

Web Programming

Unit 4: PHP

Compiled by: Prof. Ketaki Ghawali

Ketaki.ghawali@vsit.edu.in

VSIT

Vidyalankar School of
Information Technology

NAAC ACCREDITED COLLEGE

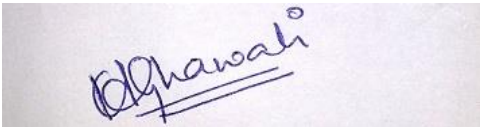
Vidyalankar School of
Information
Technology
Wadala (E), Mumbai
www.vsit.edu.in

Certificate

This is to certify that the e-book titled "Web programming" comprises all elementary learning tools for a better understating of the relevant concepts. This e-book is comprehensively compiled as per the predefined eight parameters and guidelines.



Date: 10-12-2019



Ms. Ketaki Ghawali
Department of BSc IT

⚠ DISCLAIMER: *The information contained in this e-book is compiled and distributed for educational purposes only. This e-book has been designed to help learners understand relevant concepts with a more dynamic interface. The compilers of this e-book and Vidyalankar School of Information technology give full and due credit to the authors of the contents, developers and all websites from wherever information has been sourced. We acknowledge our gratitude towards the websites YouTube, Wikipedia, and Google search engine. No commercial benefits are being drawn from this project.*

UNIT IV

Unit IV :PHP

- Contents
- Introduction
- PHP Syntax and variables
- Comments
- Control statements,branching,looping
- Functions
- Passing information with PHP, GET and POST
- Superglobal Arrays
- String and String Functions
- Regular expression
- Arrays
- Basic PHP error/problems

- Recommended Books

PHP 5.1 for Beginners, by Ivan Bayross

Prerequisites and Linking

Unit III	Pre-requisites	Linking				
Java Script	Sem-I	Sem. II	Sem. III	Sem. IV	Sem. V	Sem. VI
	C Programming	-	-	-	Project Development	Project Development

PHP INTRODUCTION

PHP is a server-side scripting language.

<https://www.youtube.com/watch?v=fT7rCWZvsD8>

What is PHP?

- PHP stands for **PHP: Hypertext Preprocessor**
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts

- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support.

Server-side Scripting

<https://www.youtube.com/watch?v=JnCLmLO9LhA>

Server-side scripting is about "programming" the behavior of the server. This is called server-side scripting or server scripting.

Client-side scripting is about "programming" the behavior of the browser. (see Web JavaScript chapter).

What is Server-side Scripting?

Normally when a browser requests an HTML file, the server returns the file, but if the file contains a server-side script, the script inside the HTML file is executed by the server before the file is returned to the browser as plain HTML.

What can Server Scripts Do?

- Dynamically edit, change or add any content to a Web page
- Respond to user queries or data submitted from HTML forms

- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- Provide security since your server code cannot be viewed from a browser

Important : Because the scripts are executed on the server, the browser that displays the ASP/PHP file does not need to support scripting at all!

PHP INSTALLATION

<https://www.youtube.com/watch?v=FisOKkixArQ>

What do you Need?

If your server supports PHP you don't need to do anything.

Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support.

However, if your server does not support PHP, you must install PHP.

Here is a link to a good tutorial from PHP.net on how to install PHP5:

<http://www.php.net/manual/en/install.php>

Download PHP

Download PHP for free here: <http://www.php.net/downloads.php>

Download MySQL Database

Download MySQL for free here: <http://www.mysql.com/downloads/>

PHP Syntax

PHP code is executed on the server, and the plain HTML result is sent to the browser.

Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with **<?** and end with **?>**.

For maximum compatibility, we recommend that you use the standard form (**<?php**) rather than the shorthand form.

```
<?php  
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>  
<body>  
  
<?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.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

Note: The file must have a .php extension. If the file has a .html extension, the PHP code will not be executed.

Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
```

```
<body>
```

```
<?php
```

```
//This is a comment
```

```
/*
```

```
This is
```

```
a comment
```

```
block
```

```
*/
```

```
?>
```

```
</body>
```

```
</html>
```

PHP Variables

A variable is used to store information.

Variables in PHP

Variables are used for storing values, like text strings, numbers or arrays. When a variable is declared, it can be used over and over again in your script. All variables in PHP start with a \$ sign symbol.

The correct way of declaring a variable in PHP:

```
$var_name = value;
```

New PHP programmers often forget the \$ sign at the beginning of the variable. In that case it will not work.

Let's try creating a variable containing a string, and a variable containing a number:

```
<?php  
$txt="Hello World!";  
$x=16;  
?>
```

PHP is a Loosely Typed Language

In PHP, a variable does not need to be declared before adding a value to it. In the example above, you see that you do 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, you have to declare (define) the type and name of the variable before using it.

