INTRODUCTION TO

A Beginner's Guide to PHP Programming Language

TABLE OF CONTENTS

INTRODUCTION

I think the best place to start is a definition of PHP for those of you who are completely new to it or don't quite get what it does. PHP is an open source server side scripting language that allows you to add functionality to the back end of any website (think login systems and connecting to databases). It's powerful, very widely used, and is a language you must know if you want to be a web developer. PHP is an acronym for PHP Hypertext Preprocessor and has had numerous updates since the days when the initial model of PHP (called Personal Home Page then) was developed.

It makes sense that the language is used worldwide because of it's application in web development, but why learn PHP? Why is it special? The specialty and the uniqueness of PHP lies in its simplicity and the how efficiently and easily this language has introduced new and remarkable features into web development. PHP in some ways is doing everything that HTML was never made to do. HTML is bounded by its own restrictions of merely outputting content on the webpage, PHP on the other hand bring a whole new level of interactivity to web development. Taking data from the user, connecting to a database, giving structure and new dimensions to webpage, the list of PHP functionalities goes on and on, but the most important one is that it makes our lives easier, and when I say we, I mean both the developer and the user.

How Do PHP and HTML Work Together?

Well, when we say PHP and HTML work together, that does not mean one intrudes into the other's territory, instead both HTML and PHP keep hold of their own functionality only to make the overall code more effective. Confused?

Let's have a look at this example:

```
<!DOCTYPE HTML>

<html>

<head>

<title> PHP and HTML </title>

<head>

<body>

Wait for it, PHP is on it's way..!!!

<?php

echo 'Hi, PHP is here';

?>

</body>

</html>
```

Wait, wait.!! What is this "<?php" all about? I have never seen anything like that before inside HTML. That must be an error. Well it isn't. There are a few important things to notice about the example above.

- The “<?php” and “?>“ are the opening and closing tags of a PHP script. All your PHP code must reside in between these two.
- While combining PHP with HTML, it must be noted that the PHP tag is closed before any HTML starts, just like in the example above.
- A PHP script is saved with a .php extension. If a file has both HTML and PHP in it, even then we need to save it with a .php extension as the embedded HTML in PHP takes care of the HTML part in the code.
- Even though PHP is regarded as a server side scripting language, its utility is not restricted to that as it brings a lot more to the plate, we will be discussing about it in the coming sections.

GETTING STARTED: CONNECTING TO A WEB SERVER

When I am usually asked about the things that one needs to start learning PHP, my answer is a textpad, willingness to explore and a desire to write code. I believe that these are the three things you need to keep hold of if you want to achieve success in your quest to be an expert in PHP. However; I have noticed that a lot of people face difficulties in setting up the web server for PHP and therefore updating my list(just for the time being), this section is completely dedicated to web servers, setting up a web server and the common issues faced during the set up.

What is a web server and why PHP uses it?

A web server is used to communicate with the client using communication protocols. In case of PHP, the client is the web browser and the protocol used is HTTP. A web server is therefore required to process the request sent by the client and then deliver on those requests. Since PHP is a server side scripting language, it needs a web server to process and deliver the content to the client.

How should I download a web server?

Downloading a web server is easier than you think. All you need to do is go to the website, download the executable file from there, install it on your system and then run it, as easy as that. Wait, but which website? Well, there are a few web servers available to run your PHP file. You can either download Apache, MySQL and PHP separately or you can download a complete package in the form of XAMPP, WAMP, EasyPHP or from any other similar source. Once you have downloaded the file, all you need to do is open the application and then start Apache on it by clicking on the start button.

- Once you have installed the web server and started the Apache services on your system, type in localhost in your browser and it should take you to the web server's page which will confirm that Apache services are turned on.
- To run your PHP file, you must place all your PHP files inside the htdocs folder (in case of XAMPP) or inside the www folder (In case of WAMP and EasyPHP). You will find these respective folders at the location where you installed your web server.
- Now once you have placed all your PHP files that you have to run in the correct folder, you can go back to localhost, browse your file from there and open it. That's it; you will see the result that your file delivers on the webpage.

- It is noteworthy that from PHP 5.4 onwards, PHP would be capable of running the web server on its own, so you might not need an external application at your behest. This can be achieved by running $ php –S localhost: 8000 in the folder from where you want to serve the pages from (just like www and htdocs in case of WAMP and XAMPP respectively.)

Common Issues faced while webs server installation:

- The conflict of port between the web servers and Skype is a common issue with a lot written about it. The best way to get over it if you face such an issue is:
    - Prevention is better than cure they say and in this case, it is actually the best option. You should download the web server on your system first so that is occupies the port that it desires and then go with Skype download. This way you can avoid the conflict.

CODING YOUR 1st PHP SCRIPT

Quite often I have heard students talking about the successful and well organized websites on the internet and dreaming of having one of their own. I wonder what stops people from pursuing their dreams, but then I quickly realize that it is always difficult to complete your first walk without falling down and it is the fear of falling down that stops people from taking the first step. So, since we are done with all the pre requisites for learning PHP, let us take that first step, the stepping stone in our quest to become experts in PHP, the step that is far easier than you think it is. So, without any more time, let us write our first PHP script, the first of many.

Now, I have already explained this in a previous section, but since we are starting afresh, I will just quickly walk you through the PHP starting and ending tags. Unlike languages like C++ or C that has headers or HTML that has the HTML tag and a whole lot of other tags before we finally reach the <body> section, a PHP script has just the <?php as the opening tag and ?> as the closing tag. So, everything that we want to write in your PHP script must be confined within the boundaries of these two tags.

