SUNY Polytechnic Institute

Overview of the PHP Programming Language

Mehdi Essaadi, Mohammad Karimi nia, Patrick Cooper, Timothy Tu, Matthew Szlucha

CS 431: Principles Programming Languages

Professor Ronald Sarner

11/30/2020

History

PHP, originally an acronym for “personal home page”, was created in 1994 by Rasmus Lerdorf as an extension, using C. His implementations to the C language allowed him to work with web forms and databases, as well as form handling and HTML embedding. As a result, PHP allowed users to create small web applications. The first version was released to the public as php tools version 1.0 so that the language could gain feedback and the code could improve.

In 1997 the second release of PHP was released turning the extension into an official language. The language had many inconsistencies in naming, but a development team had formed to help improve the code.

By 1998 The project had switched developers, led by Zeev Suraski and Andi Gutmans, they rewrote most of the PHP core, and introduced the “Zend engine” a script engine with a powerful interpreter for php. Throughout PHP 3s cycles the names acronym was changed to “Hypertext Preprocessor” because of its similarities to HTML.

PHP 4 was released in 2000, and went through multiple releases as the core was re-written. The language had introduced “superglobals”, fixed many security issues, introduced a command line interface, and fixed a few memory leaks.

In 2004 PHP 5 was released, further developing on the languages new engine, giving more performance improvements, addition of PHP data objects for accessing databases, JSON and Namespace support, support for the windows OS, replacement of multiple libraries, support for generators, and argument unpacking.

PHP 6 was scrapped and never released but planned to add Unicode support. Due to lack of understanding in Unicode and performance issues arising from converting to and from Unicode, the version was scrapped, but some non Unicode features were released with PHP 5.3

In 2015 PHP 7 was released. Led by Dmitry Stogov, Xinchen Hui, and Nikita Popov this version further worked on the PHP engine, gave more performance improvements, further support for the windows OS, better syntax, more consistency across platforms, and more error fixing.

PHP 8 is out during 2020, adding many changes to compiling and new function types, but is also not compatible with code written in previous versions of the language.

Currently many websites use this language, but about 40% are still running on no longer supported old versions such as PHP 5, and by the end of 2020 the most popular version, PHP 7.2 will lose support raising the percentage to 84% of outdated PHP coded websites.

Distinguishing Features and Application Domain

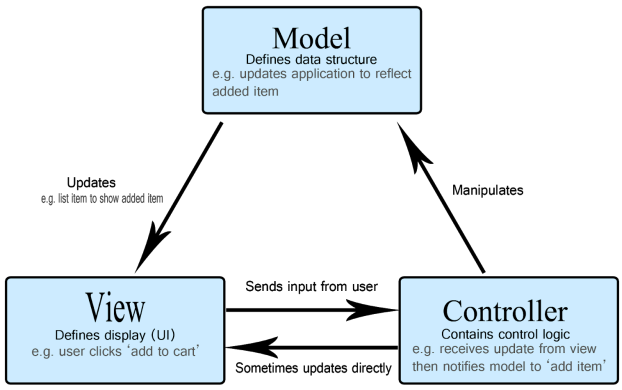
Each programming language has its own special features that set itself to be different from other languages. PHP is a server-side scripting language that is simple and easy to use, a programming language with a similar syntax to C. PHP is organized for general-purpose programming, a programming language that is capable of creating all types of programs. PHP supports object-oriented programming features which result in increased speed and introduces added features such as data encapsulation and inheritance. PHP declares its data types based on the value assigned to the variable in execution time.

PHP is known for its flexibility and embedded nature that “can be integrated with languages such as HTML, MXML, Javascript, and other languages”(PHP Unique Features). PHP can be run on many platforms including multiple operating systems such as Windows, Unix, Mac OS, Linus, and more. PHP scripts can be run on any device such as laptops, phones, on command-line as well as on machines. PHP frameworks are open source and free to use. PHP has a huge active community, with many online developers willing to help beginners with

web-based applications, as well as the large amounts of volunteers worldwide who contribute to many features and new versions for PHP libraries.