In PHP, the variable is declared automatically when you use it.

Naming Rules for Variables

- A variable name must start with a letter or an underscore "_"

- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore (\$my_string), or with capitalization (\$myString)

PHP String Variables

A string variable is used to store and manipulate text.

String Variables in PHP

String variables are used for values that contains characters.

In this chapter we are going to look at the most common functions and operators used to manipulate strings in PHP.

After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.

Below, the PHP script assigns the text "Hello World" to a string variable called \$txt:

```
<?php
$txt="Hello World";

echo $txt;
?>
```

The output of the code above will be:

Hello World

Now, lets try to use some different functions and operators to manipulate the string.

The Concatenation Operator

There is only one string operator in PHP.

The concatenation operator (.) is used to put two string values together.

To concatenate two string variables together, use the concatenation operator:

```
<?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!

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

Strings and String Functions

The strlen() function

The strlen() function is used to return the length of a string.

Let's find the length of a string:

```
<?php
echo strlen("Hello world!");
?>
```

The output of the code above will be:

12

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string).

The strpos() function

The strpos() function is used to search for character within a string.

If a match is found, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string:

```
<?php
echo strpos("Hello world!","world");
?>
```

The output of the code above will be:

6

The position of the string "world" in our string is position 6. The reason that it is 6 (and not 7), is that the first position in the string is 0, and not 1.

The str_word_count Function

The PHP str_word_count() function counts the number of words in a string.

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

The strrev function

The PHP strrev() function reverses a string.

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

The str_replace function

The PHP str_replace() function replaces some characters with some other characters in a string. The example below replaces the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

PHP Operators

Operators are used to operate on values.

PHP Operators

This section lists the different operators used in PHP.

https://www.youtube.com/watch?v=miSQP_KbwP8

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3

*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
=	x=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false

!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	And	x=6 y=3 (x < 10 && y > 1) returns true
	Or	x=6 y=3 (x==5 y==5) returns false
!	Not	x=6 y=3 !(x==y) returns true

PHP If...Else Statements

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

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*** : use this statement to execute some code only if a specified condition is true
- ***if...else statement*** : use this statement to execute some code if a condition is true and another code if the condition is false
- ***if...elseif...else statement*** : use this statement to select one of several blocks of code to be executed
- ***switch statement*** : use this statement to select one of many blocks of code to be executed

The if Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

if (condition) code to be executed if condition is true;

The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

Notice that there is no `..else..` in this syntax. The code is executed only if the specified condition is true.

The if...else Statement

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

Syntax

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
    echo "Have a nice weekend!";
else
    echo "Have a nice day!";
?>

</body>
</html>
```

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces :

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
```

```
{
echo "Hello!<br />";
echo "Have a nice weekend!";
echo "See you on Monday!";
}
?>

</body>
</html>
```

The **if...elseif...else** Statement

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

Syntax

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
    echo "Have a nice weekend!";
elseif ($d=="Sun")
```

```
    echo "Have a nice Sunday!";  
else  
    echo "Have a nice day!";  
?>  
</body>  
</html>
```

PHP Switch Statement

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

The PHP Switch Statement

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

```
<html>
<body>
<?php
switch ($x)
{
case 1:
    echo "Number 1";
    break;
case 2:
    echo "Number 2";
    break;
case 3:
    echo "Number 3";
    break;
default:
    echo "No number between 1 and 3";
}
?>

</body>
</html>
```

PHP Arrays

An array stores multiple values in one single variable.

What is an Array?

A variable is a storage area holding a number or text. The problem is, a variable will hold only one value.

An array is a special variable, which can store multiple values in one single variable.

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="Saab";  
$cars2="Volvo";  
$cars3="BMW";
```

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 best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own index so that it can be easily accessed.

In PHP, there are three kind of arrays:

- **Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

Numeric Arrays

A numeric array stores each array element with a numeric index.

There are two methods to create a numeric array.

i) In the following example the index are automatically assigned (the index starts at 0):

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

ii) In the following example we assign the index manually:

```
$cars[0]="Saab";  
$cars[1]="Volvo";  
$cars[2]="BMW";  
$cars[3]="Toyota";
```

Example

In the following example you access the variable values by referring to the array name and index:

```
<?php  
$cars[0]="Saab";  
$cars[1]="Volvo";  
$cars[2]="BMW";
```

```
$cars[3]="Toyota";  
echo $cars[0] . " and " . $cars[1] . " are Swedish cars."  
?>
```

The code above will output:
Saab and Volvo are Swedish cars.

Associative Arrays

An associative array, each ID key is associated with a value.

When storing data about specific named values, a numerical array is not always the best way to do it.

With associative arrays we can use the values as keys and assign values to them.

Example 1

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Example 2

This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";  
$ages['Quagmire'] = "30";  
$ages['Joe'] = "34";
```

The ID keys can be used in a script:

```
<?php  
$ages['Peter'] = "32";  
$ages['Quagmire'] = "30";  
$ages['Joe'] = "34";  
  
echo "Peter is " . $ages['Peter'] . " years old."  
?>
```

The code above will output:
Peter is 32 years old.

