Latest PHP Interview Questions for Quick Preparation

PHP Interview Questions & Answers.

Q-1. What does PEAR signify in PHP?

Answer.
PEAR is an acronym for ***PHP Extension and Application Repository***. It was Stig S. Bakken who introduced it in 1999 with the following features.

- A structured library of open-sourced code for PHP users.
- Simple code distribution and package management.
- A standard style for code written in PHP.
- The PHP Foundation Classes (PFC).
- The PHP Extension Community Library (PECL).
- A website, mailing lists, and download mirrors to support the PHP/PEAR community. Thus, PEAR is a community-driven project with the <PEAR Group> as its governing body.
- It provides a command-line interface that can come into use for installing the packages on demand.

Q-2. What is the difference between $name and $$name?

Answer.
$name is a variable whereas **$$name** is a reference to the variable.

Let’s see an example.

```
$name "Welcome";

$Welcome "To";

$To "TechBeamers";

echo $name;
/*prints Welcome*/
echo $name;
/*prints To*/
echo $$name;
/*prints TechBeamers*/
```

This code sample prints the following message <WelcomeToTechBeamers>.

Q-3. How do you embed PHP code in an HTML page?

Answer.
In an HTML page, all PHP code must be enclosed within either of the three special markup tags recognized by the PHP Parser.

<?php PHP code goes here **?>**

- **<?php** PHP code goes here **?>**
- **<?** PHP code goes here **?>**
- **<script language="php">** PHP code goes here **</script>**

Amongst these **<?php…?>** is the most commonly used tag.

We can chain as many statements as required separating them with semicolons as shown below.

```
<?php

  echo "Hello World";

  echo "A second statement";

?>
```

Q-4. What is the name of the scripting engine of PHP?

Answer.
ZEND Engine 2 is the name of the scripting engine that powers PHP.

Q-5. What is the difference between the GET and POST methods?

Answer.
Following are the key differences between <GET> and <POST>.

GET Method.

- It submits all the Name-Value pairs as a query string in the URL.
- This method is not secure and reveals the data transmitted with the URL.
- The allowed length of the GET string should not exceed 2048 characters.
- If the Form tag does not contain any method name, GET will take over as default.
- The payload data is in text format. It accepts only ASCII values.
- GET is beneficial for performing data retrieval operations.

POST Method.

- It submits all the Name-Value pairs in the Message Body of the request.
- Unlike the GET, the Post method is secure as the Name-Value pairs do not appear in the location bar of the web browser.
- Since the payload gets encoded into the request, it doesn't show up as part of the URL.
- There is no restriction on the length of the string (i.e. the amount of data transmitted).
- In case, the POST method is in use and page refresh happens at the same time, then a prompt will occur before processing the request.
- If the service associated with the processing of a form has side effects (for example, modification of a database or subscription to a service), the method should be POST.

- It has no restriction on data usage and permits binary data also.
- POST is beneficial for performing both insert and update operations.

Q-6. What are Magic functions in PHP?

Answer.
They are special PHP functions that start with a double underscore **<__>**. None of these are stand-alone. And it is mandatory to define them inside a Class.

Below are some facts about the Magic functions in PHP.

- It is the programmer who defines the PHP magic function. Once the programmer provides the definition, PHP enables him to achieve powerful things using the Magic function.
- PHP does not allow us to call it directly from the code. Instead, the call happens indirectly.

Let's see an example.

```
class Animal {
   // height of animal
   public $height;
   // weight of animal
   public $weight;
   // code
   public function __construct($height, $weight)
   {
      $this->height = $height;  //set the height instance variable
      $this->weight = $weight; //set the weight instance variable
   }
}
```

Now, if we instantiate the Animal Class using the following line of code.

```
Animal obj = new Animal(6, 150);
```

It will automatically call the **<__construct()>** function and create an object obj, of the Animal class, with its height as 6 and weight as 150.

Here is a list of some powerful magic functions supported by PHP.