PHP is an open-source scripting language that is free to use. It supports “many databases such as MySQL, SQLite, Oracle, and more”(PHP Unique Features). PHP provides the libraries to access these databases to interact with web servers, it has a large availability of pre-defined functions that support data encryption options which allows the implementation of dynamic websites with secured data, it even allows its users to use third-party applications to secure data. PHP includes a way to provide memory usage information so that developers can optimize their code, it has fast and efficient performance resulting in fast loading websites, provides a built-in module for an easy and efficient database management system, and also supports session management and the removal of unwanted memory allocation.

There are many PHP frameworks such as the Model View Controller (MVC) which makes the development and maintenance of the code easier. The MVC is split into 3 parts, model, view, and controller. Codecademy best describes the MVC in terms of a todo app. “The Model would define what a task is and that a list is a collection of tasks. The View would define what the todos and lists look like visually. Lastly, the Controller would define how a user adds a task or marks another as complete. In this case, the Controller connects the View’s add button to the model so that when you click the add task button, the Model adds a new task.” (Codecademy).



Data types

There are currently 8 main data types in the PHP language, all under 3 categories. The categories are scalar, compound, and special. Scalar types include strings, integers, floats, and Booleans. Compound types include arrays and objects. Special types include nulls, and resources.

Scalar data types are single unit data types like integers. They store one unit of data, up to a maximum size.

Compound data types are much like scalar but can store multiple units of data, as long as they have a consistent data type. A common example of this is Arrays.

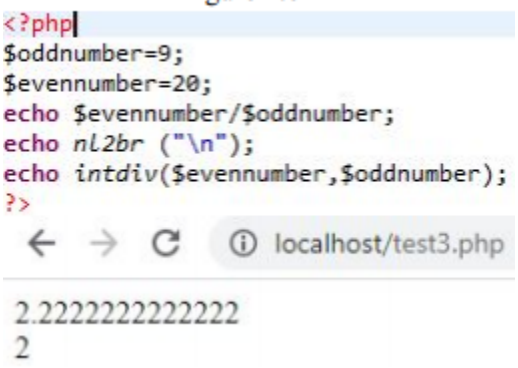
Special data types are data types that do not quite fit into either category, thus they are "Special". One example is null values. NULL does not contain any data. They are the default data type of any variable that has not been declared or has been emptied.

Primitives

Primitives are the smallest form of data processing in a computer language. They are used to build larger, more complicated programs. According to the PHP manual, there are eight primitives; four scalars (Bool, Int, Float String), two compounds (Array and Object),and two special (Resource, and Null). (Introduction)(PHP Data Types)

Scalar Types

Figure 1.0



A scalar type, simply, is a data type that contains one single value. First, Int, is a data type that can hold a single value. This value cannot be a fraction, or decimal, but can be negative. One interesting fact about int types is how integer division is handled. If

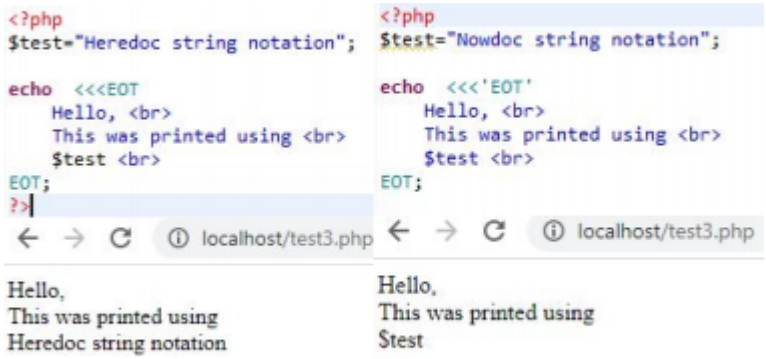
you divide two integers in PHP, the product will be assigned a float if the answer should be a decimal. If you would like to use integer division, where the answer would be rounded, and stored in an Int, you must use the INTDIV function. See figure 1.0. (Introduction)(PHP Data Types)

Second, Bool is a simple type that can only have two values. True and False. In PHP, a bool can be casted to an Int, Float, or String. The false values would be 0,0.0, and NULL respectively. True would be represented by any other value.

