Padasalai's Telegram Groups!

(தலைப்பிற்கு கீழே உள்ள லிங்கை கிளிக் செய்து குழுவில் இணையவும்!)

- Padasalai's NEWS - Group
https://t.me/joinchat/NIfCqVRBNj9hhV4wu6_NqA

- Padasalai's Channel - Group
https://t.me/padasalaichannel

- Lesson Plan - Group
https://t.me/joinchat/NIfCqVWwo5iL-21gpzrXLw

- 12th Standard - Group
https://t.me/Padasalai_12th

- 11th Standard - Group
https://t.me/Padasalai_11th

- 10th Standard - Group
https://t.me/Padasalai_10th

- 9th Standard - Group
https://t.me/Padasalai_9th

- 6th to 8th Standard - Group
https://t.me/Padasalai_6to8

- 1st to 5th Standard - Group
https://t.me/Padasalai_1to5

- TET - Group
https://t.me/Padasalai_TET

- PGTRB - Group
https://t.me/Padasalai_PGTRB

- TNPSC - Group
https://t.me/Padasalai_TNPSC

12- COMPUTER APPLICATION- LESSON 4-7
BOOK BACK QUESTIONS

NAME :

CLASS :

SCHOOL :

Prepared By

Mr.M.Dhanapal,MCA.,B.Ed.,

PG-Assist In Computer Science & CA

Literacy Mission MHSS, Samalapuram

Lesson 4-7 PHP-Book back Questions and answer

1. Difference between Client and server?

Sno	Client	Server
1	A client machine is a small computer with a basic hardware configuration.	A server machine is a high-end computer with an advanced hardware configuration.
2	A client is a simple and less powerful machine	A server is a powerful expensive machine.
3	A client is used for simple tasks	A server is used for storing huge data files and applications.
4	A client supports a single user log-in at a time.	A server supports simultaneous, multiple user log-ins

2. Write the features of server side scripting.

- Generally quicker to load than client-side scripting
- User is able to include external files to save coding.
- Scripts are hidden from view so it is more secure. Users only see the HTML output.
- The site can use a content management system which makes editing simpler.

3. Write the purpose of Web servers.

Web server software that runs on server hardware, governs the server side scripting compilation into an intermediate byte-code that is then interpreted by the runtime engine.

4. Differentiate server side and client side scripting language.

The main difference between server-side scripting and client-side scripting is that the server side scripting involves server for its processing. On the other hand, client-side scripting requires browsers to run the scripts on the client machine but does not interact with the server while processing the client-side scripts.

Mr.M.Dhanapal.,MCA.,B.Ed.,9790573672 Literacy Mission MHSS

12- Computer Application- lesson 4-7 Book Back Questions

5. What is difference between user defined and system defined functions?

Sno	User Defined function	System Defined function
1	User Defined Function (UDF) in PHP gives a privilege to user to write own specific operation inside of existing program module.	Built in function are predefined in php. Each function have specific task
2	User define functions are started with keyword function.	Here function keywords are not used in built in function and we cannot redefine the built in function.

3	Example:
	function hello()
	{
	}
		Example:
		$a=fopen('sample.txt','r')

6. Differentiate Associate array and multidimensional array.

Sno	Associative Array	Multidimensional Array
1	Associative array will have their index as string so that you can establish a strong association between key and values.	An array containing one or more arrays and values are accessed using multiple indices
2	Example: $a=array('1'=>46,'2'=>86); echo"$a['1'];	.Example: $a=array(array(12,45,66),array(13,86,89)); echo "$a[0][0]";
3	Single array can be used in associative array	Two or more arrays are nested within another array

Mr.M.Dhanapal.,MCA.,B.Ed.,9790573672 Literacy Mission MHSS

7. Compare if and if else.

Sno	If	If else
1	if statement consist one block. if condition id true if block is executed otherwise skipped the if block	If else consists of two alternative blocks. Based on the condition if block is executed otherwise else block is executed.
2	Its one way branching	It is two branching

3	Example:
	$a=1;
	if($a>=5)
	echo “$a is positive”;
		Example:
		$a=1;
		if($a>=5)
		echo “$a is positive”;
		else
		echo “$a is negative”;

8. Differentiate switch and if else.

Sno	Switch	If else
1	Switch is multi branching statement.	It is two way branching
2	Switch statement is used to perform different actions based on different conditions.	It’s based on single conditions