__construct() , __destruct() , __call() , __callStatic() , __get(), __set() , __isset() ,__unset()

, __sleep() , __wakeup(), __toString() , __invoke() , __set_state() , __clone().

Q-7. What is a .htacces file in PHP?

Answer.
The <.htaccess> is a configuration file used on the servers to run the Apache Web Server software. If this file is present in the directory, then the Apache Web Server will load it into memory and execute it.

It acts as a powerful tool to alter the configuration of the Web Server software by enabling/disabling the additional functionality and features that it offers.

These include –

- The redirection feature is like an occurrence of 404, file not found.
- More advanced functions such as content password protection or image hotlink prevention.

Q-8. How can we display the output directly to the browser?

Answer.
In order, to display the output directly to the browser, we have to use the special tags **<?= and ?>**.

Q-9. What are the different types of errors in PHP?

Answer.
There are essentially three types of errors in PHP.

Notices.

These are small, non-critical errors that PHP encounters while executing a script. For instance, accessing a variable that is not defined. By default, PHP does not display these errors to the user. However, the default behavior can be modified.

Warnings.

These are more severe errors, like attempting to include() a file that does not exist. By default, PHP displays these errors to the user. They do not result in script termination.

Fatal Errors.

These are critical errors. For example, instantiating an object of a non-existent class, or calling a non-existent function. These errors result in the immediate termination of the script. PHP’s default behavior is to display them to the user when it occurs.

Q-10. What is the maximum file size that PHP allows to upload? How can we increase it?

Answer.
The standard limit in PHP to upload a file is 2MB. However, we can change this value by making modifications in the **php.ini** file. We have to alter the value of **upload_max_filesize** and restart all the services.

Q-11. What is a final class in PHP?

Answer.
The final class is a unique object-oriented concept that prohibits a class from being inherited. We can use it to protect the methods of the base class from getting overridden by the child classes.

Final Class Example.

```
final class baseclass
{
   public function mymethod()  {
      echo  "base class method";
   }
}

class derivedclass extends baseclass
{
   public function mymethod() {
      echo   "derived class method";
   }
}

$c = new derivedclass();
$c->mymethod();
```

In the above example, we've kept the base class as final to prevent it from being derived. However, the second class in the example is trying to extend the base which will cause a compile-time error.

Q-12. What is the difference between include() and require()?

Answer.
The **<include()>** function throws a warning when a file not found error occurs. But the script continues to execute till its end. Let's see an example.

```
<?php
      include("filename.php");
      echo "Welcome";
?>
```

In the above example, if the file not found error occurs, the code will lead to a warning and print **<Welcome>**. It is because a warning in PHP neither stops the execution of the script nor blocks the echo command.

The **<require()>** function gives a fatal error if the specified file is not found and also stops the execution of the script. Let's see an example.

```
<?php
      require("filename.php");
      echo "Welcome";
?>
```

The above code will throw a fatal error and stop the script. Even the echo command will not run in this case.

Q-13. What is the use of var_dump in PHP?

Answer.
The **var_dump** function takes one or more variables as arguments and returns structural information about their types and values. It has the following syntax.

```
var_dump(variable1, variable2,.....variablen);
```

Here is an example.

```
<?php
       $m=3.1;
       $n=true;
       var_dump($m, $n);
?>
```

The above code displays the following output.

```
float(3.1)
bool(true)
```

Q-14. What are the differences between echo() and print() methods?

Answer.
Following are the key differences between **echo()** and **print()** statements.

Echo.

- It can accept multiple expressions.
- Since it does not return any value, so responds a bit faster.
- The echo statement displays the output on the user screen. Its syntax supports using echo with or without parentheses.
- Multiple arguments are allowed if separated with <,>.
- It doesn't return any value.

Print.

- It cannot accept multiple expressions.
- Returning a value makes it a little slower than the echo.
- The print statement displays the output on the user screen. Its syntax supports using print with or without parentheses.
- Multiple arguments are not allowed.
- Interestingly, it will always return the value 1.

