Introduction on PHP - Hypertext Preprocessor

Shravan Kumar[#1], Jawahar Kumar[#2], Sagar Kumar[#3]

Department Of Computer Science & Engineering

[1]*M.Tech Computer Science & Engineering*
[2]*M.Tech Computer Science & Engineering*
[3]*B.Tech Computer Science & Engineering*

AllianceTech Computer Systems Pvt. Ltd.

Bilaspur (Chhattisgarh)

[1]shravan51090@gmail.com
[2]jawahar11088@gmail.com
[3]sagarcvru@gmail.com

***Abstract*— PHP is a server scripting language, and is a powerful tool for making dynamic and interactive Web pages.**
PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

***Keywords*—Data Types, Function, Objects, Implementation, Use, Editors, Server etc...**

1. INTRODUCTION

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language.
Originally created by Rasmus Lerdorf in 1995, the reference of PHP is now produced by The PHP Group. While PHP originally stood for Personal Home Page, it now stands for PHP: Hypertext Preprocessor, a recursive acronym.
PHP code is interpreted by a web server with a PHP processor module which generates the resulting web page. PHP commands can be embedded directly into an HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone graphical applications.

SYNTAX:
The following Hello world program is written in PHP code embedded in an HTML document:

```
function getAdder($x) {
 return function($y) use ($x) {
  return $x + $y;
 };
}
 $adder = getAdder(8);
echo $adder(2); // prints "10"
```

2. DATA TYPES

- PHP stores whole numbers in a platform-dependent range, either a 64-bit or 32 bit signed integer equivalent to the C-language long type.
- Integer variables can be assigned using decimal (positive and negative), octal, hexadecimal, and binary notations.
- Floating point numbers are also stored in a platform-specific range. These are typically created by functions from a particular extension, and can only be processed by functions from the same extension; examples include file, image, and database resources.
- Arrays can contain elements of any type that PHP can handle, including resources, objects, and even other arrays.
- PHP also supports strings, which can be used with single quotes, double quotes, and nowdoc or heredoc syntax.

3. FUNCTION

PHP has hundreds of base functions and thousands more via extensions. These functions are well documented on the PHP site; however, the built-in library has a wide variety of naming conventions and inconsistencies.PHP currently has no functions for thread programming, although it does support multi process programming on POSIX systems.

Additional functions can be defined by a developer:

```
function myFunction() { // declares a function, this is named myFunction
 return 'John Doe'; // returns the value 'John Doe'
}
 echo 'My name is ' . myFunction() . '!'; //outputs the text concatenated with the return value of myFunction.

// myFunction is called as a result of this syntax.
// the result of the output will be 'My name is John Doe!'
```

PHP gained support for closures in PHP 5.3. True anonymous functions are supported using the following

SYNTAX:

```
function getAdder($x) {
 return function($y) use ($x) {
  return $x + $y;
 };
}
 $adder = getAdder(8);
echo $adder(2); // prints "10"
```

4. OBJECTS

Basic object-oriented programming functionality was added in PHP 3 and improved in PHP 4. Object handling was completely rewritten for PHP 5, expanding the feature set and enhancing performance. In previous versions of PHP, objects were handled like value types. The drawback of this method was that the whole object was copied when a variable was assigned or passed as a parameter to a method. In the new approach, objects are referenced by handle, and not by value PHP 5 introduced private and protected member variables and methods, along with abstract classes, final classes, abstract methods, and final methods. It also introduced a standard way of declaring constructors and destructors, similar to that of other object-oriented languages such as C++, and a standard exception handling model. Furthermore, PHP 5 added interfaces and allowed for multiple interfaces to be implemented. There are special interfaces that allow objects to interact with the runtime system. Objects implementing Array Access can be used with array syntax and objects implementing Iterator or Iterator Aggregate can be used with the for each language construct. There is no virtual table feature in the engine, so static variables are bound with a name instead of a reference at compile time.

If the developer creates a copy of an object using the reserved word clone, the Zend engine will check if a __clone () method has been defined or not. If not, it will call a default __clone() which will copy the object's properties. If a __clone() method is defined, then it will be responsible for setting the necessary properties in the created object. For convenience, the engine will supply a function that imports the properties of the source object, so that the programmer can start with a by-value replica of the source object and only override properties that need to be changed.

The following is a basic example of object-oriented programming in PHP:

```
class Person {
 public $firstName;
 public $lastName;
 public function __construct ($firstName, $lastName = '')
{ //Optional parameter
  $this->firstName = $firstName;
  $this->lastName = $lastName;
 }
  public function greet() {
  return "Hello, my name is " . $this->firstName . " " . $this-
>lastName . ".";
 }
  public static function staticGreet($firstName, $lastName) {
  return "Hello, my name is " . $firstName. " " . $lastName. ".";
 }
}
$he = new Person ('John', 'Smith');
$she = new Person ('Sally', 'Davis');
$other = new Person('amine');
 echo $he->greet(); // prints "Hello, my name is John Smith."
echo '<br />';
echo $she->greet(); // prints "Hello, my name is Sally Davis."
echo '<br />';
echo $other->greet(); // prints "Hello, my name is amine."
echo '<br />';
echo Person::staticGreet('Jane', 'Doe'); // prints "Hello, my
name is Jane Doe."
```

5. IMPLEMENTATION

The PHP language was originally implemented as an interpreter, and this is still the most popular implementation. Several compilers have been developed which decouple the PHP language from the interpreter. Advantages of compilation include better execution speed, static analysis, and improved interoperability with code written in other languages.

