Unit - I

INTRODUCTION

1. Origin of PHP

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including My SQL, Postgre SQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The My SQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first t.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.
- Common uses of PHP.

2. PHP is better than its alternatives

- It is faster to code and faster to execute.
- The same PHP code runs unaltered on different Web servers and different operation systems.
- Additionally, functionality that is standard with PHP is an add-on in other environments. A more detailed argument follows.

- PHP is free. Anyone may visit the PHP Web site and download the complete source code. Binaries are also available for Windows. The result is easy entry into the experience. There is very little risk in trying PHP, and its license allows the code to be used to develop works with no royalties.
- This is unlike products such as Allaire's Cold Fusion or Everyware's Tango Enterprise that charge thousands of dollars for the software to interpret and serve scripts. Even commercial giants like Netscape and IBM now recognize the advantages of making source code available.
- PHP runs on UNIX, Windows 98, Windows NT, and the Macintosh. PHP is designed to integrate with the Apache Web Server.
- Apache, another free technology, is the most popular Web server on the Internet and comes with source code for UNIX and Windows. Commercial flavors of Apache like Web Ten and Stronghold support PHP, too.
- But PHP works with other Web servers, including Microsoft's Internet Information Server. Scripts may be moved between server platforms without alteration. PHP supports ISAPI to allow for the performance benefits of tightly coupling with Microsoft Web servers.

3. How PHP works with the web server

- The PHP software works with the web server, which is the software that delivers web pages to the world.
- When you type a URL into your web browser’s address bar, you’re sending a message to the web server at that URL, asking it to send you an HTML file. The web server responds by sending the requested file. Your browser reads the HTML file and displays the web page.
- You also request a file from the web server when you click a link in a web page.

- In addition, the web server processes a file when you click a web page button that submits a form. This process is essentially the same when PHP is installed. You request a file, the web server happens to be running PHP, and it sends HTML back to the browser, thanks to the programming in PHP.
- More specifically, when PHP is installed, the web server is configured to expect certain file extensions to contain PHP language statements.
- Often the extension is .php or .phtml, but any extension can be used. When the web server gets a request for a file with the designated extension, it sends the HTML statements as is, but PHP statements are processed by the PHP software before they're sent to the requester.
- When PHP language statements are processed, only the output, or anything printed to the screen is sent by the web server to the web browser.
- The PHP language statements, those that don't produce any output to the screen, aren't included in the output sent to the browser, so the PHP code is not normally seen by the user.
- For instance, in this simple PHP statement, <?php is the PHP opening tag, and ?> is the closing tag.
- <?php echo "<p>Hello World</p>"; ?>
- Here, echo is a PHP instruction that tells PHP to output the upcoming text. The PHP software processes the PHP statement and outputs the following:
- <p>Hello World</p>
- That regular HTML statement is delivered to the user's browser. The browser interprets the statement as HTML code and displays a web page with one paragraph — Hello World.
- The PHP statement isn't delivered to the browser, so the user never sees any PHP statements. PHP and the web server must work closely together.

4. Hardware and Software Requirements and installation

These are the following the software requirements

- WAMP Server
- LAMP Server
- MAMP Server
- XAMPP Server

All these types of software automatic configure inside operating system after installation it having PHP, My SQL, Apache and operating system base configuration file, it doesn't need to configure manually.

Server	**Stands for**
WAMP	**Microsoft window o/s, Apache Mysql PHP**
LAMP	**Linux Operating System Apache Mysql PHP**
MAMP	**Mac os Apache Mysql PHP**
XAMPP	**x-os(cross operating system) Apache Mysql PHP Perl**

Hardware Requirements

- If you would like to use a dedicated hosting server for running PHPKB Knowledge Base Software or host it on a local server, you can use hardware requirements and example specifications below for a reference.
- Recommended Hardware Specifications
- On small instances, server load is primarily driven by peak visitors however you may use the following specifications for optimal performance.
- 5 Concurrent Users 25 Concurrent Users 50 Concurrent Users
- 1 GHz CPU

- 2 GB RAM
- 1 GB disk space
- Dual 2 GHz CPU
- 4 GB RAM
- 1 GB disk space
- Dual 2.6 GHz CPU
- 8 GB RAM
- 4 GB disk space

<u>The PHP Installer</u>

- ➢ Although an installer is available from php.net, I would recommend the manual installation if you already have a web server configured and running.
- ➢ Manual Installation
- ➢ Manual installation offers several benefits:
- ➢ backing up, reinstalling, or moving the web server can be achieved in seconds (see 8 Tips for Surviving PC Failure) and
- ➢ you have more control over PHP and Apache configuration.

Step 1: Download the files

- ✓ Download the latest PHP 5 ZIP package from www.php.net/downloads.php
- ✓ As always, virus scan the file and check its MD5 checksum using a tool such as fsum.

Step 2: Extract the files

- ✓ We will install the PHP files to C:\php, so create that folder and extract the contents of the ZIP file into it.
- ✓ PHP can be installed anywhere on your system, but you will need to change the paths referenced in the following steps.

Step 3: Configure php.ini

- ✓ Copy C:\php\php.ini-development to C:\php\php.ini. There are several lines you will need to change in a text editor (use search to find the current setting).
- ✓ Where applicable, you will need to remove the leading semicolon to uncomment these setting.

Define the extension directory:

extension_dir = "C:/php/ext"

Enable extensions.

This will depend on the libraries you want to use, but the following extensions should be suitable for the majority of applications:

extension=curl

extension=gd2

extension=mbstring

extension=mysql

extension=pdo_mysql

extension=xmlrpc

If you want to send emails using the PHP mail() function, enter the details of an SMTP server (your ISP’s server should be suitable):

```
[mail function]

; For Win32 only.

SMTP = mail.myisp.com

smtp_port = 25

; For Win32 only.

sendmail_from = my@emailaddress.com
```

Step 4: Add C:\php to the path environment variable

- ✓ To ensure Windows can find PHP, you need to change the path environment variable.
- ✓ Open Settings, type ‘environment variables’ into the search field and open the result.
- ✓ Select the “Advanced” tab, and click the “Environment Variables” button.
- ✓ Scroll down the System variables list and click on “Path” followed by the “Edit” button. Click “Edit text” and add ;C:\php to the end of the Variable value line (remember the semicolon).

5. PHP pros and cons

5.1. PHP : past, present, future(PHP3.O,PHP4.O,PHP5.O)

- PHP3 is oldest stable version and it was pure procedural language constructive like C
- Whereas PHP4 have some OOPs concept added like class and object with new functionality
- PHP5 approximately all major oops functionality has been added along with below thing
- Implementation of exceptions and exception handling
- Type hinting which allows you to force the type of a specific argument
- Overloading of methods through the __call function
- Full constructors and destructors etc through a __constuctor and __destructor function
- __autoload function for dynamically including certain include files depending on the class you are trying to create.
- Finality : can now use the final keyword to indicate that a method cannot be overridden by a
- child. You can also declare an entire class as final which prevents it from having any children at all.
- Interfaces & Abstract Classes
- Passed by Reference :
- An __clone method if you really want to duplicate an object
- Numbers of Functions Deprecated or removed in PHP 5.x like ereg,ereg_replace,magic_quotes, session_register,register_globals, split(), call_user_method() etc

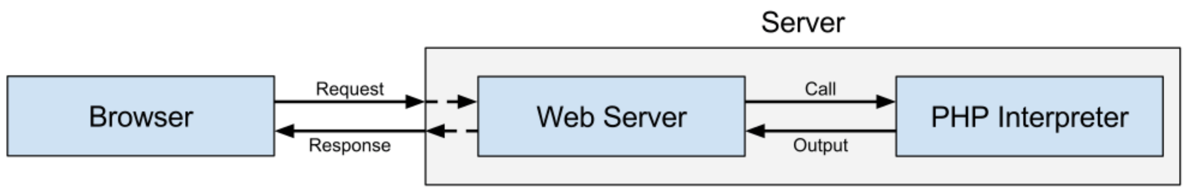
6 .Strength of Php

- Supports database connectivity. PHP can access over 20 different databases including My Sql, Oracle, and MS Access.
- Supports sessions. PHP can generate unique session IDs. The ID follows the user during a single session on a web site. This simplifies creating shopping cart applications and tracking user's behaviors.
- Eliminates client configuration problems. With PHP there is no need to worry if the client has the appropriate software installed, since the application is executed on the server.
- Reduces development time. Even a newcomer can begin developing PHP applications in hours. Yet PHP contains many advanced features for professional programmers.
- Maintains source code security. The user does not see your source code as they do with JavaScript.

