**PHP: A Brief Introduction**

PHP can be viewed as a relatively simple language to learn, as well as a difficult one. Due to the fact that the functions, algorithms and basic data types are so similar to those of languages entry-level programmers might be more acclimated to, they might analyze it and quickly recognize familiar patterns along with function and variable usage. However, the difference in the purpose of PHP as well as the nature of the PHP language might also cause newcomers to this language to cast it off as too divergent from the norm for comfort. A brief introduction to the history, use, and data types of PHP may very well help to clear the murky waters of its differences and allow those who might otherwise abandon it to learn how it might be implemented, but also, and possibly more importantly, to understand why knowing such a language would be advantageous for anyone wishing to make a place for themselves in the world of programming.

PHP originated as a personal C program that was written, and used, by Rasmus Lerdorf in 1994 in order to track the traffic to his resume which was posted online (The PHP Group, “History of PHP”). After he implemented this new program, dubbed by him “Personal Home Page Tools,” or “PHP Tools,” he recognized that it required more functions and over time he rewrote the program to incorporate more functionality, and by 1995 he had released the source code for his project to the public so that others may use it as well as develop it further. In these early years of its life, PHP was for a time renamed “Forms Interpreter,” leading to the original version of PHP frequently being referred to as PHP/FI. During this time of it’s renaming, it was defined as it’s own programming language, and was stripped of its definition as a project written in C. It went through a complete overhaul and built-in support for various databases as well as many other web based scripting support was developed into the code itself. By the end of 1998 it had gained a rather large following, considering its humble beginnings, and was implemented in “approximately 1% of all domains on the internet at the time” (The PHP Group, “History of PHP”). Even though it had considerable support and widespread use, it was clear that it could not last on its own for long, because although it had many additions, developments, and the scrutiny of many different people, it had still been written primarily by a single person, for a single purpose.

PHP/FI is often referred to as PHP 2.0, due to the major reconstruction it underwent in 1996, and the continuing upgrades it received through 1998. The third version of PHP, PHP 3.0, was the earliest version of PHP which approximates the way PHP is written and implemented today. College students Andi Gutmans and Zeev Suraski undertook another complete rewrite of the base code for the language development for use in an application they were developing for a project at school. This revamp of the original language introduced object-oriented support and much more consistent syntax. Gutmans and Suraski continued work on the development of PHP, even after PHP 3.0 was released, and introduced PHP 4.0 in mid 1999. At this point, PHP included support for “more web servers, HTTP sessions, output buffering, more secure ways of handling user input and several new language constructs,” (The PHP Group, “History of PHP”). Since that time, PHP has continued to evolve until it became what it is now, and to this day it is still evolving, including the beta a new version, PHP 7.1.0, which was released very recently.

PHP is now primarily considered a server-side scripting language, rather than a programming language. The important difference for programmers between scripting and programming languages is that scripts do not need to be compiled in order to run. They are used “to script the behavior that handles a request,” (Things That Matter Most, “Scripting vs. Programming: PHP and OOP”). PHP is usually run on a web server, which can be created for home users using programs such as WAMP, which is a software stack for Windows OS which offers support for Apache, OpenSSL, MySQL as well as PHP. Before its characterization as a server-side scripting language, it was used as a means to create more dynamic web pages. It’s use now is to act as a filter for incoming and outgoing data between client and server. This allows for functionality such as forms, and retrieving and committing information to server-side databases. It also houses the ability to write custom extensions in either C or C++, and the format for creating theses custom extensions is both defined and documented for users to easily connect their extensions to the main functionality of PHP.

The data types of PHP are highly analogous and comparable to the built-in data types of most other well-established programming languages, and also shares its object-oriented programming capabilities with most extensive languages that entry level programmers may have already had experience with. PHP has many different built-in data types, including three numeric types, one sequential type, and strings (text). PHP’s numeric data types are integers, or whole numbers, and floating-point numbers, or values with decimal digits. Contrary to many other programming languages, integers in PHP can be decimal, as a sequence of digits, octal, a sequence of digits which may have leading 0s, or hexadecimal, numbers which begin with 0x and follow by a sequence of digits or letters, (Rasmus, Tatroe and MacIntyre, p. 22-23). Floating-point numbers in PHP may be assigned as either regular decimal style numeric (i.e. 3.14, 0.07, etc), or in scientific notation, (i.e. 0.314E1, 7E-2). As with a few other languages, PHP approximates the representation of floating-point values, such that in many cases the value 3.5 is actually represented as , which means that programmers must be very attentive and conscious of these numbers when using them in mathematical equations. Booleans are also considered numeric types, they are simple truth values, defined as either true or false, but at the base of the code, this is defined as 0 or 1.

Strings, objects and arrays are three incredibly important and necessary built-in data types with ample support and documentation. There are too many functions for these data types, and those functions are too in-depth to receive the scrutiny they deserve in an introduction such as this, but to not discuss them at all would be iniquitous. Strings in PHP are well supported, due to the fact that text is so imperative and common in web usage. There are a vast number of ways to manipulate strings, including comparisons, disassembly, searches, concatenation and curtailing. An important note to cover when considering strings in PHP, is the difference between the use of single or double quotation marks. Most importantly, strings which are declared using double quotes may be interpolated, meaning that the text which comprises these strings may be expanded, or more clearly for programmers, may have variables inserted within them which appear as the value of the variable rather than the name of the variable. An example within “Programming PHP” (Rasmus, Tatroe and MacIntyre, p. 24) illustrates this point very distinctly:

Code:

$name = "Guido";

echo "Hi, $name\n";

echo 'Hi, $name';

Output:

Hi, Guido

Hi, $name

In addition to the ability to interpolate text with in double quotes, usage of double quotes also supports a number of character insertion such as newlines (\n), tabs (\t), backslashes (\\), etc., which are unsupported when used with single quotations.

Arrays in PHP are exactly what any programmer might immediately think of when imagining arrays, however two very notable characteristics of arrays in PHP is that they a) may have custom indices, such as text indices or non-numerical order even if the index is an integer, and b) that the type of data stored in an array is not defined before it’s implementation, nor even afterward. Data in arrays in PHP may be of multiple different types, so that an array may contain both string and integer values, as well as Booleans, objects, or any other data type that PHP offers. (The PHP Group, “Arrays”).

Objects are custom defined data values that are created by the programmer. Once again, the documentation and support for objects is much to far-reaching to be fully captured in such a short period, but suffice it to say that they allow for a far more customized experience with a scripting language than languages such as Python (another server-side scripting language) might offer.

With this brief overview of PHP’s usage and data typing, programmers wishing to enter the realm of web development may hopefully find their passage from application development forward easier. It is clear to see that PHP has a basis in conventional application programs, likely due to it’s origination as a C program. Since it’s initiation into the wider programming world, it has received frequent updates, abundant documentation, as well as overly sufficient support of numerous web-related applications. The few topics breached in this introduction barely scratch the surface of the capabilities and implementations of PHP, which illuminates its fantastic growth from a small, personal program into an extensive general-purpose web-development language.

Works Cited

Lerdorf, Rasmus, Kevin Tatroe, and Peter MacIntyre. *Programming PHP*. Beijing: O'Reilly, 2002. Print.

"PHP: Arrays - Manual." *PHP: Arrays - Manual*. The PHP Group, n.d. Web. 19 Aug. 2016.

"PHP: History of PHP - Manual." *PHP: History of PHP*. The PHP Group, n.d. Web. 19 Aug. 2016.