

Introductory Guide to



A beginners guide to learning PHP

Written by
Roger D. Fedor, II

Table of Contents

Introduction	2
Displaying Data	3
Variables	5
Using Conditional Statements and Loops	7
Arrays	10
Creating Functions	13
Gathering Input from the User	14
String Manipulation	16
Dealing with MySQL Databases	19
Need more help?	22
Index	23

Introduction

PHP is short for “PHP: Hypertext Preprocessor” and was developed and released in 1994 by Rasmus Lerdof as a means of tracking the people who view his resumes. Over the past years since its release, PHP has become one of the most widely used internet programming languages available. At the start of March 2000, over 2 million websites used PHP and as of March 2005, there are over 20 million websites. It also happens to be one of the easiest to learn as well. PHP offers simplicity along with the advanced features used in Perl and some other languages. Some people might think that programming is too complicated for them; however it’s quite the opposite if people think of it in terms of real life. People have a situation where they can bring some common thinking in the picture. In this user’s manual I will show how it relates to human thinking and how easy it is to create a wide variety of scripts.

PHP is a server side programming language so when people write code for it, they will have to upload it to their server to see how it looks. There are a few websites such as <http://www.tripod.lycos.co.uk> that offer free PHP web hosting along with a MySQL database. This is a great starting point for users who want to learn PHP.

There are two ways of doing PHP scripts: one is by coding the PHP around the html code for the websites; the other is straight PHP where the HTML is display using an echo command. To tell the server when the PHP code starts and ends, put “<?PHP” before the code and “?” after the code. Every command used in PHP ends with a semi colon except the commands that require brackets which will be taught later in this manual. Comments can be added to code as well which will be used to help further the understanding of what the code examples do. These comments can be declared a few different ways: first by putting either a “#” or a “//” in from; or by starting a comment with “/*” and ending with “/*”.

Displaying Data

There are a few ways users can output data to web pages. The most common command used though is **echo**. Think of **echo** as what you would want to say in real life, it's what you would like to communicate to another person. We do this sort of thing in every day life like when speaking to friends and family.

There are two ways of using this function that will be displayed here in this example piece of code.

Code

```
<?PHP

# One way to use the echo command is as follows
echo ( "This is the <b>line</b> you echoed.<br>" );

# The most common way of using the echo command is like this
echo "<b>This</b> is the line you echoed.";

?>
```

Output

```
This is the line you echoed.
This is the line you echoed.
```

When using **echo** to display data to a user, you can use either the double quotes (“) or a single quote (‘). A single quote is used when the programmer does not want the server to do any changes to the string; you want it kept as is but when using the double quote if variables are present in the code, the server will replace that variable with what is stored. Variables will be discussed in the next section.

To join two or more objects together you would use a period (.) in between the quotes or double quotes signify that you are joining to items together. An example of this would be as follows.

Code

```
<?PHP

echo "String 1<br>"
    ."String 2<br>";
echo "String 1 "."String 2";

?>
```

Output

```
String 1  
String 2  
String 1 String 2
```

The second way of communicating with a user is by using print. Print is similar to echo but the main difference between them is that print returns a value and is used in more complex equations where echo does not. I will be using the echo command throughout this instruction manual.

Variables

Variables are used to store data in and do calculations and manipulations to them. They are like a person's short term memory when they want to remember something.

Variables are remembered by the script for as long as the script is running, so once the script stops running the variables will no longer exist. In PHP they are denoted by the dollar sign (\$) at the beginning of the variable name to tell the server that it's a variable. Let's say that someone just told you something and now you want to pass this information to another person. Here is an example of variables used in conjunction with echo.

Code

```
<?PHP

$name = "Billy Johnson";
echo "Hello, my name is $name.";

?>
```

Output

```
Hello, my name is Billy Johnson.
```

The server automatically determines what type of variable you are trying to use. If you try to store a letter to a variable, the server will think of it as a string and if you store a number to a variable it will treat it as an integer. To store numbers to variables you would use it like the following example.

Code

```
<?PHP

$number = 5;
echo 'The variable stored in $number is '.$number;

?>
```

Output

```
The variable stored in $number is 5
```

With the ability to store numbers to variables, now you can do some simple mathematical calculations like addition, subtraction, multiplication and division. The following example of code will demonstrate how they are used.

