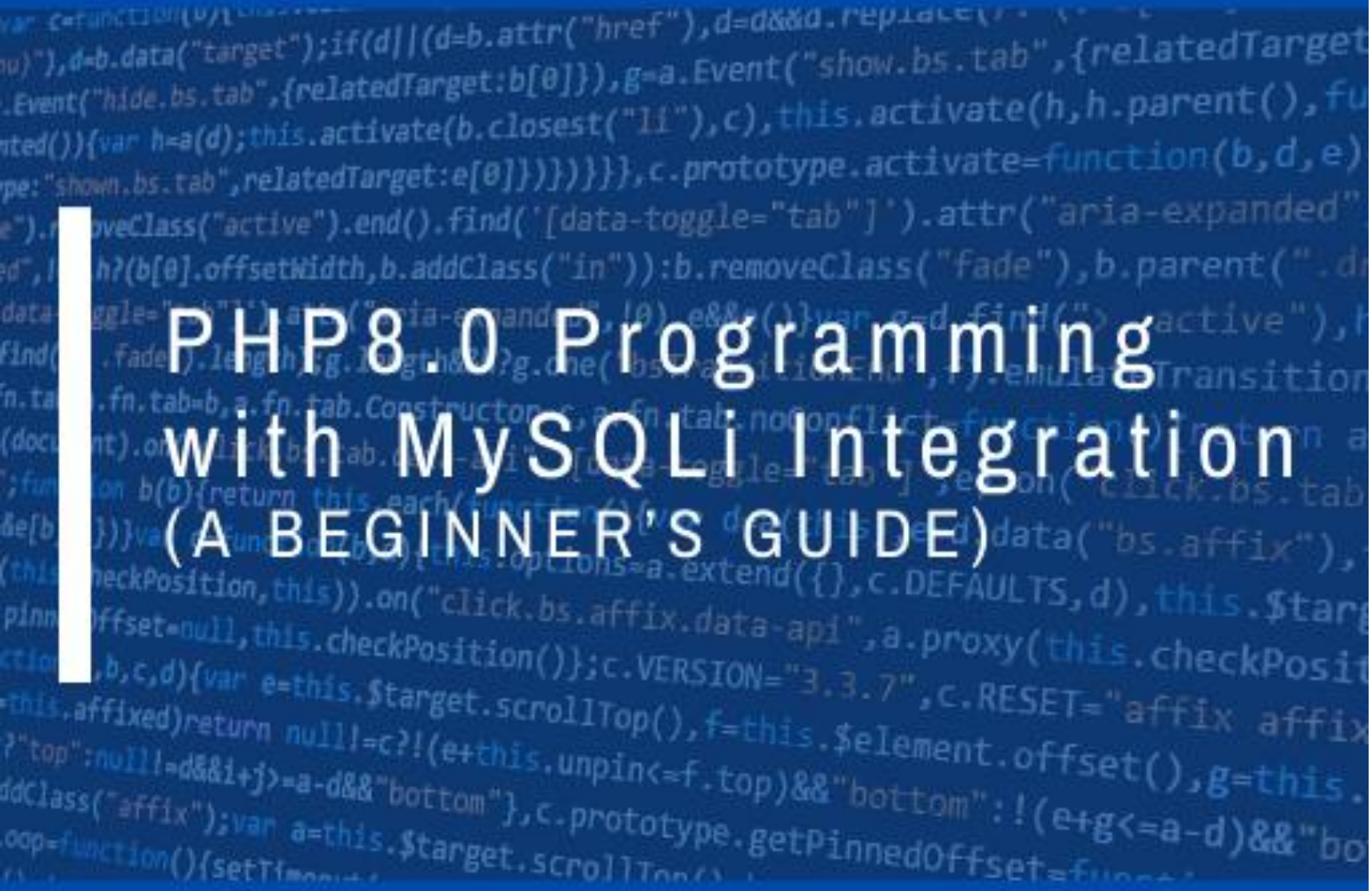
MODULE FOR
WEB SYSTEMS AND TECHNOLOGIES 3



PHP8.0 Programming with MySQLi Integration (A BEGINNER'S GUIDE)

ARNOLD P. DELA CRUZ
ALLEN PAUL L. ESTEBAN
ROLAIDA L. SONZA
JET C. AQUINO

Objectives

The objective of PHP/MySQL book is to train students in becoming proficient PHP/MySQL web developers.