Unlike HTML, there is no specific body section in PHP, therefore, to output something on the webpage a user must provide specific commands. Echo and Print are two of the most commonly used PHP commands for outputting content on the webpage, with echo

holding superiority over the latter. We will talk in detail about these two in the upcoming sections.

```php
<?php

echo "My First PHP script";

?>
```

It barely took a line of code to output content on the webpage using PHP; while on the other hand, HTML would have taken more than 6-7 lines to carry out the same operation. However; it is important to note that PHP is, by no means, an immediate upgrade to HTML. It is beyond doubt that PHP does operations that HTML was never build to perform, but HTML tags such as <form>, and others form an integral part of a good web development process and therefore, it is the integration of HTML and PHP that really does the trick. Even though in real world scenario, it is impossible to have a complete website with just echo statements in it, however before one enters the sea, it is important to teach him/her to swim and this, along with the upcoming sections are dedicated towards giving the readers a feel of the language, its functions and making them realize how PHP's ease of use comes in handy for the users.

VARIABLE

In simple words a variable is anything that can store data or values. In real world; your cupboard, a jar, a piggy bank, are all different types of variables. However; in the world of computers and programming languages, variables must be defined by the user and they can then be used to store values of a particular type.
So, how my computer differentiates between a variable and a normal text? There must be something to tell the computer that this particular entity is a variable. To overcome this hiccup, every variable in PHP starts with a '$' symbol followed by the name of the variable.

Eg:

```
1  $name
2  $whole_sale
3  $WORLDWAR
4  $_rio23
```

All these above examples are valid PHP variables.

Note:

A PHP variable is case sensitive and the name of the variable should not have any space between two strings, although we can use an underscore'_' symbol.
Further, the name of a variable cannot start with anything other than an alphabet or an underscore '_'.
A starting alphabet can be followed by any of the alpha numeric characters and underscore symbol.

Every programming language has its own style of variable declaration, however; the use of variables remains the same. In PHP, declaring a variable is arguably the easiest amongst all the programming languages because of the dynamically typed nature of PHP variables.

What exactly is this dynamically typed nature of PHP variables?

Have you noticed that we have not talked about the data type of a PHP variable anywhere in our variable declaration part discussed above? Well, it's because PHP is dynamically typed language which automatically does the job of correcting the variable to the data type of the value stored. So as a programmer you do not have to explicitly mention the data type of the variables.

Eg:

```
1  <?php
2      $name= 'MY NAME IS JOHN';
3      $x=8;
4      $y=14.6;
5  ?>
```

You can see in the above example that PHP stores a string, an integer and a float value without any data type attached to any of the three variables. Assigning values to a variable for PHP is the same as it was for C or C++. All you need to do is use the assignment operator '=' with the value to be stored in the right side of the operator and the variable on the left side. It is important to note that string data is stored inside single quotes for a PHP variable.

How to use variables in a PHP script?

Now that we have understood the basics of declaring and assigning values to a PHP variable, the next junction is using a PHP variable in the code.

Eg:

```php
<?php
    $name = 'Alex';
    $age= 21;
    echo "My name is $name and I am $age years old.";
?>
```

The above example will display, My name is Alex and I am 21 years old; on the webpage. It is quite clear that we have used double quotations in the echo statement and have directly included the names of the variables in the statement.

Eg:

```php
<?php
    $name = 'Alex';
    $age= 21;
    echo 'My name is '.$name.' and I am '.$age.' years old.';
?>
```

This PHP script will also display, My name is Alex and I am 21 years old; on the webpage, however; we have used single quotations instead of double quotes in this example. It should be noted that while using the single quotes, the variable name should be concatenated with the statement using the dot (.) symbol and the name of the variable should not be included inside the single quotes.

Even though the use of single quotations for including variables in echo statements looks a bit complicated from the outside, but actually it is the better way of doing the job when you are creating a complete website. It is because, while designing a website, we might have to include HTML and Javascript in our PHP file and using the double quotations in the echo statements along with the syntax of the tags of HTML code creates a confusion regarding the start and the end of a statement. Therefore, keeping a larger picture in mind, it is better to get used to the concatenation of variables inside the echo statement.

Apart from simply echoing the values of a variable on the webpage, we can also perform different operations on the PHP variables, just like in other languages such as C or C++. By using operators like (+, -, /, *) we can add, subtract, divide or multiply the values in a variable. It is quite obvious that these operations used directly, are not valid on the string values in a PHP variable.
Thus, PHP variables and its proper use is the basic building block towards the creation of a good website and good knowledge of PHP variables also helps in understanding the more fundamental aspects of PHP programming with ease.

IF STATEMENT

While working with PHP and displaying content on the webpage, we will come across situations where a different operation must be performed for different cases. For instance, certain social networking websites do not allow users below the age of 13.
Therefore, after the user has entered his/her age in the profile, the webpage must show a message saying " You are allowed to use this website"; if the age is above 13, similarly, the website must show the message "You are not old enough to use this website", if the age is below 13. Achieving success with such a problem without the flow control statements can be slow, time consuming and frustrating, both for the user and the programmer. In this section, we are going to discuss about the 'if' statements and the 'if else' statements.

IF Statement

If statement is used when there is a condition to be evaluated and a result to be displayed given that condition holds true.

Eg:

```php
<?php
    $age=18; //variable definition
    if ($age<20) // condition for the if statement
    {
        echo 'You are not an adult yet'; // If condition holds true
    }
?>
```

The given code initializes the value of the variable $age then gives the condition of a string getting printed if the age is less than 20.

