1. **Explain php programing beyond definition?**

PHP (Hypertext Preprocessor) is known as a general-purpose scripting language that can be used to develop dynamic and interactive websites.

It was among the first server-side languages that could be embedded into HTML,

making it easier to add functionality to web pages without needing to call external files for data.

2. **Why do we need to use php programming?**

PHP allows web developers to create dynamic content and interact with databases

It’s easy to learn and use

It’s open source (and therefore free!)

It enjoys strong community support It’s fast and secure

It is well connected with databases

There’s a lot of legacy code

3.What is the latest php version we have today and list the updated features for the latest 3 release?

8.2

Release Date2022-12-08

PHP 8.2 is the latest PHP version which brings read only classes,

DNF types, null, false, and true types, sensitive parameter redaction support,

a new random extension, and several new features along with a few deprecations.

8.1

Release Date2021-11-25

PHP 8.1, released in 2021, brings major new features such as Enums,

Fibers, never return type, Intersection Types, read only properties, and more,

while ironing out some of its undesired legacy features by deprecating them.

8.0

Release Date2020-11-26

PHP 8.0, on the 25th year of PHP history, brings several important features such as Union Types,

JIT, Constructor Property Promotion, Match Syntax, Named Parameters, and several more performance, syntax, and quality-of-life improvements.

4.What is different between new release vs stable release of a software product?

A New release is the distribution of the final version or the newest version of a software application.

A software release may be public or private and generally signifies the unveiling of a new or upgraded version of the application.

A stable release is a version that has been tested as thoroughly as possible and is as reliable as we can make it.

It does not have all the new features of a beta release and it does not have the latest fixes for problems

5.What are the main features of php programming?

Simplicity

Flexibility

Objective oriented

Interpreted language

Efficient

Fast Performance

Free and open-source

Case-sensitive

Security

Platform independent

Loosely typed language

Real-time access monitoring

Error reporting and handling

Memory and CPU use information

Active community support

6.With a help of examples explain why php is case sensitive?

PHP is a unique programming language in terms of case sensitivity.

In PHP, variables and constants are case sensitive, while functions are not case sensitive.

PHP classes are a mix between variables and functions, so they are partially case-sensitive.

Let’s see some practical example below:

// you can create two variables like this:

$num = 99;

$NUM = 20;

echo $num; // 99

echo "\n".$NUM; // 20

// but you can't have two functions like this:

function greetings(){

echo "Hello World!";

}

// Fatal error: Cannot redeclare GREETINGS()

function GREETINGS(){

echo "Hello World!";

}

As you can see in the example above, the variables $num and $NUM can have different values.

But when you declare two functions with the same name, PHP produces a fatal error: cannot redeclare the function.

This happens even though the function names use different letter cases.

7.What and why do we use comments while writing php codes, With a help of example explain

different types of php comments?

Comments in PHP

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

Let others understand your code

Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did.

Comments can remind you of what you were thinking when you wrote the code

PHP supports several ways of commenting:

Example 1

Syntax for single-line comments:

<!DOCTYPE html>

<html>

<body>

<?php

// This is a single-line comment

# This is also a single-line comment

?>

</body>

</html>

Example 2

Syntax for multiple-line comments:

<!DOCTYPE html>

<html>

<body>

<?php

/\*

This is a multiple-lines comment block

that spans over multiple

lines

\*/

?>

</body>

</html>

Example 3

Using comments to leave out parts of the code:

<!DOCTYPE html>

<html>

<body>

<?php

// You can also use comments to leave out parts of a code line

$x = 5 /\* + 15 \*/ + 5;

echo $x;

?>

</body>

</html>

8.Differentiate with real example the following php output functions:

a. Echo() vs print()

b. Print() vs printf()

c. Printf() vs print\_r()

d. Print\_r vs var\_dump()

a. echo and print are more or less the same.

They are both used to output data to the screen.