At the end of the topic, students will have basic understanding of the web technology and be able to architect, write, debug, and run complete web applications using PHP and MySQL.

Contents

- Get up to speed with tried-and-true programming strategies, for both Web and local development.
- Gain valuable tips and tricks for using the PHP language and syntax.
- See how to integrate MySQL in Windows, PHP's most popular development platform.
- Learn how to streamline PHP's interaction with MySQL, SQL, and relational databases in general.
- See how Object Orientated Programming with PHP can allow for code reuse, modularity and rapid development.
- Watch how PHP's sessions can be used to quickly build full-scale web applications.
- Begin to see how PHP Technology can be leveraged to rapidly create and maintain large enterprise web applications.

Table of Contents

Chapter

PHP is the engine behind millions of dynamic web applications. Its broad feature set, approachable syntax, and support for different operating systems and web servers have made it an ideal language for both rapid web development and the methodical construction of complex systems.

One of the major reasons for PHP's success as a web scripting language is its origins as a tool to process HTML forms and create web pages. This makes PHP very web-friendly. PHP can speak to a multitude of databases, and it knows numerous Internet protocols.

The best things in using PHP are that it is extremely simple for a new comer, but offers many advanced features for a professional programmer. Don't be afraid reading the long of PHP's features. You can jump in, in a short time, and start writing simple scripts in a few hours.

REFERENCES

PHP Programming, Don Gossellin, Diana Kokoska, Robert Easterbrooks

PHP and MySQL Web Development, Luke Welling

PHP Programming with MySQL, Don Gossellin, Diana Kokoska, Robert Easterbrooks

A Complete Guide to MySQL, Philip J. Pratt , Mary Z. Last

ABOUT THE AUTHOR

Dr. Arnold P. dela Cruz has been working as a professor in computer programming for 26 years, and he is an Associate Professor III and Dean in the College of Information and Communications Technology of the Nueva Ecija University of Science and Technology.

He received his Master's degree in Information Technology (MSIT) in Hannam University South Korea and Doctorate in Information Technology (DIT) in Angeles University Foundation. He is also a member of different organizations like Computing Society of the Philippines (CSP), Philippine Schools, Universities, and Colleges Computer Education and Systems Society, Inc (PSUCCESS), Graduate Education Association of Chartered Colleges and Universities of the Philippines (GEACCUP III), and Philippine Society of Information Technology Educators Foundation, Inc. (PSITE).

PHP 8.0 Programming with MySQLi Integration
(A Beginner's Guide)

Arnold P. dela Cruz, DIT

Author

Editor

Allen Paul L. Esteban, MSIT
Faculty Member – NEUST – Graduate School

Compiler

Jet C. Aquino, DIT
Faculty Member – NEUST – Graduate School

Rolaida L. Sonza, DIT
Faculty Member – NEUST – Graduate School

CHAPTER 1

Getting Started with PHP

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software (OSS)
- PHP is free to download and use
- **PHP 8.0** is a major update of the PHP language. It contains many new features and optimizations including named arguments, union types, attributes, constructor property promotion, match expression, nullsafe operator, JIT, and improvements in the type system, error handling, and consistency.

What is a PHP File?

- PHP files may contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (means that you can develop in Windows and serve on a Unix platform)

PHP MySQLi

- The MySQLi functions allows you to access MySQL database servers.
- Note: The MySQLi extension is designed to work with MySQL version 4.1.13 or newer.

- Basically, MySQL is the old database driver, and MySQLi is the Improved driver. The "i" stands for "improved" so it is MySQL improved. MySQLi can be done procedural and object-oriented whereas MySQL can only be used procedurally. Mysqli also supports prepared statements which protect from SQL Injection.

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

- Install an Apache server on a Windows or Linux machine
- Install PHP on a Windows or Linux machine
- Install MySQL on a Windows or Linux machine

Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled, you can start a scripting block with <? and end with ?>.

However, for maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

```
<?php
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>
<?php
     echo "Hello World";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The **semicolon** is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above, we have used the echo statement to output the text "Hello World".

Note: The file must have the .php extension. In file with the .html extension, the PHP code will not be executed.

Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>
<?php
     //This is a comment
     /*
     This is
     a comment
     block
     */
?>
</body>
</html>
```

Variables in PHP

Variables are used for storing values, like text strings, numbers or arrays.