PHP compilers of note include Phalanger, which compiles PHP into Common Intermediate Language (CIL) byte code, and Hip-hop, developed at Facebook and now available as open source, which transforms the PHP Script into C++, then compiles it, reducing server load up to 50% .

PHP source code is compiled on-the-fly to an internal format that can be executed by the PHP engine. In order to speed up execution time and not have to compile the PHP source code every time the web page is accessed, PHP scripts can also be deployed in executable format using a PHP compiler.

Code optimizers aim to enhance the performance of the compiled code by reducing its size, merging redundant instructions and making other changes that can reduce the execution time. With PHP, there are often opportunities for code optimization. An example of a code optimizer is the accelerator PHP extension.

Another approach for reducing compilation overhead for PHP servers is using an Opcode cache. Opcode caches work by caching the compiled form of a PHP script (Opcode) in shared memory to avoid the overhead of parsing and compiling the code every time the script runs. An Opcode cache, APC, is planned to be built into an upcoming release of PHP (but not 5.4 as previously planned.

Opcode caching and code optimization can be combined for best efficiency, as the modifications do not depend on each other (they happen in distinct stages of the compilation).

6. USE

- PHP is a general-purpose scripting language that is especially suited to server-side web development where PHP generally runs on a web server.
- Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or elsewhere.
- It can also be used for command-line scripting and client-side graphical user interface (GUI) applications.
- PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS).
- Most web hosting providers support PHP for use by their clients.
- Originally designed to create dynamic web pages, PHP now focuses mainly on server-side scripting, and it is similar to other server-side scripting languages that provide dynamic content.
- The LAMP architecture has become from a web server to a client, such as Microsoft's ASP.NET, Sun Microsystems'JavaServer Pages, and mod_perl.
- PHP has also attracted the development of many software frameworks that provide building blocks and a design structure to promote rapid application development (RAD).
- Some of these include CakePHP, Symfony, CodeIgniter, Yii Framework, and Zend Framework, offering features similar to other frameworks popular in the web industry as a way of deploying web applications. PHP is commonly used as the P in this bundle alongside Linux, Apache and MySQL, although the P may also refer to Python, Perl, or some mix of the three.

7. LIST OF PHP EDITORS

1. CROSS-PLATFORM

- Aptana Studio – Eclipse-based IDE, able to use PDT plug-in, visual JS editor. Open-source, free project. (Community edition merged in).
- Bluefish – A multipurpose editor with PHP syntax support, in-line PHP documentation, etc. With GVFS, supports SFTP, FTP,WebDAV, and SMB.
- Eclipse – PHP Eclipse and PHP Development Tools projects. With additional plug-in supports SVN, CVS, database modelling, SSH/FTP access, database navigation, Trac integration, and others.
- Editra – Versatile open source editor. Syntax highlighting and (partial) code completion for PHP + HTML and other IDE-like features like code browser etc.
- Emacs – Advanced text editor. The nXhtml add-on has special support for PHP (and other template languages). The major modeweb-mode.el is designed for editing mixed HTML templates.
- Geany – Syntax highlighting for HTML + PHP. Provides PHP function list.
- JEdit – Versatile free/open source editor. Supports SFTP and FTP.
- Komodo Edit – General purpose scripting language editor with support for PHP. Free version of the commercial Active State Komodo IDE.
- Netbeans – A dedicated PHP coding environment and complete integration with web standards. Supports SFTP and FTP. SVN support can be added using plug-in.
- SciTE – fast, PHP syntax highlighting, compiler integration, powerful config via Lua API.
- Sublime Text – fast, lot of features.
- Vim – provides PHP syntax highlighting, debugging.

2. WINDOWS

- Alley code HTML Editor – Shareware editor with syntax highlighting for both PHP and HTML.
- ConTEXT – *No longer under development* Freeware editor with syntax highlighting.
- Crimson Editor – Lightweight editor. Supports FTP.
- HTML-Kit – Syntax highlighting, supports FTP.
- Microsoft Web Matrix – A combined editor, server and publishing environment, syntax highlighting for HTML, PHP, Razor, node.js, C# and JavaScript and publishing through Web Deploy and FTP. Supports multiple file encodings as of version 2.
- Notepad2 – Simple editor with syntax highlighting
- Notepad++ – Supports FTP & SFTP via plug-in; syntax highlighting.
- Note Tab – Broad editor, allows you to have multiple tabs open for different files, has FTP support, comes in two trial versions (Note Tab Standard, Note Tab Pro) and one free version Note Tab Light. Note Tab Pro highlights tags.
- Programmer's Notepad
- PSPad – Supports FTP; syntax highlighting.
- RJ TextEd – Feature rich text editor with many great features
- Sublime Text – Free for evaluation IDE.