3	Example:
	$a=1;
	switch($a)
	{
	case 1:
	echo”one”;
	break;
	}
		Example:
		$a=1;
		if($a>=5)
		echo “$a is positive”;
		else
		echo “$a is negative”;

9. Difference between if statement and if elseif else.

Sno	if	if elseif else
1	if statement consist one block. if condition id true if block is executed otherwise skipped the if block	If else executes the block of statements if the condition in the corresponding if is false. After the else, if another condition is to be checked, then an if statement follows the else. This is else if and is called as if else ladder.
2	Its one way branching	It is multi branching

3	Example:
	$a=1;
	if($a>=5)
	echo "$a is positive";
		Example:
		$a=1;$b=5;$c=6;
		if(($a>b)&&($a>c))
		echo "$a is greater";
		elseif(($b>a)&&($b>c))
		echo "$b is greater";
		else
		echo "$c is greater";

10.Compare for loop and foreach.

Sno	For loop	For each loop
1	For loop is an important functional looping system which is used for iteration logics when the programmer know in advance how many times the loop should run.	foreach loop is exclusively available in PHP. It works only with arrays. The loop iteration depends on each KEY Value pair in the Array. For each, loop iteration the value of the current array element is assigned to $value variable and the array pointer is shifted by one, until it reaches the end of the array element
2	Example: for($a=5;$a<=5;$a++)	Example: $a=aray(1,5,6,8)

	echo “$a”;	foreach($a as $values) echo “$values”;

11.Differentiate for each and while loop.

Sno	While loop	For each loop
1	While loop is an important feature which is used for simple iteration logics. It is checking the condition whether true or false. It executes the loop if specified condition is true.	Foreach loop is exclusively available in PHP. It works only with arrays. The loop iteration Depends on each KEY Value pair in the Array. For each, loop iteration the value of the current array element is assigned to $value variable and the array pointer is shifted by one, until it reaches the end of the array element

2	Example:
	$a=5;
	while($a<=5)
	{
	echo “$a”;
	$a++; }
		Example:
		$a=aray(1,5,6,8)
		foreach($a as $values)
		echo “$values”;

12.Differentiate while and do while.

Sno	While loop	For each loop
1	While loop is an important feature which is used for simple iteration logics. It is checking the condition whether true or false. It executes the loop if specified condition is true.	Do whileloop always run the statement inside of the loop block at the first time execution. Then it is checking the condition whether true or false. It executes the loop.
2.	Condition is placed at beginning of the loop	Condition is placed at end of the loop

3	Example:
	$a=5;
	while($a<=5)
	{
		Example:
		$a=5;
		do{
		echo “$a”;

	echo “$a”; $a++; }	$a++; }while($a<=5);

5-Function

Function:

```
<?php
function name()
{
echo "Example for function";
}
name();
?>
```

Output:

Example for function

Parameterized Function:

```
<?php
function name($a)
{
echo "$a";
}
name("Welcome");
?>
```

Output:

Welcome

Function with return statement:

```
<?php
function name($a,$b)
{
$c=$a+$b;
return $c;;
}
echo "Sum is =".name(5,6);
?>
```

Output:

Sum is = 11

Array

Indexed array:

```
<?php
$a=array("arun","bala");
echo "Name1=".$a[0]."<br>";
echo "Name2=".$a[1]."<br>";
?>
```

Output:

Name1=arun
Name2=bala

Associative Arrays:

```
<?php
$a=array("1"=>"arun",2=>"bala");
echo "Name1=".$a[1]."<br>";
echo "Name2=".$a[2]."<br>";
?>
```

Output:

Name1=arun
Name2=bala

Multidimensional array:

```
<?php
$a=array(array("arun","XII-C"),array("bala","XII-C"));;
echo "Name =".$a[0][0]."  Class = ".$a[0][1]."<br>";
echo "Name =".$a[1][0]."  Class = ".$a[1][1]."<br>";
?>
```

Output:

Name =arun Class = XII-C
Name =bala Class = XII-C

6- Conditional Statements

If :

```
<?php
$a=5;
if($a>0)
echo "$a is postive number";
?>
```

Output:

5 is postive number

If..else:

```
<?php
$a=5;
if($a>0)
echo "$a is postive number";
else
echo "$a is zero or negative";
?>
```

Output:

5 is postive number

If..elseif..else:

```
<?php
$a=5;
$b=6;
$c=8;
if(($a>$b)&&($a>$c))
echo "$a is Greater";
elseif(($b>$a)&&($b>$c))
echo "$b is Greater";
else
echo "$c is Greater";
?>
```

Output:

8 is Greater

Switch:

```
<?php
$a=3;
switch($a)
{
case 1:
        echo "one";
        break;
case 2:
        echo "two";
        break;
```

```
case 3:
        echo "three";
        break;
default:
        echo "break";
}
?>
```

Output:

three

7-Looping

For Loop:

```
<?php
for($a=1;$a<=1;$a++)
{
echo "Welcome";
}
?>
```

Output:

Welcome

While Loop:

```
<?php
$a=1;
while($a<=1)
{
echo "Welcome";
$a++;
}
?>
```

Output:

Welcome

Do while:

```
<?php
$a=1;
do
{
echo "Welcome";
$a++;
}while($a<=1);
?>
```

Output:

Welcome

Foreach:

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value)
{
  echo "$value <br>";
}
?>
```

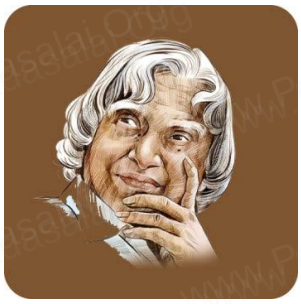
Output:

red
green
blue
yellow

“All the best”

STUDY MATERIAL PHP

12



Prepared By

Mr.M.Dhanapal,MCA.,B.Ed.,

PG-Assist In Computer Science

Literacy Mission MHSS, Samalapuram

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

PHP is an amazing and popular language!

It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
It is deep enough to run the largest social network (Facebook)!
It is also easy enough to be a beginner's first server side language!

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

PHP 5 Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

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 uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Example

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

Comments in PHP

A comment in PHP code is a line that is not read/executed as 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 what you are doing
- 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:

```
<!DOCTYPE html>
<html>
<body>
 <?php
// This is a single-line comment
# This is also a single-line comment
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>
</body>
</html>
```

Output :

10

PHP Case Sensitivity

In PHP, NO keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are case-sensitive.

In the example below, all three echo statements below are legal (and equal):

Example

However; all variable names are case-sensitive.

In the example below, only the first statement will display the value of the $color variable (this is because $color, $COLOR, and $coLOR are treated as three different variables):

```
<!DOCTYPE html>
<html>
<body>
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
</body>
</html>
```

Output :

My car is red

My house is

My boat is

PHP 5 Variables

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

Example

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

After the execution of the statements above, the variable **$txt** will hold the value**Hello world!**, the variable **$x** will hold the value **5**, and the variable **$y** will hold the value **10.5**.

Note: When you assign a text value to a variable, put quotes around the value.

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Think of variables as containers for storing data.

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

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)

Remember that PHP variable names are case-sensitive!

Output Variables

The PHP `echo` statement is often used to output data to the screen.

The following example will show how to output text and a variable:

Example

```
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

The following example will produce the same output as the example above:

Example

```
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

The following example will output the sum of two variables:

Example

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

Note: You will learn more about the `echo` statement and how to output data to the screen in the next chapter.

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 other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

```
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Example

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

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.

PHP The global Keyword

The `global` keyword is used to access a global variable from within a function.

To do this, use the `global` keyword before the variables (inside the function):

Example

```
<?php
$x = 5;
$y = 10;

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:

Example

```
<?php
$x = 5;
$y = 10;

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

myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

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.

PHP 5 echo and print Statements

In PHP there are two basic ways to get output: `echo` and `print`.

In this tutorial we use `echo` (and `print`) in almost every example. So, this chapter contains a little more info about those two output statements.

PHP echo and print Statements

`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 echostatement:

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;
?>
```

PHP 5 Data Types

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

Example

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 10.365;
var_dump($x);
?>
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

Example

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

You will learn a lot more about arrays in later chapters of this tutorial.

PHP Object

An object is a data type which stores data and information on how to process that data.

In PHP, an object must be explicitly declared.

First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example

```
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

Tip: If a variable is created without a value, it is automatically assigned a value of NULL.

Variables can also be emptied by setting the value to NULL:

Example

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

A common example of using the resource data type is a database call.

We will not talk about the resource type here, since it is an advanced topic.

PHP 5 Strings

A string is a sequence of characters, like "Hello world!".

PHP String Functions

In this chapter we will look at some commonly used functions to manipulate strings.