Third, Float is a type that can store a decimal. Depending on your system, you can store a float up to 1.8E108 bits in size. (Introduction)(PHP Data Types)

Last, string is the final scalar primitive type. Strings in PHP can be made using single or double quotes. Single quotes will not process variables, or any markup characters like newline. If you output a single quoted string, PHP will print the variable name, not its value. Double quotes will process valuables and characters. There are also two other formats to make strings in PHP: Heredoc, and Nowdoc. Heredoc is similar to double quotes, it will process variables, where Nowdoc is the same as single quotes. See figure 1.1 for Heredoc, and Nowdoc format. This figure shows both formats, and how they process variables. (Introduction)(PHP Data Types)

Figure 1.1



Compound Types

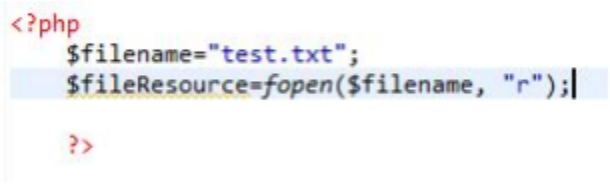
Compound types are data types that can contain more than one value. There are two compound types(Array, and Object). First,there are three types of arrays (Numeric, Associative, and Multidimensional) Numeric arrays act the same as C/C++. They use index values starting 0 to access and loop through each element in the array. Next, Associative arrays are similar, except the user defines the index keys. For example in figure 1.2, you can see the user defines an index key as a unique identifier for the value 50. To access this value in the array, the user uses the string filter as the index key. This is also seen in figure 1.2. Multi dimensional arrays act the same as in C/C++. They use numeric indexes to access each element in the array. Optionally, these arrays can also use the user defined keys in place on index values. (Introduction)(PHP Data Types)

Figure 1.2

Special Types

Special types are the primitive data types that do not fit into another category. These include resources and NULL. NULL is used the same way as C/C++. It is used to represent an element with no value. Resource is used as a pointer to any information outside of the program that is needed. This can include a file, image, other programs, etc. A resource example can be seen in figure 1.3. In this example we are making a

Figure 1.3



pointer to an open file. In order to access the information in the file, we must use the fileresourse, not filename. (Introduction)(PHP Data Types)(PHP: Arrays)

Scoping and Parameter Passing

Most of the PHP variables only have a single scope. Unlike C, where "global variables are available to functions unless it is overwritten by a local definition." (Variable Scope). Any variables that are used inside a function is limited to the local function scope.

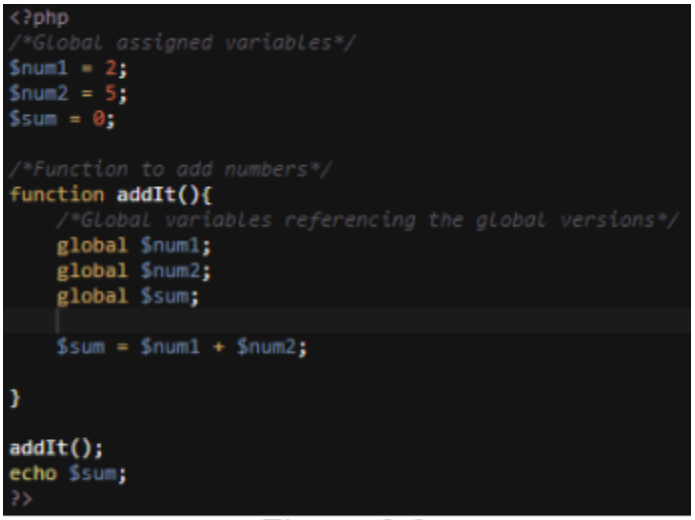
```php
<?php
$test = 2; /*Global assigned variable*/

function showVariable(){
    echo $test; /*References a variable in the local scope*/
}
?>
```

Figure 2.1

In figure 2.1, when the function showVariable() is called, there will be no result in the output. This is because the variable $test is assigned globally, but since PHP variables only have a single scope, $test has to be declared within the function in order for it to be used in the function.