3. MAC OS X

- Text Wrangler – Supports SFTP and FTP
- Text Mate
- Smultron

4. LINUX

- gedit
- gPHPEdit
- Kwrite

- Kate – Supports any protocol that is supported by KIO. This includes HTTP, FTP, SSH, SMB and WebDAV.
- KDevelop – Supports everything as Kate above with addition of references of functions and syntax parser.

5. COMMERCIAL EDITORS

- Active State Komodo IDE – Support for PHP syntax checking, debugging, trial available
- Adobe Dreamweaver – Supports SFTP and FTP; Trial available
- BBEdit – Supports SFTP and FTP; Trial available
- Coda – Supports SFTP and FTP; Trial available
- Code Charge Studio – Supports FTP
- Edit Plus – Supports SFTP and FTP; Trial available
- EmEditor
- HyperEdit – Integrates PHP, JavaScript and HTML in an only interface WYSIWYG.
- Microsoft Expression Web – Full PHP support with syntax highlighting, etc.; Trial available
- NuSphere PhpED – A PHP development environment and integration with modern web standards. Supports SFTP, WebDAV, and FTP. Native support for CVS source control system, SVN and Git support can be added using Tortoise Windows Shell plug-in.
- PHPEdit – Supports SFTP and FTP
- phpDesigner – supports FTP
- Jet Brains PhpStorm – (Cross platform) professional PHP IDE with advanced editor, on-the-fly code analysis and other web development specific tools including FTP/SFTP synchronization; Trial available
- Embarcadero RadPHP (formerly Delphi for PHP) – Focus on web (Facebook, Google) and mobile (iOS, Android) development; Trial available
- Rapid PHP Editor – Support for PHP syntax checking, auto-complete, debug and support for CSS, Javascript and HTML
- skEdit
- SlickEdit
- Sublime Text – Commercial multi-language text editor with syntax highlighting. Unlimited trial available.[2]
- Text Pad – Trial available
- Top PHP Studio – Supports FTP
- Ultra Edit – Supports SFTP and FTP; Trial available
- Web Smart PHP - Includes templates that generate the initial PHP, HTML, CSS and JavaScript. Edit any PHP file in the full-featured IDE, which includes an interactive debugger, HTML tools, code prompting and separation of the HTML from the business logic. Also includes built-in change management and supports integration with third-party change management.
- Zend Studio – Supports SFTP and FTP (Eclipse with Zend's commercial plug-in)
- Smultron
- VS.PHP – based on Visual Studio IDE, supports Intelligence for PHP, remote and local debugging. Local debugging is performed using IDE embedded web server with PHP extensions.

8. SERVER

1. ZEND SERVER

Zend Server refers to a PHP application server product line offered by Zend Technologies, released in early 2009 with production support available for Windows and Linux. It is available in two versions, Zend Server and Zend Server Community Edition. Zend Server is supported on x86 and x86-64 compatible machines with IBM iSeries support in Zend Server 5. On Linux it is supported on Red Hat Enterprise Linux, CentOS, Fedora Core, Oracle Enterprise Linux, Debian and Ubuntu. On Windows, Zend Server is supported on Windows server 2003/2008, XP, Vista, and Windows 7. At Zendcon 2009 Zend announced the public beta of Zend Server 5.0. The current version is 6.0.1 which included support for production deployment functionality.

2. FEATURES OF ZEND SERVER

- Native installation – Windows MSI, Debian deb and Red Hat yum installation.
- Certified PHP (Both 5.3 and 5.4).
- Zend Framework – ZF installation and updates as part of the native installation package.
- Apache or IIS integration.
- Java connector – Integrates, existing Java functionality as PHP code.
- Web-based administrator console.
- Debugger interface.
- Byte code acceleration – Keeps compiled code in shared memory, increasing performance by reducing disk access and CPU processing time.
- Data Caching API.
- Page caching.
- Job queuing.
- Deployment functionality.
- Application monitoring.
- Function Error.
- Database Error.
- Slow Function Execution.
- Slow Query Execution.
- Slow Request Execution.

- High Memory Usage.
- Inconsistent Output Size.
- Uncaught Java Exception.
- Custom Event.
- Fatal PHP Error.
- PHP Error.
- Application problem diagnostics.
- Zend Download Server (Linux only) - Allows for large content, such as videos, to be downloaded without tying up an Apache process.
- Software updates and hot fixes.