Get The Length of a String

The PHP `strlen()` function returns the length of a string.

The example below returns the length of the string "Hello world!":

Example

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

The output of the code above will be: 12.

Count The Number of Words in a String

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

Example

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

The output of the code above will be: 2.

Reverse a String

The PHP `strrev()` function reverses a string:

Example

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

The output of the code above will be: !dlrow olleH.

Search For a Specific Text Within a String

The PHP `strpos()` function searches for a specific text within a string.

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

The example below searches for the text "world" in the string "Hello world!":

Example

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

The output of the code above will be: 6.

Tip: The first character position in a string is 0 (not 1).

Replace Text Within a String

The PHP `str_replace()` function replaces some characters with some other characters in a string.

The example below replaces the text "world" with "Dolly":

Example

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

The output of the code above will be: Hello Dolly!

Complete PHP String Reference

For a complete reference of all string functions, go to our complete PHP String Reference.

The PHP string reference contains description and example of use, for each function!

PHP 5 Constants

Constants are like variables except that once they are defined they cannot be changed or undefined.

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Note: Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

To create a constant, use the `define()` function.

Syntax

```
define(name, value, case-insensitive)
```

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

The example below creates a constant with a **case-sensitive** name:

Example

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

The example below creates a constant with a **case-insensitive** name:

Example

```
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

Constants are Global

Constants are automatically global and can be used across the entire script.

The example below uses a constant inside a function, even if it is defined outside the function:

Example

```
<?php
define("GREETING", "Welcome to W3Schools.com!");

function myTest() {
    echo GREETING;
}

myTest();
?>
```

PHP Operators

Operators are used to perform operations on variables and values.
PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result	Show it

+	Addition	$x + $y	Sum of $x and $y	Show it »
-	Subtraction	$x - $y	Difference of $x and $y	
*	Multiplication	$x * $y	Product of $x and $y	Show it »
/	Division	$x / $y	Quotient of $x and $y	
%	Modulus	$x % $y	Remainder of $x divided by $y	Show it »
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)	

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description	Show it
x = y	x = y	The left operand gets set to the value of the expression on the right	Show it »
x += y	x = x + y	Addition	

x -= y	x = x - y	Subtraction	Show it »
x *= y	x = x * y	Multiplication	
x /= y	x = x / y	Division	Show it »
x %= y	x = x % y	Modulus	

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result	Show it
==	Equal	$x == $y	Returns true if $x is equal to $y	Show it »
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type	
!=	Not equal	$x != $y	Returns true if $x is not equal to $y	Show it »
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y	
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type	Show it »

>	Greater than	$x > $y	Returns true if $x is greater than $y	
<	Less than	$x < $y	Returns true if $x is less than $y	Show it »
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y	
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y	Show it »

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description	Show it
++$x	Pre-increment	Increments $x by one, then returns $x	Show it »
$x++	Post-increment	Returns $x, then increments $x by one	
--$x	Pre-decrement	Decrements $x by one, then returns $x	Show it »
$x--	Post-decrement	Returns $x, then decrements $x by one	

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result	Show it
and	And	$x and $y	True if both $x and $y are true	Show it »
or	Or	$x or $y	True if either $x or $y is true	
xor	Xor	$x xor $y	True if either $x or $y is true, but not both	Show it »
&&	And	$x && $y	True if both $x and $y are true	
||	Or	$x || $y	True if either $x or $y is true	Show it »
!	Not	!$x	True if $x is not true	

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result	Show it
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2	Show it »
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1	

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result	Show it
+	Union	$x + $y	Union of $x and $y	Show it »
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs	
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types	Show it »
!=	Inequality	$x != $y	Returns true if $x is not equal to $y	
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y	Show it »
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y	

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- `if` statement - executes some code if one condition is true
- `if...else` statement - executes some code if a condition is true and another code if that condition is false
- `if...elseif....else` statement - executes different codes for more than two conditions
- `switch` statement - selects one of many blocks of code to be executed

PHP - The if Statement

The `if` statement executes some code if one 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 (HOUR) is less than 20:

Example

```
<?php
$t = date("H");

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

PHP - The if...else Statement

The `if....else` statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
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 day!" if the current time is less than 20, and "Have a good night!" otherwise:

Example

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
```

```
}
?>
```

PHP - The if...elseif....else Statement

The `if....elseif...else` statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are 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 ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

The `switch` statement is 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;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

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 favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP while loops execute a block of code while the 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 code-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 as long as the specified condition is true

- `do...while` - loops through a block of code once, and then repeats the loop as long as the 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

The PHP while Loop

The `while` loop executes a block of code as long as the specified condition is true.

Syntax

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

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

Example

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

The PHP do...while Loop

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

Syntax

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

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

Example

```
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

Notice that in a `do while` loop the condition is tested AFTER executing the statements within the loop. This means that the `do while` loop would execute its statements at least once, even if the condition is false the first time.