With the use of global variables in PHP, the variable has to be declared global inside of the function if it is going to be used inside the function. "All references to either variable will refer to the global versions. There is no limit to how many global variables can be manipulated in a function"(Variable Scope).



```php
<?php
/*Global assigned variables*/
$num1 = 2;
$num2 = 5;
$sum = 0;

/*Function to add numbers*/
function addIt(){
    /*Global variables referencing the global versions*/
    global $num1;
    global $num2;
    global $sum;

    $sum = $num1 + $num2;

}

addIt();
echo $sum;
?>
```

Figure 2.2

In figure 2.2, when the function addIt() is called, since the variables inside the function sum are declared globally, the output will produce a value of 7 after the function call.

Another way to access variables in the global scope is by using the PHP-defined $GLOBALS array. “The $GLOBALS array is an associative array with the name of the global variable being the key and contents of that variable being the value of the array element”(Variable Scope). The $GLOBALS array takes the contents of the variable and expresses it as an array element and keeps the name of the variable being passed through. $GLOBALS exist in any scope because it is a superglobal. Superglobals are built-in variables that are always available in all scopes.

```php
<?php
/*Global assigned variables*/
$num1 = 2;
$num2 = 5;
$sum = 0;

/*Function to add numbers*/
function addIt(){
    /*Global variables referencing the global versions*/

    $GLOBALS['sum'] = $GLOBALS['num1'] + $GLOBALS['num2'];

}

addIt();
echo $sum;
?>
```

Figure 2.3

Figure 2.3 will also produce an output of 7, but notice that the way it is written is using $GLOBALS and the variable as an array.

PHP uses static variables, static variables exist only in a local function scope but doesn't lose the value when the program execution leaves the scope. This gives us one way that we can use to create recursive functions.

```php
<?php
function test(){
    static $num = 1;
    echo $num;
    $num++;
}
test();
test();
?>
```

Figure 2.4

In figure 2.4, variable $num is a static variable. The function will first output the number 1, then increment the variable $num to 2. So in the next call of the function, it will output the number 2 and increment the variable again.

"PHP supports passing arguments by value, pass by reference, default argument values, and also supports variable-length argument lists." (Function Arguments). Function arguments are by default set to be passed by value. To modify the function arguments, they must be passed by reference. To have an argument always pass by reference, we would use an ampersand (&).

```php
<?php
function addCheese(&$cheese){
    $cheese .= ' with cheese.';
}
$burger = 'I would like a burger';
addCheese($burger);
echo $burger;
?>
```

Figure 2.5

Figure 2.5 takes the string $burger and puts it through the function which will return the original $burger string combined with the $cheese string in the addCheese() function. This code will output the string "I would like a burger with cheese."

"A function may define C++ style default values for scalar arguments. PHP also allows the use of arrays and also the special type NULL as default values. The default value must be a constant expression. Any defaults should be on the right side of any non-default arguments; otherwise, things will not work"(Function Arguments).

```php
<?php
function burger($addOns = array("cheese"), $base = NULL){
    $buns = is_null($base) ? "brioche" : $base;
    return "Here is your burger with ".join(", ", $addOns)." and $buns buns.\n";
}
echo burger();
echo burger(array("cheese", " bacon", " ketchup"), "lettuce");
?>
```

Figure 2.6

In figure 2.6, in the first call of burger(), the output will produce the string “Here is your burger with cheese and brioche buns.”. The default references in the burger() function point to the $addOns to be cheese and the $base to be NULL. Since the $base is null, it defaults to Bioche in the sentence when being returned. In the second call of burger(), the output will produce the string “Here is your burger with cheese, bacon, ketchup and lettuce buns.”

“PHP has support for variable-length argument lists in user-defined functions by using the ...token. Argument lists can use the …token to be a sign that the function will use a variable number of arguments that will be passed into the given variable as an array. The ...token can also be used when calling functions, to unpack an array or traversable variable into the argument list” (Function Arguments). The … token basically allows the developer to have a sign that shows that a number of arguments will pass through a function as an array variable. This is effective when needing to pass by many things at once, such as the example code below, where we pass 4 numbers through the function all at once.

```php
<?php
function sum(...$number){
    $sum = 0;
    foreach ($number as $num){
        $sum += $num;
    }
    return $sum;
}
echo sum(1,2,3,4)
?>
```