3. ZEND SERVER CE

- Native installation – Windows MSI, Debian deb, Red Hat yum installation
- Certified PHP
- Zend Framework – ZF installation and updates as part of the native installation package
- Apache or IIS integration
- Java connector – Integrates existing Java functionality as PHP code
- Web-based administrator console
- Debugger interface
- Byte code acceleration – Keeps compiled code in shared memory, increasing performance by reducing disk access and CPU processing time
- Data Caching API

4. ZEND SERVER CLUSTER MANAGER

- High Availability Session Clustering with Graceful Shutdown
- Event Monitoring Aggregation
- Code Tracing Aggregation
- Cluster-wide configuration
- Notifications when machines are added or removed from the cluster
- Notifications when cluster node configuration is out of sync
- Deployment functionality for cross-cluster application management

9. PHP INSTALLATION

1. What Do I Need?

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

2. Use a Web Host with PHP Support

If your server has activated support for PHP you do not need to do anything.
Just create some .php files, place them in your web directory, and the server will automatically parse them for you.
You do not need to compile anything or install any extra tools.
Because PHP is free, most web hosts offer PHP support.

3. Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- install a web server
- install PHP
- install a database, such as MySQL

The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

10. PHP SYNTAX

The PHP script is executed on the server, and the plain HTML result is sent back to the browser.
A PHP script can be placed anywhere in the document.
A PHP script starts with <?php and ends with ?>:

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php".
A PHP file normally contains HTML tags, and some PHP scripting code.
Below, we have an example of a simple PHP file, with a PHP script that sends the text "Hello World!" back to the browser:
Example :

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.
With PHP, there are two basic statements to output text in the browser: echo and print.

1. Comments in PHP:

```
<! DOCTYPE html>
<html>
<body>
<? Php
//This is a PHP comment line
/*
this is
a PHP comment
block
```

```
*/
?>
</body>
</html>
```

11. PHP VARIABLES

Variables are "containers" for storing information:

Example:

```
<? Php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

1. Much like Algebra

```
x=5
y=6
z=x+y
```

In algebra we use letters (like x) to hold values (like 5).
From the expression z=x+y above, we can calculate the value of z to be 11.
In PHP these letters are called variables.

2. PHP Variables

As with algebra, PHP variables can be used to hold values (x=5) or expressions (z=x+y).
Variable can have short names (like x and y) or more descriptive names (age, car name, total volume).
Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name should not contain spaces
- Variable names are case sensitive ($y and $Y are two different variables)

3. Creating (Declaring) PHP Variables

PHP has no command for declaring a variable.
A variable is created the moment you first assign a value to it:

```
$txt="Hello world!";
$x=5;
```

After the execution of the statements above, the variable txt will hold the value Hello world!, and the variable x will hold the value 5.
Note: When you assign a text value to a variable, put quotes around the value.

4. PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.
PHP automatically converts the variable to the correct data type, depending on its value.
In a strongly typed programming language, we will have to declare (define) the type and name of the variable before using it.

5. PHP Variable Scopes

The scope of a variable is the part of the script where the variable can be referenced/ used.
PHP has four different variable scopes:

- local
- global
- static
- parameter

6. Local Scope

A variable declared within a PHP function is local and can only be accessed within that function:

Example:

```
<? Php
$x=5; // global scope
function myTest()
{
echo $x; // local scope
}
myTest();
?>
```

The script above will not produce any output because the echo statement refers to the local scope variable $x, which has not been assigned a value within this scope.
You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
Local variables are deleted as soon as the function is completed.

7. Global Scope

A variable that is defined outside of any function has a global scope.
Global variables can be accessed from any part of the script, EXCEPT from within a function.
To access a global variable from within a function, use the global keyword:

Example:

```
<? Php
$x=5; // global scope
$y=10; // global scope
function myTest()
{
global $x,$y;
```

```
$y=$x+$y;
}
myTest();
echo $y; // outputs 15
?>
```

PHP also stores all global variables in an array called $GLOBALS [index]. The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this:

```
<?php
$x=5;
$y=10;
function myTest()
{
$GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y'];
}
myTest();
echo $y;
?>
```

8. Static Scope

When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted. To do this, use the static keyword when you first declare the variable:

Example:

```
<?php
function myTest()
{
static $x=0;
echo $x;
$x++;
}
myTest();
myTest();
myTest();
?>
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

9. Parameter Scope

A parameter is a local variable whose value is passed to the function by the calling code.
Parameters are declared in a parameter list as part of the function declaration:

Example:

```
<? Php
function myTest($x)
{
echo $x;
}
myTest(5);
?>
```

Parameters are also called arguments. We will discuss it in more details in our PHP functions chapter.

12. PHP STRING VARIABLES

A string variable is used to store and manipulate text.