When a variable is set, it can be used over and over again in your script

All variables in PHP start with a **$** sign symbol.

The correct way of setting a variable in PHP:

```
$var_name = value;
```

New PHP programmers often forget the $ sign at the beginning of the variable. In that case it will not work.

Let's try creating a variable with a string, and a variable with a number:

```
<?php
     $txt = "Hello World!";
     $number = 16;
?>
```

PHP is a Loosely Typed Language

In PHP a variable does not need to be declared before being set.

In the example above, you see that you do not have to tell PHP which data type the variable is.

PHP automatically converts the variable to the correct data type, depending on how they are set.

In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

In PHP the variable is declared automatically when you use it.

Variable Naming Rules

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with underscore ($my_string), or with capitalization ($myString)
- Variables, on the other hand, are case-sensitive. That is, $name, $NAME, and $NaME are three different variables

Strings in PHP

String variables are used for values that contains character strings.

After we create a string, we can manipulate it. A string can be used directly in a function or it can be stored in a variable.

Below, the PHP script assigns the string "Hello World" to a string variable called $txt:

```
<?php
     $txt="Hello World";
     echo $txt;
?>
```

The output of the code above will be:

```
Hello World
```

The Concatenation Operator

The **concatenation operator** (.) is used to put two string values together.

To concatenate two variables together, use the dot (.) operator:

```
<?php
     $txt1="Hello World";
     $txt2="1234";
     echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be:

```
Hello World 1234
```

If we look at the code above, you will see that we used the concatenation operator two times. This is because we had to insert a third string. Between the two string variables, we added a string with a single character, an empty space, to separate the two variables.

Using the strlen() function

The **strlen()** function is used to find the length of a string.

Let's find the length of our string "Hello world!":

```
<?php
     echo strlen("Hello world!");
?>
```

The output of the code above will be:

```
12
```

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string)

Using the strpos() function

The **strpos()** function is used to search for a string or character within a string.

If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string:

```
<?php
     echo strpos("Hello world!","world");
?>
```

The output of the code above will be:

```
6
```

Using the strrev() function

The **strrev()** function reverses a string.

Example

```
<?php
      echo strrev("Hello World!");
?>
```

The output of the code above will be:

```
!dlroW olleH
```

Name : _____ Section : _____

Exercise 1

- Store your name, address, phone number and email address in variables with valid names.
- Display the value of the variables in different lines.

Exercise 2

- Given a string, create a PHP program that will create a new palindrome-like string based on the original string.
- Example Output:

```
The String is  ABCDE
The Output is  ABCDEEDCBA
```

PHP Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

CHAPTER 2

Functions and Control Structures

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

- **if...else statement** - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** - is used with the if...else statement to execute a set of code if **one** of several condition is true

The if...else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

```
if (<condition>)
  <code to be executed if condition is true>;
else
  <code to be executed if condition is false>;
```

Example

The example will output “a is bigger than b”, otherwise it will output “a is smaller or equal to b” :

```
<html>
<body>
<?php
     $a = 8;
     $b = 7;
     if ($a > $b)
        echo “a is bigger than b”;
     else
        echo “a is smaller or equal to b”;
?>
</body>
</html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>
<body>
<?php
      $a = 5;
      if ($a==5)
         {
          echo "Your Grade is 5! <br>" ;
          echo "Have a nice weekend!";
          echo "See you on next semester";
         }
?>
</body>
</html>
```

The elseif Statement

If you want to execute some code if one of several conditions are true use the elseif statement

Syntax

```
if (<condition>)
  <code to be executed if condition is true>;
elseif (<condition>)
  <code to be executed if condition is true>;
else
  <code to be executed if condition is false>;
```

Example

The following example will output "*a is equal to b*", and "*a is bigger than b*" if the value of $a is greater than 7, otherwise it will output "*a is smaller than b*":

```
<html>
<body>
<?php
      $a = 7; $b = 7;
      if ($a > $b)
         echo "a is bigger than b";
      elseif ($a == $b)
         echo "a is equal to b";
      else
         echo "a is smaller than b";
?>
```

```
</body>
</html>
```

The Switch Statement

If you want to select one of many blocks of code to be executed, use the switch statement.

The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

```
switch (<expression>)
{
case label1:
  code to be executed if expression = label1;
  break;
case label2:
  code to be executed if expression = label2;
  break;
default:
  code to be executed
  if expression is different
  from both label1 and label2;
}
```

Example