IF-ELSE Statement

Now, there is a very high probability that while designing a webpage or any other PHP program for that matter, we might have to consider the case when a condition applied does not hold true. For instance; when a user is required to enter his age on a webpage, the age entered in this case is bound to be a whole number, but what happens if the user enters an alphabet instead of a number. You can't just let the webpage crash or not show an error message, as it would make your page look unorganized and unplanned. So what should be done then? Well since IF statement only covers the very basic aspect of a program flow, PHP has the IF-ELSE statement that allows the developer to embed a section for the case when the condition provided in the IF clause does not hold true.

Eg:

```php
<?php
    $age = 30; //variable definition
    if ($age<18) //If condition
    {
        echo 'You are not an adult yet'; // If the condition in the 'IF clause' holds true
    }
    else // else clause defined
    {
        echo 'Congrats! You are an adult now'; // if condition is false then this statement gets executed

    }
?>
```

The example specified above, initialises the variable $age to a value and then gives the condition of a string getting printed if the age is less than 18 and a different string being printed if the value of the variable is more than 18.

IF-ELSE IF Statement

The IF and IF-Else statements covers majority of the problems which have either one or two possible outcomes; however what happens if the number of possible outcomes are more than two. To ensure that our webpage does not show any unexpected error even if the numbers of outcomes are more than two, we must use the IF-ELSE IF statement in our code. The IF-ELSE IF statement works the same way as the IF-ELSE statement with just more than two conditions to deal with. Let us have a look at an example for better understanding.

Eg:

```php
<?php
    $day = 'Monday'; //variable declaration
    if ($day == 'Monday') //If condition
    {
        echo 'Today is Monday.';
    }
    elseif ($day == 'Tuesday') // ELSE IF condition used for a different outcome.
    {
        echo 'Today is Tuesday.';
    }
    elseif ($day == 'Wednesday') // ELSE IF condition used for a different outcome.
    {
        echo 'Today is Wednesday.';
    }
    elseif ($day == 'Thursday')
    {
        echo 'Today is Thursday.';
    }
    elseif ($day == 'Friday')
    {
        echo 'Today is Friday.';
    }
    elseif ($day == 'Saturday')
    {
        echo 'Today is Saturday.';
    }
    else
    {
        echo 'Invalid Entry.';
    }
?>
```

In the above example, there are seven possibilities for the seven days of the week and therefore, if else if statement has been used to solve the problem and derive the desired result.

Note:

Brackets must be used after the completion of the 'if' clause and the completion of the else clause. It is important to note that the else keyword remains outside the brackets of the 'if' clause. Following the syntax is important to ensure that the program runs as expected.

While the condition of the 'if' statement is specified, it must be noted that the equality operator ('==') and not the assignment operator ('='); is used for making the comparison in the condition.

Thus, the need for a better and simplified approach to overcome the more realistic and probable hurdles that might occur during the designing of our website, is solved by using the if, if else and if else if flow control statements in PHP.

FOR LOOP

In order to understand PHP For Loops, we must quickly recollect our knowledge of programming languages and how the flow control statements are executed in the basic programming languages. PHP For Loops follow the same structure for the execution of For Loops as is in the case of the other programming languages like C or C++.

'For loops' are particularly useful when a programmer intends to execute an operation or a command for some specific number of times. The syntax for PHP for loops primarily consists of three expressions in the declarations phase.
The *first expression* initializes the counter to a specific value. In easier terms, this expression is the assignment of a value to a variable which is to be used in the next expressions.
The *second expression* is the condition expression which evaluates a certain specified condition on the first expression and performs the operation if it holds true.
The *third expression* deals with assignment and updating of loop counter at the end of each cycle. This expression can be incrementing or decrementing the counter values.

Eg:

```php
<?php
    for ($num=1;$num<=10;$num++){
        echo 'This<br>';
    }
?>
```

In the example of PHP For Loops, a for loop is initiated inside the PHP mainframe and the operation to be performed is specified. It is important to note that the operation to be performed by any For Loop must be included inside the brackets just after specifying the 'for' keyword.

```
{
    echo 'This<br>'; /*The operation of printing 'This' is included inside the brackets*/
}
```

Note:

In case the brackets are not used and there are more than one line in the operation phase; then the compiler will only execute the first line of command and will neglect the other lines as part of the For Loop.

Inside the 'for loop', three expressions are specified. The first expression assigns the value 1 to the variable $num. It is to be noted that in PHP, variables are declared by using the '$' symbol before the name of the variable. The second expression, which is '$num<=10'; specifies the condition for the execution of this PHP For Loop. According to it, the operation will get executed until the value of the $num variable becomes more than 10. The third and final expression shows the value of $num variable incremented by 1. Using '$num++' is the same as writing $num=$num +1.

This For Loop performs the action of printing the word 'This' 10 times after the execution of the code. It is important to note that the third expression in the for loop which is performing the duty of incrementing the $num variable in the example specified above, will be functional only after the PHP For Loop operation has been executed once. For instance, in the above example, after the $num

variable is specified to the value 1 and the condition '$num<=10' is checked and satisfied, the operation of printing the word 'This' will be executed and then the value of $num variable will be incremented by 1. The same process will continue until the value of $num variable reaches the value 11 and the condition is not satisfied for the operation to be executed.

Instead of incrementing the $num variable, a programmer can also use the decrement function. However; it goes without saying that if the decrement function is used, the first two expressions of the 'for loop' have to be adjusted accordingly. For instance, in the above example, the same result can be obtained by adjusting the expressions in this way:

```
<?php
    for ($num=10;$num>;=1;$num--){
        echo 'This<br>';
    }
?>
```

PHP For Loops are particularly useful in making the code smaller, more organised and easier to understand. Further, For Loops, just like other flow control statements are useful in obtaining desired results for real time applications of any programming code.