Code

```
<?PHP

    $var1 = 1;
    $var2 = $var1 + 1;
    $var3 = $var2 * 3;
    $var4 = $var3 / 2;
    $var5 = (($var1 + 1) * 3) / 2;
    echo "$var1<br>$var2<br>$var3<br>$var4<br>$var5";

?>
```

Output

```
1
2
6
3
3
```

If you just want to increment a variable by 1 or decrease a variable by one, you can use the ++ and -- operators after the variable.

Code

```
<?PHP

    $number = 1;
    echo "$number<br>";
    $number++;
    echo "$number<br>";
    $number--;
    echo "$number<br>";

?>
```

Output

```
1
2
1
```

Using Conditional Statements and Loops

Conditional statements are one of the most widely used decision making functions in programming and in life as well. We make decisions on what time we eat, when we do homework, whether or not we want to go to work that day, etc. In PHP it's no different, but we are normally using variables in the decision. One of the most basic conditional statements used in all scripts and programming languages is the **if else** statement. When using the if else statements, you are comparing two variables together and if it matches it's true, else it's false. Here is an example of the **if else** statement in action.

Code

```
<?PHP

    $var1 = 1;
    If ($var1 >= 1) {
        echo "We are Borg, ";
    } else {
        echo "Hello ";
    }
    $var--;
    If ($var >= 1) {
        echo "my name is Wilber.";
    } else {
        echo "you will be assimilated.";
    }

?>
```

Output

```
We are Borg, you will be assimilated.
```

If you don't want to do another set of instructions when your if statement is false, you can just leave off the else statement

Code

```
<?PHP

    $var1 = 1;
    If ($var1 == 1) {
        Echo "We are Borg, ";
    }
    echo "you will be assimilated.";

?>
```


Output

```
We are Borg, you will be assimilated.
```

There are two kinds of loops people use in programming and scripting which are **while** and **for** loops. **While** loops are used when you don't really know when it's going to end, you just want it to run till the expression you give it is false. **While** loops though are a bit messy when the expression never reaches an ending and if that happens then the script will run without stopping. The following loop will continue running while \$var1 is less than or equal to 5.

Code

```
<?PHP

    $var1 = 0;
    While ($var1 <= 5) {
        echo "$var1<br>";
        $var1++;
    }

?>
```

Output

```
0
1
2
3
4
5
```

For loops are a bit different in the sense that you know how many times you want to run through the code. The following example does the same exact thing that the previous example did, but with fewer lines of code.

Code

```
<?PHP

    for ($x = 0; $x <= 5; $x++) {
        echo "$x<br>";
    }

?>
```

Output

```
0  
1  
2  
3  
4  
5
```

Switches are another kind of conditional statement where it matches a variable to a value. If the value is found, it will run the code in that it says but if it's not there, you have a choice to run some standard code.

Code

```
<?PHP  
    $var = 1;  
    switch($var) {  
        default:  
            echo "Value not found.";  
            break;  
        case 1:  
            echo "Value is 1.";  
            break;  
        case 2:  
            echo "Value is 2.";  
            break;  
    }  
?>
```

Output

```
Value is 1.
```

In the example above, the variable var is set to 1. Now it goes through the switch saying ok, is the variable var equal to 1? Yes it is so let's run the code. If the variable var was set to 2, it would have asked ok, is the variable var equal to 1? No so let's move on, is it equal to 2? Yes so let's run the code in there. If the variable was set to anything besides 1 or 2, it would then run the code that was in default and if default is not there, it won't run any code at all.

Arrays

An array is a container for variables and they can be thought of like a tree. You have the base of the tree, which is the address of the variable then that base branches off to containers to store data like strings or numbers. You could branch those containers out as far as you want with data. The square brackets is used in arrays is the location to where you want to access the data. If an address is not put into the brackets, then it adds the variable or string on the end of the array. In the following example, it will show how you can create an array and how to get the information from that array.

Code

```
<?PHP

$array[] = 0;
$array[] = 1;

$array[ 10 ] = "Location 10";

$array[ "ten" ] = "Location ten";

$array[ "another" ][ 1 ] = "Multi-dimensional Array";

echo $array[ "ten" ]."$array[10]";

echo "<pre>";
print_r( $array );
echo "</pre>";

?>
```

Output

```
Location 10
Array
(
    [0] => 0
    [1] => 1
    [10] => Location 10
    [ten] => Location ten
    [another] => Array
        (
            [1] => Multi-dimensional Array
        )
)
```

As you can see in the previous example, there are different ways of putting data into an array. You can also put an array into an array creating a multi-dimensional array as you see above. The code in the above example that says “<pre>” is HTML for preformatted

text so it displays like a text file. Outside of that is not unless you use HTML code to format it. The **print_r** command is used to display arrays on pages. It comes preformatted so that is why there are the “<pre>” tags around it.