```
<html>
<body>
<?php
      switch ($x)
      {
      case 1:
        echo "Number 1";
        break;
      case 2:
        echo "Number 2";
        break;
      case 3:
        echo "Number 3";
        break;
      default:
        echo "No number between 1 and 3";
}
?>
</body> </html>
```

Exercise 3

- Write a PHP program that will output the *Total*, *Average* and *Remarks* of the Prelim, Midterm and Final grades, if the average is less than 75 the remark is "Failed" else "Passed".

Name : _____ Section : _____

Exercise 4

- Give the student final grade with the ff. grading scheme: there are three quizzes worth 10 points each; a laboratory worth 100 point; and exam worth 100 points. The grade is based on the ff. weights: 50% on the exam; 25% on the laboratory; and 25% on the quiz average. The grade is determined from the weighted average in the traditional way: 90 or over is "A"; below 90 to 80 is "B"; and below 80 to 70 is "C", and below 70 is "F".

Name : _____ Section : _____

Looping

Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.

In PHP we have the following looping statements:

- **while** - loops through a block of code if and as long as a specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as a special condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

The while Statement

The while statement will execute a block of code **if and as long as** a condition is true.

Syntax

```
while (<condition>)
   <code to be executed>;
```

Example

The following example demonstrates a loop that will continue to run as long as the variable i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>
<?php
     $i=1;
     while($i<=5)
       {
       echo "The number is " . $i . "<br />";
       $i++;
       }
?>
</body>
</html>
```

The do...while Statement

The do...while statement will execute a block of code **at least once** - it then will repeat the loop **as long as** a condition is true.

Syntax

```
do
{
  <code to be executed>;
}
while (<condition>);
```

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 5:

```
<html>
<body>
<?php
     $i=0;
     do
       {
       $i++;
       echo "The number is " . $i . "<br />";
       }
     while ($i<5);
?>
</body>
</html>
```

The for Statement

The **for statement** is the most advanced of the loops in PHP.

In its simplest form, the for statement is used when you know how many times you want to execute a statement or a list of statements.

Syntax

```
for (<init; cond; incr>)
{
  <code to be executed>;
}
```

Parameters:

- **init**: Is mostly used to set a counter, but can be any code to be executed once at the beginning of the loop statement.
- **cond**: Is evaluated at beginning of each loop iteration. If the condition evaluates to TRUE, the loop continues and the code executes. If it evaluates to FALSE, the execution of the loop ends.
- **incr**: Is mostly used to increment a counter, but can be any code to be executed at the end of each loop.

Note: Each of the parameters can be empty or have multiple expressions separated by commas.

- **cond:** All expressions separated by a comma are evaluated but the result is taken from the last part. This parameter being empty means the loop should be run indefinitely. This is useful when using a conditional break statement inside the loop for ending the loop.

Example

The following example prints the text "Hello World!" five times:

```
<html>
<body>
<?php
     for ($i=1; $i<=5; $i++)
     {
       echo "Hello World!<br />";
     }
?>
</body>
</html>
```

The foreach Statement

For every loop, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop, you'll be looking at the next element.

Syntax

```
foreach (<array as value>)
{
    <code to be executed>;
}
```

Example

```
<html>
<body>
<?php
$arr=array("one", "two", "three");
foreach ($arr as $value)
{
  echo "Value: " . $value . "<br />";
}
?>
</body> </html>
```

Name : _____ Section : _____

Exercise 5

- Read 5 scores and output the total and average of 5 scores.

Name : _____ Section : _____

Exercise 6

- Write a multiplication table using loop.

PHP Functions

Create a PHP Function

A **function** is a block of code that can be executed whenever we need it.

Creating PHP functions:

- All functions start with the word "function()"
- Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Add a "{" - The function code starts after the opening curly brace
- Insert the function code
- Add a "}" - The function is finished by a closing curly brace

Example

A simple function that writes my name when it is called:

```
<html>
<body>
<?php
     function writeMyName()
       {
        echo "Arnold Dela Cruz";
       }
     writeMyName();
?>
</body>
</html>
```

Use a PHP Function

Now we will use the function in a PHP script:

```
<html>
<body>
<?php
     function writeMyName()
       {
       echo "Arnold Dela Cruz";
```

```
      }

    echo "Hello world!<br />";
    echo "My name is ";
    writeMyName();
    echo ".<br />That's right, ";
    writeMyName();
    echo " is my name.";
?>
</body>
</html>
```

The output of the code above will be:

```
Hello world!
My name is Arnold Dela Cruz.
That's right, Arnold Dela Cruz is my name.
```

PHP Functions - Adding parameters

Our first function (writeMyName()) is a very simple function. It only writes a static string.