7. Basic PHP Development

7. 1 How PHP script work

- PHP always start with a browser making a request for a web page.
- This request is going to hit the web server.
- The web server will then analyze it and determine what to do with it.
- If the web server determines that the request is for a PHP file (often `index.php`), it'll pass that file to the PHP interpreter.
- The PHP interpreter will read the PHP file, parse it (and other included files) and then execute it.
- Once the PHP interpreter finishes executing the PHP file, it'll return an output. The web server will take that output and send it back as a response to the browser.



7. 2 Basic PHP syntax

- ✓ PHP script starts with the <?php and ends with the ?> tag.
- ✓ The PHP delimiter <?php and ?> in the following example simply tells the PHP engine to treat the enclosed code block as PHP code, rather than simple HTML.

Syntax :

```
<?php
// Some code to be executed
echo "Hello, world!";
?>
```

- ✓ Every PHP statement end with a semicolon (;) — this tells the PHP engine that the end of the current statement has been reached.

Comment Lines

- ➢ A comment is simply text that is ignored by the PHP engine. The purpose of comments is to make the code more readable.
- ➢ It may help other developer (or you in the future when you edit the source code) to understand what you were trying to do with the PHP.
- ➢ PHP support single-line as well as multi-line comments.
- ➢ To write a single-line comment either start the line with either two slashes (//) or a hash symbol (#).

<u>For example:</u>

```
<?php
// This is a single line comment
# This is also a single line comment
echo "Hello, world!";
?>
```

However to write multi-line comments, start the comment with a slash followed by an asterisk (/*) and end the comment with an asterisk followed by a slash (*/).

<u>Case Sensitivity in PHP</u>

Variable names in PHP are case-sensitive.

As a result the variables $color, $Color and $COLOR are treated as three different variables.

```
<?php
// Assign value to variable
$color = "blue";
// Try to print variable value
echo "The color of the sky is " . $color . "<br>";
echo "The color of the sky is " . $Color . "<br>";
echo "The color of the sky is " . $COLOR . "<br>";
?>
```

<u>7.3 PHP variables</u>

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.

- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

7.4.PHP Data types

PHP has a total of eight data types which we use to construct our variables –

- Integers − are whole numbers, without a decimal point, like 4195.
- Doubles − are floating-point numbers, like 3.14159 or 49.1.
- Booleans − have only two possible values either true or false.
- NULL − is a special type that only has one value: NULL.
- Strings − are sequences of characters, like 'PHP supports string operations.'
- Arrays − are named and indexed collections of other values.
- Objects − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- Resources − are special variables that hold references to resources external to PHP (such as database connections).

Integers

- They are whole numbers, without a decimal point, like 4195.
- They are the simplest type.
- They correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like
- $int_var = 12345;
- $another_int = -12345 + 12345;
- Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format.
- Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.
- **Doubles**

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed.

For example, the code

```
<?php

$many = 2.2888800;

$many_2 = 2.2111200;

$few = $many + $many_2;

print("$many + $many_2 = $few <br>");

?>
```

- It produces the following browser output −
- 2.28888 + 2.21112 = 4.5

Boolean

- They have only two possible values either true or false.
- PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so −
- **if (TRUE)**
- **print("This will always print<br>");**
- **else**
- **print("This will never print<br>"**

NULL

- NULL is a special type that only has one value: NULL.
- To give a variable the NULL value, simply assign it like this −
- $my_var = NULL;
- The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed −
- $my_var = null;
- A variable that has been assigned NULL has the following properties −
- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with IsSet() function.

Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

- $string_1 = "This is a string in double quotes";
- $string_2 = 'This is a somewhat longer, singly quoted string';
- $string_39 = "This string has thirty-nine characters";
- $string_0 = ""; // a string with zero characters.
- 8.Displaying Type Information with var_dump():
- var_dump() provides information about all the types contained within the variable, as well as about the variable itself.
- $testing = 5;
- var_dump($testing);
- Output
- int(5)

7.5. Displaying type information

- **This function displays structured information about one or more expressions that includes its type and value.**
- **Arrays and objects are explored recursively with values indented to show structure.**

Example #1 var_dump()

```
<?php
$a = array(1, 2, array("a", "b", "c"));
var_dump($a);
?>
```

The above example will output:

```
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}
```

```
<?php

$b = 3.1;
$c = true;
var_dump($b, $c);
```

```
?>
```

The above example will output:

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

7.6. Testing for a Specific Data Type

- PHP provides a special function corresponding to each data type.
- These functions accept a variable or value and return a Boolean

Function

- is_array()
- Returns true if the argument is an array
- is_bool()
- Returns true if the argument is boolean
- is_double()
- Returns true if the argument is a double
- is_int()
- Returns true if the argument is an integer
- is_object()
- Returns true if the argument is an object

- is_string()
- Returns true if the argument is a string
- is_null()
- Returns true if the argument is null
- is_resource()
- Returns true if the argument is a resource

7.7.Operators

- PHP Operator is a symbol i.e used to perform operations on operands
- In simple words, operators are used to perform operations on variables or values.

For example:

$num=10+20;//+ is the operator and 10,20 are operands

In the above example, + is the binary + operator, 10 and 20 are operands and $num is variable.

PHP Operators can be categorized in following forms:

a. Arithmetic Operators
b. Assignment Operators
c. Bitwise Operators
d. Comparison Operators
e. Incrementing/Decrementing Operators
f. Logical Operators
g. String Operators
h. Array Operators

Arithmetic operators:

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

Operator	Name	Example	Explanation
+	Addition	$a + $b	Sum of operands
-	Subtraction	$a - $b	Difference of operands
*	Multiplication	$a * $b	Product of operands
/	Division	$a / $b	Quotient of operands
%	Modulus	$a % $b	Remainder of operands
**	Exponentiation	$a ** $b	$a raised to the power $b

Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	**Name**	**Example**	**Explanation**
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a += $b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b
/=	Divide then Assign (quotient)	$a /= $b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Find remainder same as $a = $a % $b

Bitwise operators

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise 0.
|	Or (Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor (Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1

Comparison Operators

Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b

Return 0 if $a is equal $b

Return 1 if $a is greater than $b

<u>Incrementing/Decrementing Operators</u>

The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
$a++	Return $a, then increment the value of $a by one		
--	decrement	--$a	Decrement the value of $a by one, then return $a
$a--	Return $a, then decrement the value of $a by one		

<u>Logical Operators</u>

<u>T</u>he logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
and	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

String Operators

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenate both $a and $b
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

Array Operators

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order
!==	Non-Identity	$a !== $b	Return TRUE if $a is not identical to $b
<>	Inequality	$a <> $b	Return TRUE if $a is not equal to $b

7.7 Variable manipulation

★ Variable manipulation is a method of specifying or editing variables in a computer program.

