languages.
3. HTML is a presentation language while XML is neither a programming language nor a presentation language.

Notes

4. HTML is case insensitive while XML is case sensitive.
5. HTML is used for designing a web-page to be rendered on the client side while XML is used basically to transport data between the application and the database.
6. HTML has its own predefined tags while what makes XML flexible is that custom tags can be defined and the tags are invented by the author of the XML document.
7. HTML is not strict if the user does not use the closing tags but XML makes it mandatory for the user to close each tag that has been used.
8. HTML does not preserve white space while XML does.
9. HTML is about displaying data, hence static but XML is about carrying information, hence dynamic.

Thus, it can be said that HTML and XML are not competitors but rather **complement** to each other and clearly serving altogether different purposes.

13.1.3 Is XML just Like SGML?

It is true up to some extent. SGML (Standard Generalized Markup Language) is the standard for encoding paper documents into an electronic format. With the evolution of the internet, it became clear that HTML is no longer able to provide the need for more dynamic content as it has reached its limitations. XML (Extensible Markup Language) is a language that was derived from SGML and contains a more limited feature set in order to make it simpler for coders to use as SGML is too comprehensive and complex for the intended use. Since XML is simply a subset of SGML, SGML parsers are capable of reading and decoding valid XML files. The reverse is not necessarily true though as SGML files might have features that the XML parser does not understand.

Being a subset, there would be no feature in XML that does not exist in SGML. Here is a short list of what's been removed.

The following SGML declarations are no longer allowed in XML:

- DATATAG
- OMITTAG
- RANK
- LINK
- CONCUR
- SUBDOC
- FORMAL

The following SGML constructs are no longer allowed in XML:

- Empty start tags
- Empty end tags
- Unclosed start tags
- Unclosed end tags

Attribute specifications with no name Directly entered attribute values in attribute specification are not allowed and should be entered in literals. The following SGML entity declarations are no longer allowed in SGML:

- External SDATA entities
- Internal SDATA entities
- External CDATA entities
- Internal CDATA entities
- #DEFAULT entities
- PI entities
- Bracketed text entries

You are no longer allowed to specify attributes for entities. Changes have also been made in the way that coders place comments into their code. A comment declaration is no longer allowed to have more than a single comment. An empty comment declaration has also been disallowed. A parameter separator is disallowed to contain any comment. This includes any markup declaration, excluding comment declarations, of course.

13.1.4 Why XML?

XML (eXtensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. There are many advantages of using XML to transfer data as shown below.

1. ***XML is a good choice for data transfer using text-based documents supporting Unicode.***

The beauty of XML tags is that it permits lay readers to understand the data and its meaning. XML is written in Unicode so its symbols and special characters are included as part of the data. No more need to scour the data to find the single symbol that threw an error.

In the example below for contact information, the first shows the correct XML code. This provides clarity for computers and for those working with data.



Example:

```
<Person>
<Name>Jim Jones</Name>
  <Email>jim.jones@gmail.com</Email>
  <Phone>123-456-7890</Phone>
</Person>
```

2. ***XML is a platform, network and system independent.***

The biggest dilemma any company faces is adhering to internal data rules and standards, external industry standards, plus their own customers' requirements. Being text-based, XML can be read by any coding language, on any platform, or on any system. There is no longer a long drawn-out process to ensure compatibility, because data can be transferred quickly and seamlessly.

3. ***Users define their own tags in XML.***

These days, companies have their own data requirements and standards. Developers don't have wait for emerging new standards, with XML tags they can proceed based upon known requirements.

Notes

4. *Searching data in XML is easy.*

Not only are XML tags written in an easy-to-read format, but utilizing an XML parser permits users to search for a particular tag. The days of plodding through endless code are over!

5. *XML provides easy communication of the hierarchy of data elements using the tree structure.*

Data elements are often found in multiple tables and structures, see example below:



Example:

```
<item>Item ID</item>
<attribute>Voltage</attribute>
<value>15V</value>
<attribute>Brand</attribute>
<value>EnergySaving1</value>
</item>
```

In this example, there are multiple attributes and values describing one item in a manufacturer's catalog. This item, for example, is 15 volts and is branded 'EnergySaving1'. Unlimited attributes can be assigned to identify each item, just as there are unlimited branches emanating from each tree trunk.

6. *Can be saved in XML or text format, in any database.*

SQL, Oracle, or Documentum. With XML, data can be saved in any database and modified for any requirements, context, or syntax.

7. *User interface can be rearranged for cosmetic reasons or better usability, without needing to change the underlying data structure.*

Frequently data specialists are asked to rearrange the interface of the data. Without XML, this means the underlying data structure must also be changed. This is costly in terms of time and manpower, and it increases the probability of errors. When the data structure does not have to be touched, costs are minimized.

8. *Developers have greater flexibility as they design and develop websites or user display formats.*

Developers can easily revamp a website, or change the way data is displayed creating a design-driven site with changing the supporting data structures. XML continues to grow in popularity as the language of choice for transmitting data between different platforms, devices or data targets. It is a platform independent language which can simplify complex data transfers like attributes for catalog data or multiple table structures supporting your website.

13.1.5 XML Development Goals

XML was developed by the W3C as a way to create markup languages that met the needs of developers but were still fairly easy for humans to read and understand. These 10 design goals, as defined at the W3C, explain the basics of XML and how you should write elements and attributes when using XML. If you follow these design goals, you'll have an XML schema that ultimately meets the goals of the XML working group.

1. *XML shall be straightforwardly usable over the Internet.*

When you write XML it should be readily available over the Internet. XML is not intended as a programming language for stand-alone systems, but rather to be used across the Internet for a wide variety of sources.

2. *XML shall support a wide variety of applications.*

The beauty of XML is that it was intended to be used for as many things as possible. This flexibility can sometimes make it more difficult to understand, but ultimately, XML can be used to describe a Web page about flowers or a database of car parts or nearly anything you can imagine.

3. *XML shall be compatible with SGML.*

SGML or Standard Generalized Markup Language, is the ISO standard on which all XML and thus XHTML documents are based upon. If a document isn't compatible with SGML, then it cannot be called XML.

4. *It shall be easy to write programs which process XML documents.*

XML was always intended to be easy to use and process. Because XML is based on human-readable text, this makes it a lot easier for programmers to figure out what is meant by the XML tags. This in turn makes it easier to write a program that processes those tags.

5. *The number of optional features in XML is to be kept to the minimum.*

Ideally, there would be zero optional features. Optional features cause problems because they are not guaranteed to be in any given situation. The more optional features there are in a system the more combinations there are for the system and so the more difficult the programming becomes.

6. *XML documents should be human-legible and reasonably clear.*

Rather than having elements that are named a3209zd you would have an element named <first_name>. Someone reading your XML should be able to make an educated guess about what the data is that's being tagged.

7. *The XML design should be prepared quickly.*

It is better to spend time building the data than it is on building an XML design.

8. *The design of XML shall be formal and concise.*

Only include as many elements as you need to be clear, not more and not less.

9. *XML documents shall be easy to create.*

XML is intended to not require a special editor or tool to create. And in fact, most XML documents can be edited in a text editor like Notepad or TextEdit.

10. *Terseness in XML markup is of minimal importance.*

When you're creating XML element names, first_name is better than fname because it's clearer and more human readable. While you do want to keep elements names short, the shortness should not be at the sacrifice of human-readability.

13.1.6 How is XML Defined?

XML is not a programming language like C++ or Java, but a markup specification language. A browser or other application must read the XML document in order to make it do something. Development of XML documents begins with the identification and definition of the data elements that will be displayed or exchanged. The data elements are defined using tags that indicate what a data element represents. For example, in the simplest explanation, a person's name might contain three tags:

Notes



Example:

```
<Name>
<First>Fred</First>
<MiddleInitial>M</MiddleInitial>
<Last>Smith</Last>
</Name>
```

Developers create a schema that contains the tag name, their data formats, and the relationships to one another.

Self Assessment

State whether the following statements are true or false:

1. XML is a markup language.
2. HTML is about carrying information and is dynamic.
3. XML was developed by the W3C as a way to create markup languages that met the needs of developers.

13.2 Lightning Guide to XML

Most XML consists of elements (like HTML tags), entities, and regular data. For example:



Example:

```
<book>
<title>php</title>
  <authors>
<author>ABC</author>
  <author>XYZ</author>
  </authors>
</book>
```

In HTML, you often have an open tag without a close tag. The most common example of this is:

```
<br>
```

In XML, that is illegal. XML requires that every open tag be closed. For tags that do not enclose anything, such as the line break `
`, XML adds this syntax:

```
<br />
```

Tags can be nested but cannot overlap. For example, this is valid:

```
<book><title>Programming PHP</title></book>
```

but this is not valid, because the book and title tags overlap:

```
<book><title>Programming PHP</book></title>
```

XML also requires that the document begin with a processing instruction that identifies the version of XML being used (and possibly other things, such as the text encoding used). For example: being used (and possibly other things, such as the text encoding used). For example:

```
<?xml version="1.0" ?>
```

The final requirement of a well-formed XML document is that there be only one element at the top level of the file.



Example:

```
<?xml version="1.0" ?>
<library>
<title>Programming PHP</title>
  <title>Programming Perl</title>
<title>Programming C#</title> </library>
```

But this is not well formed, as there are three elements at the top level of the file:

```
<?xml version="1.0" ?> <title>Programming PHP</title> <title>Programming
Perl</title>
<title>Programming C#</title>
```

There are two ways to write down this structure: the Document Type Definition (DTD) and the Schema. DTDs and Schemas are used to validate documents; that is, to ensure that they follow the rules for their type of document.

Most XML documents do not include a DTD. Many identify the DTD as an external with a line that gives the name and location (file or URL) of the DTD:

```
<!DOCTYPE rss PUBLIC 'My DTD Identifier' 'http://www.abc.com/my.dtd'>
```

Sometimes it is convenient to encapsulate one XML document in another. For example, an XML document representing a mail message might have an attachment element that surrounds an attached file. If the attached file is XML, it is a nested XML document. What if the mail message document has a body element (the subject of the message), and the attached file is an XML representation of a dissection that also has a body element, but this element has completely different DTD rules? How can you possibly validate or make sense of the document if the meaning of body changes partway through?