The example below sets the $x variable to 6, then it runs the loop, **and then the condition is checked**:

Example

```
<?php
$x = 6;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

PHP `for` loops execute a block of code a specified number of times.

The PHP for Loop

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

Syntax

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

Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

The example below displays the numbers from 0 to 10:

Example

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

The PHP foreach Loop

The `foreach` loop works only on arrays, and is used to loop through each key/value pair in an array.

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, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

The real power of PHP comes from its functions; it has more than 1000 built-in functions.

PHP User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute immediately when a page loads.

A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user-defined function declaration starts with the word `function`:

Syntax

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

Note: A function name can start with a letter or underscore (not a number).

Tip: Give the function a name that reflects what the function does!

Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code, and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name followed by brackets ():

Example

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
```

```
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

The following example has a function with two arguments ($fname and $year):

Example

```
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Functions - Returning values

To let a function return a value, use the `return` statement:

Example

```
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

PHP 5 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] . ".";
?>
```

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.

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

PHP Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

```
$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] . ".";
?>
```

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);
?>
```

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>";
}
?>
```

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.";
?>
```

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>";
}
?>
```

PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array you need two indices to select an element
- For a three-dimensional array you need three indices to select an element

PHP - Two-dimensional Arrays

A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

First, take a look at the following table:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.

To get access to the elements of the $cars array we must point to the two indices (row and column):

Example

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

We can also put a `for` loop inside another `for` loop to get the elements of the $cars array (we still have to point to the two indices):

Example

```
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

PHP 5 File Handling

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files

PHP has several functions for creating, reading, uploading, and editing files.

Be careful when manipulating files!

When you are manipulating files you must be very careful.

You can do a lot of damage if you do something wrong. Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.

PHP readfile() Function

The `readfile()` function reads a file and writes it to the output buffer.

Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

The PHP code to read the file and write it to the output buffer is as follows (the `readfile()` function returns the number of bytes read on success):

Example

```
<?php
echo readfile("webdictionary.txt");
?>
```

The `readfile()` function is useful if all you want to do is open up a file and read its contents.

PHP 5 File Open/Read/Close

In this chapter we will teach you how to open, read, and close a file on the server.

PHP Open File - fopen()

A better method to open files is with the `fopen()` function. This function gives you more options than the `readfile()` function.

We will use the text file, "webdictionary.txt", during the lessons:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

The first parameter of `fopen()` contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified file:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

Tip: The `fread()` and the `fclose()` functions will be explained below.

The file may be opened in one of the following modes:

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists

r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

PHP Read File - fread()

The `fread()` function reads from an open file.

The first parameter of `fread()` contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

The following PHP code reads the "webdictionary.txt" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

PHP Close File - fclose()

The `fclose()` function is used to close an open file.

It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!

The `fclose()` requires the name of the file (or a variable that holds the filename) we want to close:

```
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
```

```
fclose($myfile);
?>
```

PHP Read Single Line - fgets()

The `fgets()` function is used to read a single line from a file.

The example below outputs the first line of the "webdictionary.txt" file:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

Note: After a call to the `fgets()` function, the file pointer has moved to the next line.

PHP Check End-Of-File - feof()

The `feof()` function checks if the "end-of-file" (EOF) has been reached.

The `feof()` function is useful for looping through data of unknown length.

The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

PHP Read Single Character - fgetc()

The `fgetc()` function is used to read a single character from a file.

The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

Example

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
```

Note: After a call to the `fgetc()` function, the file pointer moves to the next character.

PHP 5 File Create/Write

In this chapter we will teach you how to create and write to a file on the server.

PHP Create File - fopen()

The `fopen()` function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

If you use `fopen()` on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Example

```
$myfile = fopen("testfile.txt", "w")
```

PHP File Permissions

If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

PHP Write to File - fwrite()

The `fwrite()` function is used to write to a file.