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

You may have noticed the parentheses after the function name, like: writeMyName(). The parameters are specified inside the parentheses.

Example 1

The following example will write different first names, but the same last name:

```
<html>
<body>
<?php
    function writeMyName($fname)
      {
       echo $fname . "Dela Cruz.<br />";
      }
    echo "My name is ";
    writeMyName("Joseph");
    echo "My name is ";
    writeMyName("Gloria");
    echo "My name is ";
```

```
     writeMyName("Pinoy");
?>
</body>
</html>
```

The output of the code above will be:

```
My name is Joseph Dela Cruz.
My name is Gloria Dela Cruz.
My name is Pinoy Dela Cruz.
```

Example 2

The following function has *two parameters*:

```
<html>
<body>
<?php
     function writeMyName($fname,$punctuation)
       {
        echo $fname . "Dela Cruz" . $punctuation . "<br />";
       }
     echo "My name is ";
     writeMyName("Joseph",".");
     echo "My name is ";
     writeMyName("Gloria","!");
     echo "My name is ";
     writeMyName("Pinoy","...");
?>
</body>
</html>
```

The output of the code above will be:

```
My name is Joseph Dela Cruz.
My name is Gloria Dela Cruz!
My name is Pinoy Dela Cruz...
```

PHP Functions - Return values

Functions can also be used to return values.

Example

```
<html>
```

```
<body>
<?php
     function add($x,$y)
       {
        $total = $x + $y;
        return $total;
       }
     echo "1 + 16 = " . add(1,16);
?>
</body>
</html>
```

The output of the code above will be:

```
1 + 16 = 17
```

The PHP Date() Function

The PHP date() function formats a timestamp to a more readable date and time.

Syntax

```
date(format,timestamp)
```

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time (as a timestamp)

PHP Date - Format the Date

The first parameter in the date() function specifies how to format the date/time. It uses letters to represent date and time formats. Here are some of the letters that can be used:

- d - The day of the month (01-31)
- m - The current month, as a number (01-12)
- Y - The current year in four digits

Other characters, like"/", ".", or "-" can also be inserted between the letters to add additional formatting:

```
<?php
	echo date("Y/m/d");
	echo "<br />";
	echo date("Y.m.d");
	echo "<br />";
	echo date("Y-m-d");
?>
```

The output of the code above could be something like this:

```
2006/07/11
2006.07.11
2006-07-11
```

PHP Date - Adding a Timestamp

The second parameter in the date() function specifies a timestamp. This parameter is optional. If you do not supply a timestamp, the current time will be used.

In our next example we will use the mktime() function to create a timestamp for tomorrow.

The mktime() function returns the Unix timestamp for a specified date.

Syntax

```
mktime(hour,minute,second,month,day,year,is_dst)
```

To go one day in the future we simply add one to the day argument of mktime():

```
<?php
$tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y"));
echo "Tomorrow is ".date("Y/m/d", $tomorrow);
?>
```

The output of the code above could be something like this:

```
Tomorrow is 2006/07/12
```

The include() Function

The **include() function** takes all the text in a specified file and copies it into the file that uses the include function.

Example 1

Assume that you have a standard header file, called "header.php". To include the header file in a page, use the include() function, like this:

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

Example 2

Now, let's assume we have a standard menu file that should be used on all pages (include files usually have a ".php" extension). Look at the "menu.php" file below:

```
<html>
<body>
<a href="http://www.sites.com/default.php">Home</a> |
<a href="http://www.sites.com/about.php">About Us</a> |
<a href="http://www.sites.com/contact.php">Contact Us</a>
```

The three files, "default.php", "about.php", and "contact.php" should all include the "menu.php" file. Here is the code in "default.php":

```
<?php include("menu.php"); ?>
<h1>Welcome to my home page</h1>
<p>Some text</p>
</body>
</html>
```

If you look at the source code of the "default.php" in a browser, it will look something like this:

```
<html>
<body>
```

```
<a href="default.php">Home</a> |
<a href="about.php">About Us</a> |
<a href="contact.php">Contact Us</a>
<h1>Welcome to my home page</h1>
<p>Some text</p>
</body>
</html>
```

Of course, we would have to do the same thing for "about.php" and "contact.php". By using include files, you simply have to update the text in the "menu.php" file if you decide to rename or change the order of the links or add another web page to the site.