This problem is solved with the use of namespaces. Namespaces let you qualify the XML tag for example, email:body and human:body.

XML documents generally are not completely ad hoc. The specific tags, attributes, and entities in an XML document, and the rules governing how they nest, comprise the structure of the document.



Did u know? XML documents generally are not completely ad hoc. The specific tags, attributes, and entities in an XML document, and the rules governing how they nest, comprise the structure of the document.

Self Assessment

Fill in the blanks:

4. XML requires that every open tag be.....
5. The final requirement of a well-formed XML document is that there be only one element at thelevel of the file.
6.let you qualify the XML tag.

13.3 Generating XML

Following example shows how we generate dynamic XML.



Example:

```
<?php
//simulate a remote connection
sleep(2);

//include any libraries you want to use here.

//get the $mode var
$mode = stripslashes($_GET['mode']);
//Set the content-type header to xml
header("Content-type: text/xml");
//echo the XML declaration
echo chr(60).chr(63).'xml version="1.0" encoding="utf-8"
'.chr(63).chr(62);
?>
<xmlresponse>
  <?php
    //make a decision based on $mode
    switch ($mode) {
      case 'getitems':
        //set items in a test array
        $items_array[] = array(
          'id'=>15,
          'name'=>'Finger Bun',
          'price'=>'$0.80 ea'
        );
        $items_array[] = array(
          'id'=>16,
          'name'=>'Donuts',
          'price'=>'$0.50 ea'
        );
        $items_array[] = array(
          'id'=>17,
          'name'=>'Apple Pie',
          'price'=>'$1.20 slice'
        );
        $items_array[] = array(
          'id'=>18,
          'name'=>'Double Choc Chip Cup Cakes',
          'price'=>'$1.00 ea'
        );

        //echo a count of items
        echo '<item_count>'.count($items_array).'</item_count>';
        //echo the array in XML style
        for ($x=0;$x<count($items_array);$x++) {
          echo '<item>';
            echo '<id>'.htmlspecialchars($items_array[$x]['id']).'</id>';
            echo '<name>'.htmlspecialchars($items_array[$x]['name']).'</
name>';
```

```

        echo '<price>'.htmlspecialchars($items_array[$x]['price']).'</
price>';
        echo '</item>';
    }

    //set a No Error response - function defined below
    SetErrorNode();
    break;

default:
    //incorrect mode, let the hacker know!
    SetErrorNode(404,'Stop hacking. Get a real job.');
```

```

    break;
} //end switch
?>
</xmlresponse><?php

//FUNCTIONS
//this sets the error code node - essential for every xml document to be
returned.
function SetErrorNode($code=0,$text='') {
    echo "<error_code>" . $code . "</error_code>\n<error>" .
htmlspecialchars($text) . "</error>";
}
?>
```

Explanation

First we simulate an internet connection delay (for those using local development servers) in case you've got any features such as a waiting screen on the front-end that you want to admire for two seconds.

Next we get the \$mode variable from the GET array so we can decide what to do with it later. Then we send the content-type headers to the browser, so that it knows this is an XML document. Then we echo the XML declaration. You might be wondering why I've used chars everywhere there. This is so we don't have to turn off PHP short tags just to make this thing work, a tip for all you out there.

Then we echo the root tag, xmlresponse.

The rest of the script is echoing data for the application to use. The only thing to mention is that any data you want to send within XML tags should be coded using htmlspecialchars. Even HTML code should be coded.

Finally, we use the function SetErrorNode to echo our error tags as we want to.

Self Assessment

Fill in the blanks:

7. We use the functionto echo our error tags.
8. We simulate anconnection delay for those using local development servers in case you've got any features.
9. Getting the \$mode variable from thearray help you to decide what to do with it later.

13.4 Parsing XML

Parsing XML essentially means navigating through an XML document and returning the relevant data. An increasing number of web services return data in JSON format, but a large number still return XML, so you need to master parsing XML if you really want to consume the full breadth of APIs available.

Using PHP's SimpleXML extension that was introduced back in PHP 5.0, working with XML is very easy to do.

Let's start with the following example.



Example:

```
<?xml version="1.0" encoding="utf-8"?>

<languages>

    <lang name="C">
        <appeared>1972</appeared>

        <creator>Dennis Ritchie</creator>
    </lang>

    <lang name="PHP">
        <appeared>1995</appeared>

        <creator>Rasmus Lerdorf</creator>
    </lang>

    <lang name="Java">
        <appeared>1995</appeared>

        <creator>James Gosling</creator>
    </lang>

</languages>
```

The above XML document encodes a list of programming languages, giving two details about each language: its year of implementation and the name of its creator.

The first step is to loading the XML using either `simplexml_load_file()` or `simplexml_load_string()`. As you might expect, the former will load the XML file a file and the later will load the XML from a given string.

```
<?php
2 $languages = simplexml_load_file("languages.xml");
```

Both functions read the entire DOM tree into memory and returns a SimpleXMLElement object representation of it. In the above example, the object is stored into the `$languages` variable. You can then use `var_dump()` or `print_r()` to get the details of the returned object if you like.

```
SimpleXMLElement Object
(
    [lang] => Array
        (
```

```

[0] => SimpleXMLElement Object
  (
    [@attributes] => Array
      (
        [name] => C
      )
    [appeared] => 1972
    [creator] => Dennis Ritchie
  )
[1] => SimpleXMLElement Object
  (
    [@attributes] => Array
      (
        [name] => PHP
      )
    [appeared] => 1995
    [creator] => Rasmus Lerdorf
  )
[2] => SimpleXMLElement Object
  (
    [@attributes] => Array
      (
        [name] => Java
      )
    [appeared] => 1995
    [creator] => James Gosling
  )
)
)

```

The XML contained a root language element which wrapped three lang elements, which is why the SimpleXMLElement has the public property lang which is an array of three SimpleXMLElements. Each element of the array corresponds to a lang element in the XML document.

You can access the properties of the object in the usual way with the -> operator. For example, \$languages->lang[0] will give you a SimpleXMLElement object which corresponds to the first lang element. This object then has two public properties: appeared and creator.



Example:

```
<?php
```

```
$languages->lang[0]->appeared;
```

```
    $languages->lang[0]->creator;
```

Iterating through the list of languages and showing their details can be done very easily with standard looping methods, such as foreach.

Notes*Example:*

```
<?php
foreach ($languages->lang as $lang) {

    printf(
        "<p>%s appeared in %d and was created by %s.</p>",
        $lang["name"],
        $lang->appeared,
        $lang->creator
    );

}
```

Notice that the lang element's name attribute to retrieve the name of the language. You can access any attribute of an element represented as a SimpleXMLElement object using array notation like this.

13.4.1 Element Handlers

XML_Parse, as it's parsing along the incoming data, calls an element handler each time it encounters an element. There are two handlers for elements; one is called when the element begins, the other when the element ends. Between the two calls there are handler calls for the various contents of the element (if any).



Caution An unparsed entity must be accompanied by a notation declaration, and while you can define handlers for declarations of unparsed entities and notations.

13.4.2 Character Encoding

It is very important that the character encoding of any XML or (X)HTML document is clearly labeled, so that clients can easily map these encodings to Unicode. This can be done in the following ways:

- Send the 'charset' parameter in the Content-Type header of HTTP. Example:

```
Content-Type: text/html; charset=utf-8
```

To do this you will need to have access to server settings or serve your document via scripting

- For XML (including XHTML), use the encoding pseudo-attribute in the XML declaration at the start of a document or the text declaration at the start of an entity. Example:

```
<? xml version="1.0" encoding="utf-8" ?>
```

There are potential issues you should be aware of when using this with XHTML 1.0 served as HTML.

- For HTML or XHTML served as HTML, you should always use the <meta> tag inside <head>. Example:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" >
```


- For XHTML, you need a slash at the end:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

For a discussion of which approach is best for which type of (X)HTML document, see the tutorial Character sets & encodings in XHTML, HTML and CSS.

The examples above show declarations for UTF-8 encoded content. This is likely to be the best choice of encoding for most purposes, but it is not the only possibility.

If not using UTF-8 you should replace the utf-8 text in the examples above with the name of the encoding you have chosen. You can see the full list of character encoding names registered by IANA (long). In practice, a few encodings will be preferred, most likely: ISO-8859-1 (Latin-1), US-ASCII, UTF-16, the other encodings in the ISO-8859 series, iso-2022-jp, euc-kr, and so on.

Ensuring the Declaration Works

It is important to not only use the encoding declarations above in HTTP or content, but also:

- Save your data in the appropriate encoding from your editing environment.
- Ensure that there is no conflict between what you declare in the document and what the server automatically applies, since server settings override in-document declarations.

13.4.3 Case Folding

The element handler functions may get their element names case-folded. Case-folding is defined by the XML standard as “a process applied to a sequence of characters, in which those identified as non-uppercase are replaced by their uppercase equivalents”. In other words, when it comes to XML, case-folding simply means uppercasing.

By default, all the element names that are passed to the handler functions are case-folded. This behaviour can be queried and controlled per XML parser with the `xml_parser_get_option()` and `xml_parser_set_option()` functions, respectively.

13.4.4 Using the Parser

To use the XML parser, create a parser with `xml_parser_create()`, set handlers and options on the parser, then hand chunks of data to the parser with the `xml_parse()` function until either the data runs out or the parser returns an error. Once the processing is complete, free the parser by calling `xml_parser_free()`.

The `xml_parser_create()` function returns an XML parser:

```
$parser = xml_parser_create([encoding]);
```

The optional encoding parameter specifies the text encoding (“ISO-8859-1”, “US-ASCII”, or “UTF-8”) of the file being parsed.

The `xml_parse()` function returns TRUE if the parse was successful or FALSE if it was not:

```
$success = xml_parse(parser, data [, final]);
```

The data argument is a string of XML to process. The optional final parameter should be true for the last piece of data to be parsed.

To easily deal with nested documents, write functions that create the parser and set its options and handlers for you. This puts the options and handler settings in one place, rather than duplicating them in the external entity reference handler. Example has such a function is as follows:

Notes



Example:

Creating a Parser

```
function create_parser ($filename) {
    $fp = fopen('filename', 'r');
    $parser = xml_parser_create( );
    xml_set_element_handler($parser, 'start_element', 'end_element');
    xml_set_character_data_handler($parser, 'character_data');
    xml_set_processing_instruction_handler($parser, 'processing_instruction');
    xml_set_default_handler($parser, 'default');
    return array($parser, $fp);
}

function parse ($parser, $fp) {
    $blockSize = 4 * 1024; // read in 4 KB chunks
    while($data = fread($fp, $blockSize)) { // read in 4 KB chunks
        if(!xml_parse($parser, $data, feof($fp))) {
            // an error occurred; tell the user where
            echo 'Parse error: ` . xml_error_string($parser) . ` at line ` .
                xml_get_current_line_number($parser);
            return FALSE;
        }
    }
    return TRUE;
}

Notes

if (list($parser, $fp) = create_parser('test.xml')) {
    parse($parser, $fp);
    fclose($fp);
    xml_parser_free($parser);
}
```

PHP's XML parser is event-based, meaning that as the parser reads the document, and it calls various handler functions you provide as certain events occur, such as the beginning or end of an element.



Did u know? PHP's XML parser is event-based, meaning that as the parser reads the document, and it calls various handler functions you provide as certain events occur, such as the beginning or end of an element.

13.4.5 Errors

Objects with this interface are used to receive error and warning information from the XMLReader. If you create an object that implements this interface, then register the object with your XMLReader, the parser will call the methods in your object to report all warnings and errors. There are three levels of errors available: warnings, (possibly) recoverable errors, and unrecoverable errors. All methods take a SAXParseException as the only parameter. Errors and warnings may be converted to an exception by raising the passed-in exception object.

```
ErrorHandler.error(exception)
```

Called when the parser encounters a recoverable error. If this method does not raise an exception, parsing may continue, but further document information should not be expected by the application. Allowing the parser to continue may allow additional errors to be discovered in the input document.

```
ErrorHandler.fatalError(exception)
```

Called when the parser encounters an error it cannot recover from; parsing is expected to terminate when this method returns.

```
ErrorHandler.warning(exception)
```

Called when the parser presents minor warning information to the application. Parsing is expected to continue when this method returns, and document information will continue to be passed to the application. Raising an exception in this method will cause parsing to end.

13.4.6 Methods as Handlers

Because functions and variables are global in PHP, any component of an application that requires several functions and variables is a candidate for object orientation. XML parsing typically requires you to keep track of where you are in the parsing (e.g., “just saw an opening title element, so keep track of character data until you see a closing title element”) with variables, and of course you must write several handler functions to manipulate the state and actually do something. Wrapping these functions and variables into a class provides a way to keep them separate from the rest of your program and easily reuse the functionality later.

Use the `xml_set_object()` function to register an object with a parser. After you do so, the XML parser looks for the handlers as methods on that object, rather than as global functions:

```
xml_set_object(object);
```

13.4.7 Sample Parsing Application

```
<?php
    $g_books = array();
    $g_elem = null;

    function startElement( $parser, $name, $attrs )
    {
        global $g_books, $g_elem;
        if ( $name == 'BOOK' ) $g_books []= array();
        $g_elem = $name;
    }

    function endElement( $parser, $name )
    {
        global $g_elem;
        $g_elem = null;
    }

    function textData( $parser, $text )
    {
        global $g_books, $g_elem;
        if ( $g_elem == 'AUTHOR' ||
```

Notes

```
$g_elem == 'PUBLISHER' ||
$g_elem == 'TITLE' )
{
    $g_books[ count( $g_books ) - 1 ][ $g_elem ] = $text;
}
}

$xml_parser_create();

$xml_set_element_handler( $parser, "startElement", "endElement" );
$xml_set_character_data_handler( $parser, "textData" );

$f = fopen( 'books.xml', 'r' );

while( $data = fread( $f, 4096 ) )
{
    xml_parse( $parser, $data );
}

$xml_parser_free( $parser );

foreach( $g_books as $book )
{
    echo $book['TITLE'] . " - " . $book['AUTHOR'] . " - ";
    echo $book['PUBLISHER'] . "\n";
}
?>
```

The script starts by setting up the `g_books` array, which holds all the books and their information in memory, and a `g_elem` variable, which stores the name of the tag the script is currently processing. The script then defines the callback functions. In this example, the callback functions are `startElement`, `endElement`, and `textData`. The `startElement` and `endElement` functions are called when tags are opened and closed, respectively. The `textData` function is called on the text between the start and end of the tags.

In this example, the `startElement` tag is looking for the `book` tag to start a new element in the `book` array. Then, the `textData` function looks at the current element to see if it's a publisher, title, or author tag. If so, the function puts the current text into the current book.

To get the parsing going, the script creates the parser with the `xml_parser_create` function. Then, it sets the callback handlers. After that, the script reads in the file and sends off chunks of the file to the parser. After the file is read, the `xml_parser_free` function deletes the parser. The end of the script dumps out the contents of the `g_books` array.

Self Assessment

State whether the following statements are true or false:

10. The XML contained a root language element which wrapped five lang elements.
11. The character encoding of any XML or (X)HTML document should be clearly labeled, so that clients can easily map these encodings to Unicode.
12. The element handler functions may get their element names case-folded.

13.5 Transforming XML with XSLT

Notes

The Extensible Stylesheet Language (XSL) is a W3C consortium standard for describing presentation rules that applies to XML document. It includes both a transformation language (XSLT) and a formatting language.

XSLT is an xml based language and W3C specification that describes how to transform an xml document into another xml document or into HTML,PDF or some other format.

An XSL document should be saved with an extension of .xsl and can be integrated in an xml document in the same way as .css.



Example:

: illustrating XSL.

XSL1.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/" >
<html>
<body>
<table border="2" bgcolor="cyan">
<tr>
<th>TITLE</th>
<th>ARTIST</th>
<th>COUNTRY</th>
<th>PRICE</th>
<th>YEAR</th>
</tr>
<xsl:for-each select="CATALOG/CD">
<tr>
<td>
<xsl:value-of select="TITLE" />
</td>
<td>
xsl:value-of select="ARTIST" />
</td>
<td>
<xsl:value-of select="COUNTRY" />
</td>
<td>
<xsl:value-of select="PRICE" />
</td>
<td>
<xsl:value-of select="YEAR" />
</td>
</tr>
</xsl:for-each>
```

Notes

```
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
CDCatalog.xml
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="XSL1.xsl"?>
<CATALOG>
  <CD>
    <TITLE> PAA </TITLE>
    <ARTIST> Amitabh Bachan</ARTIST>
    <COUNTRY>India</COUNTRY>
    <COMPANY>ABCL</COMPANY>
    <PRICE>Rs 160</PRICE>
    <YEAR>2009</YEAR>
  </CD>
  <CD>
    <TITLE> 3 Idiots </TITLE>
    <ARTIST>Aamir Khan</ARTIST>
    <COUNTRY>India</COUNTRY>
    <COMPANY>Percept</COMPANY>
    <PRICE>Rs 140</PRICE>
    <YEAR>2010</YEAR>
  </CD>
  <CD>
    <TITLE>Namastey London</TITLE>
    <ARTIST> Akshay Kumar</ARTIST>
    <COUNTRY>India</COUNTRY>
    <COMPANY>New Motion</COMPANY>
    <PRICE>RS 130</PRICE>
    <YEAR>2008</YEAR>
  </CD>
  <CD>
    <TITLE> Taare Zameen Par </TITLE>
    <ARTIST>Aamir Khan</ARTIST>
    <COUNTRY>India</COUNTRY>
    <COMPANY>Percept</COMPANY>
    <PRICE>Rs 140</PRICE>
    <YEAR>2010</YEAR>
  </CD>
  <CD>
    <TITLE>Bhool Bhulaiya</TITLE>
    <ARTIST> Akshay Kumar</ARTIST>
```

```

<COUNTRY>India</COUNTRY>
<COMPANY>New Motion</COMPANY>
<PRICE>RS 130</PRICE>
<YEAR>2007</YEAR>
</CD>
<CD>
  <TITLE> Deewar </TITLE>
  <ARTIST> Amitabh Bachan</ARTIST>
  <COUNTRY>India</COUNTRY>
  <COMPANY>ABCL</COMPANY>
  <PRICE>Rs 160</PRICE>
  <YEAR>1970</YEAR>
</CD>
</CATALOG>

```

Self Assessment

Fill in the blanks:

13. includes both a transformation language (XSLT) and a formatting language.
14. XSLT is anbased language.
15. An XSL document should be saved with an extension of .xsl and can be integrated in an xml document in the same way as

13.6 Web Services

A web service is a collection of protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. In XML web services, all data to be exchanged is formatted with XML tags.

XML-RPC and SOAP are two of the standard protocols used to create web services. XML-RPC is the older (and simpler) of the two, while SOAP is newer and more complex. Microsoft's .NET initiative is based around SOAP, while many of the popular web journal packages, such as Frontier and blogger, offer XML-RPC interfaces.

PHP provides access to both SOAP and XML-RPC through the xmlrpc extension, which is based on the xmlrpc-epi project.



Notes The xmlrpc extension is not compiled in by default, so you will need to add with-xmlrpc to your configure line.

13.6.1 Servers

An XML-RPC Server acts as a traffic cop of sorts, waiting for incoming requests and redirecting them to the appropriate functions for processing.