1. String Variables in PHP

String variables are used for values that contain characters.
After we have created a string variable we can manipulate it.
A string can be used directly in a function or it can be stored in a variable.
In the example below, we create a string variable called txt, then we assign the text "Hello world!" to it. Then we write the value of the txt variable to the output:

Example:

```
<? Php
$txt="Hello world!";
echo $txt;
?>
```

2. The PHP Concatenation Operator

There is only one string operator in PHP.
The concatenation operator (.) Is used to join two string values together.
The example below shows how to concatenate two string variables together:

Example:

```
<? Php
$txt1="Hello world!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be: Hello world! What a nice day!

13. PHP IF...ELSE STATEMENTS

Conditional statements are used to perform different actions based on different conditions.

1. PHP Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.
In PHP we have the following conditional statements:

- if statement - executes some code only if a specified condition is true
- if...else statement - executes some code if a condition is true and another code if the condition is false
- if...else if....else statement - selects one of several blocks of code to be executed
- switch statement - selects one of many blocks of code to be executed

2 .PHP - The if Statement

The if statement is used to execute some code only if a specified condition is true.

Syntax:

```
if (condition)
  {
  code to be executed if condition is true;
  }
```

The example below will output "Have a good day!" if the current time is less than 20:

Example:

```
<? Php
$t=date("H");
if ($t<"20")
  {
  echo "Have a good day!";
  }
?>
```

3. PHP - The if...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is false.

Syntax:

```
if (condition)
 {
  code to be executed if condition is true;
 }
else
 {
  code to be executed if condition is false;
 }
```

Example:

```
<?php
$t=date("H");
if ($t<"20")
  {
  echo "Have a good day!";
  }
else
  {
  echo "Have a good night!";
  }
?>
```

4. PHP - The if...else if....else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax:

```
if (condition)
  {
  code to be executed if condition is true;
  }
else if (condition)
  {
  code to be executed if condition is true;
 }
else
  {
  code to be executed if condition is false;
 }
```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!"

Example:

```
<?php
$t=date("H");
if ($t<"10")
  {
  echo "Have a good morning!";
  }
else if ($t<"20")
  {
  echo "Have a good day!";
  }
else
  {
  echo "Have a good night!";
  }
?>
```

14. PHP - THE SWITCH STATEMENT

The switch statement is used to perform different actions based on different conditions.

Use the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch (n)
{
case label1:
  code to be executed if n=label1;
```

```
 break;
case label2:
 code to be executed if n=label2;
 break;
default:
 code to be executed if n is different from both label1 and label2;
}
```

This is how it works: First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.
Example:

```
<?php
$favcolor="red";
switch ($favcolor)
{
case "red":
 echo "Your favourite color is red!";
 break;
case "blue":
 echo "Your favourite color is blue!";
 break;
case "green":
 echo "Your favourite color is green!";
 break;
default:
 echo "Your favourite color is neither red, blue, or green!";
}
?>
```

15. PHP ARRAYS

An array stores multiple values in one single variable:
Example:

```
<? Php
$cars=array ("Volvo","BMW","Toyota");
echo "I like " . $cars [0]. ", " . $cars [1]. "And ". $cars [2]. ".";
?>
```

1. What is an Array?

An array is a special variable, which can hold more than one value at a time.
If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Volvo";
$cars2="BMW";
$cars3="Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
The solution is to create an array!
An array can hold many values under a single name, and you can access the values by referring to an index number.

2. Create an Array in PHP

In PHP, the array () function is used to create an array:

```
array ();
```

In PHP, there are three types of arrays:

- Indexed arrays - Arrays with numeric index
- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

3. PHP Indexed Arrays

There are two ways to create indexed arrays:
The index can be assigned automatically (index always starts at 0):

```
$cars=array ("Volvo","BMW","Toyota");
```

Or the index can be assigned manually:

```
$cars [0] ="Volvo";
$cars [1] ="BMW";
$cars [2] ="Toyota";
```

The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:
Example

```
<? Php
$cars=array ("Volvo","BMW","Toyota");
Echo "I like". $cars [0]. ", " . $cars [1]. "And ". $cars [2]. ".";
?>
```

4. Get the Length of an Array - The count () Function

The count () function is used to return the length (the number of elements) of an array:
Example

```
<? Php
$cars=array ("Volvo","BMW","Toyota");
echo count($cars);
?>
```

5. Loop through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:
Example

```
<? Php
$cars=array ("Volvo","BMW","Toyota");
$arrlength=count ($cars);

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

```
?>
```

6. PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.
There are two ways to create an associative array:

```
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
```
Or:
```
$age ['Peter'] ="35";
$age ['Ben'] ="37";
$age ['Joe'] ="43";
```

The named keys can then be used in a script:
Example
```
<? Php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

7. Loop through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:
Example
```
<? Php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
foreach($age as $x=>$x_value)
  {
  echo "Key=" . $x. ", Value=". $x_value;
  echo "<br>";
  }
?>
```

8. PHP Sorting Arrays
The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.

9. PHP - Sort Functions for Arrays
In this chapter, we will go through the following PHP array sort functions:

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

10. Sort Array in Ascending Order - sort()

The following example sorts the elements of the $cars array in ascending alphabetical order:

Example:

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

The following example sorts the elements of the $numbers array in ascending numerical order:
Example
```
<?php
$numbers=array(4,6,2,22,11);
sort($numbers);
?>
```