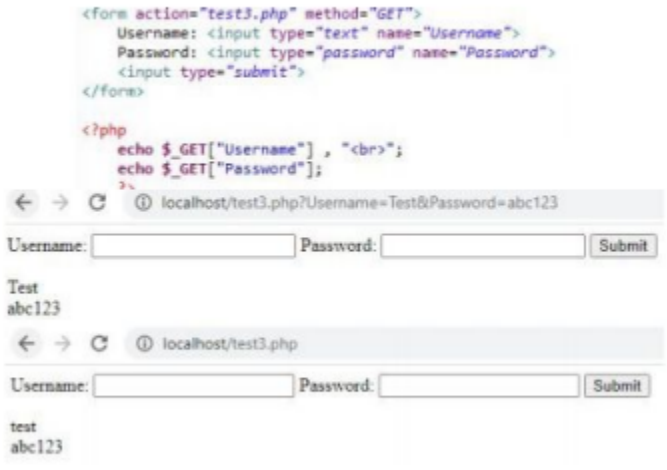
Figure 2.7

In figure 2.7, the function will push 1, 2, 3, 4 into the function variable $numbers as an array. Then inside the function, $acc will add each of the 4 numbers inside the for loop created, producing the output of 10.

I/O Functionality

The simplest form of I/O in PHP is print and echo. Both of these functions print text to the screen. Echo and print are very similar, however echo is faster, and can process multiple parameters, while print can process only one. Both can process HTML markup. This can include font changes and newlines.

Figure 3.0



The easiest way to get data from the user on the front end is a text box. This can be done using a HTML form. The syntax of this form can be seen in the top image in figure 3.0. Put simply, you must specify the method you would like to grab the data (POST or GET). You must also specify a type and name for each textbox. In our example we have two text boxes, one named username and one password. They use the "text" and "password" textbox types. Text is a normal textbox, while password will only display small dots in place of the users cleartext password within the textbox. Next, the lower portion of the image has a command called $_GET. This is used along with the name of the text box to get any data entered into it. We are printing this information back to the screen for testing purposes. In our second image, you can see that when the user enters a username and password, they are passed to PHP, and then printed back to the screen. The second image also has a major security flaw. The URL is printing the user's username and password in clear text. This flaw can be fixed by changing the form method from GET to POST. The only difference between GET and POST as used in the first two pictures, is that POST will hide the entered information from the URL, as

seen in the third picture in figure 3.0. Syntactically, the only changes needed transition from GET to post, is to change the method in the HTML form from ‘GET’ to ‘POST’, and change $_GET to $_POST. (PHP Programming)

Next, File I/O can be used to access and read a file outside of a PHP program. The function used to open a file is fopen(file resource, mode). There are many modes such as read, write, append, etc. An example of opening a file can be seen in figure 3.1. Next, once a file is opened in the correct mode, you can read or write to it. To read a file you can use the function fgets(file resource). This function is similar to getline from C/C++. It will grab a whole line in a file and stop at a newline. An example of this can be seen in figure 3.1. There is also fread(file resource, length), where you can specify the amount from the file you want to pull, and fgetc(file resource) will pull one char at a time from the file. (EdurekaIN)(PHP File Open).

Figure 3.1

There is also a function to write to a file. You must make sure to open the file in write or append mode first. Then the function fwrite(fil resource, pointer to info to be written). An example of this can be seen in figure 3.2. This figure also utilized the fclose(file resource) function, used to close a file. Last, feof(file resource), can be used to

Figure 3.2

find the end of a file. This can be useful when looping through a file.(EdurekaIN)(PHP File Create)

Control and Data Structures

PHP has several structures to control the flow of a program. It has multiple comparison structures. If, will only run its contents if an expression is true. Else is a companion structure that will run when the if's expression is not true. Elseif is a combination of the two which will run its contents if the preceding expression is not true but its own is. A switch is another comparison structure, which compares a single value to multiple options, and runs the option that it matches, or a default if there is none. The break structure will escape the switch, or the next structure, the match. Match is a similar structure that will return a value instead of running a statement.