Notes

To create your own XML-RPC server involves initializing the XML-RPC Server class in your controller where you expect the incoming request to appear, then setting up an array with mapping instructions so that incoming requests can be sent to the appropriate class and method for processing.

Here is an example to illustrate:

```
$this->load->library('xmlrpc');
$this->load->library('xmlrpcs');

$config['functions']['new_post'] = array('function' =>
    'My_blog.new_entry'),
$config['functions']['update_post'] = array('function' =>
    'My_blog.update_entry');
$config['object'] = $this;

$this->xmlrpcs->initialize($config);
$this->xmlrpcs->serve();
```

The above example contains an array specifying two method requests that the Server allows. The allowed methods are on the left side of the array. When either of those are received, they will be mapped to the class and method on the right.

The 'object' key is a special key that you pass an instantiated class object with, which is necessary when the method you are mapping to is not part of the CodeIgniter super object.

In other words, if an XML-RPC Client sends a request for the new_post method, your server will load the My_blog class and call the new_entry function. If the request is for the update_post method, your server will load the My_blog class and call the update_entry function.

The function names in the above example are arbitrary. You'll decide what they should be called on your server, or if you are using standardized APIs, like the Blogger or MetaWeblog API, you'll use their function names.

There are two additional configuration keys you may make use of when initializing the server class: debug can be set to TRUE in order to enable debugging, and xss_clean may be set to FALSE to prevent sending data through the Security library's xss_clean function.

Processing Server Requests

When the XML-RPC Server receives a request and loads the class/method for processing, it will pass an object to that method containing the data sent by the client.

Using the above example, if the new_post method is requested, the server will expect a class to exist with this prototype:

```
class My_blog extends CI_Controller {

    function new_post($request)
    {

    }

}
```

The \$request variable is an object compiled by the Server, which contains the data sent by the XML-RPC Client. Using this object you will have access to the request parameters enabling you to process the request. When you are done you will send a Response back to the Client.

Below is a real-world example, using the Blogger API. One of the methods in the Blogger API is `getUserInfo()`. Using this method, an XML-RPC Client can send the Server a username and password, in return the Server sends back information about that particular user (nickname, user ID, email address, etc.). Here is how the processing function might look:



Example:

```
class My_blog extends CI_Controller {

    function getUserInfo($request)
    {
        $username = 'smitty';
        $password = 'secretsmittypass';

        $this->load->library('xmlrpc');

        $parameters = $request->output_parameters();

        if ($parameters['1'] != $username AND $parameters['2'] !=
$password)
        {
            return $this->xmlrpc->send_error_message('100', 'Invalid
Access');
        }

        $response = array(array('nickname' => array('Smitty','string'),
                                'userid'   => array('99','string'),
                                'url'      => array('http://
yoursite.com','string'),
                                'email'    =>
array('jsmith@yoursite.com','string'),
                                'lastname' => array('Smith','string'),
                                'firstname' => array('John','string')
                                ),
                            'struct');

        return $this->xmlrpc->send_response($response);
    }
}
```



Notes The `output_parameters()` function retrieves an indexed array corresponding to the request parameters sent by the client. In the above example, the output parameters will be the username and password. If the username and password sent by the client were not valid, an error message is returned using `send_error_message()`. If the operation was successful, the client will be sent back a response array containing the user's info.

13.6.2 Clients

An XML-RPC client issues an HTTP request and parses the response. The `xmlrpc` extension that ships with PHP can work with the XML that encodes an XML-RPC request, but it does not know how to issue HTTP requests. For that functionality, this file contains a function to perform the HTTP request.

Notes

Using a text editor, create a controller called `xmlrpc_client.php`. In it, place this code and save it to your `applications/controllers/` folder:



Example:

```
<?php

class Xmlrpc_client extends CI_Controller {

    function index()
    {
        $this->load->helper('url');
        $server_url = site_url('xmlrpc_server');

        $this->load->library('xmlrpc');

        $this->xmlrpc->server($server_url, 80);
        $this->xmlrpc->method('Greetings');

        $request = array('How is it going?');
        $this->xmlrpc->request($request);

        if ( ! $this->xmlrpc->send_request() )
        {
            echo $this->xmlrpc->display_error();
        }
        else
        {
            echo '<pre>';
            print_r($this->xmlrpc->display_response());
            echo '</pre>';
        }
    }
}
?>
```

Task Write the PHP code which shows a client for the multiply function for the XML-RPC service.

Self Assessment

State whether the following statements are true or false:

16. Software applications written in only one programming language.
17. XML-RPC and SOAP are two of the standard protocols used to create web services.
18. An XML-RPC client issues an HTTP request and parses the response.



Case Study

Moving to an XML based Website

In early 2007, it started the task of reworking the ageing HyperWrite Website. The site was originally created in 1995. It underwent a major rework (to a frames-based design) in 1997, and was reworked in 1999, 2000 and 2002. In the decade since the Website was launched, not only has Web technology moved on, but HyperWrite's activities, focus and business direction are now quite different.

Screen capture of HyperWrite Website circa 1995



Time and budget were allocated to renovate the site to better serve HyperWrite's business needs, and to serve as a practical example of the company's capabilities.

Reasons for the Site Renovation

When it was decided to completely revise the site in 2007, one of the prime motivators was to move to being fully standards-based (XHTML and CSS). But the reasons for updating the site were not only technical. Analysis of the Website logs over a 12 month period showed the most popular area of the site was the knowledge part (where magazine-style articles relating to technical documentation and Help technologies were published). It was decided to give that area greater prominence. The services offered by HyperWrite were better categorised into training, consultancy and conferences (rather than lumped together as services). The Web logs also showed that Firefox was used on average by 15% of site visitors. Considering the rate of Firefox adoption is increasing, the percentage for the last month of the year would be a lot higher. Previously, when HyperWrite was mainly providing Windows Help systems consultancy, we could assume that our target audience nearly all used Internet Explorer for browsing. The greater importance of open systems in our business was another argument towards fully embracing XHTML.

The site had many inconsistencies, accidentally introduced over the years via editing tool changes and style changes. Any site revamp will provide the opportunity to standardize the pages, but it was keen to find a way to reduce the likelihood of the site *drifting* in future.

The Role of XML

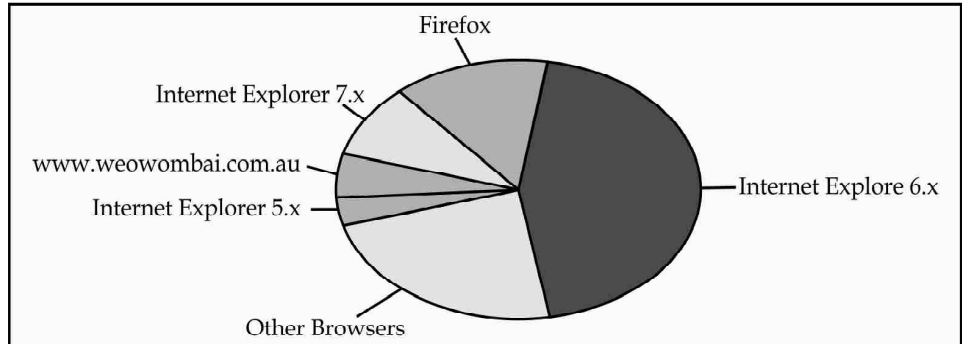
XML is great for enforcing standards; if a document does not conform to its XML rules, it would not save! But there are hundreds of XML languages. For Web sites and similar types of content, the most appropriate XML applications are RSS, DITA, DocBook and, of course, XHTML. DITA plays an increasingly important role in HyperWrite's consultancy and training business, so it wanted to include DITA content in the site. As the project developed, a site map format was required to store information about the structure of the site.

Contd...

Notes

As ASP.Net was the technology platform on which the site would be deployed, the ASP.Net "sitemap" format was an option, as was the "ditamap" format.

ASP.Net and Visual Web Developer



Some portions of the site, such as the newsletter subscription page, required server-side processing. Previously, the site had used Microsoft's ASP technology for this purpose. The site was, and would continue to be, hosted on a Windows 2003 Server with Internet Information Server, that supports both ASP and ASP.Net. For page editing, FrontPage was discounted as an option, because of its inability to work in pure XHTML. Adobe DreamWeaver was considered, but Microsoft Visual Web Developer® (VWD) was selected as the page editing software. VWD is a solid XHTML and generic XML editor, with an integrated CSS editor. Its primary role, though, is as a Web application development tool for ASP.Net. A single editor could therefore be used for programming of server-side logic, and for any static Web pages. ASP.Net allows easy server-side XSL transformations of XML content that provided an XSL-T file is already available for the transformation. The task of creating an ASP.Net page to turn an XML data file into HTML can take as little as 30 seconds. Similarly, if a site map is available (in the ASP.Net "sitemap" XML format), a dynamic table of contents (TOC) for the site can be created instantly. As soon as the sitemap is updated, the TOC is automatically updated.

Architecture

The Website's new architecture is essentially a three column design, with major navigation buttons in the left column, the main content in the centre, and sidebar information in the right. A branding banner and a breadcrumbs trail run across the top of the design, and a footer block along the bottom. The branding banner is an ASP.Net "included page". As the server delivers a page to the browser, it inserts the included page content at the top. The actual banner code only occurs once, in the included page itself. It is re-used on every page in the site. If the banner needs to be changed, only the included page needs to be altered. The breadcrumb trail is automatically generated through a standard ASP.Net design-time control. The design-time control simply references the sitemap XML file, and automatically generates the breadcrumb trail.

Likewise, the main navigation buttons in the left column are derived through a design-time control referencing the same sitemap XML file. The ASP.Net sitemap XML file format follows a simple sitemap/sitemapnode/sitemapnode structure. For the HyperWrite site, the sitemap XML is generated (through an XSL-T file) from a ditamap file. To further simplify matters, ASP.Net provides a "master page" feature, which allows common (repeated) elements of a page to be locked into a template-like skeleton. The new site uses a master page to set the banner, breadcrumbs, navigation, sidebar and the footer block. This leaves just the main content to be composed for each page.