★ Variable manipulation can be used to create dynamic content in HTML and programming languages. However, the method can also be used by an attacker, to alter data that a browser sends to a Web server.

- ★ A variable, in this context, is a string of text symbols and/or numerals that can change, depending on conditions or on information passed to the program.
- ★ In mathematics, variable manipulation is the process of rearranging a multivariable equation to express a specific variable as a function of other variables.
- ★ The variable thus singled-out is known as the dependent variable, while the other variables are called independent variables.
- ★ In this context, a variable is a mathematical quantity whose numerical value can change, in contrast to constants that have fixed numerical values.

7.8 .Dynamic variables

- ➢ PHP allows us to use dynamic variable.
- ➢ Dynamic variable is variable variables.
- ➢ We can name a variable with the value stored in another variable.
- ➢ That is, one variable contains the name of another variable.

Example:

```
$text = ‘Hello World’;
$dynamicText = “text”;
echo ‘The dynamic ‘.$dynamicText. ‘ is ‘.${$dynamicText};
```

Output : The dynamic text is Hello World

We can implement it ${$dynamicText} or $$dynamicText .

Here we can associate variable to another variable. We can use it as alternative of associative array and list. The time complexity also less then array and list also.

7.8 String in PHP

- A string is a collection of characters. String is one of the data types supported by PHP.
- The string variables can contain alphanumeric characters. Strings are created when;
- You declare variable and assign string characters to it
- You can directly use them with echo statement.
- String are language construct, it helps capture words.
- Learning how strings work in PHP and how to manipulate them will make you a very effective and productive developer.

8 .Control structures

- ➢ Control structure allows you to control the flow of code execution in your application.
- ➢ Generally, a program is executed sequentially, line by line, and a control structure allows you to alter that flow, usually depending on certain conditions.

PHP supports a number of different control structures:

- ➢ if

- else
- elseif
- switch
- while
- do-while
- for
- foreach

8.1. The If statement

The if construct allows you to execute a piece of code if the expression provided along with it evaluates to true.

Example:

```
<?php

$age = 50;

if ($age > 30)

{

  echo "Your age is greater than 30!";

}

?>
```

8.2 Using the Else clause with if Statement

if the expression evaluates to true. On the other hand, if the expression evaluates to false, it won't do anything.

More often than not, you also want to execute a different code snippet if the expression evaluates to false.

Syntax

```
if (expression)
{
  // code is executed if the expression evaluates to TRUE
}
else
{
  // code is executed if the expression evaluates to FALSE
}
```

Example

```
<?php
$age = 50;
if ($age < 30)
{
  echo "Your age is less than 30!";
}
else
{
  echo "Your age is greater than or equal to 30!";
}
?>
```

8.3. Multiple if, nested If Statement

We can consider the elseif statement as an extension to the if-else construct.

If you've got more than two choices to choose from, you can use the elseif statement.

Syntax

```
if (expression1)

{

  // code is executed if the expression1 evaluates to TRUE

}

elseif (expression2)

{

  // code is executed if the expression2 evaluates to TRUE

}

elseif (expression3)

{

  // code is executed if the expression3 evaluates to TRUE

}

else

{

  // code is executed if the expression1, expression2 and expression3 evaluates to
FALSE, a default choice
```

Example

```
<?php
$age = 50;

if ($age < 30)
{
  echo "Your age is less than 30!";
}
elseif ($age > 30 && $age < 40)
{
  echo "Your age is between 30 and 40!";
}
elseif ($age > 40 && $age < 50)
{
  echo "Your age is between 40 and 50!";
}
else
{
  echo "Your age is greater than 50!";
}
?>
```

8.4 Switch Statement

- The switch statement is somewhat similar to the elseif statement which we've just discussed in the previous section.
- The only difference is the expression which is being checked.
- In the case of the elseif statement, you have a set of different conditions, and an appropriate action will be executed based on a condition.
- On the other hand, if you want to compare a variable with different values, you can use the switch statement.

```
<?php

$favourite_site = 'Code';

switch ($favourite_site) {

  case 'Business':

    echo "My favourite site is business.tutsplus.com!";

    break;

  case 'Code':

    echo "My favourite site is code.tutsplus.com!";

    break;

  case 'Web Design':

    echo "My favourite site is webdesign.tutsplus.com!";

    break;

  case 'Music':

    echo "My favourite site is music.tutsplus.com!";

    break;

  case 'Photography':
    echo "My favourite site is photography.tutsplus.com!";

    break;

  default:

    echo "I like everything at tutsplus.com!";

}

?>
```

8.5 ? Operators

This is called the Ternary Operator, and it's common to several languages, including PHP, Javascript, Python, Ruby...

```
$x = $condition ? $trueVal : $falseVal;

// same as:
```

Example:

```
if ($condition) {

    $x = $trueVal;

} else {

    $x = $falseVal;

} error
```

Unit - II

ARRAY

14. PHP Library Function

- **14.1 Array function**
- **14.2 String Function**
- **14.3 Date and Time function**

15. User defined function

- **15.1. Defining a Functions with Parameters and without Parameters**
- **15.2. Returning value from the function**
    - **15.3. Dynamic function calls accessing variable with global statement**
    - **15.4 setting default values for the arguments**
    - **15.5.Passing argument to function by value**
    - **15.6.Passing argument to function by reference**

1. Single dimensional array:

1.1How to create an array

Array creation is pretty simple. First, you need to create a variable and then tell PHP that you want that variable to be an array:

- ❖ $theVar = array();
- ❖ Now, $theVar is an array. However, it's an empty array waiting for you to come along and fill it.
- ❖ Technically, you can skip the variable creation step.
- ❖ It's still a good idea to explicitly define an array because it helps you remember the element is an array, and there are a few special cases (such as passing an array into a function) where the definition really matters.
- ❖ How to fill an array
- ❖ An array is a container, so it's a lot more fun if you put something in it.
- ❖ You can refer to an array element by adding an index (an integer) representing which element of the array you're talking about.

2. Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example

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

```
<?php
   $marks = array(
      "mohammad" => array (
         "physics" => 35,
         "maths" => 30,
         "chemistry" => 39
      ),

      "qadir" => array (
         "physics" => 30,
         "maths" => 32,
         "chemistry" => 29
      ),

      "zara" => array (
         "physics" => 31,
         "maths" => 22,
         "chemistry" => 39
      )
   );

   /* Accessing multi-dimensional array values */
   echo "Marks for mohammad in physics : " ;
   echo $marks['mohammad']['physics'] . "<br />";

   echo "Marks for qadir in maths : ";
```

```
        echo $marks['qadir']['maths'] . "<br />";
        echo "Marks for zara in chemistry : " ;
        echo $marks['zara']['chemistry'] . "<br />";
      ?>
   </body>
</html>
```

Output:

```
Marks for mohammad in physics : 35
Marks for qadir in maths : 32
Marks for zara in chemistry : 39
```

3.Casting arrays

- ★ The meaning of type casting is to use the value of a variable with different data type.
- ★ In other word typecasting is a way to utilize one data type variable into the different data type.
- ★ Typecasting is the explicit conversion of data type because user explicitly defines the data type in which he wants to cast.
- ★ In this explore various aspects of PHP Type casting.
- ★ PHP does not require or support type definition of the variable. In PHP we never define data type while declaring the variable.
- ★ In PHP variables automatically decide the data type on the basis of the value assignment or context.