WHILE LOOP

Flow control statements are basically loops aimed at performing a particular set of operation for a period. Writing condition for let's say 100 integers knowing that the operation to be performed on each one of them is the same is both ineffective and time consuming.

Therefore we need a flow control statement that can customise our code and group all the elements together, and then write the operation to be performed for the group instead of doing it for individual elements. To deal with the repetitive nature of any code or program, we can use either the 'for loop' or the while loop. However; if the operation to be performed in each case is the same, while loop is a better alternative than the 'for loop'.

There are quite a few interesting points to note about the while loop. First and foremost, a while loop provides a condition and as long as that condition holds true, performs a specified operation. That sounds very familiar to IF statement or the FOR statement, isn't it? The while loop is essentially used to optimize a given code and to ensure the perfect use of flow control statements in it, therefore it has a few similarities to both, the IF statement and the FOR statement.

Eg:

```php
<?php
    $x=0; //Variable declaration
    while($x<=10) //While loop initialised and condition provided
    {
        echo 'The number '.$x.' is smaller than our required value<br>';
        /* An echo statement is printed every time the condition holds true. */
        $x++; // The value inside the variable $x is incremented
    }
?>
```

As it is quite clear in the example given above, the while loop performs the same operation upon a variable until a given condition holds true. The while keyword is used to initiate a while loop and the condition to be checked is included inside closed parenthesis, followed by the operation to be performed, which is included inside closed brackets.

Note:

The condition included inside the while statement can be more than one with each separated by operators for comparison, such as '&&' or ||.

The statement $x++ is the same as $x= $x+1 and must be included inside the while loop to ensure that the operation does not turn into an infinite loop.

We can also decrement the value of the variable in the loop according to the requirement of the problem and the values desired on the webpage.

There is another way of performing the while operation on any variable. Instead of including the operation to be performed inside the curly brackets, we can also use the endwhile keyword to specify the end of any while loop.

The syntax of a while loop involving endwhile is slightly different from the conventional while loop, as explained in the example below.

Eg:

```php
<?php
    $num = 10; // Variable is initialised
    while($num>=0) : // Note that the while statement is followed by a colon (:) sign
        echo $num.'<br>'; //The operation is written without the curly brackets
        $num-;
    endwhile;
    /*endwhile keyword marks the end of the while loop and eliminates the use of brackets */
?>
```

It is important to note that the use of endwhile keyword can only give desired results if the colon sign is used after the while statement. Programmers around the globe are more comfortable with the curly brackets as it stands for all the other flow control statements as well.

Just like the while loop, there is another flow control statement, namely the do while loop that follows the same rules with a slight modification in the syntax. As we just saw, the WHILE loop provides the condition first and then the operation to be performed; DO..WHILE on the other hand employs an exactly opposite approach. In a DO...WHILE loop, the operation to be performed is given first followed by the condition at the end of every loop.

Along with its utility in simple PHP codes, 'While loop' is used extensively in cases where a particular data is to be fetched from a database. Names of people above a certain age or other related data can be fetched with ease and accuracy from a big database using the while loop.

FUNCTION

A function is a block of code that can be used or called from anywhere in the PHP script once it has been declared in it. The major use of function is to make the script smaller, easier to understand and more compact. Imagine that we are given the task of adding two numbers using PHP.

Now as programmers, our objective is to write a code that can be used for any two numbers on the number line. Without using functions, we will have to write down the same code again and again for the different set of numbers; an impossible task to achieve. However, what if we could write the code just once and then use this single code again and again for different numbers? Clearly the second option sounds more optimal and can be achieved using functions in PHP.

There are two important terms associated with functions in any programming language, the first one being 'declaring a function' and the second one is 'calling a function'. Basically, declaring a function is nothing but creating the content inside the function, while calling a function is giving the green light for the function to run. Now to declare a function in PHP, the keyword 'function' is required before the name of the function which is to be created. The content or the operation that the function will perform is included inside the braces, { and }. After creating the function, we must give the green light for the function to be used in the code i.e call the function. To call a function we must simply write the name of the function followed by parenthesis and a semicolon.

Note:

A function can be called inside its declaration part, however; this practice must be avoided by beginners as it involves recursion. Further, even if the function is called in itself, it must be called outside once to instruct the compiler to look for the declaration part.

Eg:

```php
<?php
    function name() // This part is the declaration of the function
    {
        echo 'Alex';
    }
    name(); // This is the function call.
?>
```

Functions in PHP can be executed with parameters or arguments passed to it. These parameters act like variables inside the declaration part of the function. We can pass as many arguments as we want to the function, however the function must know about the number of arguments passed. If for a function, we pass more arguments and specify less variables, or vice versa, then the PHP script will show an error message complaining about the missing value or variable accordingly.

Eg:

```php
<?php
    function add($num1, $num2) // Takes two arguments from the function call
    {
        $sum = $num1 + $num2; // Adds the two arguments
        return $sum;  // Returns the value received after adding the numbers
    }
    echo add( 10, 30);
?>
```

It is important to note that in the above example, $num1 and $num2 are used as variables for the function add and the values 10 and 30 are fed to these two variables during the function call. The variable

$num1 thus contains the value 10 and the variable $num2 contains the value 30.

Note:

The return keyword is used to return the value inside the variable $sum from the function add. This value is then printed on the page as echo keyword is used before the function call.