Contd...

The main content can be normal XHTML, typed directly in Visual Web Developer RSS, transformed on the server by an XSL-T file DocBook or DITA XML, transformed on the server by an XSL-T file. The transformed RSS, DITA and DocBook content is dynamically placed within the master page template. Like the banner, the footer block is an included page.

The sidebar was used in the previous design, and was intended to carry snippets of news, hints and related links. However, experience shows that the material was very rarely updated, and was often stale. This was probably because we focussed on keeping the main content up-to-date. If we did not happen to notice that the sidebar information was obsolete, it would never get changed. The new approach is to make the sidebar information a "conditional included page". If the master page script finds a file with a .inc extension and the same name as the current page, it displays that .inc page in the sidebar. If it cannot find a specific .inc page, it looks in the current folder for a file named sidebar.inc, and places that file's content in the sidebar column. The .inc file can be XHTML or RSS; if it is RSS, it will be transformed (by an XSL-T file) to XHTML on the server.

For example, when a page within the /Training folder is requested, the server pulls in the sidebar.inc within the /Training folder. Likewise, a page within the /Conferences folder will pull in the /Conferences/sidebar.inc file. This approach meant that the master page could still be used for sidebar content, and that any changes to sidebar content would only have to be made once per section in the applicable sidebar.inc file. As you can deduce, the whole idea is highly dependent upon XSL-T. This should make perfect sense, because the source content needs to be turned into XHTML before it reaches the browser. Additionally, it is more efficient to transform to XHTML on-the-fly, as required, rather than pre-transform the content using an XSL processor.

Questions

1. Write the key concepts XML used in above case study.
2. What were the main reasons for the site renovation?

13.7 Summary

- XML (eXtensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- An XML-RPC client issues an HTTP request and parses the response.
- An XML-RPC Server acts as a traffic cop of sorts, waiting for incoming requests and redirecting them to the appropriate functions for processing.
- Web service is a collection of protocols and standards used for exchanging data between applications or systems.
- The Extensible Stylesheet Language (XSL) is a W3C consortium standard for describing presentation rules that applies to XML document.
- It is very important that the character encoding of any XML or (X)HTML document is clearly labeled, so that clients can easily map these encodings to Unicode.
- Parsing XML essentially means navigating through an XML document and returning the relevant data.

13.8 Keywords

Case-folding: It is defined by the XML standard as “a process applied to a sequence of characters, in which those identified as non-uppercase are replaced by their uppercase equivalents”.

Character Data Handler: The character data handler takes in a reference to the XML parser that triggered the handler and a string containing the character data itself.

Markup Language: A markup language is a mechanism to identify structures in a document.

Processing Instructions: Processing instructions are used in XML to embed scripts or other code into a document.

SGML (Standard Generalized Markup Language): It is the standard for encoding paper documents into an electronic format.

Web Service: This is a collection of protocols and standards used for exchanging data between applications or systems.

XML: XML is a markup language for documents containing structured information.

XSLT: Extensible Stylesheet Language Transformations (XSLT) is a language for transforming XML documents into different XML, HTML, or any other format.

13.9 Review Questions

1. What is XML? What are the main development goals of XML?
2. Write the features of XML.
3. How do we generate XML document in PHP? Explain with example.
4. How do we parse XML in PHP? Which parsers are used in PHP?
5. How do we create a parser? Explain with example.
6. What are the unparsed entities? How does it declare?
7. How do we transform XML with XSLT?
8. Write the PHP code for XSL transformation from variables.
9. Write about the web services used in PHP.

Answers: Self Assessment

- | | |
|-------------------------------------|---------------|
| 1. False | 2. False |
| 3. True | 4. Closed |
| 5. Top | 6. Namespaces |
| 7. SetErrorNode | 8. Internet |
| 9. GET | 10. False |
| 11. True | 12. True |
| 13. Extensible Style sheet Language | 14. Xml |
| 15. .css | 16. False |
| 17. True | 18. True |

13.10 Further Readings

Notes



Books

Bangia, Ramesh (2008). *“Web Technology (including HTML, CSS, XML, ASP, JAVA).”* Firewall Media.

Jackson (2007). *“Web Technologies: A Computer Science Perspective.”* Pearson Education India.

Kamal, Raj (2002). *“Internet and Web Technologies.”* Tata McGraw-Hill Education.

Puntambekar, A.A. (2009). *“Web Technologies.”* Technical Publications.

Sarukkai, Ramiesh R. (2002). *“Foundations of Web Technology.”* Springer.

Xavier, C. (2007). *“Web Technology and Design.”* New Age International.



Online links

<http://nwalsh.com/docs/articles/xml/>

<http://www.epa.gov/cdx/about/xmlfactsheet.pdf>

<http://techforum4u.com/content.php/318-What-is-the-difference-between-HTML-and-XML>

<http://www.differencebetween.net/technology/difference-between-sgml-and-xml/>

<http://www.datazenengineering.com/data-content-management/data-management/integration-services/26-8-reasons-why-xml-is-right-for-data-transfer>

<http://webdesign.about.com/od/xml/tp/aa022706.htm>

<http://blog.jc21.com/2006-09-07/how-to-dynamically-return-xml-data-using-php/>

<http://phpmaster.com/parsing-xml-with-simplexml/>

https://developer.mozilla.org/en/docs/XML_Web_Services

<http://www.w3.org/International/O-charset.en.php>

<http://ciisa.isa.utl.pt/docs/php3/ref.xml.html>

<http://docs.python.org/2/library/xml.sax.handler.html>

<http://www.ibm.com/developerworks/library/os-xmlDOMphp/>

Unit 14: Security

CONTENTS

Objectives

Introduction

14.1 Global Variables and Form Data

14.1.1 Initialize Variables

14.1.2 Set Variables Order

14.1.3 Data Filtering

14.2 Filenames

14.2.1 Check for Relative Paths

14.2.2 Restrict Filesystem Access to a Specific Directory

14.3 File Uploads

14.3.1 Beware of Filling Your Filesystem

14.3.2 Surviving register_globals

14.3.3 Distrust Browser-Supplied Filenames

14.4 File Permissions

14.4.1 Do not Use Files

14.4.2 Get It Right the First Time

14.4.3 Session Files

14.4.4 Safe Mode

14.5 PHP Code

14.6 Shell Commands

14.7 Summary

14.8 Keywords

14.9 Review Questions

14.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Discuss about Global Variables and Form Data
- Understand File Uploads in PHP
- Explain about File Permissions in PHP
- Discuss about the PHP Code
- Explain the Shell Commands in PHP

Introduction

Notes

PHP is the most commonly used server-side programming language and 72% of web servers deploy PHP. PHP is open source. The core of PHP is reasonably secure, but its plug-ins, libraries and third party tools are often insecure. Also no default security mechanism is included in PHP (there were some in the old days, but they usually broke things). PHP developers are usually better informed than ASPX or JSP developers on how the web and HTTP works, and that makes for better coding practices, but they both lack basic security knowledge. Other languages have built-in security mechanisms, which is why PHP websites often have more flaws.

14.1 Global Variables and Form Data

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword GLOBAL in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example:



Example:

```
<?
$somevar = 15;
function addit() {
GLOBAL $somevar;
$somevar++;
print "Somevar is $somevar";
}
addit();
?>
```

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

The predefined `$_GET` variable is used to collect values in a form with `method="get"`

Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.



Example:

```
<form action="welcome.php" method="get">
Name: <input type="text" name="fname">
Age: <input type="text" name="age">
<input type="submit">
</form>
```

The predefined `$_POST` variable is used to collect values from a form sent with `method="post"`.

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

Notes



Notes However, there is an 8 MB max size for the POST method, by default (can be changed by setting the `post_max_size` in the `php.ini` file).



Example:

```
<form action="welcome.php" method="post">
Name: <input type="text" name="fname">
Age: <input type="text" name="age">
<input type="submit">
</form>
```

14.1.1 Initialize Variables

It is best to always initialize your variables before using them, because variables that are not initialized have a default value of their type depending on the context in which they are used. And it can also lead to script errors if certain variables are not initialized before a script runs.

Variables should have names that identify them well when you scan your code. Common sense variable naming helps you and others decipher your code after you write it.

Here is an example of valid variable names with prefilled dummy values, and display them on a web page:



Example:

```
<?php
// Setting variable names and values upon creation, names cannot start
with a number
$age = 37;
$name = "Joe";
$_car_color = "green";
// Now echo the sentence to display in the web browser
echo "My name is $name. I am $age years old and my car is $_car_color.";
?>
```

14.1.2 Set Variables Order

The default PHP configuration automatically creates global variables from the environment, cookies, server information, and GET and POST parameters. The `variables_order` directive in `php.ini` controls the order and presence of these variables. The default value is "EGPCS", meaning that first the environment is turned into global variables, then GET parameters, then POST parameters, then cookies, then server information.

Allowing GET requests, POST requests, and cookies from the browser to create arbitrary global variables in your program is dangerous. A reasonable security precaution is to set `variables_order` to "ES":

```
variables_order = "ES"
```

For maximum safety, you can disable `register_globals` in your `php.ini` file to prevent any global variables from being created. However, changing `register_globals` or `variables_order` will break scripts that were written with the expectation that form parameters would be accessible as

global variables. To fix this problem, add a section at the start of your code to copy the parameters into regular global variables:

```
$name = $_REQUEST['name']; $age = $_REQUEST['age']; // ... and so on for all
incoming form parameters
```

14.1.3 Data Filtering

PHP is often characterized as a ‘weak’ programming language. This is mostly because PHP is known to be an easy to learn language that is used as a footstep into web programming. Most of this misunderstanding is down to authors and tutorials that write about PHP and often concentrate only on how easy it is to write programs in PHP that simply collect data and then send it on through email or to a database, all the while forgetting to mention data validation. Beginners then go on to write these ‘easy’ scripts and find themselves subject to SQL injections and other forms of attacks that are easily preventable. One of the reasons why data validation is not mentioned in these tutorials and books is because validating user input is too ‘complicated’ for beginners and won’t comply with the notion that PHP is supposed to be ‘easy’ to program with. In reality, it only takes a few simple steps to validate user input. So what exactly do we mean by data validation and why is it so important? Validating data becomes important when your application starts to accept user input.