For example:

```
<?php
```

```
$i =1;
var_dump($i); //$i is integer
$i = 2.3;
var_dump($i); //$i is float
$i = "php type casting";
var_dump($i); //$i is string
?>
```

In above example, you can see that variable $i type is getting changed on the different type of value assignment.

So due to this flexible nature of the PHP, we do not need to type cast variable always.

But Sometimes when we need extra security in the variable we cast type. For example, if we are taking some integer input from user then we should type cast.

Type casting in PHP works same as C programming.

Desired data type name with parenthesis before the variable which we need to cast. For example, if we need to cast the string to the integer then below will work:

```
<?php
$string_var = "string value for php type";
$int_var = (int)$string_var;
var_dump($int_var);
?>
```

We can cast following data type variable in PHP

(int), (integer) - cast to integer

(bool), (boolean) - cast to boolean

(float), (double), (real) - cast to float (string) - cast to string

(array) - cast to array

(object) - cast to object

(unset) - cast to NULL (PHP 5)

PHP type Casting to Integer

Using (int) or (integer) keyword we can cast/convert any data type value to the integer. If we need to take integer casting then we can also use intval() function.

If we will convert boolean to an integer then False will output 0 and true will output 1. For example

```
<?php
$bool_false = false;
$int_val =  (int) $bool_false ;
var_dump($int_val); //Output will be 0
$bool_true = true;
$int_val =  (int) $bool_true ;
var_dump($int_val); //Output will be 1
?>
```

If we will convert resource data type to an integer then it will return unique resource ID. For example

```
<?
$fp = fopen("filename.txt", "w");
$int_cast = (int) $fp;
var_dump($int_cast);
?>
```

Llll

4. Associative Array

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index.

- Associative array will have their index as string so that you can establish a strong association between key and values.
- To store the salaries of employees in an array, a numerically indexed array would not be the best choice.
- Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

Example:

```
<html>
  <body>
    <?php
      /* First method to associate create array. */
      $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);
      echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
      echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
      echo "Salary of zara is ".  $salaries['zara']. "<br />"
      /* Second method to create array. */
      $salaries['mohammad'] = "high";
      $salaries['qadir'] = "medium";
```

```
         $salaries['zara'] = "low";
         echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
         echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
         echo "Salary of zara is ".  $salaries['zara']. "<br />";
      ?>
   </body>
</html>
```

This will produce the following result −

```
Salary of mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low.
```

<u>**5.Accessing array elements**</u>

Now that we have created an associative array and assigned names to each element we can use those names to access the corresponding array values. We can, therefore, extend our previous example to extract the customer name from our $customerArray:

```
<?php
      $customerArray = array('customerName'=>'John Smith',
'customerAddress'=>'1 The Street', 'accountNumber'=>'123456789');
      echo $customerArray['customerName'];
?>
```

The result will be the customer name "John Smith" appearing in the browser window.

6.Looping through an array and associative array.

- It is often necessary to loop through each element of an array to either read or change the values contained therein. There are a number of ways that this can be done.
- One such mechanism is to use the foreach loop. The foreach loop works much like a for or while loop and allows you to iterate through each array element. There are two ways to use foreach. The first assigns the value of the current element to a specified variable which can then be accessed in the body of the loop

The syntax for this is:

```
foreach ($arrayName as $variable)
```

For example:

```
    $colorArray = array("Red", "Yellow", "Green", "Blue", "Indigo");

    foreach ($colorArray as $color)
    {
       echo "$color <br>";
    }
```

This will result in the following output:

```
Red
Yellow
Green
Blue
Indigo
```

9.Looping through an associative array

For associative arrays the foreach keyword allows you to iterate through both the keys and the values using the following syntax:

foreach ($arrayName as $keyVariable=>$valueVariable)

For example:

```
    $customerArray = array('customerName'=>'John Smith',
'customerAddress'=>'1 The Street', 'accountNumber'=>'123456789');

      foreach ($customerArray as $key=>$value)
      {
            echo "Key = $key <br>";
            echo "Value = $value <br>";
      }
```

This will result in the following output:

```
Key = customerName
Value = John Smith
Key = customerAddress
Value = 1 The Street
Key = accountNumber
```

Value = 123456789

8.Examining array

- Array creation is pretty simple. First, you need to create a variable and then tell PHP that you want that variable to be an array:
- $theVar = array();
- Now, $theVar is an array. However, it’s an empty array waiting for you to come along and fill it.
- Technically, you can skip the variable creation step. It’s still a good idea to explicitly define an array because it helps you remember the element is an array, and there are a few special cases (such as passing an array into a function) where

9.Joining array

Now that we have created an associative array and assigned names to each element we can use those names to access the corresponding array values.

We can, therefore, extend our previous example to extract the customer name from our $customerArray:

array_merge($arr1, $arr2)

If you want to combine two different arrays into a single array, you can do so using this function.

It doesn't matter whether the arrays to be combined are of same type(indexed, associative etc) or different types, using this function we can combine them into one single array.

Let's take an example where we will merge an indexed array and an associative array.

```
<?php
$hatchbacks = array(
        "Suzuki" => "Baleno",
        "Skoda" => "Fabia",
        "Hyundai" => "i20",
        "Tata" => "Tigor"
    );
// friends who own the above cars
$friends = array("Vinod", "Javed", "Navjot", "Samuel");
// let's merge the two arrays into one
$merged = array_merge($hatchbacks, $friends);
print_r($merged);
?>
```

```
Array (
[Suzuki] => Baleno
[Skoda] => Fabia
[Hyundai] => i20
[Tata] => Tigor
[0] => Vinod
[1] => Javed
[2] => Navjot
[3] => Samuel
)
```

10.Sorting array

- Array can be sorted using the sort function.

- A number of different sorts are possible using the sort function. The function takes two arguments.

- The first argument is the name of the array.
- The second indicates the sort algorithm to use.
- The available algorithms are SORT_NUMERIC, SORT_STRING and SORT_REGULAR. If no sort type is specified, SORT_REGULAR is used.
- Similarly array items can be sorted in descending order using the rsort function.

For example we can sort our array of color names:

```
$colorArray = array("Red", "Yellow", "Green", "Blue", "Indigo");
  sort($colorArray, SORT_STRING);
```

It is also possible to perform a Natural Sort using the natsort function.

11.Sorting associative array

- Associative arrays can be sorted in two ways, either by key or by value.

- To sort by key use ksort and krsort (reverse sort). To sort by value use the asort and arsort functions.
- The syntax and options for these functions are as outlined for the sort and rsort functions.
- Below we have a list of some commonly used array functions in PHP:
- sizeof($arr)
- This function returns the size of the array or the number of data elements stored in the array.

- It is just like count($arr) method, that we used in previous tutorials while traversing the array.

Example

```
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
echo "Size of the array is: ". sizeof($lamborghinis);
?>
```

```
Size of the array is: 3
```

is_array($arr)

To check whether the provided data is in form of an array, we can use the is_array() function. It returns True if the variable is an array and returns False otherwise.

```
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
// using ternary operator
echo is_array($lamborghinis) ? 'Array' : 'not an Array';
$mycar = "Urus";
// using ternary operator
echo is_array($mycar) ? 'Array' : 'not an Array';
?>
```

```
Array
not an Array
```

in_array($var, $arr)

When using an array, we may often want to check whether a certain value is present in the array or not. For example, if get a list of certain cars, like we do in almost all our examples, to check if a certain car is added into the array, we can use the in_array function.