The differences are small:

echo has no return value while print has a return value of 1 so it can be used in expressions.

echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

Example

<?php

echo "<h2>PHP is Fun!</h2>";

echo "Hello world!<br>";

echo "I'm about to learn PHP!<br>";

echo "This ", "string ", "was ", "made ", "with multiple parameters.";

?>

Display Variables

The following example shows how to output text and variables with the echo statement:

Example

<?php

$txt1 = "Learn PHP";

$txt2 = "W3Schools.com";

$x = 5;

$y = 4;

echo "<h2>" . $txt1 . "</h2>";

echo "Study PHP at " . $txt2 . "<br>";

echo $x + $y;

?>

The PHP print Statement

The print statement can be used with or without parentheses: print or print().

Display Text

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

Example

<?php

print "<h2>PHP is Fun!</h2>";

print "Hello world!<br>";

print "I'm about to learn PHP!";

?>

Display Variables

The following example shows how to output text and variables with the print statement:

Example

<?php

$txt1 = "Learn PHP";

$txt2 = "W3Schools.com";

$x = 5;

$y = 4;

print "<h2>" . $txt1 . "</h2>";

print "Study PHP at " . $txt2 . "<br>";

print $x + $y;

?>

b. PHP print() Function

Example

Write some text to the output:

<?php

print "Hello world!";

?>

PHP printf() Function

Example

Output a formatted string:

<?php

$number = 9;

$str = "Beijing";

printf("There are %u million bicycles in %s.",$number,$str);

?>

c. PHP printf() Function

Example

Output a formatted string:

<?php

$number = 9;

$str = "Beijing";

printf("There are %u million bicycles in %s.",$number,$str);

?>

PHP print\_r() Function

Example

Print the information about some variables in a more human-readable way:

<?php

$a = array("red", "green", "blue");

print\_r($a);

echo "<br>";

$b = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

print\_r($b);

?>

d. PHP var\_dump() Function

Example

Dump information about different variables:

<?php

$a = 32;

echo var\_dump($a) . "<br>";

$b = "Hello world!";

echo var\_dump($b) . "<br>";

$c = 32.5;

echo var\_dump($c) . "<br>";

$d = array("red", "green", "blue");

echo var\_dump($d) . "<br>";

$e = array(32, "Hello world!", 32.5, array("red", "green", "blue"));

echo var\_dump($e) . "<br>";

Example: Say we have got the following array and we want to display its contents.

$arr = array ('xyz', false, true, 99, array('50'));

// Dump two variables

echo var\_dump($a, $b) . "<br>";

?>

var\_dump() function - Displays values and types

array(5) {

[0]=>

string(3) "xyz"

[1]=>

bool(false)

[2]=>

bool(true)

[3]=>

int(100)

[4]=>

array(1) {

[0]=>

string(2) "50"

}

}

print\_r() function - Displays human-readable output

Array

(

[0] => xyz

[1] =>

[2] => 1

[3] => 99

[4] => Array

(

[0] => 50

)

)

PHP print\_r() Function

Example

Print the information about some variables in a more human-readable way:

<?php

$a = array("red", "green", "blue");

print\_r($a);

echo "<br>";

$b = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

print\_r($b);

?>

9. List and Describe different datatype we have in php by categorizing them in scalar, compound

and special datatypes.

PHP Data Types: Scalar Types

In simple words, a variable is called scalar type if it holds singular value only. There are 4 scalar data types in PHP.

boolean

integer

float

string

PHP Data Types: Compound Types

In contrast to Scalar data types, a variable is called compound if it holds multiples values within. There are 2 compound data types in PHP.

array

object

PHP Data Types: Special Types

There are 2 special data types in PHP.

resource

NULL

10. What is php variable, list the variable naming rules you have to obey while defining a variable inphp?

Variables are "containers" for storing information.

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, A variable name must start with a letter or the underscore character.

A variable name cannot start with a number.

A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )

followed by the name of the variable:

Example

<?php

$txt = "Hello world!";

$x = 5;

$y = 10.5;

Rules for PHP variables:

A variable starts with the $ sign, followed by the name of the variable

A variable name must start with a letter or the underscore character

A variable name cannot start with a number

A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )

Variable names are case-sensitive ($age and $AGE are two different variables)

11.List and explain at least 10 super global variables?

Superglobals are built-in variables that are always available in all scopes.