The first parameter of `fwrite()` contains the name of the file to write to and the second parameter is the string to be written.

The example below writes a couple of names into a new file called "newfile.txt":

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the `fclose()`function.

If we open the "newfile.txt" file it would look like this:

```
John Doe
Jane Doe
```

PHP Overwriting

Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.

In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

```
Mickey Mouse
Minnie Mouse
```

PHP 5 File Upload

With PHP, it is easy to upload files to the server.

However, with ease comes danger, so always be careful when allowing file uploads!

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the `file_uploads` directive, and set it to On:

```
file_uploads = On
```

Create The HTML Form

Next, create an HTML form that allow users to choose the image file they want to upload:

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

Some rules to follow for the HTML form above:

- Make sure that the form uses method="post"
- The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form

Without the requirements above, the file upload will not work.

Other things to notice:

- The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

The form above sends data to a file called "upload.php", which we will create next.

Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType =strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
?>
```

PHP script explained:

- $target_dir = "uploads/" - specifies the directory where the file is going to be placed
- $target_file specifies the path of the file to be uploaded
- $uploadOk=1 is not used yet (will be used later)
- $imageFileType holds the file extension of the file (in lower case)
- Next, check if the image file is an actual image or a fake image

Note: You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

Check if File Already Exists

Now we can add some restrictions.

First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

Limit File Size

The file input field in our HTML form above is named "fileToUpload".

Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:

```
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

Limit File Type

The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

Complete Upload File PHP Script

The complete "upload.php" file now looks like this:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType =strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

```
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been
uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

PHP 5 MySQLi Functions

Function	Description
mysqli_affected_rows()	Returns the number of affected rows in the previous MySQL operation
mysqli_autocommit()	Turns on or off auto-committing database

	modifications
mysqli_change_user()	Changes the user of the specified database connection
mysqli_character_set_name()	Returns the default character set for the database connection
mysqli_close()	Closes a previously opened database connection
mysqli_commit()	Commits the current transaction
mysqli_connect_errno()	Returns the error code from the last connection error
mysqli_connect_error()	Returns the error description from the last connection error
mysqli_connect()	Opens a new connection to the MySQL server
mysqli_data_seek()	Adjusts the result pointer to an arbitrary row in the result-set
mysqli_debug()	Performs debugging operations
mysqli_dump_debug_info()	Dumps debugging info into the log

mysqli_errno()	Returns the last error code for the most recent function call
mysqli_error_list()	Returns a list of errors for the most recent function call
mysqli_error()	Returns the last error description for the most recent function call
mysqli_fetch_all()	Fetches all result rows as an associative array, a numeric array, or both
mysqli_fetch_array()	Fetches a result row as an associative, a numeric array, or both
mysqli_fetch_assoc()	Fetches a result row as an associative array
mysqli_fetch_field_direct()	Returns meta-data for a single field in the result set, as an object
mysqli_fetch_field()	Returns the next field in the result set, as an object
mysqli_fetch_fields()	Returns an array of objects that represent the fields in a result set
mysqli_fetch_lengths()	Returns the lengths of the columns of the current row in the result set

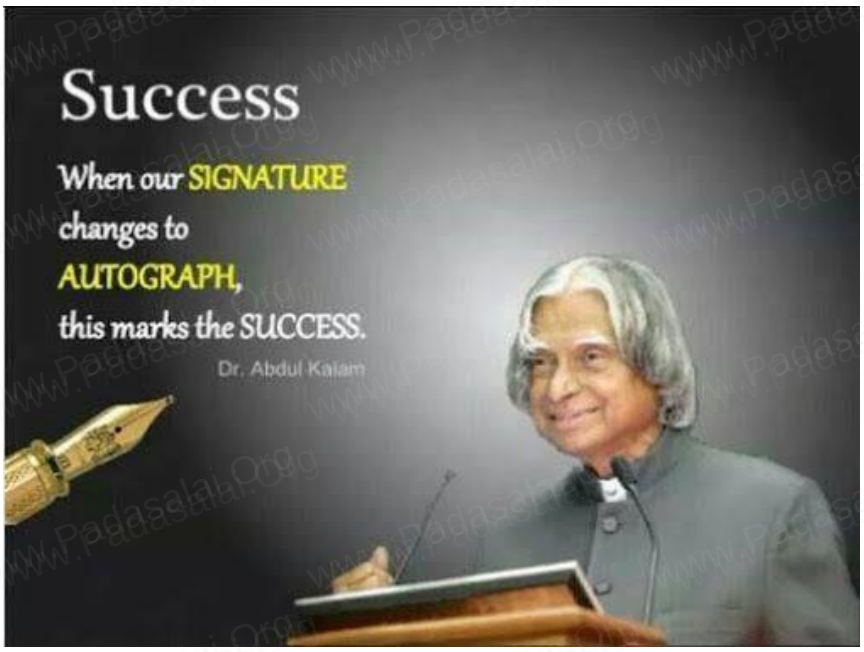
mysqli_fetch_object()	Returns the current row of a result set, as an object
mysqli_fetch_row()	Fetches one row from a result-set and returns it as an enumerated array
mysqli_field_count()	Returns the number of columns for the most recent query
mysqli_field_seek()	Sets the field cursor to the given field offset
mysqli_field_tell()	Returns the position of the field cursor
mysqli_free_result()	Frees the memory associated with a result
mysqli_get_charset()	Returns a character set object
mysqli_get_client_info()	Returns the MySQL client library version
mysqli_get_client_stats()	Returns statistics about client per-process
mysqli_get_client_version()	Returns the MySQL client library version as an integer
mysqli_get_connection_stats()	Returns statistics about the client connection

mysqli_get_host_info()	Returns the MySQL server hostname and the connection type
mysqli_get_proto_info()	Returns the MySQL protocol version
mysqli_get_server_info()	Returns the MySQL server version
mysqli_get_server_version()	Returns the MySQL server version as an integer
mysqli_info()	Returns information about the most recently executed query
mysqli_init()	Initializes MySQLi and returns a resource for use with mysqli_real_connect()
mysqli_insert_id()	Returns the auto-generated id used in the last query
mysqli_kill()	Asks the server to kill a MySQL thread
mysqli_more_results()	Checks if there are more results from a multi query
mysqli_multi_query()	Performs one or more queries on the database
mysqli_next_result()	Prepares the next result set from mysqli_multi_query()

mysqli_num_fields()	Returns the number of fields in a result set
mysqli_num_rows()	Returns the number of rows in a result set
mysqli_options()	Sets extra connect options and affect behavior for a connection
mysqli_ping()	Pings a server connection, or tries to reconnect if the connection has gone down
mysqli_prepare()	Prepares an SQL statement for execution
mysqli_query()	Performs a query against the database
mysqli_real_connect()	Opens a new connection to the MySQL server
mysqli_real_escape_string()	Escapes special characters in a string for use in an SQL statement
mysqli_real_query()	Executes an SQL query
mysqli_reap_async_query()	Returns the result from async query
mysqli_refresh()	Refreshes tables or caches, or resets the replication server information

mysqli_rollback()	Rolls back the current transaction for the database
mysqli_select_db()	Changes the default database for the connection
mysqli_set_charset()	Sets the default client character set
mysqli_set_local_infile_default()	Unsets user defined handler for load local infile command
mysqli_set_local_infile_handler()	Set callback function for LOAD DATA LOCAL INFILE command
mysqli_sqlstate()	Returns the SQLSTATE error code for the last MySQL operation
mysqli_ssl_set()	Used to establish secure connections using SSL
mysqli_stat()	Returns the current system status
mysqli_stmt_init()	Initializes a statement and returns an object for use with mysqli_stmt_prepare()
mysqli_store_result()	Transfers a result set from the last query
mysqli_thread_id()	Returns the thread ID for the current connection

mysqli_thread_safe()	Returns whether the client library is compiled as thread-safe
mysqli_use_result()	Initiates the retrieval of a result set from the last query executed using the mysqli_real_query()
mysqli_warning_count()	Returns the number of warnings from the last query in the connection



ALL THE BEST

1. **Difference between Get and Post method**

GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
4) Get request is **idempotent** . It means second request will be ignored until response of first request is delivered	Post request is **non-idempotent.**
5) Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.

2. **Define HTML form Controls.**
   - ✓ An **HTML form** is a section of a document containing normal content, markup, special elements called **controls**(checkboxes, radio buttons, menus, etc.), and labels on those **controls**.
   - ✓ HTML controls and changes their appearance by modifying their properties.
3. **Define Form handling method in PHP.**
   - ✓ When the user keying the input data in HTML controls and clicks the submit button the request will be generated and reaches a PHP file which is mentioned in the FORM tag under the Action attribute.
   - ✓ All input values are synchronized and sent to the server via POST method or GET method.
4. **What is form validation in php?**
   - ✓ Validation is a process of checking the input data submitted by the user from client machine.
   - ✓ There are two types of validation available in PHP.
   - ✓ **Client-Side Validation**: The input data validations are performed on the client machine's web browsers using client side scripts like Java script or adding
   - ✓ "required" attribute in HTML input tags.
   - ✓ **Server Side Validation**: After the submission of data, validations are performed on the server side using the programming like PHP, ASP or JSP etc, available in the server machine.

5. List out the html control to support PHP language?

The HTML form controls are

- ✓ Text inputs
- ✓ Buttons
- ✓ Checkbox
- ✓ Radio box
- ✓ File Select
- ✓ Form Tag

6. Write the syntax of text box in html.