To make arrays easier to work with, there are a few functions, like the **print_r** function above, that help a user determine the size of the array and its contents. To tell the size of an array, you would use the **sizeof** function in PHP. When you know the size of an array, then you can use the **for** loop to go through the array manipulating the array as you want.

Code

```
<?PHP

$array[] = "Array location 0";
$array[] = "Array location 1";
$array[] = "Array location 2";

echo "Size of array: ".sizeof($array)."<br>";
For ($x = 0; $x < sizeof($array); $x++) {
    echo $array[ $x ]."<br>";
}

?>
```

Output

```
Size of array: 3
Array location 0
Array location 1
Array location 2
```

The array above uses 3 spaces, but it is started from 0, not 1 so whenever you use conditional loops, you do not want to include the last location, 3.

Let us relate this to a business. You are the accountant of a small business of about 3 employees and they all have the different hourly wage. You need to store their name, their social security number, their hourly wage and the total hours worked. How would you go around displaying this information along with how much they have earned so far? It can be easily accomplished by using a multi-dimensional array.

Code

```
<?PHP

$employee[ 0 ][ 'name' ] = "Billy Montgomery";
$employee[ 0 ][ 'ssn' ] = "1234567890";
$employee[ 0 ][ 'wage' ] = 7.98;
$employee[ 0 ][ 'hours' ] = 5;
```

```

$employee[ 1 ][ 'name' ] = "Martha Stewart";
$employee[ 1 ][ 'ssn' ] = "0987654321";
$employee[ 1 ][ 'wage' ] = 10.00;
$employee[ 1 ][ 'hours' ] = 10;

$employee[ 2 ][ 'name' ] = "Bob Richenson";
$employee[ 2 ][ 'ssn' ] = "3216540987";
$employee[ 2 ][ 'wage' ] = 12;
$employee[ 2 ][ 'hours' ] = 15;

for ($x = 0; $x < sizeof( $employee ); $x++ ) {
    $salary = $employee[ $x ][ 'hours' ] * $employee[ $x ][ 'wage' ];
    echo "Employee Number: $x"
        ."Name: ".$employee[ $x ][ 'name' ]."<br>"
        ."SSN: ".$employee[ $x ][ 'ssn' ]."<br>"
        ."Wage: ".$employee[ $x ][ 'wage' ]."<br>"
        ."Worked: ".$employee[ $x ][ 'hours' ]."<br>"
        ."Total: $$salary<br><br>";
}

?>

```

Output

```

Employee Number: 0
Name: Billy Montgomery
SSN: 1234567890
Wage: 7.98
Worked: 5
Total: $39.9

Employee Number: 1
Name: Martha Stewart
SSN: 0987654321
Wage: 10
Worked: 10
Total: $100

Employee Number: 2
Name: Bob Richenson
SSN: 3216540987
Wage: 12
Worked: 15
Total: $180

```

Creating Functions

One of the nice things about PHP is that you can create your own functions which can cut down on the amount of code people write. With the ability of creating functions, you can create libraries to be used in code later on.

Let us create a simple function where we can add two arrays of the same size together and then use the **return** to send an array back with the sum of both arrays in it.

Code

```
<?PHP

function sum( $array1, $array2 ) {
    for ( $x = 0; $x < sizeof( $array1 ); $x++ ) {
        $arr_return[ $x ] = $array1[ $x ] + $array2[ $x ];
    }
    return $arr_return;
}

$arr1[ 0 ] = 2;
$arr1[ 1 ] = 5;
$arr1[ 2 ] = 3;
$arr1[ 3 ] = 1;

$arr2[ 0 ] = 3;
$arr2[ 1 ] = 2;
$arr2[ 2 ] = 2;
$arr2[ 3 ] = 4;

$arr3 = sum( $arr1, $arr2 );

echo "<pre>";
print_r( $arr3 );
echo "</pre>";

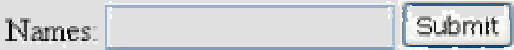
?>
```

Output

```
Array
(
    [0] => 5
    [1] => 7
    [2] => 5
    [3] => 5
)
```


Gathering Input from the User

One of the most basic functions of PHP is its ability to interact with a user and get input from them. Prior to this section, we were talking to the user like it was a one way street, from the server to its user. Now we can turn this one way street into a two way street by asking a user for input and then interpreting that input as we see fit. Let us create a page that will ask for a users name, and then reply by saying hello when they hit a submit button. In this example we will be using a form and we will be posting the data the user enters to itself. All data using post will be put into a variable called **\$_POST** and the data that is in the URL is in the variable called **\$_GET**. **\$_POST** and **\$_GET** are both arrays and they can be treated as arrays. The keys in the array are the names of the fields.