Did u know? A PHP script cannot trust any variable it has not explicitly set. When you have got a rather large number of variables, this can be a much harder task than it may sound.

The rule of thumb is not to trust any data that comes from outside your application i.e. from forms or through the browser. While any data that originate from within your application is ‘safe’. Any data that comes from outside needs to be ‘sanitized’ before it is accepted into your application. Example of ‘safe’ data is:

```
$myvar = "A safe variable";
```

The code above contains a variable that is defined within your application and can therefore be trusted. While the following data cannot be trusted:

```
$user = $_POST['username'];
$ID = $_GET['id'];
```

The code above shows a variable called \$user that comes from a form that is used to collect the user name. This data cannot be trusted since it comes from outside our application. On its own the variable is not harmful, but if it is used in a database it could potentially present a security problem (as we will demonstrate shortly). The second variable called \$ID is contained in a query string that can easily be tampered with when shown in a browser. For example if your query string was generated like this:

```
delete.php?id = echo $row['id'];
```

Then on the browser it will show the following line when the delete script is run:

```
delete.php?id=2
```

Any attacker will then be able to simply change the number to a letter to crash our application, which surprisingly in many cases reveals more security information about the application.

Let’s take a practical look at one of the most common and serious attacks that take place when data validation is not implemented, SQL injections. Below is a sample of a login script. Assume that a form takes the user name and password and sends it to a processing script that grants a user access to the rest of the application if their login details are correct:

Notes

*Example:*

```
<?php
$user = $_POST['username'];
$pass= $_POST['password'];  $result = "SELECT * FROM users WHERE name =
'$user' AND password = '$password'";
if($row = mysql_fetch_assoc($result)) {
    //authenticated
}
?>
```

As it is, there is nothing wrong with the above code, but consider if the variables contain the following information:

```
$user = "Dantago !Noabes";
$pass = "x' OR 'a'='a";
```

What does the above information do? The data contained in the \$pass variable is trying to fool your MySQL into thinking that the user is authenticated and that they should have access to the rest of the application. How does it do that? Take a look at what the MySQL code looks like with the above variables:

```
SELECT * FROM users WHERE name = 'Dantago !Noabes' AND password = 'x' OR
'a'='a'
```

So how does PHP help to validate data? You can create your own filters or use PHP's data filters that come with PHP version 5.2 and above. In addition, you can also prevent SQL injection using a function available in PHP called `mysql_real_escape_string()`. Let's look at how we can avoid SQL injection using the code from our previous example:

*Example:*

```
$user = "Dantago !Noabes";
$pass = "x' OR 'a'='a";
$clean_user = mysql_real_escape_string($user);
$clean_pass = mysql_real_escape_string($pass);
$q = "SELECT * FROM users WHERE name = '$clean_user.' AND password =
'$clean_pass.'";
$result = mysql_query($q);
if($row = mysql_fetch_assoc($result)) {
    //authenticated
}
```

Again, there is nothing wrong with the code above. It will now look like this in MySQL:

```
SELECT * FROM users WHERE name = 'Dantago !Noabes' AND password = 'x' OR
'a'='a' and will fail(unless 'a' is the correct password).
```

Data validation does not only revolve around SQL attacks. Other data can be validated, such as checking to see that an email address or URL is written in the proper format or ensuring that a particular value is of the right type. This can be particularly useful when checking that a query string value that is passed on to the application is what it is supposed to be. For example, a user ID is usually an integer and not a letter or string. This can be validated by using `int()` or `is_numeric()`.

Table 14.1: PHP Filter Functions

Function	Description
<code>filter_has_var()</code>	Checks if a variable of a specified input type exists
<code>filter_id()</code>	Returns the ID number of a specified filter
<code>filter_input()</code>	Get input from outside the script and filter it
<code>filter_input_array()</code>	Get multiple inputs from outside the script and filters them
<code>filter_list()</code>	Returns an array of all supported filters
<code>filter_var_array()</code>	Get multiple variables and filter them
<code>filter_var()</code>	Get a variable and filter it

Table 14.2: PHP Filters

Field	Description
<code>FILTER_CALLBACK</code>	Call a user-defined function to filter data
<code>FILTER_SANITIZE_STRING</code>	Strip tags, optionally strip or encode special characters
<code>FILTER_SANITIZE_STRIPPED</code>	Alias of "string" filter
<code>FILTER_SANITIZE_ENCODED</code>	URL-encode string, optionally strip or encode special characters
<code>FILTER_SANITIZE_SPECIAL_CHARS</code>	HTML-escape "<>&" and characters with ASCII value less than 32
<code>FILTER_SANITIZE_EMAIL</code>	Remove all characters, except letters, digits and !#\$%&*+ /=?^_`{ }~@.[]
<code>FILTER_SANITIZE_URL</code>	Remove all characters, except letters, digits and \$-_.+!*()' \ \ ^ ~ [] ` < > # % " ' / ? : @ & =
<code>FILTER_SANITIZE_NUMBER_INT</code>	Remove all characters, except digits and +-.
<code>FILTER_SANITIZE_NUMBER_FLOAT</code>	Remove all characters, except digits, +- and optionally .eE
<code>FILTER_SANITIZE_MAGIC_QUOTES</code>	Apply addslashes()
<code>FILTER_UNSAFE_RAW</code>	Do nothing, optionally strip or encode special characters
<code>FILTER_VALIDATE_INT</code>	Validate value as integer, optionally from the specified range
<code>FILTER_VALIDATE_BOOLEAN</code>	Return TRUE for "1", "true", "on" and "yes", FALSE for "0", "false", "off", "no", and "", NULL otherwise
<code>FILTER_VALIDATE_FLOAT</code>	Validate value as float
<code>FILTER_VALIDATE_REGEXP</code>	Validate value against regexp, a Perl-compatible regular expression
<code>FILTER_VALIDATE_URL</code>	Validate value as URL, optionally with required components
<code>FILTER_VALIDATE_EMAIL</code>	Validate value as e-mail
<code>FILTER_VALIDATE_IP</code>	Validate value as IP address, optionally only IPv4 or IPv6 or not from private or reserved ranges



Did u know? A good way to ensure that `security.inc` is always included at the top of every PHP script is to use the `auto_prepend_file` directive.

Self Assessment

State whether the following statements are true or false:

1. A global variable must be explicitly declared to be global in the function in which it is to be modified.

Notes

2. Information sent from a form with the GET method is not visible to everyone.
3. Information sent from a form with the POST method is visible to others.

14.2 Filenames

It is fairly easy to construct a filename that refers to something other than what you intended. For example, say you have a \$username variable that contains the name the user wants to be called, which the user has specified through a form field. Now let's say you want to store a welcome message for each user in the directory /user/local/lib/greetings, so that you can output the message any time the user logs into your application. The code to print the current user's greeting is:

```
<?php include("/usr/local/lib/greetings/$username") ?>
```

This seems harmless enough, but what if the user chose the username "../../../../etc/passwd"?

The code to include the greeting now includes /etc/passwd instead. Relative paths are a common trick used by hackers against unsuspecting scripts.

Another trap for the unwary programmer lies in the way that, by default, PHP can open remote files with the same functions that open local files. The fopen() function and anything that uses it (e.g., include() and require()) can be passed an HTTP or FTP URL as a filename, and the document identified by the URL will be opened. Here's some exploitable code:

```
<?php chdir("/usr/local/lib/greetings"); $fp = fopen($username, "r"); ?>
```

If \$username is set to "http://www.example.com/myfile", a remote file is opened, not a local one.

The situation is even more dire if you let the user tell you which file to include():

```
<?php $file = $_REQUEST['theme']; include($file); ?>
```

If the user passes a theme parameter of "http://www.example.com/badcode.inc" and your variables_order includes GET or POST, your PHP script will happily load and run the remote code. Never use parameters as filenames like this.

14.2.1 Check for Relative Paths

When you need to allow the user to specify a filename in your application, you can use a combination of the realpath() and basename() functions to ensure that the filename is what it ought to be. The realpath() function resolves special markers such as "." and "..". After a call to realpath(), the resulting path is a full path on which you can then use basename(). The basename() function returns just the filename portion of the path.

Going back to our welcome message scenario, here's an example of realpath() and basename() in action:

```
$filename = $_POST['username'];  
$vetted = basename(realpath($filename));  
if ($filename !==$vetted)  
{ die("$filename is not a good username"); }
```

In this case, we have resolved \$filename to its full path and then extracted just the filename. If this value does not match the original value of \$filename, we have got a bad filename that we do not want to use. Once you have the completely bare filename, you can reconstruct what the file path ought to be, based on where legal files should go, and add a file extension based on the actual contents of the file:

```
include("/usr/local/lib/greetings/$filename");
```

14.2.2 Restrict Filesystem Access to a Specific Directory

Notes

If your application must operate on the filesystem, you can set the `open_basedir` option to further secure the application by restricting access to a specific directory. If `open_basedir` is set in `php.ini`, PHP limits filesystem and I/O functions so that they can operate only within that directory or any of its subdirectories.



Example:

```
open_basedir = /some/path
With this configuration in effect, the following function calls succeed:
unlink("/some/path/unwanted.exe");
include("/some/path/less/travelled.inc");
But these generate runtime errors:
$fp = fopen ("/some/other/file.exe", "r");
$dp = opendir("/some/path/../other/file.exe");
```

Of course, one web server can run many applications, and each application typically stores files in its own directory. You can configure `open_basedir` on a per-virtual host basis in your `httpd.conf` file like this:



Example:

```
<VirtualHost 1.2.3.4>
    ServerName domainA.com
    DocumentRoot /web/sites/domainA
    php_admin_value open_basedir /web/sites/domainA
</VirtualHost>
```

Similarly, you can configure it per directory or per URL in `httpd.conf`:

```
# by directory
<Directory /home/httpd/html/app1>
    php_admin_value open_basedir /home/httpd/html/app1
</Directory>

# by URL
<Location /app2>
    php_admin_value open_basedir /home/httpd/html/app2
</Location>
```

The `open_basedir` directory can be set only in the `httpd.conf` file, not in `.htaccess` files, and you must use `php_admin_value` to set it.