Q-15. What are visibility keywords in PHP?

Answer.
We can set the visibility of a Property or Method by prefixing the declaration with the following keywords public, protected, or private.

- A public specifier signifies that the members are accessible from everywhere within your application.
- A protected specifier indicates that the members are accessible from within the class, the one that inherits it, and from the parent as well.
- Private specifier restricts that the members should only be accessible from within the class.

Q-16. What is the importance of `htmlentities()`?

Answer.
The **<htmlentities()>** function ensures that the text is correctly printed into HTML. It does this by converting any character that might be read as part of the HTML markup so that it gets displayed, rather than parsed.

It takes a normal string and converts any character like **<** and **>** into their respective HTML entity. Here is the prototype of this function.

```
htmlentities(string, flags, character-set, double_encode)
```

Let's take an example.

```
<?php
   $str = '<a href="https://techbeamers.com">Go to techbeamers.com</a>';
   echo htmlentities($str);
?>
```

The HTML output of the above code above will look like the one shown below.

```
&lt;a href=&quot;https://techbeamers.com&quot;&gt;Go to techbeamers.com&lt;/a&gt;
```

However, the browser will render the above code in the following style.

```
<a href="https://techbeamers.com">Go to techbeamers.com</a>
```

Q-17. What is Type Juggle in PHP?

Answer.
In PHP, mentioning the type of the variable is not required for declaring a variable. The data type is determined implicitly, by the value/context of the variable. If we assign an integer value to a variable $num, then it becomes of type integer, implicitly. If we assign a string value to the variable **$str**, it becomes of type string.

Let's take an example.

```
$num3 = $num1 + $num2
```

Here, if **$num1** is an integer, PHP also treats $num2 and $num3 as integers.

Q-18. What is the use of the header() function in PHP?

Answer.
The **header()** function sends a raw HTTP header to the client browser. Its rule is to call this function always, before sending the actual output.

For example, we don't print any HTML element, before using the **header()** function.

Q-19. What is a PHP filter and why is it used? Also, list down different PHP filter functions.

Answer.
The primary use of the PHP filter is to validate the data coming from different sources.

- User input from a form
- Cookies
- Web services data
- Server variables
- Database query results

Filtering is an important feature for a web application to validate and sanitize the data coming from insecure sources. PHP supports the following filter functions.

- **<filter_has_var()>**
  Check if a variable of a specified input type exists.
- **<filter_id()>**
  Returns the ID number of a specified filter.
- **<filter_var_array()>**
  Get multiple variables and filter them.
- **<filter input()>**
  Get input from outside the script and filter it.
- **<filter_input_array>**
  Get multiple inputs from outside the script and filter them.

- **<filter_list()>**
  Returns an array of all filters it supports.
- **<filter_var()>**
  Get a variable and filter it.

Q-20. How can we get all the properties of a browser in PHP?

Answer.
We can get the browser properties by using the following code snippet.

```
<?php
   $_SERVER ['HTTP_USER_AGENT'];
?>
```