Let's take an example and see,

```
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
// new concept car by lamborghini
$concept = "estoque";
echo in_array($concept, $lamborghinis) ? 'Added to the Lineup' : 'Not yet!'
?>
```

print_r($arr)

Although this is not an array function, but it deserves a special mention here, as we can use this function to print the array in the most descriptive way possible. This function prints the complete representation of the array, along with all the keys and values.

```
<?php
$lamborghinis = array("Urus", "Huracan", "Aventador");
print_r($lamborghinis);
?>
```

```
Array (
[0] => "Urus"
[1] => "Huracan"
[2] => "Aventador"
)
```

array_merge($arr1, $arr2)

If you want to combine two different arrays into a single array, you can do so using this function. It doesn't matter whether the arrays to be combined are of same type(indexed, associative etc) or different types, using this function we can combine them into one single array.

Let's take an example where we will merge an indexed array and an associative array.

```
<?php
$hatchbacks = array(
        "Suzuki" => "Baleno",
        "Skoda" => "Fabia",
        "Hyundai" => "i20",
        "Tata" => "Tigor"
    );
// friends who own the above cars
$friends = array("Vinod", "Javed", "Navjot", "Samuel");
// let's merge the two arrays into one
$merged = array_merge($hatchbacks, $friends);
print_r($merged);
?>
```

```
Array (
[Suzuki] => Baleno
[Skoda] => Fabia
[Hyundai] => i20
[Tata] => Tigor
[0] => Vinod
[1] => Javed
[2] => Navjot
[3] => Samuel
)
```

12.Loops

A Loop is an Iterative Control Structure that involves executing the same number of code a number of times until a certain condition is met.

12.1While loop

- They are used to execute a block of code a repeatedly until the set condition gets satisfied
- When to use while loops
- While loops are used to execute a block of code until a certain condition becomes true.
- You can use a while loop to read records returned from a database query.
- Types of while loops
- Do… while - executes the block of code at least once before evaluating the condition
- While… - checks the condition first. If it evaluates to true, the block of code is executed as long as the condition is true. If it evaluates to false, the execution of the while loop is terminated.

While loop

It has the following syntax

```
   <?php
while (condition){
block of code to be executed;
}
?>
```

HERE,

"while(…){…}" is the while loop block code

"condition" is the condition to be evaluated by the while loop

"block of code…" is the code to be executed if the condition gets satisfied

How it works

The flow chart shown below illustrates how the while… loop works

PHP Loop and Control Structures

Practical example

The code below uses the while… loop to print numbers 1 to 5.

```
<?php
$i = 0;
while ($i < 5){
echo $i + 1 . "<br>";
$i++;
}
?>
```

Output:

```
1
2
3
4
5
```

12.2 Do While

The difference between While… loop and Do… while loop is do… while is executed at-least once before the condition is evaluated.

Let's now look at the basic syntax of a do… while loop

```
<?php
do{
block of code to be executed
}
?>
  while(condition);
```

HERE,

"do{…} while(…)" is the do… while loop block code

"condition" is the condition to be evaluated by the while loop

"block of code…" is the code that is executed at least once by the do… while loop

How it works

The flow chart shown below illustrates how the while… loop works

PHP Loop and Control Structures

We are now going to modify the while… loop example and implement it using the do… while loop and set the counter initial value to 9.

The code below implements the above modified example

```
<?php
$i = 9;
do{
    echo "$i is"." <br>";
}
while($i < 9);
?>
```

The above code outputs:

```
9
```

<u>**12.3 For Loop**</u>

The above code outputs “21 is greater than 7” For loops For... loops execute the block of code a specifiednumber of times. There are basically two types of for loops;
for
for… each.
Let’s now look at them separately. For loop It has the following basic syntax

```
<?php
for (initialize; condition; increment){

//code to be executed


}
?>
```

HERE,
“for…{…}” is the loop block
“initialize” usually an integer; it is used to set the counter’s initial value.
“condition” the condition that is evaluated for each php execution. If it evaluates to true then execution of the for... loop is terminated. If it evaluates to false, the execution of the for... loop continues.
“increment” is used to increment the initial value of counter integer.
The flowchart shown below illustrates how for loop in php works

12.4. For Each loop

The php foreach loop is used to iterate through array values. It has the following basic syntax

```
<?php
```

```
foreach($array_variable  as $array_values){
block of code to be executed
}
?>
```

HERE,

"foreach(…){…}" is the foreach php loop block code

"$array_data" is the array variable to be looped through

"$array_value " is the temporary variable that holds the current array item values.

"block of code…" is the piece of code that operates on the array values

How it works The flowchart shown below illustrates how the for… each… loop works

PHP Loop and Control Structures

Practical examples

The code below uses for… each loop to read and print the elements of an array.

```
<?php
$animals_list = array("Lion","Wolf","Dog","Leopard","Tiger");
foreach($animals_list as $array_values){
echo $array_values . "<br>";
}
?>
```

Output:

```
Lion
Wolf
Dog
Leopard
Tiger
```

13.Function in php

13.1 Introduction to function

- PHP functions are similar to other programming languages.
- A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- You already have seen many functions like fopen() and fread() etc.
- They are built-in functions but PHP gives you option to create your own functions as well.

There are two parts which should be clear to you

- Creating a PHP Function
- Calling a PHP Function

In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

14.Library function

14.1 Array function

```
<html>
<body bgcolor="megenta">
<?php
$a=Array("crypt","php","dm");
$b=Array("mm","wap","php");
echo'<table align="center"  bgcolor="white" border="2"><tr><td>';
echo'<h1 align="center"><u>Array Manipulation</u></h1>';
echo'<b><u>Given Array:</u><br> &nbsp; a=';
```

```
print_r($a);
echo'<br> &nbsp; b=';
print_r($b);
echo'</b><br><h3><font color="darkbrown">Array  Ascending
a()&nbsp;&nbsp;:</font></h3>';
sort($a);
print_r($a);
echo'<br><h3><font color="fuchsia">Array  Decending
a()&nbsp;&nbsp;:</font></h3>';
rsort($a);
print_r($a);
echo'<br><h3><font color="navy">Intersection of a(),b()&nbsp;:</font></h3>';
print_r(Array_intersect($a,$b));
echo'<br><h3><font color="darkred">Difference of a() & b():</font></h3>';
print_r(Array_diff($a,$b));
echo'<h3><font color="darkorange">Merging of a() &
b()&nbsp;&nbsp;&nbsp;:</font></h3>';
print_r(Array_merge($a,$b));
echo'<br><h3><font color="darkgreen">Array Slicing in a()
&nbsp;&nbsp;&nbsp;&nbsp;:</font></h3>';
print_r(Array_slice($a,2));
echo'<br><h3><font color="dodgerblue">Array Splicing in a()
&nbsp;&nbsp;:</font></h3>';
print_r(Array_splice($a,2));
echo'<br><h3><font color="chocolate">Array Poping from a():</font></h3>';
echo'<b>Before  :</b>';
print_r($a);
```

```
Array_pop($a);
echo'<br><b>After &nbsp; :</b>';
print_r($a);
echo'<br><h3><font color="lime">Array Pushing into b():</font></h3>';
echo'<b>Before  :</b>';
print_r($b);
Array_push($b,"50","30");
echo'<br><b>After &nbsp; :</b>';
print_r($b);
echo'<br><h3><font color="gold">Addition of b() Element:</font></h3>Sum=';
$i=Array_sum($b);
print_r($i);
echo"</td></tr></table>";
?>
</body></html>
```