11. Sort Array in Descending Order - rsort()
The following example sorts the elements of the $cars array in descending alphabetical order:
Example
```
<? Php
$cars=array ("Volvo","BMW","Toyota");
rsort($cars);
?>
```

The following example sorts the elements of the $numbers array in descending numerical order:
Example
```
<? Php
$numbers=array (4, 6, 2, 22, 11);
rsort($numbers);
?>
```

12. Sort Array in Ascending Order, According to Value - asort()
The following example sorts an associative array in ascending order, according to the value:
Example
```
<? Php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
asort($age);
?>
```

13. Sort Array in Ascending Order, According to Key - ksort()

The following example sorts an associative array in ascending order, according to the key:

Example:

```
<? Php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
ksort($age);
?>
```

14. Sort Array in Descending Order, According to Value - arsort()

The following example sorts an associative array in descending order, according to the value:

Example

```
<? Php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
arsort($age);
?>
```

16. PHP LOOPING - WHILE LOOPS

Loops execute a block of code a specified number of times, or while a specified condition is true.

1. PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In PHP, we have the following looping statements:

While - loops through a block of code while a specified condition is true

Do...while - loops through a block of code once, and then repeats the loop as long as a specified condition is true

For - loops through a block of code a specified number of times

foreach - loops through a block of code for each element in an array.

2. The while Loop

The while loop executes a block of code while a condition is true.

Syntax:

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

Example:

The example below first sets a variable i to 1 ($i=1;).
Then, the while loop will continue to run as long as i is less than, or equal to 5. I will increase by 1 each time the loop runs:

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

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

3. The do...while Statement

The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.

Syntax:

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

Example

The example below first sets a variable I to 1 ($I=1 ;).
Then, it starts the do...while loop. The loop will increment the variable i with 1, and then write some output. Then the condition is checked (is i less than, or equal to 5), and the loop will continue to run as long as i is less than, or equal to 5:

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

Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

17. PHP LOOPING - FOR LOOPS

Loops execute a block of code a specified number of times, or while a specified condition is true.

1. The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax:

```
for (init; condition; increment)
  {
  code to be executed;
  }
```

Parameters:
Init: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
Condition: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
Increment: Mostly used to increment a counter (but can be any code to be executed at the end of the iteration)
Note: The init and increment parameters above can be empty or have multiple expressions (separated by commas).
Example
The example below defines a loop that starts with i=1. The loop will continue to run as long as the variable i is less than, or equal to 5. The variable i will increase by 1 each time the loop runs:

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

Output:
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

2. The foreach Loop

The foreach loop is used to loop through arrays.
Syntax:

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

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

Example:

The following example demonstrates a loop that will print the values of the given array:

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

Output:
One
two
three

18. PHP FUNCTIONS

The real power of PHP comes from its functions.
In PHP, there are more than 700 built-in functions.

1. PHP Built-in Functions
For a complete reference and examples of the built-in functions, please visit our PHP Reference.

2. PHP Functions
In this chapter we will show you how to create your own functions.
To keep the script from being executed when the page loads, you can put it into a function.
A function will be executed by a call to the function.
You may call a function from anywhere within a page.

3. Create a PHP Function
A function will be executed by a call to the function.

Syntax:

```
function functionName()
{
code to be executed;
}
```

PHP function guidelines:
Give the function a name that reflects what the function does
The function name can start with a letter or underscore (not a number)

Example

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

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

```
<?php
function writeName()
{
echo "Kai Jim Ram";
}
echo "My name is ";
writeName();
?>
</body>
</html>
```

Output:
My name is Kai Jim Ram

4. PHP Functions - Adding parameters
To add more functionality to a function, we can add parameters. A parameter is just like a variable.
Parameters are specified after the function name, inside the parentheses.

Example 1

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

```
<html>
<body>
<?php
function writeName($fname)
{
echo $fname . “Ram.<br>";
}
echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Helm");
echo "My brother's name is ";
writeName("Stale");
?>
</body>
</html>
```

Output:
My name is Kai Jim Ram.
My sister's name is Helm Ram.
My brother's name is Stale Ram.

Example 2
The following function has two parameters:

```
<html>
<body>
<?php
function writeName($fname,$punctuation)
{
echo $fname . “Ram”. $ Punctuation. "<br>";
}
echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Helm","!");
echo "My brother's name is ";
writeName("Santo","?");
?>
</body>
</html>
```

Output:
My name is Kai Jim Ram.
My sister's name is Helm Ram!
My brother's name is Santo Ram?

5. PHP Functions - Return values
To let a function return a value, use the return statement.
Example

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

Output:
1 + 16 = 17

19. PHP FORMS AND USER INPUT

The PHP $_GET and $_POST variables are used to retrieve information from forms, like user input.

1. PHP Form Handling
The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

Example

The example below contains an HTML form with two input fields and a submit button:

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

When a user fills out the form above and clicks on the submit button, the form data is sent to a PHP file, called "welcome.php":

"welcome.php" looks like this:
<html>
<body>
Welcome <?php echo $_POST["fname"]; ?>!
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
Output could be something like this:
Welcome John!
You are 28 years old.