Code	Web Browser
<pre> <?PHP if (\$_POST['name']) echo "Hello " . \$_POST['name']; ?> <form name="form1" method="post" action=""> Names: <input type="text" name="name"> <input type="submit" name="Submit" value="Submit"> </form> </pre>	

Let us say that in the name field we enter Billy and we hit submit, and then the browser would look like the following.

Output



Let us say we have a file on a server called print.php and we want to see all the data in the **\$_GET** array when we enter the URL <http://www.example.com/print.php?name=Billy&type=test>. In that file print.php we have the following code and let's see what happens when the page comes up. When doing variables like this, make sure to put a question mark after the filename to say that you are declaring variables.

Code

```
<?PHP
    echo "<pre>";
    print_r( $_GET );
    echo "</pre>";
?>
```

Output

```
Array
(
    [name] => Billy
    [type] => test
)
```


String Manipulation

As users input data, there maybe a need to extract specific pieces of data from the users input. One way to extract data from a string is by using the **substr** command in PHP. Whenever the **substr** is used, certain pieces of information need to be given in order for it to return a result. Some of the information needed is the string you want to examine, the position of where you want to start reading, and the length of text you want to read. The length of text is not required so if you do not enter it, it will just read from the given starting point you give it, to the end of the string.

Code

```
<?PHP

$string = "abcdef";
echo substr($string ,1,2)."<br>";
echo substr($string ,3)."<br>";
echo substr($string ,0,2)."<br>";

?>
```

Output

```
bc
cdef
ab
```

Let us say that you are trying to make a script to validate a phone number. You know that the format of the number needs to be xxx-xxx-xxxx which is 12 characters long. The first way you can validate this string is by the checking the length. The command to find out the length of a string is **strlen** and the only variable it needs is the string to check the size for. The script would basically be asking itself, "Does this string have the right amount of characters to be a phone number?" and if it is, then you want to display, "This is a valid phone number" else display "This is not a valid phone number."

Code

```
<?PHP

$number = "505-237-2066";
$length = strlen( $number );
echo $length."<br>";
if (strlen( $number ) == 12) {
    echo "This is a valid number";
} else {
```

```

        echo "This is not a valid number";
    }
?>

```

Output

```

12
This is a valid number

```

Now, if we were to change the \$number variable in the above example to "237-2066", then the output would be as follows.

Output

```

8
This is not a valid number

```

A very useful command is known as **explode**. **Explode** is used when you have a string and you have a separator within the string to separate each field. There are only two required variables to use the **explode** command and that's the separator and the string you want to separate. The return value of this command will return your individual strings, but in an array. Here is an example of a string separated by commas.

Code

```

<?PHP

$string = "abc,123,def,456";
$array = explode(",",$string);
echo "<pre>";
print_r( $array );
echo "</pre>";

?>

```

Output

```

Array
(
    [0] => abc
    [1] => 123

```

```
[2] => def
[3] => 456

)
```

Sometimes, rather than breaking strings apart, people may need to replace strings. Replacing strings can be used to change shortened phrases or special tags into its rightful tag. The command from replacing strings is **str_replace** and the required fields in it are search for, replace with, in string and it is used as follows.

Code

```
<?PHP

$string1 = "This is a red apple.";
$string2 = str_replace("red","green",$string1);
echo $string1."<br>".$string2;

?>
```

Output

```
This is a red apple.
This is a green apple.
```

Dealing with MySQL Databases

MySQL databases or tables that websites use that act as long term memory. It can be put into use many different ways for example you could make yourself an online address book, or you could keep inventory of all the movies you have and where they are at. This is just a short list of the things you can do and with a bit of imagination you can make much bigger things.

Normally when you find a host for your website, or with the free websites, they will provide you with a tool called PHPMyAdmin which is a script used to manage and maintain databases. It is widely used among and makes it very easy to create tables.