$GLOBALS

$\_SERVER

$\_GET

$\_POST

$\_FILES

$\_COOKIE

$\_SESSION

$\_REQUEST

$\_ENV

**PHP $GLOBALS**

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS:

Example

<?php

$x = 75;

$y = 25;

 function addition() {

  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];

}

 addition();

echo $z;

?>

In the example above, since z is a variable present within the $GLOBALS array, it is also accessible from outside the function!

**PHP $\_SERVER**

$\_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $\_SERVER:

Example

<?php

echo $\_SERVER['PHP\_SELF'];

echo "<br>";

echo $\_SERVER['SERVER\_NAME'];

echo "<br>";

echo $\_SERVER['HTTP\_HOST'];

echo "<br>";

echo $\_SERVER['HTTP\_REFERER'];

echo "<br>";

echo $\_SERVER['HTTP\_USER\_AGENT'];

echo "<br>";

echo $\_SERVER['SCRIPT\_NAME'];

?>

$\_SERVER['SCRIPT\_URI'] Returns the URI of the current page

**PHP $\_REQUEST**

PHP $\_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $\_REQUEST to collect the value of the input field:

Example

<html>

<body>

<form method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">

Name: <input type="text" name="fname">

<input type="submit">

</form>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// collect value of input field

  $name = $\_REQUEST['fname'];

if (empty($name)) {

echo "Name is empty";

  } else {

echo $name;

  }

}

?>

</body>

</html>

**PHP $\_POST**

PHP $\_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $\_POST is also widely used to pass variables.

Example

<html>

<body>

<form method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">

Name: <input type="text" name="fname">

<input type="submit">

</form>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

  // collect value of input field

$name = $\_POST['fname'];

if (empty($name)) {

echo "Name is empty";

  } else {

    echo $name;

  }

}

?>

</body>

</html>

**PHP $\_GET**

PHP $\_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$\_GET can also collect data sent in the URL.

Assume we have an HTML page that contains a hyperlink with parameters:

<html>

<body>

<a href="test\_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>

</html>

When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test\_get.php", and you can then access their values in "test\_get.php" with $\_GET.

The example below shows the code in "test\_get.php":

Example

<html>

<body>

<?php

echo "Study " . $\_GET['subject'] . " at " . $\_GET['web'];

?>

</body>

</html>

**$\_FILES**

When a user submits a form which supports file uploads, the $\_FILES superglobal variable will be populated with the information of the files that are uploaded. It’s a two-dimensional array which holds the following attributes of uploaded files:

name: name of the file

type: type of the file

size: size of the file

tmp\_name: server path of the uploaded file

error: error if file upload has failed

Let’s have a look at the following example to see how file uploads work in PHP.

<!DOCTYPE html>

<html>

<body>

<div>

<form action="upload.php" method="POST" enctype="multipart/form-data">

Upload File:<input type="file" name="document">

<input type="submit" value="Upload" name="Submit">

</form>

</div>

</body>

</html>

In the above example, we’ve set the enctype attribute to multipart/form-data, and thus it enables file uploads. When a user submits a file, you can use the $\_FILES superglobal variable to access the uploaded file information, as shown in the following snippet.

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

if (isset($\_FILES["document"]) && $\_FILES["document"]["error"] === UPLOAD\_ERR\_OK) {

$fileName = $\_FILES["document"]["name"];

$fileType = $\_FILES["document"]["type"];

$fileSize = $\_FILES["document"]["size"];

$fileTmpName = $\_FILES["document"]["tmp\_name"];

$fileNameCmps = explode(".", $fileName);

$fileExtension = strtolower(end($fileNameCmps));

// sanitize file-name

$newFileName = md5(time() . $fileName) . '.' . $fileExtension;

// check if file has one of the following extensions

$allowedfileExtensions = array('jpg', 'gif', 'png', 'zip', 'txt', 'xls', 'doc');

if (in\_array($fileExtension, $allowedfileExtensions))

{

$uploadFileDir = './uploaded\_files/';

$dest\_path = $uploadFileDir . $newFileName;

if(move\_uploaded\_file($fileTmpPath, $dest\_path))