PHP Array Functions

PHP Array Introduction

The array functions allow you to manipulate arrays.

PHP Array Functions

PHP: indicates the earliest version of PHP that supports the function.

Function	Description	PHP
<u>array()</u>	Creates an array	3
<u>array_reverse()</u>	Returns an array in the reverse order	4
<u>count()</u>	Counts elements in an array, or properties in an object	3
<u>sort()</u>	Sorts an array	3

PHP array()

Definition and Usage

array() creates an array, with keys and values. If you skip the keys when you specify an array, an integer key is generated, starting at 0 and increases by 1 for each value.

Syntax

array(key => value)

Parameter	Description
Key	Optional. Specifies the key, of type numeric or string. If not set, an integer key is generated, starting at 0

value	Required. Specifies the value
-------	-------------------------------

Example 1

```
<?php
$a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse");
print_r($a);
?>
```

The output of the code above will be:

```
Array ( [a] => Dog [b] => Cat [c] => Horse )
```

Example 2

```
<?php
$a=array("Dog","Cat","Horse");
print_r($a);
?>
```

The output of the code above will be:

```
Array ( [0] => Dog [1] => Cat [2] => Horse )
```

PHP array_reverse() Function

Definition and Usage

The array_reverse() function returns an array in the reverse order.

Syntax

array_reverse(array,preserve)	
Parameter	Description
array	Required. Specifies an array
preserve	Optional. Possible values:

	<ul style="list-style-type: none"> • true • false <p>Specifies if the function should preserve the array's keys or not.</p>
--	---

Example

```
<?php
$a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse");
print_r(array_reverse($a));
?>
```

The output of the code above will be:

```
Array ( [c] => Horse [b] => Cat [a] => Dog )
```

PHP sort() Function

Definition and Usage

The sort() function sorts an array by the values.

This function assigns new keys for the elements in the array. Existing keys will be removed.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
sort(array,sorttype)
```

Parameter	Description
array	Required. Specifies the array to sort
sorttype	Optional. Specifies how to sort the array values. Possible values: <ul style="list-style-type: none"> • SORT_REGULAR - Default. Treat values as they are (don't change types) • SORT_NUMERIC - Treat values numerically • SORT_STRING - Treat values as strings

- | | |
|--|---|
| | <ul style="list-style-type: none">• SORT_LOCALE_STRING - Treat values as strings, based on local settings |
|--|---|

Example

```
<?php
$my_array = array("a" => "Dog", "b" => "Cat", "c" => "Horse");

sort($my_array);
print_r($my_array);
?>
```

The output of the code above will be:

```
Array
(
    [0] => Cat
    [1] => Dog
    [2] => Horse
)
```

PHP Looping - While Loops

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

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

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 defines a loop that starts with i=1. The 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

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 defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, 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

The for loop and the foreach loop will be explained in the next chapter.

PHP Looping - For Loops

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

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 loop)

Note: Each of the 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 i is less than, or equal to 5. 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

PHP Functions

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

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.

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 Refsnes";  
    }  
    echo "My name is ";  
    writeName();  
?>  
</body>  
</html>
```

Output:

My name is Kai Jim Refsnes

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 . " Refsnes.<br />";
}
echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>
</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

Example 2

The following function has two parameters:

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

```
<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}
echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>
</body>
</html>
```

Output:

My name is Kai Jim Refsnes.

My sister's name is Hege Refsnes!

My brother's name is Ståle Refsnes?

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

PHP Forms and User Input

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

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.

<https://www.youtube.com/watch?v=YcUH31YI2wE>

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 click 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 functions will be explained in the next chapters.

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.

PHP \$_GET Function

The built-in \$_GET function is used to collect values in a form with method="get".

The \$_GET Function

The built-in \$_GET function is used to collect values from a form sent 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 (max. 100 characters).

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 function to collect form data (the names of the form fields will automatically be the keys in the \$_GET array):

```
Welcome <?php echo $_GET["fname"]; ?>.<br />  
You are <?php echo $_GET["age"]; ?> years old!
```

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.

Note: The get method is not suitable for large variable values; the value cannot exceed 100 characters.

PHP \$_POST Function

The built-in \$_POST function is used to collect values in a form with method="post".

The \$_POST Function

The built-in \$_POST function 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 function to collect form data (the names of the form fields will automatically be the keys in the \$_POST array):

```
Welcome <?php echo $_POST["fname"]; ?>!<br />  
You are <?php echo $_POST["age"]; ?> years old.
```

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.