14.2.String function

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = 'This is a somewhat longer, singly quoted string';
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters.
```

14.3.Date and Time function

```
<html>
<body bgcolor="green">
<?php
echo'<table align="center" bgcolor="white"><tr><td>';
echo'<h1><u><font color="crimson" face="cooper black">Date & Time
Function</font></u></h1><br>';
echo'<b>Current Date, i.e. 16-12-31</b><br>';
echo date("y-m-d");
echo'<br><br><b>Current Month, i.e. 08 </b><br>';
echo date("m");
echo'<br><br><b>Current Month Name, i.e. Jan </b><br>';
echo date("M");
echo'<br><br><b>Current Timestamp</b><br>';
echo time();
$d=strtotime("tomorrow");
echo'<br><br><b>Tomorrow Date is</b><br>';
echo date("y-m-d h:i:sa",$d);
$d=strtotime("Next Saturday");
echo'<br><br><b>Next Saturday is</b><br>';
echo date("y-m-d h:i:sa",$d);
$d=strtotime("+3 months");
echo'<br><br><b>After 3 Month</b><br>';
echo date("y-m-d h:i:sa",$d);
$d=mktime(11,14,54,18,12,2014);
echo"<br><br><b>Created Date is</b><br>".date("y-m-d h:i:sa",$d);
```

```
echo'<br><br><b>Increase 1 day From Current Date </b><br>';
echo"Date is".date("y-m-d",time()+86400);
echo'<br><br><b>Decrease 15 days From Current Date </b><br>';
echo"Date is".date("y-m-d",time()-(86400*15));
echo'<br><br><b>Current Time in 12hr Format,i.e. 08:50:55pm</b><br>';
echo date("h:i:sa");
echo'<br><br><b>Current Time in 24hr Format,i.e. 18:50:55pm</b><br>';
echo date("H:i:s");
echo'</td></tr></table>';
?>
</body>
</html>
```

15.User defined function

Creating PHP Function

Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it.

Following example creates a function called write Message() and then calls it just after creating it.

Note that while creating a function its name should start with keyword function and all the PHP code should be put inside { and } braces as shown in the following example below

```
<html>

   <head>
      <title>Writing PHP Function</title>
```

```
  </head>

  <body>

    <?php
      /* Defining a PHP Function */
      function writeMessage() {
        echo "You are really a nice person, Have a nice time!";
      }

      /* Calling a PHP Function */
      writeMessage();
    ?>
  </body>
</html>
```

This will display following result −

```
You are really a nice person, Have a nice time!
```

15.1. Defining a Functions with Parameters and without Parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters like.

These parameters work like variables inside your function.

Following example takes two integer parameters and add them together and then print them.

```
<html>

   <head>
      <title>Writing PHP Function with Parameters</title>
   </head>

   <body>

      <?php
         function addFunction($num1, $num2) {
            $sum = $num1 + $num2;
            echo "Sum of the two numbers is : $sum";
         }

         addFunction(10, 20);
      ?>
   </body>
</html>
```

This will display following result −

```
Sum of the two numbers is : 30
```

15.2.Returning value from the function

A function can return a value using the return statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

You can return more than one value from a function using return array(1,2,3,4).

Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that return keyword is used to return a value from a function.

Example:

```
<html>
  <head>
    <title>Writing PHP Function which returns value</title>
  </head>

  <body>

    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        return $sum;
      }
      $return_value = addFunction(10, 20);

      echo "Returned value from the function : $return_value";
    ?>

  </body>
</html>
```

This will display following result −

```
Returned value from the function : 30
```

15.3 Dynamic Function Calls accessing variables with global statement

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself. Following example depicts this behavior.

```
<html>
  <head>
    <title>Dynamic Function Calls</title>
  </head>
  <body>
    <?php
      function sayHello() {
        echo "Hello<br />";
      }

      $function_holder = "sayHello";
      $function_holder();
    ?>

  </body>
</html>
```

This will display following result −

```
Hello
```

Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that return keyword is used to return a value from a function.

```
<html>

   <head>
      <title>Writing PHP Function which returns value</title>
   </head>

   <body>

      <?php
         function addFunction($num1, $num2) {
            $sum = $num1 + $num2;
            return $sum;
         }
         $return_value = addFunction(10, 20);

         echo "Returned value from the function : $return_value";
      ?>

   </body>
</html>
```

This will display following result −

```
Returned value from the function : 30
```

16.5 Setting Default Values for Function Parameters

We can set a parameter to have a default value if the function's caller doesn't pass it.

Following function prints NULL in case use does not pass any value to this function.

```
<html>

   <head>
      <title>Writing PHP Function which returns value</title>
   </head>

   <body>

      <?php
         function printMe($param = NULL) {
            print $param;
         }

         printMe("This is test");
         printMe();
      ?>

   </body>
</html>
```

This will produce following result −

This is test

Setting Default Values for Function Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it.

Following function prints NULL in case use does not pass any value to this function.

```
<html>

   <head>
      <title>Writing PHP Function which returns value</title>
   </head>

   <body>

      <?php
         function printMe($param = NULL) {
            print $param;
         }

         printMe("This is test");
         printMe();
      ?>

   </body>
</html>
```

This will produce following result −

This is test

15.5.Passing argument to function by value

- PHP has a large number of built-in functions such as mathematical, string, date, array functions etc.
- It is also possible to define a function as per specific requirement. Such function is called user defined function.
- A function is a reusable block of statements that performs a specific task. This block is defined with function keyword and is given a name that starts with an alphabet or underscore.
- This function may be called from anywhere within the program any number of times.

Syntax

```
//define a function
function myfunction($arg1, $arg2, ... $argn)
{
   statement1;
   statement2;
   ..
   ..
   return $val;
}
//call function
$ret=myfunction($arg1, $arg2, ... $argn);
```

Function may be defined with optional but any number of arguments. However, same number of arguments must be provided while calling. Function's body can contain any valid PHP code i.e. conditionals, loops etc. (even other functions or classes may be defined inside a function). After executing statements in the block, program control goes back to the location from which it was invoked irrespective of presence of last statement of function block as return. An expression in front of return statement returns its value to calling environment.

function with arguments

In following example, a function is defined with two formal arguments.

Example

```
<?php
function add($arg1, $arg2){
   echo $arg1+$arg2 . "\n";
}
add(10,20);
add("Hello", "World");
?>
```

Output

This will produce following result. −

```
30
```

15.6 Passing Arguments to a function by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

Any changes made to an argument in these cases will change the value of the original variable.

You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Following example depicts both the cases.