Task Develop a PHP program to access a specific directory from your system.

Self Assessment

Fill in the blanks:

- The code to include the greeting now includes `/etc/passwd` instead. Relative paths are a common trick used by hackers against

Notes

5. Thefunction resolves special markers such as "." and "..".
6. The open_basedir directory can be set only in thefile, not in .htaccess files.

14.3 File Uploads

A very useful aspect of PHP is its ability to manage file uploads to your server. Allowing users to upload a file to your server opens a whole can of worms, so please be careful when enabling file uploads.

When the uploader.php file is executed, the uploaded file exists in a temporary storage area on the server. If the file is not moved to a different location it will be destroyed. To save our precious file we are going to need to make use of the \$_FILES associative array.

The \$_FILES array is where PHP stores all the information about files. There are two elements of this array that we will need to understand for this example.

- uploadedfile - uploadedfile is the reference we assigned in our HTML form. We will need this to tell the \$_FILES array which file we want to play around with.
- \$_FILES['uploadedfile']['name'] - name contains the original path of the user uploaded file.
- \$_FILES['uploadedfile']['tmp_name'] - tmp_name contains the path to the temporary file that resides on the server. The file should exist on the server in a temporary directory with a temporary name.

Now we can finally start to write a basic PHP upload manager script! Here is how we would get the temporary file name, choose a permanent name, and choose a place to store the file.

```

/ Where the file is going to be placed
$target_path = "uploads/";

/* Add the original filename to our target path.
Result is "uploads/filename.extension" */
$target_path = $target_path . basename( $_FILES['uploadedfile']['name']);

```

You will need to create a new directory in the directory where uploader.php resides, called "uploads", as we are going to be saving files there.

We now have all we need to successfully save our file to the server. \$target_path contains the path where we want to save our file to.

Now all we have to do is call the move_uploaded_file function and let PHP do its magic. The move_uploaded_file function needs to know: (1) The path of the temporary file (check!) (2) The path where it is to be moved to (check!).

```

$target_path = "uploads/";

$target_path = $target_path . basename( $_FILES['uploadedfile']['name']);

if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path))
{
    echo "The file ". basename( $_FILES['uploadedfile']['name']).
    " has been uploaded";
} else{

```



```
    echo "There was an error uploading the file, please try again!";
}
```

If the upload is successful, then you will see the text “The file filename has been uploaded”. This is because `move_uploaded_file` returns true if the file was moved, and false if it had a problem.

If there was a problem then the error message “There was an error uploading the file, please try again!” would be displayed.



Notes This script is for education purposes only. We do not recommend placing this on a web page viewable to the public.

These few lines of code we have given you will allow anyone to upload data to your server. Because of this, we recommend that you do not have such a simple file uploader available to the general public. Otherwise, you might find that your server is filled with junk or that your server’s security has been compromised.

14.3.1 Beware of Filling Your Filesystem

Another trap is the size of uploaded files. Although you can tell the browser the maximum size of file to upload, this is only a recommendation and it cannot ensure that your script won’t be handed a file of a larger size. The danger is that an attacker will try a denial of service attack by sending you several large files in one request and filling up the filesystem in which PHP stores the decoded files.

Set the `post_max_size` configuration option in `php.ini` to the maximum size (in bytes) that you want:

```
post_max_size = 1024768          ; one megabyte
```

The default 10 MB is probably larger than most sites require.

14.3.2 Surviving `register_globals`

Perhaps the most controversial change in PHP is when the default value for the PHP directive `register_globals` went from ON to OFF in PHP \approx 4.2.0. Reliance on this directive was quite common and many people didn’t even know it existed and assumed it’s just how PHP works. It will explain how one can write insecure code with this directive but keep in mind that the directive itself isn’t insecure but rather it’s the misuse of it.

When on, `register_globals` will inject your scripts with all sorts of variables, like request variables from HTML forms. This coupled with the fact that PHP doesn’t require variable initialization means writing insecure code is that much easier. It was a difficult decision, but the PHP community decided to disable this directive by default. When on, people use variables yet really don’t know for sure where they come from and can only assume. Internal variables that are defined in the script itself get mixed up with request data sent by users and disabling `register_globals` changes this. Let’s demonstrate with an example misuse of `register_globals`:



Example:

```
<?php
// define $authorized = true only if user is authenticated
if (authenticated_user()) {
```

Notes

```
        $authorized = true;
    }

    // Because we didn't first initialize $authorized as false, this might
    be
    // defined through register_globals, like from GET auth.php?authorized=1
    // So, anyone can be seen as authenticated!
    if ($authorized) {
        include "/highly/sensitive/data.php";
    }
?>
```

When `register_globals = on`, our logic above may be compromised. When off, `$authorized` can't be set via request so it'll be fine, although it really is generally a good programming practice to initialize variables first. For example, in our example above we might have first done `$authorized = false`. Doing this first means our above code would work with `register_globals` on or off as users by default would be unauthorized.

14.3.3 Distrust Browser-Supplied Filenames

Be careful using the filename sent by the browser. If possible, do not use this as the name of the file on your filesystem. It's easy to make the browser send a file identified as `/etc/passwd` or `/home/rasmus/.forward`. You can use the browser-supplied name for all user interaction, but generate a unique name yourself to actually call the file.



Example:

```
$browser_name = $_FILES['image']['name'];
$temp_name = $_FILES['image']['tmp_name'];
echo "Thanks for sending me $browser_name.";

$counter++; // persistent variable
$my_name = "image_$counter";
if (is_uploaded_file($temp_name)) {
    move_uploaded_file($temp_name, "/web/images/$my_name");
} else {
    die("There was a problem processing the file.");
}
```

Self Assessment

State whether the following statements are true or false:

7. The `$_FILES` array is where PHP stores all the information about files.
8. If the upload is successful, then you will see the text "The file filename has not been uploaded".
9. If there was a problem then the error message "There was an error uploading the file, please try again!" would be displayed.

14.4 File Permissions

The `chmod()` function changes permissions of the specified file. Returns `TRUE` on success and `FALSE` on failure.

Syntax

```
chmod(file,mode)
```

Parameter Description

`file` Required. Specifies the file to check

Parameter	Description
<code>file</code>	Required. Specifies the file to check
<code>mode</code>	Required. Specifies the new permissions. The mode parameter consists of four numbers: <ul style="list-style-type: none"> • The first number is always zero • The second number specifies permissions for the owner • The third number specifies permissions for the owner's user group • The fourth number specifies permissions for everybody else Possible values (to set multiple permissions, add up the following numbers): <ul style="list-style-type: none"> • 1 = execute permissions • 2 = write permissions • 4 = read permissions



Example:

```
<?php
// Read and write for owner, nothing for everybody else
chmod("test.txt",0600);

// Read and write for owner, read for everybody else
chmod("test.txt",0644);

// Everything for owner, read and execute for everybody else
chmod("test.txt",0755);

// Everything for owner, read for owner's group
chmod("test.txt",0740);
?>
```

14.4.1 Do not Use Files

Because all scripts running on a machine run as the same user, a file that one script creates can be read by another, regardless of which user wrote the script. All a script needs to know to read a file is the name of that file.

There is no way to change this, so the best solution is to not use files. As with session stores, the most secure place to store data is in a database.

A complex workaround is to run a separate Apache daemon for each user. If you add a reverse proxy such as Squid in front of the pool of Apache instances, you may be able to serve 100+ users on a single machine. Few sites do this, however, because the complexity and cost are much greater than those for the typical situation, where one Apache daemon can serve web pages for thousands of users.

Notes

14.4.2 Get It Right the First Time

Do not create a file and then change its permissions. This creates a race condition, where a lucky user can open the file once it is created but before it is locked down. Instead, use the `umask()` function to strip off unnecessary permissions. For example:

```
umask(077); // disable -rwxrwx $fp = fopen("/tmp/myfile", "w");
```

By default, the `fopen()` function attempts to create a file with permission 0666 (rw-rw-rw-). Calling `umask()` first disables the group and other bits, leaving only 0600 (rw---). Now, when `fopen()` is called, the file is created with those permissions.

14.4.3 Session Files

A PHP session file is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc.). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.

Sessions work by creating a Unique ID (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

Before you can store user information in your PHP session, you must first start up the session.



Notes The `session_start()` function must appear BEFORE the `<html>` tag:

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

```
<html>
```

```
<body>
```

```
</body>
```

```
</html>
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

The correct way to store and retrieve session variables is to use the PHP `$_SESSION` variable:



Example:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
```

```
?>

<html>
<body>

<?php
//retrieve session data
echo "Pageviews=". $_SESSION[ 'views' ];
?>

</body>
</html>
```

If you wish to delete some session data, you can use the `unset()` or the `session_destroy()` function.

The `unset()` function is used to free the specified session variable:

```
<?php
session_start();
if(isset($_SESSION[ 'views' ]))
    unset($_SESSION[ 'views' ]);
?>
```

You can also completely destroy the session by calling the `session_destroy()` function:

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

14.4.4 Safe Mode

Many ISPs have scripts from several users running on one web server. Since all the users who share such a server run their PHP scripts as the same user, one script can read another's data files. Safe mode is an attempt to address this and other problems caused by shared servers. If you are not sharing your server with other users that you do not trust, you do not need to worry about safe mode at all.