PHP has two main types of loops. While loops are simpler, they repeat for as long as a condition is true. As with most loops, this requires a condition to avoid infinitely looping until a crash. Do-while loops are similar, but do not check the expression. For loops are slightly more complicated, they have three expressions to evaluate, one checked at the beginning of each iteration, and one performed at the end of each iteration. Until the second expression is false, the loop will run. Foreach is a related structure that iterates over arrays specifically. Break will also escape any type of loop. Continue, by comparison, will skip the current iteration of the loop but not break out of it.

Declare is a unique control structure, it will set one of several directives, such as ticks, which runs an event after a certain number of processes are performed. Return will do something different depending on the scope. If used within a function, return will stop the execution of that function and return the value it is given. If used outside of any function, return will cease execution of the entire program.

Require and include are two similar structures. Both require a specific file to be present and will include it in the script if it is, but include will only send out a warning if the file is absent while require will create a fatal error that crashes the program. Require_once and include_once are similar to their respective structures but will only include the program once, rather than repeating if it shows up again.

Goto is a structure that will jump around in a program. It can only remain within the same scope. It can be used to exit a loop or switch structure, but not enter it. It is not an ideal control structure to use. A program would be better off flowing properly without jumping around, but goto is there for use regardless. PHP supports recursion as well, just as C does.

The PHP standard library supports several data structures. These structures include doubly linked lists, heaps, maps, and fixed arrays.

Tuckers Criteria

Expressivity

Tucker would rate PHP's expressivity as great/excellent. PHP includes a number of control structures such as if-then-else, while loops, and for loops. PHP does follow the algebraic rules in math equations. The meaning of the symbols inside PHP does not vary greatly as it holds its meaning across the language.

Well-definedness

By well-definedness, Tucker looks at the syntax and semantics of a language and checks if they are clear, consistent, and complete. This would help compiler or interpreter writers translate all of the functions a language offers without skipping any. PHP is a very clear language. For example: echo $_GET[“name”] will get the name stored in the name field. Tucker

would rate the PHP well-definedness as great since anyone with minimum knowledge of PHP would excel in it in the matter of few hours

Data Types and Structures

PHP just like most of the programming languages, it offers a variety of data types such as real, integer, boolean, characters, and classes/objects. Although PHP supports primarily arrays, which could be used in a flexible way and solve most of the problems we could face. PHP comes with a Standard PHP Library that extends its data structure to include Doubly Linked Lists, Heaps, and maps. It also supports passing parameters by value and passing them by reference. With all these data types and structures, Tucker would rate PHP's data types and structures as great since it offers almost everything a programmer needs to achieve a goal within PHP's domain.

Modularity

Tucker assesses a language's modularity by looking at the language's ability to split complex programs into smaller more manageable programs for the ease of readability and maintenance, as well as enhancing the reliability and the verifiability of the program. Languages could achieve this by including procedures, subroutines and function. In my opinion Tucker would rate PHP as fair since it only includes functions, but you can achieve all the said characteristics by linking PHP to an SQL database.

Input/Output Facilities

Tucker would assess PHP's Input/Output functionalities as good. You could use PHP to display output to screen or printer and scan-in input from mouse and keyboard. You can also read from a file, make changes to a file, and write to a file. Without partnering with a different database/program there is no indexing offered by PHP.

Portability

Portability is one of the most important features of a language. Having the ability to easily undertake cross-platform development is a valuable one, especially when operating in a multi-platform corporate environment or when trying to address multiple markets. PHP can run on multiple platforms such as UNIX, WINDOWS, MAC OS, and OS X, this allows for a program created in WINDOWS using PHP to be run in UNIX with minimum to no changes. This ability made PHP one of the top competing programming languages in the domain. Tucker would give PHP a great mark for its portability.

Efficiency

A language's efficiency is measured by the time it takes to compile and execute programs. Languages that take more time to compile and execute the same program are considered to be less efficient. PHP comes at the top, head to head with Python when efficiency is involved, especially when compared with other competing scripting languages such as ASP.NET, JSP and PERL. Thanks to 'the Zend engine', PHP code is parsed and turned into opcodes(instructions that specify the operations to be performed) which are then interpreted. Given all the information above, Tucker would rate PHP's efficiency as excellent.