```
<html>

   <head>
      <title>Passing Argument by Reference</title>
   </head>

   <body>

      <?php
         function addFive($num) {
            $num += 5;
         }

         function addSix(&$num) {
            $num += 6;
         }

         $orignum = 10;
         addFive( $orignum );

         echo "Original Value is $orignum<br />";
         addSix( $orignum );
         echo "Original Value is $orignum<br />";
      ?>
   </body>
</html>
```

This will display following result −

```
Original Value is 10
Original Value is 16
```

Unit-III

Working with the file system

1. Creating and Deleting files
2. Reading and writing text Files
3. Working with directories
4. Checking for existence file
5. Determinig filesize
6. Opening a file for writing ,reading,appending
7. writing data to the Files and reading characters
8. working with Forms
   - 8.1 Forms
   - 8.2 Super Global Variables in PHP
   - 8.3 Server Array
   - 8.4 A script to acquire user input
   - 8.5.Combine html with php code
   - 8.6 Using Hidden Field
   - 8.7 Redirect to the user
   - 8.8 File Upload and script
9. Validation
   - 9.1 Server side validation and client side validation
   - 9.2 Working with Regular Expressions

1. Creating and deleting files

File handling is needed for any application. For some tasks to be done file needs to be processed.

File handling in PHP is similar as file handling is done by using any programming language like C. PHP has many functions to work with normal files.

Those functions are:

1) fopen() – PHP fopen() function is used to open a file. First parameter of fopen() contains name of the file which is to be opened and second parameter tells about mode in which file needs to be opened, e.g.,

```
<?php

$file = fopen(“demo.txt”,'w');

?>
```

Files can be opened in any of the following modes :

“w” – Opens a file for write only. If file not exist then new file is created and if file already exists then contents of file is erased.

“r” – File is opened for read only.

“a” – File is opened for write only. File pointer points to end of file. Existing data in file is preserved.

“w+” – Opens file for read and write. If file not exist then new file is created and if file already exists then contents of file is erased.

“r+” – File is opened for read/write.

“a+” – File is opened for write/read. File pointer points to end of file. Existing data in file is preserved. If file is not there then new file is created.

“x” – New file is created for write only.

Deleting Files

To delete a file, you use the unlink() function. You need to pass the file name that you want to delete to the unlink() function. The function returns true on success or false on failure

```
<?php

$fn = './backup/test.bak';

if(unlink($fn)){

 echo sprintf("The file %s deleted successfully",$fn);

}else{

 echo sprintf("An error occurred deleting the file %s",$fn);

}
```

2. Reading and writing text Files

1) fread() — After file is opened using fopen() the contents of data are read using fread(). It takes two arguments. One is file pointer and another is file size in bytes, e.g.,

```
<?php

$filename = "demo.txt";

$file = fopen( $filename, 'r' );

$size = filesize( $filename );

$filedata = fread( $file, $size );

?>
```

2) fwrite() – New file can be created or text can be appended to an existing file using fwrite() function. Arguments for fwrite() function are file pointer and text that is to written to file. It can contain optional third argument where length of text to written is specified, e.g.,

```
<?php

$file = fopen("demo.txt", 'w');
```

```
$text = "Hello world\n";


fwrite($file, $text);

?>
```

4) fclose() – file is closed using fclose() function. Its argument is file which needs to be closed, e.g.,

```
<?php

$file = fopen("demo.txt", 'r');

//some code to be executed

fclose($file);

?>
```

Notice that copy(), rename() and unlink() functions raise warning-level errors if the file cannot be found therefore it is good practice to check the file exists using the file_exists() function before copying, renaming or deleting.

3. Working with Directories

A new directory can be created in PHP using the mkdir() function. This function takes a path to the directory to be created.

To create a directory in the same directory as your PHP script simply provide the directory name. To create a new directory in a different directory specify the full path when calling mkdir().

A second, optional argument allows the specification of permissions on the directory (controlling such issues as whether the directory is writable):

```
<?php

$result = mkdir ("/path/to/directory", "0777");

?>
```

4.Checking for existence file

The file_exists() function in PHP is an inbuilt function which is used to check whether a file or directory exists or not.

The path of the file or directory you want to check is passed as a parameter to the file_exists() function which returns True on success and False on failure.

Syntax:

file_exists($path)

Parameters: The file_exists() function in PHP accepts only one parameter $path. It specifies the path of the file or directory you want to check.

Return Value: It returns True on success and False on failure.

Errors And Exception:

The file_exists() function returns False if the path specified points to non-existent files.

For files larger than 2gb, some of the filesystem functions may give unexpected results since PHP’s integer type is signed and many platforms use 32bit integers.

Examples:

Input : echo file_exists('/user01/work/gfg.txt');

Output : 1

```
Input : $file_pointer = '/user01/work/gfg.txt';
        if (file_exists($file_pointer)) {
            echo "The file $file_pointer exists";
        }else {

            echo "The file $file_pointer does

                                  not exists";

        }
```

Output : 1

Below programs illustrate the file_exists() function.

Program 1:

```
<?php

// checking whether file exists or not

echo file_exists('/user01/work/gfg.txt');

?>
```

Output:

```
1
```

5. Determinig file size

The filesize() function in PHP is an inbuilt function which is used to return the size of a specified file. The filesize() function accepts the filename as a parameter and returns the size of a file in bytes on success and False on failure.

The result of the filesize() function is cached and a function called clearstatcache() is used to clear the cache.

Syntax:

```
filesize($filename)
```

Parameters: The filesize() function in PHP accepts only one parameter $filename. It specifies the filename of the file whose size you want to check.

Return Value: It returns the size of a file in bytes on success and False on failure.

Program 1:

```
<?php
// displaying file size using
// filesize() function
echo filesize("gfg.txt");
?>
```

Output:

```
256
```

Program 2:

```
<?php
// displaying file size using
// filesize() function
$myfile = 'gfg.txt';
echo $myfile . ': ' . filesize($myfile) . ' bytes';
?>
```

Output:

```
gfg.txt : 256 bytes
```

6. Opening a file for writing,reading, appending

File handling in PHP is similar as file handling is done by using any programming language like C. PHP has many functions to work with normal files.

Those functions are:

1) fopen() – PHP fopen() function is used to open a file. First parameter of fopen() contains name of the file which is to be opened and second parameter tells about mode in which file needs to be opened, e.g.,

```
<?php

$file = fopen(“demo.txt”,'w');

?>
```

Files can be opened in any of the following modes :

“w” – Opens a file for write only. If file not exist then new file is created and if file already exists then contents of file is erased.

“r” – File is opened for read only.

“a” – File is opened for write only. File pointer points to end of file. Existing data in file is preserved.

“w+” – Opens file for read and write. If file not exist then new file is created and if file already exists then contents of file is erased.

“r+” – File is opened for read/write.

“a+” – File is opened for write/read. File pointer points to end of file. Existing data in file is preserved. If file is not there then new file is created.

“x” – New file is created for write only.

Deleting Files

To delete a file, you use the unlink() function. You need to pass the file name that you want to delete to the unlink() function. The function returns true on success or false on failure

```
<?php

$fn = './backup/test.bak';

if(unlink($fn)){

 echo sprintf("The file %s deleted successfully",$fn);

}else{
 echo sprintf("An error occurred deleting the file %s",$fn);

}
```

1) fread() — After file is opened using fopen() the contents of data are read using fread(). It takes two arguments. One is file pointer and another is file size in bytes, e.g.,

```
<?php

$filename = "demo.txt";

$file = fopen( $filename, 'r' );

$size = filesize( $filename );

$filedata = fread( $file, $size );

?>
```

2) fwrite() – New file can be created or text can be appended to an existing file using fwrite() function. Arguments for fwrite() function are file pointer and text that is to written to file. It can contain optional third argument where length of text to written is specified, e.g.,

```
<?php

$file = fopen("demo.txt", 'w');

$text = "Hello world\n";


fwrite($file, $text);

?>
```

4) fclose() – file is closed using fclose() function. Its argument is file which needs to be closed, e.g.,

```
<?php

$file = fopen("demo.txt", 'r');

//some code to be executed
```

```
fclose($file);

?>
```

Notice that copy(), rename() and unlink() functions raise warning-level errors if the file cannot be found therefore it is good practice to check the file exists using the file_exists() function before copying, renaming or deleting.

7. Writing data to the Files and reading characters

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
```

8. Forms

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST.