When enabled through the `safe_mode` directive in your `php.ini` file, or on a per-directory or pervirtual host basis in your `httpd.conf` file, the following restrictions are applied to PHP scripts:

- PHP looks at the owner of the running script and pretends to run as that user.
- PHP cannot switch the user ID via a `setuid()` call because that would require the web server to run as root and on most operating systems it would be impossible to switch nback.
- Any file operation (through functions such as `fopen()`, `copy()`, `rename()`, `move()`, `unlink()`, `chmod()`, `chown()`, `chgrp()`, `mkdir()`, `file()`, `flock()`, `rmdir()`, and `dir()`) checks to see if the affected file or directory is owned by the same user as the PHP script.
- If `safe_mode_gid` is enabled in your `php.ini` or `httpd.conf` file, only the group ID needs to match.
- `include` and `require` are subject to the two previous restrictions, with the exception of `includes` and `requires` of files located in the designated `safe_mode_include_dir` in your `php.ini` or `httpd.conf` file.
- Any system call (through functions such as `system()`, `exec()`, `passthru()`, and `popen()`) can access only executables located in the designated `safe_mode_exec_dir` in your `php.ini` or `httpd.conf` file.

Notes

- If `safe_mode_protected_env_vars` is set in your `php.ini` or `httpd.conf` file, scripts are unable to overwrite the environment variables listed there.
- If a prefix is set in `safe_mode_allowed_env_vars` in your `php.ini` or `httpd.conf` file, scripts can manipulate only environment variables starting with that prefix.
- When using HTTP authentication, the numerical user ID of the current PHP script is appended to the realm string to prevent cross-script password sniffing, and the authorization header in the `getallheaders()` and `phpinfo()` output is hidden.
- This realm-mangling took a little vacation in PHP 4.0.x but is back in PHP 4.1 and later.
- The functions `set_time_limit()`, `dl()`, and `shell_exec()` are disabled, as is the backtick (") operator.

To configure `safe_mode` and the various related settings, you can set the serverwide default in your `php.ini` file like this:

```
safe_mode = On safe_mode_include_dir = /usr/local/php/include
safe_mode_exec_dir = /usr/local/php/bin safe_mode_gid = On
safe_mode_allowed_env_vars = PHP_ safe_mode_protected_env_vars =
LD_LIBRARY_PATH
```

Alternately, you can set these from your `httpd.conf` file using the `php_admin_value` directive. Remember, these are system-level settings, and they cannot be set in your `.htaccess` file.

```
<VirtualHost 1.2.3.4> ServerName domainA.com DocumentRoot /web/sites/domainA
php_admin_value safe_mode On php_admin_value safe_mode_include_dir /usr/
local/php/includephp_admin_value safe_mode_exec_dir /usr/local/php/bin </
VirtualHost>
```

Self Assessment

Fill in the blanks:

10. All a script needs to know to read a file is theof that file.
11. A PHP session file is used to store information about, or change settings for asession.
12. Theis either stored in a cookie or is propagated in the URL.

14.5 PHP Code

The `eval()` function evaluates a string as PHP code.

The string must be valid PHP code and must end with semicolon.

This function returns `NULL` unless a return statement is called in the code string. Then the value passed to return is returned. If there is a parse error in the code string, `eval()` returns `FALSE`.

```
Syntax
eval (phpcode)
```



Example:

```
<?php
$string = "beautiful";
$time = "winter";
```

```
$str = 'This is a $string $time morning!';
echo $str. "<br />";

eval("\$str = \"\$str\";");
echo $str;
?>
```



Caution eval() is a useful but very dangerous function that allows you to execute a string as PHP code. There are not many occasions where this is necessary, and being realistic you should avoid its usage, especially if you want to use user input in the string.

Self Assessment

State whether the following statements are true or false:

13. The string must not be valid PHP code.
14. The eval() function evaluates a string as PHP code.
15. The string must end with semicolon.

14.6 Shell Commands

The command shell is a separate software program that provides direct communication between the user and the operating system. The non-graphical command shell user interface provides the environment in which you run character-based applications and utilities. The command shell executes programs and displays their output on the screen by using individual characters similar to the MS-DOS command interpreter Command.com. The Windows XP command shell uses the command interpreter Cmd.exe, which loads applications and directs the flow of information between applications, to translate user input into a form that the operating system understands.

You can use the command shell to create and edit batch files (also called scripts) to automate routine tasks. For example, you can use scripts to automate the management of user accounts or nightly backups. You can also use the Windows Script Host, CScript.exe, to run more sophisticated scripts in the command shell. You can perform operations more efficiently by using batch files than you can by using the user interface. Batch files accept all commands that are available at the command line. For more information about batch files and scripting, You can customize the command prompt window for easier viewing and to increase control over how you run programs.

Be very wary of using the exec(), system(), passthru(), and popen() functions and the backtick (") operator in your code. The shell is a problem because it recognizes special characters (e.g., semicolons to separate commands). For example, suppose your script contains this line:

```
system("ls $directory");
```

If the user passes the value "/tmp;cat /etc/passwd" as the \$directory parameter, your password file is displayed because system() executes the following command:

```
ls /tmp;cat /etc/passwd
```

In cases where you must pass user-supplied arguments to a shell command, use escapeshellarg() on the string to escape any sequences that have special meaning to shells:

```
$cleaned_up = escapeshellarg($directory); system("ls $cleaned_up");
```

Now, if the user passes "/tmp;cat /etc/passwd", the command that's actually run is:

```
ls `tmp;cat /etc/passwd`
```

Notes The easiest way to avoid the shell is to do the work of whatever program you are trying to call. Built-in functions are likely to be more secure than anything involving the shell.

Self Assessment

Fill in the blanks:

16. Theis a separate software program that provides direct communication between the user and the operating system.
17. The shell is a problem because it recognizes special.....
18. Thecommand shell user interface provides the environment in which you run character-based applications and utilities.



Case Study

Cyber Security

A provider of online prescriptions recently experienced a security breach where account information was stolen out of the company’s database, including patient’s social security numbers. You have been hired as a consultant to conduct a thorough analysis of the information system in order to develop recommendations for improved security. You need to develop a thorough understanding of the existing system, and of which security tools, security measures and intrusion detection systems are currently in place. You also need to gain knowledge of which internal and external “users” of the information system have access to what information, what level of privilege they hold, and why they need the information and what they do with it. The research process will involve examining detailed technical specifications and system administration procedures, interviewing users of the system, reviewing security procedures and information flow diagrams. As part of the proposed solution, you will run scenarios to test system vulnerabilities. You will need to educate yourself on the regulations that are pertinent to the management of information in the context of online pharmacies.

Recommendations will most likely include technical upgrade to the system, revisions of information access protocols and upgrade to user authentication. It may include training of company personnel at all levels of the organization. You may recommend improved system maintenance and regular security tests of the system for vulnerabilities. Some of these solutions may require significant investment of money and time and you will need to clearly show the necessities of these investments against the potential cost of non-compliance.

Questions

1. Give some example of Cyber Security.
2. What do you understand by cyber crime?

14.7 Summary

- The command shell is a separate software program that provides direct communication between the user and the operating system.
- The eval() function evaluates a string as PHP code.

Notes

- A PHP session file is used to store information about, or change settings for a user session.
- The `chmod()` function changes permissions of the specified file. Returns TRUE on success and FALSE on failure.
- A very useful aspect of PHP is its ability to manage file uploads to your server.
- In contrast to local variables, a global variable can be accessed in any part of the program.

14.8 Keywords

Command Shell: The command shell is a separate software program that provides direct communication between the user and the operating system.

Data Filtering: Data filtering is the cornerstone of web application security, and this is independent of programming language or platform.

File Uploads: File uploads are potentially the biggest security risk in web development.

Safe Mode: Safe mode is an attempt to address this and other problems caused by shared servers.

Session Files: With PHP's built-in session support, session information is stored in files in the/`tmp` directory.

14.9 Review Questions

1. What are the global variables and form data? How does it use in PHP?
2. How do we initialize the variable and set their orders in PHP?
3. What is the data filtering? Describe the different methods of data filtering.
4. How the filenames are defined with PHP code? Explain with example.
5. Explain how the file uploads are potentially the biggest security risk in web development.
6. What are the precautions when uploading a file?
7. Write a PHP program to upload an image in your web application.
8. What about the file permission in PHP? What is the safe mode in PHP?
9. How PHP allows a script to execute arbitrary PHP code?
10. How the shell commands are used in PHP?

Answers: Self Assessment

- | | |
|----------------------------|----------------------------|
| 1. True | 2. False |
| 3. False | 4. Unsuspecting Scripts |
| 5. <code>Realpath()</code> | 6. <code>httpd.conf</code> |
| 7. True | 8. False |
| 9. True | 10. Name |
| 11. User | 12. UID |
| 13. False | 14. True |

Notes

- | | |
|----------------|-------------------|
| 15. True | 16. Command Shell |
| 17. Characters | 18. Non-graphical |

14.10 Further Readings



Books

Bangia, Ramesh (2008). *Web Technology (including HTML, CSS, XML, ASP, JAVA)*. Firewall Media.

Jackson (2007). *Web Technologies: A Computer Science Perspective*. Pearson Education India.

Kamal, Raj (2002). *Internet and Web Technologies*. Tata McGraw-Hill Education.

Puntambekar, A.A. (2009). *Web Technologies*. Technical Publications.

Sarukkai, Ramiesh R. (2002). *Foundations of Web Technology*. Springer.

Xavier, C. (2007). *Web Technology and Design*. New Age International.



Online links

<http://www.w3schools.com/php>

<http://www.tutorialspoint.com/php>

<http://www.webreference.com/programming/php/DataFiltering/index.html>

http://docstore.mik.ua/oreilly/webprog/php/ch12_02.htm

<http://www.tizag.com/phpT/fileupload.php>

<http://php.net/manual/en/security.globals.php>

LOVELY PROFESSIONAL UNIVERSITY

Jalandhar-Delhi G.T. Road (NH-1)

Phagwara, Punjab (India)-144411

For Enquiry: +91-1824-300360

Fax.: +91-1824-506111

Email: odl@lpu.co.in