Pedagogy

PHP is one of the easiest web development languages out there to learn especially if you have previous programming experience. In a few hours you can start developing your own web applications. Some features that set PHP apart is the need to prefix all variables with a dollar sign $ and the use of the -> operator to indicate that a method was called on an object, besides that, it is pretty much similar to a lot of different languages out there today. One thing that is worth noting is that PHP relies heavily on the use of arrays, so if you need to do something that doesn't

fit an array, you might have to come up with your own data type which is not that big of a deal, but it could add a level of complexity for beginners. Tucker would assess PHP's pedagogy as great since it is straightforward and doesn't require a lot of programming experience to be learnt.

Generality

A language generality is expressed by how much it can achieve regardless of the domain. Using PHP there is practically no end to the possibilities that could be achieved. Some of the creative uses for PHP are: E-Commerce, building Graphical user interfaces, online communities, web applications, parsing XML files, generating mailing lists, creating website templates and the list goes on. Tucker would rate PHP generality as excellent giving all the flexibilities that it carries across domains.

Conclusion

Nowadays, PHP is one of the top competing scripting languages out there. It's simplicity and fruitful domain structures and data types, quick compile and execute times, flexible portability, endless domains, and much more has made it the first choice for many companies and programmers around the globe.

References

Boersma. Eric, “PHP vs Python: Is There a Clear Choice in 2020?”, Feb. 2020, https://stackify.com/php-vs-python-which-should-you-choose-in-2019/

Codecademy. “MVC: Model, View, Controller.” *codecademy*, N/A, https://www.codecademy.com/articles/mvc.

“Control Structures - Manual’, Php, The PHP Group, https://www.php.net/manual/en/language.control-structures.php

“Creating an Index - Manual.” Php, The PHP Group, https://www.php.net/manual/en/mongo.tutorial.indexes.php

EdurekaIN. “PHP File Tutorial | File in PHP | PHP File Handling Tuorial | Working with Files in PHP.” *YouTube*, YouTube, 18 Mar. 2015, www.youtube.com/watch?v=e7NvwnWaOZw.

GeeksforGeeks. “PHP Unique Features.” *Geeksforgeeks.org*, GeeksforGeeks, 2019, https://www.geeksforgeeks.org/php-unique-features/.

“Introduction - Manual.” *Php*, The PHP Group, www.php.net/manual/en/language.types.intro.php.

Miller, Richard, “PHP is getting Faster”, https://www.acquia.com/blog/php-getting-faster

“Objects and references- Manual.” Php, The PHP Group,

https://www.php.net/manual/en/language.oop5.references.php

"PHP Data Types." *TutorialRepublic*, Tutorial Republic, www.tutorialrepublic.com/php-tutorial/php-data-types.php#:~:text=The values assigned to a,, Object, resource and NULL.

PHP File Create/Write, Refsnes Data, www.w3schools.com/php/php_file_create.asp.

PHP File Open/Read/Close, Refsnes Data, www.w3schools.com/php/php_file_open.asp.

"PHP Programming Language Tutorial - Full Course." FreeCodeCamp.org, 20 Jan. 2018, https://www.youtube.com/watch?v=OK_JCtrrv-c

PHP Manual. "Function Arguments." *PHP*, N/A, https://www.php.net/manual/en/functions.arguments.php.

PHP Manual. "Variable Scope." *PHP*, N/A, https://www.php.net/manual/en/language.variables.scope.php.

"PHP." *Wikipedia*, Wikimedia Foundation, 27 Nov. 2020, en.wikipedia.org/wiki/PHP.

"PHP: Arrays." *GeeksforGeeks*, 1 July 2020, www.geeksforgeeks.org/php-arrays/.

Reyes. Joel, "15 Wonderfully Creative Uses for PHP", May 2009, https://code.tutsplus.com/tutorials/15-wonderfully-creative-uses-for-php--net-4714

"Stored Procedures - Manual." Php, The PHP Group,

https://www.php.net/manual/en/mysqli.quickstart.stored-procedures.php

Techie, Clever. "PHP Data Types." *PHP Data Types | Clever Techie*, Clever Techie, 17 Jan.

2017, clevertechie.com/php/11/php-data-types.