The require()

The **require() function** takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the require() function generates a fatal error and halt the execution of the script.

Remember, when using require that it is a statement, not a function. It's not necessary to write:

```
<?php
	require('somefile.php');
?>
```

Typically the require() statement operates like include() . The only difference is — the include() statement will only generate a PHP warning but allow script execution to continue if the file to be included can't be found, whereas the require() statement will generate a fatal error and stops the script execution.

Name : _____ Section : _____

Exercise 7

- Write a PHP program that may be used to compute the area of *square* (area = side * side) or a *triangle* (area = ½ * base * height) after prompting the user to type the first character of the figure name (S or T) if "S" compute the area of square or if "T" compute the area of triangle, use the ***function*** command for the computation of 2 areas.

CHAPTER 3

Manipulating Arrays

What is an array?

When working with PHP, sooner or later, you might want to create many similar variables.

Instead of having many similar variables, you can store the data as elements in an array.

Each element in the array has its own ID so that it can be easily accessed.

There are three different kind of arrays:

- **Numeric array** - An array with a numeric ID key
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

Numeric Arrays

A **numeric array** stores each element with a numeric ID key.

There are different ways to create a numeric array.

Example 1

In this example the ID key is automatically assigned:

```
$names = array("Arnold","Allen","Jet","Rolaida");
```

Example 2

In this example we assign the ID key manually:

```
$names[0] = "Arnold";
$names[1] = "Allen";
$names[2] = "Jet";
$names[3] = "Rolaida";
```

The ID keys can be used in a script:

```
<?php
$names[0] = "Arnold";
$names[1] = "Allen";
$names[2] = "Jet";
$names[3] = "Rolaida";
echo $names[1] . ", ".$names[2]. " and " . $names[3] .
" are ". $names[0] . "'s neighbors";
?>
```

The code above will output:

```
Allen, Jet and Rolaida are Arnold's neighbors
```

Associative Arrays

An associative array, each ID key, is associated with a value.

When storing data about specific named values, a numerical array is not always the best way to do it.

With associative arrays we can use the values as keys and assign values to them.

Example 1

In this example, we use an array to assign ages to the different persons:

```
$ages = array("Onack"=>32, "Joshua"=>30, "Sandy"=>34);
```

Example 2

This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Juan'] = 32;
$ages['Pedro'] = 30;
$ages['Sandy'] = 34;
```

The ID keys can be used in a script:

```
<?php
$ages['Juan'] = 32;
$ages['Pedro'] = 30;
$ages['Sandy'] = 34;
echo "Juan is " . $ages['Juan'] . " years old.";
?>
```

The code above will output:

```
Juan is 32 years old.
```

Sorting Arrays

PHP - Sort Functions for Arrays

In this chapter, we will go through the following PHP array sort functions:

sort() - sort arrays in ascending order
rsort() - sort arrays in descending order
asort() - sort associative arrays in ascending order, according to the value
ksort() - sort associative arrays in ascending order, according to the key
arsort() - sort associative arrays in descending order, according to the value
krsort() - sort associative arrays in descending order, according to the key

Sort Array in Ascending Order - sort()

Example 3

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

$clength = count($cars);
for($x = 0; $x <  $clength; $x++) {
     echo $cars[$x];
     echo "<br>"; }
?>
```

Output:

```
BMW
Toyota
Volvo
```

Sort Array in Descending Order - rsort()

Example 4

```
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
$arrlength = count($numbers);
for($x = 0; $x <  $arrlength; $x++) {
     echo $numbers[$x];
     echo "<br>"; }
 ?>
```

Output:

```
22
11
6
4
2
```

Name : _____ Section : _____

Exercise 8

- Write a program array that will read 5 scores and output the highest score, lowest score, total, average and sort the 5 scores in ascending order.

CHAPTER 4

PHP Form Handling

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.

Form example:

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

The example HTML page above contains two input fields and a submit button. When the user fills in this form and click on the submit button, the form data is sent to the "welcome.php" file.