The loosely type nature of PHP variable automatically adjusts the data type of the variable to the data type of the value passed to it. Therefore, you can even pass a string or a Boolean value to the function as an argument in the above example.
In case, we pass more than two values to the function add, then during the declaration part, we must subsequently assign more variables to store these values.
Using functions is a common and highly recommended practice in PHP scripts. Apart from making the code smaller and giving an easy flow to it, functions also allows the programmer to write the code for complex problems with lesser difficulty. Functions are of high utility in writing PHP codes for problems that are aimed at real life scenarios and hence, it is an attractive option for a PHP script.

SWITCH STATEMENT

Taking our discussion on the flow control statements to the next level, in this chapter on PHP Programming, we are going to discuss how switch statements are executed in PHP and how useful are they in the websites that we see on the internet. A lot of students usually have this doubt that, 'When I already have IF, ELSEIF, WHILE, so why do I need switch statement in my codes?' The reason why PHP or any other programming language has multiple looping structures for performing the same operation is because performance and efficiency have become important aspects of web programming in the current time. Imagine you type the URL of a website on your browser and then scrolling through the different pages on that website needs a minute or two. The popularity of internet has seen unprecedented growth in the past 10 years, however; with this growth the expectations of users to get a fast and accurate response from their web service, has also grown at a similar pace. Therefore, instead of checking the same condition multiple times on a 'IF-ELSEIF' statement, it is better to have a switch statement in your PHP script and do the operation swiftly and cleanly.

Why do I need switch statement?

As we mentioned in the above paragraph, even though 'IF', 'ELSE-IF' are often sufficient to check any condition and execute the operation corresponding to it, but in search of an optimized solution, switch statement are required. Looking at the execution cycle of IF-ELSEIF statement in the previous chapter, you might have felt that even though the flow control statement is of great utility in case of multiple outcome problems but its use has made the problem look clumsy, and for bigger problems it might get difficult to find which ELSEIF scenario tracks to which condition. Well, you got such a thought in your mind then pat your back because the developers of PHP got a similar one and therefore they introduced the Switch statement for making a PHP script easier to write and to understand. Let us see the same example that we used for IF-ELSEIF statement and see how it changes the PHP code when we impose the switch statement to it.

Eg:

```php
<?php

$day = 'Monday';  //variable declaration

switch ($day)     //Switch statement initialisation
{
case "Monday":    // Case for each outcome possible
echo 'Today is Monday.';
break;
case "Tuesday":
echo 'Today is Tuesday.';
break;
case "Wednesday":
echo 'Today is Wednesday.';
break;
case "Thursday":
echo 'Today is Thursday.';
break;
case "Friday":
echo 'Today is Friday.';
break;
case "Saturday":
echo 'Today is Saturday.';
break;
default:  // Default condition for any outcome other than the one mentioned
echo 'Invalid Entry';
}
?>
```

Note:

1. Each condition in the script can be grouped as a different case using the switch statement and then if required, a different operation can be set to each of these cases.
2. The script looks cleaner, more organised and structured after switch statement is used.
3. The default condition in the switch statement is optional and can be skipped.
4. Using the keyword ‘break’ is a critical part of the switch statements and it allows the processor to understand that the current case has ended and a new case will start now.

ARRAYS

We have talked about PHP variables in one of our previous chapter and surely by this time you might have realized the importance of variables in PHP. In fact, being the storing blocks of values in a code, variables are essential and necessary in any programming language. We already know that a variable can be used to store data or values, but what if we need to store multiple values? Let say, I need something that can store the names of all the students in a class. Since the number of students will be around 30-40, assigning these many variables and then calling each one of them separately every time will be a tedious task to do, even on the computer. So what's the better approach? Before we take a dip in the world of array, it is important to understand that array and variables are not substitutes of each other, but different storage containers which are used depending upon the task in hand. In simple words, an array is a special kind of variable that can store multiple numbers of values.

Eg:

```php
<?php
 $fruits=array("Apple","Banana","Cherry");
 echo "I like " . $fruits[0] . ", " . $fruits[1] . " and " . $fruits[2] . ".";
?>
```

It is important to note that while the name of an array is declared in the same way as in case of variables, the values are assigned inside closed parenthesis, in a comma separated format with each value included inside quotes. The keyword 'array' is critical in declaring an array as it informs PHP that $fruits is an array and not a simple variable.

Now you must be thinking,"*Alright, the array declaration part is pretty clear, but what is this $fruits [0], $fruits [1] and $fruits [2]? I dint give numbers in the array, I gave strings.*" Well, you are right, you gave strings but each value in the array is stored at a particular location and the number 0, 1, 2 etc (which is called index of an array) gives the location of that value in the array variable. So, if I need to select the 27th guy in an array variable of 60 students, I do not have to get all the first 27 students, instead I can simply write student [26] and get the 27th student in the array. "Wait, wait, you said 27th student and you wrote student [26]? I am sure you made an error there." Well, actually no, array stores value starting from index value 0, followed by 1, 2, 3 and so on and therefore the 27th guy will have an index value 26.

Note:

1. The values inside an array must belong to the same data type. Therefore, you can have all integer values or all string values in an array, but you cannot have an array with values from both integer and strings.
2. An array is a continuous block and it assigns memory as continuous blocks as well. This means that if value $fruits [0] is assigned memory location 2200, then $fruits [1] will be assigned the memory exactly next to this one.

Basically, there are three types of array:

1. Indexed Arrays
2. Associative Arrays
3. Multidimensional Arrays

We just discussed about the indexed arrays in the first example where each value in the array is assigned an index value which is then later used to refer to that array element.

Associative Array

From our knowledge of arrays, it's easy to understand that any array needs some token (index in case of indexed arrays) to be assigned to them so that these tokens can later be used to select a particular value from the array. Obviously, associative array needs a token assigned to it as well, however what makes it different from indexed array is the way and the kind of tokens assigned.