```
<input type="text" name=”name of the text box“>
```

7. Define file handling in php.

File handling is an important activity of all web application development process. Files are processed for different tasks using the following events:

- ✓ PHP Open a File,
- ✓ PHP Read a File,
- ✓ PHP Close a File,
- ✓ PHP Write a File,
- ✓ PHP Appending a File and
- ✓ PHP uploading a File.

8. Define browse button in html.

Defines a file upload box with a browse button.

<input type="file">

Example:

<input type=”file” name=”fileToUpload” id=”fileToUpload”>

In a form, the file value of the type attribute allows you to define an input element for file uploads.

This displays a browse button, which the user can click on to select a file on their local computer.

9. Write the syntax of browse button in html.

<input type="file">

Example:

<input type=”file” name=”fileToUpload” id=”fileToUpload”>

In a form, the file value of the type attribute allows you to define an input element for file uploads.

This displays a browse button, which the user can click on to select a file on their local computer.

10. Usage of file open ().

fopen() is a system function available in PHP. This function helps to open a file in the server.

It contains two parameters one for the file and the other one specifies in which mode the file should be opened (Read/Write).

11. Write the features form handling.

12. Write the purpose Get method and post method.

Post Method: The input data sent to the server with POST method is stored in the request body of the client's HTTP request.

Get Method: The input data sent to the server with POST method via URL address is known as query string. All input data are visible by user after they clicks the submit button.

13. Write short notes on File handling.

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP has several functions for creating, reading, uploading, and editing files.

Files are processed for different tasks using the following events:

- ✓ PHP Open a File,
- ✓ PHP Read a File,
- ✓ PHP Close a File,
- ✓ PHP Write a File,
- ✓ PHP Appending a File and
- ✓ PHP uploading a File

14. Compare Text box and text area.

Sno	Text Box	Text Area
1	The <input type="text"> tag defines a one-line text input control.	The <textarea> tag defines a multi-line text input control.

2	`<input type="text" name="name of the text box">`	`<textarea rows="10" cols="10 ">`
		
		`</textarea>`

15. Write short notes on File handling functions.

PHP Open a File

fopen() is a system function available in PHP. This function helps to open a file in the server. It contains two parameters one for the file and the other one specifies in which mode the file should be opened (Read/Write).

Syntax:

$file_Object= fopen("FileName", "Read/WriteMode") or die("Error Message!");

Example:

```
<?php
$myfile = fopen("Student.txt", "r") or die("Unable to open file!");
?>
```

PHP Read a File:

The fread() function reads from an open file. The file object comes from fopen function.

Syntax:

fread($file_Object,filesize("FileName"));

Example:

```
<?php
fread($myfile,filesize("Student.txt"));
?>
```

PHP Close a File:

The fclose() function is used to close an opened file. The file object comes from fopen function.

Syntax:

fclose($file_Object);

Example:

```
<?php
$myfile = fopen("student.txt", "r");
fclose($myfile);
?>
```

PHP write a File:

The fwrite() function is used to write to a file.

Syntax:

fwrite($myfile, $txt);

Example:

```
<?php
$myfile = fopen("new_school_file.txt", "w") or die("Unable to open file!");
$txt = "School Name\n";
fwrite($myfile, $txt);
$txt = "Student Name\n";
fwrite($myfile, $txt);
```

```
fclose($myfile);
?>
```

PHP Appending a File

The file_put_contents() function is used to Append to a file. Look at Table 8.1 for various parameters used in appending a file.

Syntax:

file_put_contents(file,data,mode,context)