The "welcome.php" file looks like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

A sample output of the above script may be:

```
Welcome John.
You are 28 years old.
```

The $_GET Variable

The **$_GET variable** is an array of variable names and values sent by the HTTP GET method, it is used to collect values from 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 it has limits on the amount of information to send (max. 100 characters).

Example

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

When the user clicks the "Submit" button, the URL sent could look something like this:

The "welcome.php" file can now use the $_GET variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $_GET array):

```
Welcome <?php echo $_GET["name"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

Why use $_GET?

Note: When using the $_GET variable all variable names and values are displayed in the URL. This method should not be used when sending passwords or other sensitive information! However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

Note: The HTTP GET method is not suitable on large variable values; the value cannot exceed 100 characters.

The $_REQUEST Variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE.

The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

Example

```
Welcome <?php echo $_REQUEST["name"]; ?>.<br />
You are <?php echo $_REQUEST["age"]; ?> years old!
```

The $_POST Variable

The **$_POST variable** is used to collect values from a form 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.

Example

```
<form action="welcome.php" method="post">
Enter your name: <input type="text" name="name" />
Enter your age: <input type="text" name="age" />
<input type="submit" />
</form>
```

When the user clicks the "Submit" button, the URL will not contain any form data, and will look something like this:

```
http://www.sites.com/welcome.php
```

The "welcome.php" file can now use the $_POST variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $_POST array):

```
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old!
```

Why use $_POST?

- Variables sent with HTTP POST are not shown in the URL
- Variables have no length limit

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

The $_REQUEST Variable

The **PHP $_REQUEST variable** contains the contents of both $_GET, $_POST, and $_COOKIE.

Example

```
Welcome <?php echo $_REQUEST["name"]; ?>.<br />
You are <?php echo $_REQUEST["age"]; ?> years old!
```

Example

Filename : **Post.php**

```
<html>
<head>
<title>Personal INFO</title>
</head>

<body>

<form action="output.php" method="POST">

First Name:<input type="text" size="12" maxlength="12" name="Fname"><br/>
Last Name:<input type="text" size="20" maxlength="36"
name="Lname"><br/><br/>

Gender :<br />
Male:<input type="radio" value="Male" name="gender"><br/>
Female:<input type="radio" value="Female" name="gender"><br/><br/>

Choose your favorite subject:<br />
Computer:<input type="checkbox" value="Computer" name="subject[]"><br/>
Math:<input type="checkbox" value="Math" name="subject[]"><br/>
English:<input type="checkbox" value="English" name="subject[]"><br/><br/>

Level of Education :
<select name="education">
  <option value="Elementary">Elemetary</option>
  <option value="HighSchool">HighSchool</option>
  <option value="College">College</option>
</select><br/><br/>

Favorite Quote: <br/>
<textarea rows="3" cols="30" name="quote"> </textarea><br/><br/>

<input type="submit" value="Submit na Po" >

</form>

</body>
</html>
```

Filename : **Output.php**

```
<?php

      $Fname = $_POST['Fname'];
      $Lname = $_POST['Lname'];
      $sex = $_POST['gender'];
      $subject = $_POST['subject'];
      $educ = $_POST['education'];
      $quote = $_POST['quote'];

      echo "Name : ".$Fname." ".$Lname."<br/>";
      echo "Status : ".$sex."<br/>";

      echo "Your favorite Subject(s) : "."<br/>";
      foreach ($subject as $subj)
          echo $subj."<br/>";

      echo "Education : ".$educ."<br/>" ;
      echo "Quotes: "."<i>".$quote."</i><br />";

?>
```

Name : _____ Section : _____

Exercise 9

An hourly employee is paid at a rate of **P** 30.00 per hour. From the workers Grosspay, 6% is withheld for Social Security Tax, 4% is withheld for Federal Income Tax and **P** 15.00 is withheld for Union Dues.

Civil Status	State Income Tax
Single	5% of Grosspay
Married	6% of Grosspay
Head of the Family	7% of Grosspay

Make a program using POST that will input the employee's number, employee's name, number of hours worked, civil status, and overtimepay and output the employee's number, name, civil status, each withholding amount, basicpay, grosspay, and net take home pay.

Computation:

Grosspay = BasicPay + Overtime Pay

Basicpay = Rate per Hour * No. of hours worked

NETPAY = Grosspay – Deduction