Let us say we have a table of much like the multi-dimensional array we had in one of the previous sections where we had the name, social security number, their hourly pay and the hours worked. With all the data in it, it would look something like this:

Table (employees)

id	name	ssn	wage	hours
0	Billy Montgomery	1234567890	7.98	5
1	Martha Stewart	0987654321	10.00	10
2	Bob Richenson	3216540987	12.00	15

Each column has its own attributes sort to speak. For example, id in this table would be known as the primary key, there can only be one of them and it is set for auto-increment which means that when you add a new entry into the table, it will automatically increase one point as you add. The name is known as a char field and it would have a length of only 30 characters. When creating tables, just remember that the amount of space your table takes up depends on the size of your variables. Let's say that you have a variable with a length of 30 spaces and one with 50, the variable with the length of 50 spaces will take up more space because it has to set 50 fields available. Now let's make a script to do the exact same thing we did with the multi-dimension array in one of the previous sections.

Code

```
<?PHP

mysql_connect("localhost","username","password");
mysql_select_db( "database" );
$result = mysql_query( "select * from employees" );

while ($row = mysql_fetch_array( $result )) {
    $salary = $row[ 'hours' ] * $row[ 'wage' ];
    echo "Employee Number: $x"
        ."Name: ".$row[ 'name' ]."<br>"
        ."SSN: ".$row[ 'ssn' ]."<br>"
        ."Wage: ".$row[ 'wage' ]."<br>"
        ."Worked: ".$row[ 'hours' ]."<br>"
}
```

```

        . "Total: $$salary<br><br>";
    }
?>

```

Output

```

Employee Number: 0
Name: Billy Montgomery
SSN: 1234567890
Wage: 7.98
Worked: 5
Total: $39.9

Employee Number: 1
Name: Martha Stewart
SSN: 0987654321
Wage: 10
Worked: 10
Total: $100

Employee Number: 2
Name: Bob Richenson
SSN: 3216540987
Wage: 12
Worked: 15
Total: $180

```

To explain the commands used a bit in detail, **mysql_connect** tells the web server to “Ok, I would like to connect to the follow address (localhost) and use this username (username) and this password (password) to log into it.” The following command, **mysql_select_db**, selects the database where you would like to retrieve the data from which in our case is called database. Databases can be named anything for as long as you refer your code to the proper location. After we have connected to our server and selected our database, now we can gather the information we need for our script to run using the **mysql_query** command. We are telling the server, “We need every (*) from the table called employees.” Now when we start needing certain rows of information, that is when we add the where clause onto our query. Let us use the **\$_GET** variable to select a particular employee rather than just displaying everyone.

Let’s say that we put all the code into the file employee.php under the domain example.com. In the following code we are using the id variable to look for the employee so the URL we will use is <http://www.example.com/employee.php?id=1>.

Code

```
<?PHP

mysql_connect("localhost","username","password");
mysql_select_db( "database" );
$result = mysql_query( "select * from employees where id=".$_GET['id'] );

while ($row = mysql_fetch_array( $result )) {
    $salary = $row[ 'hours' ] * $row[ 'wage' ];
    echo "Employee Number: $x"
        ."Name: ".$row[ 'name' ]."<br>"
        ."SSN: ".$row[ 'ssn' ]."<br>"
        ."Wage: ".$row[ 'wage' ]."<br>"
        ."Worked: ".$row[ 'hours' ]."<br>"
        ."Total: $$salary<br><br>";
}

?>
```

Output

```
Employee Number: 1
Name: Martha Stewart
SSN: 0987654321
Wage: 10
Worked: 10
Total: $100
```

Need more help?

Need more clarification on some commands or want to continue learning all the capabilities of PHP? Some good sources are the

- <http://www.php.net> – The official website for PHP is a great reference guide, has a search where you can look up the command you want to use and many examples by users on how it is used. It is an excellent resource and I use it constantly.
- <http://www.phpbuilder.net> – Has many articles and tutorials written by PHP Coders on how you write PHP. They have a forum as well where you can post any questions you have concerning the coding and you can get some good help there.
- <http://www.google.com> – It's a bit more difficult to use, because you don't know what kind of results you'll get on your search, but it can return coding examples or if you have problems using a command you can probably find an answer here.

Index

Arrays	10
echo	2
explode	17
for	8
if else	7
mysql_connect	20
mysql_fetch_array	20
mysql_query	20
mysql_select_db	20
print_r	10
sizeof	11
strlen	16
str_replace	18
substr	16
switch	9
variables	5
while	8
\$_GET	14
\$_POST	14