An associative array named or string tokens for the assignment and there are two ways to create an associative array:

Eg:

1. $fruits=array("Apple"=>"35","Banana"=>"20","Cherry"=>"60");
   echo "The cost of Apple is .$fruits['Apple'].";
2. $fruits['Apple']="35";
   $fruits['Banana']="20";
   $fruits['Cherry']="60";

MULTI-DIMENSIONAL ARRAYS

Remember in the last chapter we talked about arrays and how arrays are initialized in PHP? This chapter continues on the same lane and describes 'Multi-dimensional array', a topic that we did not cover in the last part. Having discussed indexed array and associative arrays, it must be noted that multi-dimensional arrays are conceptually a little different, however; if you have understood arrays well, then you will find that multi-dimensional arrays are just a walk in the park.

What are multi-dimensional arrays?

As the name suggests, multi-dimensional array refers to an array that points to multiple dimensions. In simpler words, if you have arrays inside an array, then you are actually dealing with multi-dimensional arrays. Confused? Let us explain with the help of a real time scenario. Assume you run a company that has the records of every employee in every department inside an array named $company_data. The array '$company_data' will have department names, name of the employee and then their Employee Ids. How will you accommodate so much data inside an array? You can have the name of the employee in an indexed array. To accommodate their employee ids, you can use an associative array. However; still you won't be able to group them according to their departments. This is where the multi-dimensional arrays come into picture. In the scenario that I mentioned, if inside the array '$company_name', you have arrays of each department, and then inside the arrays of department, you can keep the employee data in associative array

format. This all might seem a little too heavy to digest, but once you get the concept behind arrays, you will automatically understand the process that goes on in multi-dimensional arrays.

Eg:

```php
<?php $company_date = array ( 'IT' => array ("Alex_IT" => "1", "Barry_IT" => "2", "Greg_IT" => "3"),
'HR' => array ("Soniya_HR" => "9", "Mohammed_HR" => "6"),
'Sales' => array ("Mike_Sales" => "4", "Nathan_Sales" => "7")
);
?>
```

Note:

1. PHP syntax does not rule out any number of dimensional arrays in it. So you can have four, five or even six dimensional arrays in your PHP script. However; it gets very clumsy and difficult to understand as the number on the dimension level goes higher. Therefore, in practice, you will only find two dimensional or in some cases three dimensional arrays in PHP script.
2. If a problem needs more than 3 dimensions in the array, it is better to go for database tables and store the values in it in specific columns and rows. Integrating PHP with databases is a part of this course and will be covered later.

DO WHILE LOOP

Taking a break from the concepts of array and topics related to that, in this chapter, we are going to have a look back at the loop statements. We have already studied about For Loops and While Loops and we hope that you have started using them in your PHP scripts to get more comfortable with them. The shortcut behind learning PHP is to stop looking for shortcuts and try investing as much time as you can in exploring on the text editor. The more you succeed, the more you will try to learn.

The flow control statement that we are going to discuss today is the 'Do.. While' loop. The do while loop is an extension to the WHILE loop with a small difference in the syntax part of both the statements. As we saw in the chapter on the WHILE loop, it provides the condition first and then the operation to be performed, however; DO..WHILE on the other hand employs an exactly opposite approach. In a DO...WHILE loop, the operation to be performed is given first followed by the condition at the end of every loop.

Eg:

```php
<?php
$x=0;
do // the do statement which will provide the operation to be performed
{
  echo "The number $x is smaller than our required value
";
$x++;
}
while($x<=10) //While loop condition provided at the end
 ?>
```

FOREACH

We have almost covered all the flow control statements and loop structures as part of this lesson on PHP Programming. In this last chapter on loops before we move on to functions parameters and formatting numbers, we will discuss about the 'foreach' statement. You must be wondering, 'we have already learnt about the 'for' loops, so why 'foreach' now? Can we just not solve our problems with the loops that we have already learnt?" The answer to this is simple and one that I have already mentioned in a previous chapter. Optimization is the need of the hour and you need to be fast to make sure that you do not lag behind. For optimization in your WebPages, you must use the right loops in your PHP script, and to use the right loop in your PHP script, you must have the knowledge of which loop is right at which point.

What is 'foreach' statement and how is it different from the 'for' loops?

I am sure you must be more excited and eager about the second part of this question, but we will still start with the first part as we believe, having a strong base is more important than building a high tower. As the name suggests, 'foreach' is used to iterate through each and every value in an array. Now, the important thing to note in that definition is the use of the word array as 'foreach' only works on arrays and not on any other element. Now, I am going to counter my own statement, in fact just the previous statement, as this statement that 'foreach' only works on array is

not entirely true, but we will keep it to that for the time being, until you are comfortable with PHP and array pointers and how the values are fetched in the 'foreach' statement.

Eg:

```
<?php $names = array ('Alex => 21, 'Billy' => 16, 'Dale' => 49);

foreach($names as $key => $value)

{

echo $key. 'is'.$value. '
;

}

?>
```

In the above example, the values inside the array named '$names' are iterated one at a time and are printed in the same sequence. You might notice the difference in the syntax of a 'for' loop to that of a 'foreach' loop. It is important to note that even the most experienced programmers sometimes do mistakes with the syntax of the 'for' loop and the 'foreach' loop.

Note:

1. The internal array pointer is automatically reset to the first element of the array when the 'foreach' loop starts for the first time.
2. Given the first statement, you do not have to call reset() in a 'foreach' statement.
3. Changing the internal array pointer within the loop may lead into unexpected results from the loop. Therefore, such a practice must be avoided.