The PHP \$_REQUEST Function

The PHP built-in \$_REQUEST function contains the contents of both \$_GET, \$_POST, and \$_COOKIE.

The \$_REQUEST function 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.

Super – Global Arrays

Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- \$GLOBALS
- \$_SERVER
- \$_REQUEST
- \$_POST
- \$_GET
- \$_FILES
- \$_ENV
- \$_COOKIE

- `$_SESSION`

\$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.

```
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

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

```
<?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'];
?>
```


PHP \$_REQUEST is used to collect data after submitting an HTML form.

```
<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 is widely used to collect form data after submitting an HTML form with method="post". \$_POST is also widely used to pass variables.

```
<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 can also be used to collect form data after submitting an HTML form with method="get". \$_GET can also collect data sent in the URL.

```
<html>
```

```
<body>
```

```
<?php
```

```
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];  
?>
```

```
</body>
```

```
</html>
```

PHP Error Handling

The default error handling in PHP is very simple. An error message with filename, line number and a message describing the error is sent to the browser.

PHP Error Handling

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

This tutorial contains some of the most common error checking methods in PHP.

We will show different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Basic Error Handling: Using the die() function

The first example shows a simple script that opens a text file:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

If the file does not exist you might get an error like this:

Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in **C:\webfolder\test.php** on line **2**

To avoid that the user gets an error message like the one above, we test if the file exist before we try to access it:

```
<?php
if(!file_exists("welcome.txt"))
{
    die("File not found");
}
else
{
    $file=fopen("welcome.txt","r");
}
?>
```

Now if the file does not exist you get an error like this:

File not found

The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

However, simply stopping the script is not always the right way to go. Let's take a look at alternative PHP functions for handling errors.

QUESTIONS

1. What is cookie?
2. Why PHP is loosely typed language?
3. Explain different PHP operator?
4. Explain PHP Array Function.
5. Explain PHP Function.
6. Explain \$_GET function and \$_POST function.
7. Explain Error Handling.
8. Explain Server Side Scripting.
9. How to add PHP to HTML? Explain with example.
10. Write naming rules & conventions for declaring variable.
11. Explain any two string functions.
12. What is Array? What are different types of array. Explain with example.
13. Explain any two array functions.

MULTIPLE CHOICE QUESTIONS

1. If a Boolean variable \$alive=5;
a. \$alive is false. **b. \$alive is true.**

- c. \$alive is overflow d. The statement is not valid.
2. Which of the following comment is supported by PHP?
a. Single line C++ syntax - // b. Shell syntax - #
c. Both of the above
3. Which of the following data type is compound data type supported by PHP?
a. Array b. String
c. Float d. Boolean
4. In mail(\$param1,\$param2,\$param3,\$param4), the \$param2 contains:
a. Message b. Recipient
c. Header **d. Subject**
5. PHP supports all four different ways of delimiting. In this context identify the false statement.
a. You can use any of the delimiting style b. You can use different delimiting style in one page.
c. You can use any delimiting style but must use a single style. d. none of these
6. In PHP, string data are
a. delimited by single quote b. delimited by double quote
c. delimited by <<< identifier **d. All of the above**
7. Which of the following assignment is **by value** assignment in PHP?
a. \$value1=\$value2 b. \$value1=&\$value?

c. None of the above

8. The output of the following script would be:

```
$somerar=15;  
Function ad(){  
    GLOBAL $somerar;  
    $somerar++;  
    Echo "somerar is $somerar";  
}
```

a. Somerar is 15

c. Somerar is 1

b. Somerar is 16

d. Somerar is \$somerar

9. Trace the false statement.

a. Because the included code will be embedded in a PHP execution block, the PHP escape tags are not required on the file to be included.

c. All of the above

b. Any code found within an included file will inherit the variable scope of the location of its caller.

d. None of the above

10. Identify the variable scope that is not supported by PHP.

a. Local variables

c. Hidden variables

b. Function parameters

d. Global variables

11. Which of the following delimiting method is known as string interpolation?

a. Delimited by single quote

c. Delimited by <<< identifier

b. Delimited by double quote

d. All of the above

12. Variable scope on which a variable does not lose its value when the function exists and use that value if the function is called again is:

a. Local

c. static

b. Function parameter

d. none of the above

13. The left association operator % is used in PHP for
- a. **percentage**
 - b. bitwise or
 - c. division
 - d. modulus
14. On failure of which statement the script execution stops displaying error/warning message?
- a. Rinclude()
 - b. **Require()**
 - c. Both of above
 - d. None of the above
15. Casting operator used in PHP is:
- a. (array)
 - b. **(int64)**
 - c. (real) or (double) or (float)
 - d. (object)