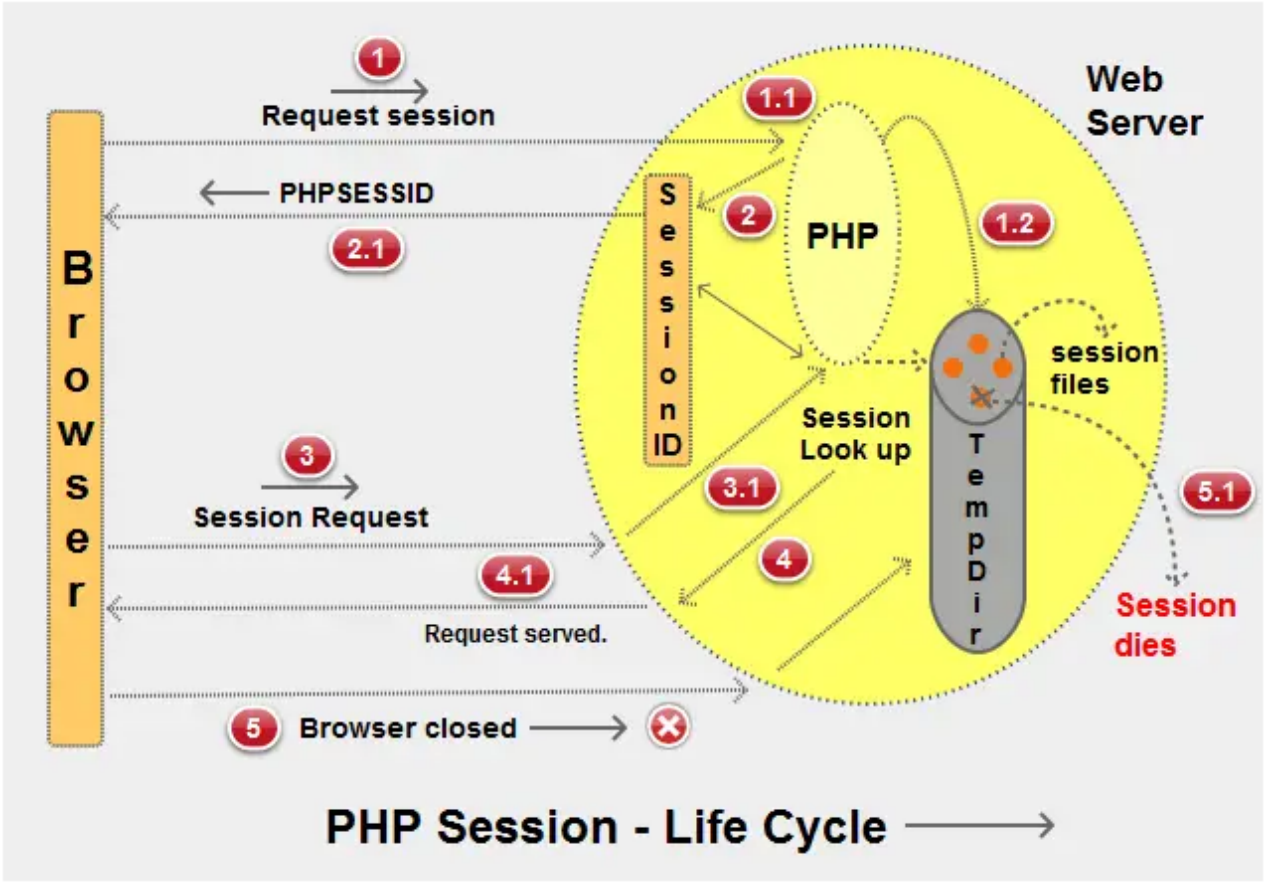
Q-21. What is a Session in PHP? How does PHP manage the lifecycle of a session?

Answer.
Sessions are data structures that provide a way to store user-specific data and associate it with a unique session ID. However, PHP sessions also solve the following problems.

A session creates a file in a temporary directory on the server to cache the registered session variables and their values. This data will be accessible to all pages on the site during that stay.

PHP uses a php.ini file to get hold of the server settings. This file also specifies the location of the session file under the **<session.save_path>**. The developers can make sure that this setting points to a correct value before using the session variable.

PHP Session Lifecycle.



While building up a session following events happen.

1. PHP generates PHPSESSID such as **"c69dc153f003406b9a0242ad4c505baa"**.

2. The PHP engine creates a file for every new session at the designated location. It bears the name of the unique identifier prefixed by **<sess_>**.

e.g. sess_c69dc153f003406b9a0242ad4c505baa.

3. When PHP gets a request from the browser or a client to access the session variable, it retrieves the session string from the PHPSESSID cookie.

4. A session comes to an end if the user decides to close the browser. However, the server could also terminate it after a certain inactivity period.

Q-22. What does accessing a class via "::" mean?

Answer.
<::> operator allows accessing static methods that do not require initializing the object. Its name is Scope Resolution Operator.