The PHP $_GET and $_POST variables will be explained in the next chapters.

2. Form Validation

User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and reduces the server load.

You should consider server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page. The user will then get the error messages on the same page as the form. This makes it easier to discover the error.

20. PHP $_GET VARIABLE

In PHP, the predefined $_GET variable is used to collect values in a form with method="get".

1. The $_GET Variable

The predefined $_GET variable is used to collect values in a form with method="get"
Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.
Example

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

When the user clicks the "Submit" button, the URL sent to the server could look something like this:
http://www.w3schools.com/welcome.php?fname=Peter&age=37
The "welcome.php" file can now use the $_GET variable to collect form data (the names of the form fields will automatically be the keys in the $_GET array):
Welcome <? Php echo $_GET ["fname"] ;?>.
You are <? Php echo $_GET ["age"] ;?> years old!

2. When to use method="get"?

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

21. PHP $_POST FUNCTION

1. The $_POST Variable

The predefined $_POST variable is used to collect values from a form sent with method="post".
Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
Note: However, there is an 8 MB max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file).
Example

<form action="welcome.php" method="post">
Name: <input type="text" name="fname">
Age: <input type="text" name="age">
<input type="submit">
</form>
When the user clicks the "Submit" button, the URL will look like this:
http://www.w3schools.com/welcome.php
The "welcome.php" file can now use the $_POST variable to collect form data (the names of the form fields will automatically be the keys in the $_POST array):
Welcome <? Php echo $_POST ["fname"] ;?>!
You are <? Php echo $_POST ["age"] ;?> years old.

2. When to use method="post"?

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

3. The PHP $_REQUEST Variable

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

The $_REQUEST variable can be used to collect form data sent with both the GET and POST methods.
Example
Welcome <? Php echo $_REQUEST ["fname"] ;?>!
You are

<? Php echo $_REQUEST ["age"] ;?> years old.

REFERENCES

- Lerdorf, Rasmus (2007-04-26). "PHP on Hormones – history of PHP presentation by Rasmus Lerdorf given at the MySQL Conference in Santa Clara, California". The Conversations Network. Retrieved 2009-12-11.
- "PHP Usage Stats". Retrieved 2013-04-01.
- "History of PHP and related projects". The PHP Group. Retrieved 2008-02-25.
- "History of PHP". php.net.
- PHP Manual: Preface, www.php.net
- "Introduction: What can PHP do?". PHP Manual. Retrieved 2009-03-05.
- "GPL-Incompatible, Free Software Licenses". Various Licenses and Comments about Them. Free Software Foundation. Retrieved 2012-03-11.
- "Embedding PHP in HTML". O'Reilly. 2001-05-03. Retrieved 2008-02-25.
- Lerdorf, Rasmus (2007-04-26). "PHP on Hormones" (mp3). The Conversations Network. Retrieved 2009-06-22.
- Lerdorf, Rasmus (2007). "Slide 3". Slides for 'PHP on Hormones' talk. The PHP Group. Retrieved 2009-06-22.
- "Lerdorf, Rasmus (June 8, 1995). "Announce: Personal Home Page Tools (PHP Tools)". Retrieved 7 June 2011.
- Lerdorf, Rasmus (1995-06-08). "Announce: Personal Home Page Tools (PHP Tools)".comp.infosystems.www.authoring.cgi. Web link. Retrieved 2006-09-17.
- "Zend Engine version 2.0: Feature Overview and Design".Zend Technologies Ltd. Retrieved 2006-09-17.
- "php.net 2007 news archive". The PHP Group. 2007-07-13. Retrieved 2008-02-22.
- Kerner, Sean Michael (2008-02-01). "PHP 4 is dead—Long Live PHP 5". Internet News. Retrieved 2008-03-16.
- Trachtenberg, Adam (2004-07-15). "Why PHP 5 Rocks!". O'Reilly. Retrieved 2008-02-22.
- "Late Static Binding in PHP". Digital Sandwich. 2006-02-23. Retrieved 2008-03-25.
- "Static Keyword". The PHP Group. Retrieved 2008-03-25.
- "PHP 6". The PHP project. Retrieved 2010-03-27.
- "Using Register Global". PHP. Retrieved 2008-04-04.
- "Prepare for PHP 6". Core PHP. 2005-11-23. Retrieved 2008-03-24.
- "PHP 5.3 migration guide". The PHP project. Retrieved 2009-07-03.
- "GoPHP5".
- GoPHP5. "PHP projects join forces to Go PHP 5" (PDF).GoPHP5 Press Release. Retrieved 2008-02-23.
- "GoPHP5". GoPHP5. Retrieved 2008-02-22.
- The PHP Group. "PHP For Windows snapshots". PHP Windows Development Team. Retrieved 2009-05-25.
- "PHP: PHP 4 Change Log". The PHP Group. 2008-01-03. Retrieved 2008-02-22.
- " Using PHP from the command line". PHP Manual. The PHP Group. Retrieved 2009-09-11.
- "PHP: PHP 5 Change Log". The PHP Group. 2007-11-08. Retrieved 2008-02-22.
- "PHP manual: PDO". The PHP Group. 2011-11-15. Retrieved 2011-11-15.
- "Built-in web server". Retrieved March 26, 2012.
- "PHP 6: Features, Release Date, Hosting and Download". Retrieved May 6, 2011.
- "PHP: Release Process". 2011-07-23. Retrieved 2008-02-22.
- "PHP: Basic syntax". The PHP Group. Retrieved 2008-02-22.
- "Your first PHP-enabled page". The PHP Group. Retrieved 2008-02-25.