FUNCTIONS WITH UNDEFINED PARAMETERS

Beginners in PHP or any other programming language, usually shy away from including functions in their codes as they believe that it makes coding difficult." If I am using the same operation, why can't I just copy paste the lines instead of including the operation in a function and then calling the function?" It is quite normal if that question comes to your mind as well, and it seems a better option if you are using a function just twice in a code that runs barely a 100 lines. However; there will be cases, in fact, in the real world applications, you will only be dealing with codes that run into thousands of lines and have the same operation carried out multiple number of times. Will it not be inefficient and quite painful to move back up in the code and then come back to paste the lines of code and then continue with your program. That is not all, every time you copy-paste a certain section of the code, the number of lines in the program go up as well. So not using functions means hard work for you and hard work for the processor as well. Now, even though we have started talking about 'Why we need to use functions in our codes', this chapter is not about the benefits of function, but about a certain area of functions that might be highly useful in PHP programming.

We have already discussed functions and passing parameters to the functions, however; there is a catch here. When using parameters, a user must mention the parameters as arguments to the function at the time of function call, now, what if we are unsure about the parameters? What if we need undefined parameters in our

functions? Confused? Let me explain in a more non-technical manner. Imagine you pass 8 parameters to your functions. Now, you know what you need in the first parameter and the 8th parameter, but at this point in time, you are not sure about the values in the parameters in between. The other parameters will be used some time during the program, but you are just not sure of them right now. What should I do in these cases? This is where the significance of this chapter comes in picture. You might have noticed that we have given a lot of text in this chapter without any example, something that we do not usually do. We did it this time because it is quite common to neglect this topic as people think they have already studied functions and this will anyways not be used a lot. However; as they say and we truly believe, you are not and expert if you have neglected things on your way up.

Now, we will directly take you to the implementation of the concept that we discussed in the above paragraph.

Eg:

```php
<?php

function add()

{

$total = 0;

foreach (func_get_args() as $arg)

{

total += (int)args;

}

return $total;

}

echo add (5,10,11,6,8,45,6,9);

?>
```

Note:

1. Notice the use of the functions func_get_args() in this example. It ensures that the arguments from the function call are provided to the foreach loop. There is another function called 'func_num_args' that can also be used in some cases.
2. Passing on the parameters in this fashion is called 'named arguments' in other programming languages and it is commonly referred as params array in PHP.
3. There are a few downsides of using the params array when we consider the PHP documentation notion and the IDE, you will realize those as you become more comfortable with PHP scripts and the behind the scenes operations on a PHP script.

FORMATTING NUMBERS

In this chapter, we will be drifting towards an aspect that is less on the technical side and more on the final result in the form of display that you see on your websites. Imagine you visit a website to buy electronics for your home. After scrolling through different items, you come down to three of them to decide upon. Now you check the price tag of each item and suddenly you realize that the mathematics classes that you slept in during school days are coming back to haunt you. The pricing of the products is something similar to this, $32456785. Well we know that it is a very big number for electronic items, but just assume it reaches that point. Do you find the format in which the amount is displayed to be user-friendly? Would you not have liked if it was displayed in the standard format? Even though this might look like a very small issue, however; in the long run, user satisfaction from the smallest of details, determine the success level of the website. As they say, no matter how good your content is, if you do not present it in a good way, you will never get the fruits out of it.

How to format numbers in PHP?

Building on our discussion in the previous paragraph, let us discuss how numbers can be actually formatted to be displayed in an organized manner on the browser. PHP has a 'number_format()' function that groups the value passed to it in thousands format.
The syntax of the 'number_format()' function is :
"number_format (number, decimal, decimal point, separator)"

We will have a closer look at the syntax and the parameters passed to it.

Number: The first parameter is the number value that needs to be formatted. It is mandatory to have the number parameter inside the number_format function and if no other parameter is given, the number is formatted without the decimals and in a comma separated format.

Decimal: The decimal format specifies the number of values that will show up after the decimal point. By default, if the decimal parameter is set, the separation for the decimal value is (.).

Decimal point: This is an optional parameter and specifies the kind of string that needs to be used for decimal points.

Separator: Another optional entity that gives the string to be used for the thousands separator.

Note:

1. Even though the separator parameter is optional, if it is set, it becomes mandatory to set the other three parameters as well.
2. Generally, we stick with just the first two parameters and the next two are not really required in the standard number format followed across the globe.

Eg:

```
1  <?php
2
3  $num = 23456.789
4
5  echo 'I have &pound;', number_format($num,2);
6
7  ?>
```

$_GET

PHP is most effective and useful when it is used to take the information from the users and then use those details to authenticate a particular user or to get access to a website or any other services provided by a particular organization. Why haven't we discussed it yet? Well, the prime reason why we did not discuss how values are fetched from the user because it is very important to get a grip of the basic concepts and functionalities of PHP before we shift towards the more influential aspects of the language. What is the first thing that you do when you visit an ecommerce website or any social networking website? You register with the website and during the registration process; you have to give your basic details and then later you can log-in to the website using the credentials that you get. Have you ever wondered how the website get your details when you click the submit button on the browser? If you think about it from a layman's perspective, it is like your data getting delivered to a database server that is located millions of kilometers away with just a click of the submit button. Isn't it amazing? Well, you know that it is happening with the help of PHP, but the bigger question is how? The answer to that is by using the $_GET and the $_POST. In this chapter, we will discuss about the $_GET and decide how it stands as compared to the $_POST method.

What is $_GET and how is it used?

The $_GET method is used to deliver information from a PHP form. The $_GET appends the users' information to the page URL, where in each detail is separated by a '?' symbol.