Q-23. How can we register the variables into a session?

Answer.
In order, to register a variable into the session, we must start the session by using the command.

```
session_start();
```

To assign a value to the variable with the name **<myvar>**, we use the global variable **<$_SESSION>**.

```
$_SESSION['myvar'] = "value";
```

Let's see the complete example.

```
<?php
/*
 * PHP Session - Register / Unregister session variables
 */

//start the session
session_start();

//if not exists, ads the myvar variable into the session array
if (!isset($_SESSION['myvar']))
    $_SESSION['myvar'] = "value";

//echo the myvar session variable
echo "The myvar variable value is : ".$_SESSION['myvar'];

//unset the myvar session variable
unset($_SESSION['myvar']);
?>
```

Q-24. How can we destroy a session in PHP?

Answer.
First of all, you should make a judgment about when to destroy a session in PHP. However, here are the three possible ways to close a session gracefully.

1. If there is only a single session and you want to delete its value, then calling the **<unset()>** function would be enough.

```
<?php
session_start();
// Destroy a particular session variable.
unset($_SESSION['myvar']) ;
?>
```

2. If you want to clear all session variables **(i.e. $_SESSION array)**, then call the **<session_unset()>** function.

```
<?php
session_start();
// Frees all session variables.
session_unset();
?>
```

Please note that the **session_unset()** function only reset the local **$_SESSION** variable instance but not the data inside the session storage.

3. The commands in the previous points only affect the **$_SESSION** variable or its values but leave the part in the session storage as is. However, there are cases when you have to remove all data bound to a session.

For example.

If a user logs out, then he won't need the session anymore.

In such a case, you should call the **session_destroy()** function. It destroys the session data cached inside the persistent storage (i.e. removes the session file from the file system).

```
<?php
// Removes all data registered to a session.
 session_destroy() ;
 ?>
```

Q-25. What is the difference between implode() and explode() functions?

Answer.

Implode() Function.

The purpose of **<implode()>** is to join the elements of an array using a string delimiter. The resulting string acts as a return value for the function, which gets stored in a variable.

Suppose we have the following array.

```
$arr = Array ("A", "E", "I", "O", "U");
```

Following is the command to combine it into a single string after applying the separator **'-'** between every element of the array.

```
$str = implode("-", $arr);
```

After executing the above code, the resulting string variable, **$str** will contain the following value.

```
A-E-I-O-U
```

The syntax for **<implode()>** is as follows.

```
implode(separator, array)
```

Explode() Function.

The **explode()** function breaks a string into an array, however, the implode function returns a string from the elements of an array.

Let's take the example of having a string.

```
$str = "A E I O U";
```

Now, to make each alphabet an element of an array, so that we can access it individually, we use the following code.

```
$arr = explode(",", $str);
```

The above code breaks the string **$str** based on separator **‘,’** and puts the resulting array in variable **$arr**.

```
Array(
[0] => A
[1] => E
[2] => I
[3] => O
[4] => U
)
```

It is equivalent to the following.

```
$arr = Array ("A", "E", "I", "O", "U");
```

The syntax for **explode()** is as follows.

```
explode (separator, string, limit)
```

Q-26. How will you redirect a web page to another page using PHP?

Answer.
Following is the piece of code which redirects a web page to another page.

```
header("Location:url_of_the_page_to_redirect.php");
exit();
```

Q-27. What is the difference between <strstr()> and <stristr()> functions in PHP?

Answer.

The <strstr()> function.

This function locates the first occurrence of a string.

It has the following syntax.

```
strstr ($string, string)
```

Let’s take an example to understand it better.

```
<?php
$email_id = 'meenakshi@gmAil.com';
$domain_name = strstr($email_id, 'A');
echo $domain_name;
?>

// Output: Ail.com
```

The <stristr()> function.

It locates the first occurrence of a string. The search here is case-insensitive.