{

$message ='File is successfully uploaded.';

}

else

{

$message = 'There was some error moving the file to upload directory. Please make sure the upload directory is writable by web server.';

}

}

else

{

$message = 'Upload failed. Allowed file types: ' . implode(',', $allowedfileExtensions);

}

echo $message;

} else {

echo 'There was some error uploading the file:' . $\_FILES["document"]["error"];

}

}

?>

The important thing is that the $\_FILES["document"]["tmp\_name"] variable contains the path of the uploaded file, which you could use to move the uploaded file to the desired location. If there’s an error, the $\_FILES["document"]["error"] variable will be populated with it.

Advertisement

**$\_COOKIE**

As the name suggests, the $\_COOKIE superglobal variable is used to read cookies that are available to the current script. Basically, it allows you to access cookies that are already set by the setcookie function in PHP. The $\_COOKIE variable is an associate array which holds all cookie variables that are sent via HTTP cookies.

Let’s assume that you’ve already created the lastVisitedSection cookie with the following snippet.

<?php

setcookie('lastVisitedSection', 'codeTutsPlus', time() + 3600, "/");

?>

Now, you can access the lastVisitedSection cookie, as shown in the following snippet.

<?php

if(isset($\_COOKIE['lastVisitedSection'])) {

echo 'Last visited section:' . htmlspecialchars($\_COOKIE['lastVisitedSection']);

} else {

echo 'We encourage you to explore different sections of tutsplus.com!';

}

?>

So that’s how you can access cookies in your script.

**$\_SESSION**

If you’ve already worked with sessions in PHP, you'll be aware of the $\_SESSION superglobal variable. A session variable allows you to share information across the different pages of a single site or app—thus it helps maintain state. The $\_SESSION variable holds an associative array of session variables that are available to the current script.

Let’s have a look at the following example, which demonstrates how to set and get a session variable.

<?php

session\_start();

// set a session variable

$\_SESSION['loggedInUserName'] = 'John';

// get a session variable later in the script or in subsequent requests

If (isset($\_SESSION['loggedInUserName'])) {

echo 'Hello, '. $\_SESSION['loggedInUserName'];

} else {

echo 'Login Now!';

}

?>

The important thing is that the session\_start function must be called at the beginning of the script to start a session and initialize the $\_SESSION variable.

**$\_REQUEST**

The $\_REQUEST superglobal variable is an associative array which holds HTTP request variables. Basically, it’s a combination of the$\_GET, $\_POST, and $\_COOKIE superglobal variables. And thus, it’s convenient to use the $\_REQUEST variable, specifically if you don’t want to use the aforementioned superglobal variables.

The presence and order of variables in this array depend on the values of the request\_order and variables\_order configuration directives in the php.ini file. You should be always careful when you use the $\_REQUEST variable, and in fact, it’s recommended that you go with $\_GET, $\_POST, and $\_COOKIE instead of using this superglobal variable.

**$\_ENV**

The $\_ENV superglobal variable is an associative array of variables that are passed to the script by the environment method. It’s useful when you want to set different values for different environments like local, staging, and production. In your application, you would have database credentials and configuration variables that are different for each environment, and thus, you could use the $\_ENV variable to access them in your script since they are initialized dynamically. There are different ways that you could use to set the environment variables.

In PHP, you can use the putenv function to initialize it, as shown in the following snippet.

putenv('HOSTNAME=localhost');

On the other hand, if you want to set an environment variable in the Apache virtual host file, use:

SetEnv HOSTNAME localhost

You should always prefer the getenv function to get the value of an environment variable.

$hostname = getenv('HOSTNAME');

You should always use the getenv function to retrieve the value of an environment variable,

since the $\_ENV variable might be empty if you haven’t enabled it via the variables\_order configuration directive in the php.ini file.

The variables\_order configuration directive defines the order in which the EGPCS (Environment, Get, Post, Cookie, and Server) variables will be initialized.

# References

<https://www.w3schools.com/php/php_superglobals.asp>

<https://www.sitesbay.com/php/php-features-of-php>

<https://codeberryschool.com/blog/en/beginners-guide-to-php-programming/>