Eg:

```php
<?php
  if( $_GET["name"] || $_GET["age"] )
  {
     echo "Welcome ". $_GET['name']. "<br />";
     echo "You are ". $_GET['age']. " years old.";
     exit();
  }
?>
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="GET">
  Name: <input type="text" name="name" />
  Age: <input type="text" name="age" />
  <input type="submit" />
  </form>
</body>
</html>
```

Note:

1. There are quite a few things that can be observed from this example of the $_GET statement. For starters, the method = "GET" in the form section defines the type of delivery option that PHP will use in its webpage.
2. Notice how the variables in are accessed using the $_GET in the PHP section of the code.
3. User information in the form of variables can also be delivered to any other PHP file. The only change in the above code will be the inclusion of an action ="xyz.php", where xyz is the name of the PHP file where the variables need to be delivered.

4. We have already talked that on using the $_GET option, the user data will reflect in the URL of the web browser once the submit button is clicked. The URL of the next webpage will look something like:
   http://www.test.com/index.htm?name1=value1&name2=value2
   Where value1 and value2 are the user information and name1 and name2 are the names of the variables they are fed into.

5. The problem with using the $_GET method is the security threat that it poses as the user information is directly displayed in the URL of the webpage and the possibilities of injections are pretty high. $_POST is considered to be a safer and better approach to deliver values.
6. $_GET finds great utility for debugging purposes, primarily to check if data is actually getting transferred or not.

$_POST

In our last chapter, we discussed how $_GET can be used to deliver user data from a HTML form to a PHP file. In this chapter, we will discuss about the other method to perform the same operation, the $_POST method. As we mentioned at the end of our last chapter in this series that the $_POST method is more secure and reliable as compared to the $_GET method, through this chapter, we will try to give enough reasons to justify our claim and also give you a detailed insight about how $_POST actually works in PHP.

Eg:

```php
<?php
  if( $_POST["name"] || $_POST["age"] )
  {
     echo "Welcome ". $_POST['name']. "<br />";
     echo "You are ". $_POST['age']. " years old.";
     exit();
  }
?>
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="POST">
  Name: <input type="text" name="name" />
  Age: <input type="text" name="age" />
  <input type="submit" />
  </form>
</body>
</html>
```

Now what just happened there? You will notice that the example that we have written here is the exact same as the example that we gave for the chapter on $_GET, with the only difference being the

use of $_POST instead of $_GET. Well, before you start wondering how that is possible, let us erase all your doubts and say that this is precisely the difference between two PHP scripts written using $_GET and $_POST. It goes without saying that most of the points of notes regarding $_GET in the previous chapter, are applicable for $_POST as well. However; there is one point that is not applicable for $_POST and that is where the difference between the two delivery options comes into light.

Unlike, $_GET, $_POST does not include the user data sent in the URL. What it means is while we were able to see all the variables and values assigned against them appended to the URL in case of $_GET, with $_POST, only the URL is displayed on the browser tab. With no information leak, $_POST does give a more secure and reliable option as compared to $_GET and also reduces the possibility of injections.

Both $_GET and $_POST are called superglobals, which means that they exist inside every function by default and do not have to be passed in as a parameter. A superglobal is available to each and every line in the script and anything that is pushed into another variable from a superglobal, must have been first fetched from the HTML form with the help of either the POST or the GET method.

ECHO VS PRINT

Now that we have crossed the initial hurdle and have written our first PHP program, I am sure questions would be emerging in your brains thick and fast. Why are headers not required? Can I compile my PHP code before I open the file in the browser? However; there is one question that might not have necessarily clicked in your mind, but discussing this right at the start clears out a lot of confusion for beginners in PHP. It is amusing that a lot of people who have studied or are studying PHP, believe that there is no print statement in PHP and echo does the job that print used to do in programming languages such as C. Even though they are not completely wrong in believing that echo statement does what print was doing all along, however; it is important to know that PHP does have a print statement of its own and just like the echo statement, it is also used to output content on the webpage.

How is print different from echo?

Why is PHP having two statements to output content on the webpage? Why not just have one and make it easier for people who want to learn the language? I would be lying if I say that this question did not come in my mind when I just started learning PHP, in fact I always believe that it is a great thing to question the

established, because only when you raise questions, you give yourself a chance to learn more.

1. Number of Parameters

The first difference between print and echo is the number of parameters that can be passed to the two. While echo accepts more than on parameter to be passed to it, print restricts the number of parameters to just one. If a echo statement is taking more than one parameter, then comma(,) is used to separate them.

```
1 | echo “I love”,”PHP”;
2 | print “Me too”;     // print “Me”,”too”; is not accepted in PHP.
```

2. Usage

Even though both print and echo are used to output the content of the webpage, it is important to understand that both these statements do not perform their job in the exact same way. Print is a language construct and it can be used to return a value, a feature that is missing from echo. Going into the details of expressions and statements, Echo expr is a statement while print expr is an expression and as such echo cannot have an expression alongside it. Therefore, something like $x = 5 + echo 6; will not be accepted by PHP, however; $x=5 + print(6); is a totally valid statement.

3. Performance

Echo is slightly faster than the print statement, but as it stands, this difference in performance does not really come into play unless you are working on a very big application that would/is expected to receive heavy traffic. Further, more than the

performance, it is the ease of use that is the major criteria in this case.

So, what should I use, print or echo?

As it is mentioned in the paragraph above, performance is not the major criteria to choose between echo and print, however; given the fact that most of applications ardently use echo in their webpages, the ease of use factor surely favours echo.

Thank you for reading!

We invite you to share your thoughts and reactions

STONE RIVER
eLEARNING