- “Bray, Tim; et al (26 November 2008). "Processing Instructions". Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C. Retrieved 2009-06-18.
- “Variables". The PHP Group. Retrieved 2008-03-16.
- “Instruction separation". The PHP Group. Retrieved 2008-03-16.
- “Comments". The PHP Group. Retrieved 2008-03-16.
- “Integers in PHP, running with scissors, and portability". MySQL Performance Blog. March 27, 2007. Retrieved 2007-03-28.
- “Types". The PHP Group. Retrieved 2008-03-16.
- “Strings". The PHP Group. Retrieved 2008-03-21.
- "SPL — Standard PHP Library". PHP.net. March 16, 2009. Retrieved 2009 – 03 -1 6.
- "Problems with PHP". Retrieved 20 December 2010.
- "PHP.NET: Process Control". Retrieved 2009-08-06.
- "Functions". The PHP Group. Retrieved 2008-03-16.
- "PHP 5 Object References". mjtsai. Retrieved 2008-03-16.
- "Classes and Objects (PHP 5)". The PHP Group. Retrieved 2008-03-16.
- "Object cloning". The PHP Group. Retrieved 2008-03-16.
- "Visibility". PHP Manual. Theserverpages.com. 2005-05-19. Retrieved 2010-08-26.
- Gervasio, Alejandro. "More on Private Methods with PHP 5 Member Visibility". devshed. com. Retrieved 24 November 2010.
- "Visibility in PHP: Public, Private and Protected". Aperiplus.sourceforge.net. Retrieved 2010-08-26.
- Favre, Nicolas (2010-02-16). "A review of PHP compilers and their outputs". Technow.owlient.eu. Retrieved 2010-05-20.More than one of |author= and |last= specified (help)[dead link]
- "How do computer languages work?". Retrieved 2009-11-04.
- (Gilmore 2006, p. 43)
- "PHP Accelerator 1.2 (page 3, Code Optimisation)" (PDF). Nick Lindridge. Retrieved 2008-03-28.
- "[PHP-DEV] APC in 5.4". Retrieved March 6, 2012.
- "PHP Internals discussion". Retrieved July 12, 2011.
- "The PHP License, version 3.01". Retrieved 2010-05-20.
- "GPL-Incompatible, Free Software Licenses". Various Licenses and Comments about Them. Free Software Foundation. Retrieved 2011-01-03.
- "PHP Function List". The PHP Group. Retrieved 2008-02-25.
- Cross Reference: /PHP_5_4/ext/standard/
- "Developing Custom PHP Extensions". devnewz. 2002-09-09. Archived from the original on 2008-02-18. Retrieved 2008-02-25.
- http://www.php.net/manual/en/intro.pdo.php
- O'Reilly Networks - ONLamp
- http://www-128.ibm.com/developerworks/library/os-php-dbmistake/index.html
- IBM Redbooks
- "News Archive: PHP 5.3.3 Released!". 2010-07-22.
- "PHP Manual Image Processing and GD;". Php.net. Retrieved 2011-04-09.
- Archived June 11, 2008 at the Wayback Machine[not in citation given]
- "PHP and MySQL". University of Alabama. Archived from the original on 2008-02-28. Retrieved 2008-02-25.
- "PHP Server-Side Scripting Language". Indiana University. 2007-04-04. Retrieved 2008-02-25.
- "JavaServer Pages Technology — JavaServer Pages Comparing Methods for Server-Side Dynamic Content White Paper". Sun Microsystems. Retrieved 2008-02-25.
- "PHP: PHP Usage Stats". SecuritySpace. 2007-04-01. Retrieved 2008-02-24.
- "Usage of server-side programming languages for websites". W3Techs. 2010-10-29. Retrieved 2010-10-29.
- "PHP and Perl crashing the enterprise party".
- "Manual: Installation requirements#PHP". Media Wiki. 2010-01-25. Retrieved 2010-02-26. "PHP is the programming language in which Media Wiki is written [...]"
- "System requirements of Silver Stripe". Retrieved 2012-03-05. "Silver Stripe requires PHP 5.2+"
- "About Word Press". Retrieved 2010-02-26. "Word Press was [...] built on PHP"