Syntax:

```
stristr ($string, string)
```

Let’s take an example to understand it better.

```
<?php
$email_id = 'meenakshi@gmAil.com';
$domain_name = strstr($email_id, 'A');
echo $domain_name;
?>

Output: akshi@gmAil.com
```

Q-28. How will you enable error reporting in PHP?

Answer.

In PHP, we use the **<error_reporting>** function, to turn on error reporting for a particular file. The other way is to enable error reporting for all the files on the web server by editing the **php.ini** file.

Let’s get more clarity on how to enable error reporting in PHP.

```
<?php

//Disable all error reports
error_reporting(0);

//Enable basic runtime errors
error_reporting(E_ERROR | E_WARNING | E_PARSE);

//Allow E_NOTICE to catch uninitialized variables or variable name
//or misspellings
error_reporting(E_ERROR | E_WARNING | E_PARSE | E_NOTICE);

//Allow all PHP errors
error_reporting(-1);

//Allow all PHP errors
 error_reporting(E_ALL);

//Alternate way to allow all errors in PHP
ini_set('error_reporting', E_ALL);
?>
```

Display Errors in PHP.

The **<display_error>** setting determines whether errors get printed on the screen or hidden from the user.

It gets used in conjunction with the **error_reporting** function as shown in the example below.

```
ini_set('display_errors',1);
error_reporting(E_ALL);
```

Modify the php.ini file.

If we want to see error reports for all our files, then just go to the web server and add the following option in the php.ini file for the website.

```
error_reporting=E_ALL
```

Q-29. How do you send an email using PHP?

Answer.
PHP provides the mail() function to send emails. However, you first need to modify the php.ini file accordingly.

For Windows, make the following change:

```
[mail function]
; For Win32 only.
SMTP = smtp.secureserver.net

; For win32 only
sendmail_from = techbeamers.com
```

On Linux, you only make one change as you can see below.

```
; For Unix only
sendmail_path = /usr/sbin/sendmail -t -i
```

Now, check the mail() function syntax.

```
mail(to, subject, message, headers, parameters);
```

Let’s see a sample code to send an email.

```
<?php
// the message
$msg = "First line of text\nSecond line of text";

// use wordwrap() if lines are longer than 70 characters
$msg = wordwrap($msg,70);

// send email
mail("someone@example.com","My subject",$msg);
?>
```

Q-30. What are Superglobals in PHP?

Answer.
Superglobals are specially-defined array variables in PHP that make it easy for the user to get information about a request or its context. This data can come either from different URLs, HTML forms, cookies, sessions, and the Web server itself.

`$_GET`

_GET represents the data forwarded to the PHP script in a URL. It applies to URLs that are directly accessible and also to form submissions that use the GET method.

`$_POST`

_POST It represents the data forwarded to the PHP script via HTTP POST. It is a form that includes a method POST.

`$_COOKIE`

_COOKIE represents the data available to a PHP script via HTTP cookies.

`$_REQUEST`

_REQUEST It is a combination of $_GET, $_POST, and $_COOKIE.

`$_SESSION`

It represents the data available to a PHP script that was earlier, stored in a session.

`$_SERVER`

_SERVER Superglobal represents the data available to a PHP script, from the Web server itself.

`$_ENV`

_ENV represents the data available to a PHP script from the environment in which PHP is running.

`$_FILES`

_FILES represents the data available to a PHP script from HTTP POST file uploads. Using $_FILES is currently the most preferred way to handle uploaded files in PHP.

`$GLOBALS`

GLOBALS It stores all variables with a global scope. It includes all of the above. Unlike the other Superglobals, $GLOBALS is part of PHP from the very beginning (PHP 3).

Wrapping Up: The Latest PHP Interview Questions

We are hopeful that the above PHP interview questions will lead you to the door of success.

In this post, we've tried to touch upon a combination of core PHP concepts, programming and object-oriented constructs, and built-in functions.

However, if there is anything that is not clear or not covered here, then do write to us. We'll try our best to answer your queries.

Best,

ParvaM