The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

form1.html

```
<form action="welcome.php" method="get">

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

<input type="submit" value="visit"/>

</form>
```

File: welcome.php

```
<?php

$name=$_GET["name"];//receiving name field value in $name variable

echo "Welcome, $name";

?>
```

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

File: form1.html

```
<form action="login.php" method="post">

<table>

<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>

<tr><td>Password:</td><td> <input type="password"
name="password"/></td></tr>

<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>

</table>

</form>
```

File: login.php

```
<?php

$name=$_POST["name"];//receiving name field value in $name variable

$password=$_POST["password"];//receiving password field value in $password
variable

echo "Welcome: $name, your password is: $password";

?>
```

Output:

php form

8.2 Super global variables

- $_REQUEST
- $_POST
- $_GET
- $_FILES

- $_ENV
- $_COOKIE
- $_SESSION

This chapter will explain some of the superglobals, and the rest will be explained in later chapters.

PHP $GLOBALS

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

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

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

Example

```
<?php

$x = 75;

$y = 25;

function addition() {

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

}

addition();

echo $z;

?>
```

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

PHP $_SERVER

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

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

Example

```
<?php

echo $_SERVER['PHP_SELF'];

echo "<br>";

echo $_SERVER['SERVER_NAME'];

echo "<br>";

echo $_SERVER['HTTP_HOST'];

echo "<br>";

echo $_SERVER['HTTP_REFERER'];

echo "<br>";

echo $_SERVER['HTTP_USER_AGENT'];

echo "<br>";

echo $_SERVER['SCRIPT_NAME'];

?>
```

The following table lists the most important elements that can go inside $_SERVER:

Element/Code	Description
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server

$_SERVER['SERVER_NAME'] Returns the name of the host server (such as www.w3schools.com)

$_SERVER['SERVER_SOFTWARE'] Returns the server identification string (such as Apache/2.2.24)

$_SERVER['SERVER_PROTOCOL'] Returns the name and revision of the information protocol (such as HTTP/1.1)

PHP $_REQUEST

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

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

Example

```
<html>

<body>


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

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

  <input type="submit">

</form>


<?php

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>


</body>
</html>
```

PHP $_POST

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

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

Example

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
```

```
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body>
</html>
```

PHP $_GET

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.

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

```
<html>
<body>
<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>
```

```
</body>

</html>
```

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

The example below shows the code in "test_get.php":

Example

```
<html>

<body>

<?php

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

?>

</body>

</html>
```

8.3 The Server Array:

$_SERVER is an array which holds information of headers, paths, script locations.

Web server creates the entries in the array.

This is not assured that every web server will provide similar information; rather some servers may include or exclude some information which are not listed here.

$_SERVER has following basic properties:

1. Set by web server.

2. Directly related to the runtime environment of the current php script.

3. It does the same job as $HTTP_SERVER_VARS used to do in previous versions of PHP

Example

```
<?php

echo $_SERVER['PHP_SELF'];

echo "<br>";

echo $_SERVER['SERVER_NAME'];

echo "<br>";

echo $_SERVER['HTTP_HOST'];

echo "<br>";

echo $_SERVER['HTTP_USER_AGENT'];

echo "<br>";

echo $_SERVER['SCRIPT_NAME'];

echo "<br>";
```

8.4 A script to acquire a user input

```
<html>
<head>
<title>Listing 9.2 A simple HTML form</title>
</head>
<body>
<form action="listing9.3.php">
<input type="text" name="user">
<br>
<textarea name="address" rows="5" cols="40">
</textarea>
<br>
<input type="submit" value="hit it!">
</form>
</body>
</html>
```

- We define a form that contains a text field with the name "user" on line 7, a text area with the name "address" on line 9, and a submit button on line 12.
- It is beyond the remit of this book to cover HTML in detail.
- If you find the HTML in these examples hard going, take a look at *Sams Teach Yourself HTML in 24 Hours* or one of the numerous online HTML tutorials.
- The FORM element's ACTION argument points to a file called listing

- 9.3.php, which processes the form information. Because we haven't added anything more than a filename to the ACTION argument, the file listing9.3.php should be in the same directory on the server as the document that contains our HTML.
- Listing 9.3 creates the code that receives our users' input.

Listing 9.3 Reading Input from the Form in Listing 9.2

```
<html>
<head>
<title>Listing 9.3 Reading input from the form in Listing 9.2</title>
</head>
<body>
<?php
print "Welcome <b>$user</b><P>\n\n";
print "Your address is:<P>\n\n<b>$address</b>";
?>
</body>
</html>
```

8.5. Combine html and PHP code

The first is to embed the PHP code in your HTML file itself with the .html extension—this requires a special consideration which we’ll discuss in a moment.

The other option, the preferred way, is to combine PHP and HTML tags in .php files.

Since PHP is a server-side scripting language, the code is interpreted and run on the server side.

For example, if you add the following code in your index.html file, it won’t run out of the box.

```
<!DOCTYPE html>

<html>

  <head>

    <title>Embed PHP in a .html File</title>

  </head>

  <body>

    <h1><?php echo "Hello World" ?></h1>

  </body>

</html>
```

First of all, don’t worry if you haven’t seen this kind of mixed PHP and HTML code before, as we’ll discuss it in detail throughout this article. The above example outputs following in your browser:

```
<?php echo "Hello World" ?>
```

So as you can see, by default, PHP tags in your .html document are not detected, and they're just considered plain-text, outputting without parsing.

That's because the server is usually configured to run PHP only for files with the .php extension.

If you want to run your HTML files as PHP, you can tell the server to run your .html files as PHP files, but it's a much better idea to put your mixed PHP and HTML code into a file with the .php extension.

How to Add PHP Tags in Your HTML Page

When it comes to integrating PHP code with HTML content, you need to enclose the PHP code with the PHP start tag <?php and the PHP end tag ?>. The code wrapped between these two tags is considered to be PHP code, and thus it'll be executed on the server side before the requested file is sent to the client browser.

```
<!DOCTYPE html>

    <title>How to put PHP in HTML - Simple Example</title>

  </head>

  <body>

    <h1><?php echo "This message is from server side." ?></h1>

  </body>

</html
```

8.6. Using Hidden Field

hidden field is not displayed on the page.

- It simply stores the text value specified in the value attribute.
- Hidden fields are great for passing additional information from the form to the server.

<label for="hiddenField">A hidden field</label>

<input type="hidden" name="hiddenField" id="hiddenField" value="" />

Example

The following form uses hidden field to store number of tries.

```
<?php//from w  w w .  j av a2s  .c  om

  $num_to_guess = 42;
  $message = "";
  if (! isset ( $_POST ['guess'] )) {
    $message = "Welcome!";
  } else if ($_POST ['guess'] > $num_to_guess) {
    $message = $_POST ['guess'] . " is too big!";
  } else if ($_POST ['guess'] < $num_to_guess) {
    $message = $_POST ['guess'] . " is too small!";
  } else {
    $message = "Well done!";
  }
  $guess = ( int ) $_POST ['guess'];
  $num_tries = ( int ) $_POST ['num_tries'];
  $num_tries ++;
  ?>
<html>
<body>
 <?php print $message?>
 Guess number: <?php print $num_tries?><br />
<form method="post" action="<?php
print $_SERVER ['PHP_SELF']?>">
<input type="hidden" name="num_tries"
  value="<?php
  print $num_tries?>" /> Type your guess here: <input type="text" name="guess"
value="<?php
  print $guess?>" />
```

```
</form>

</body>

</html>
```

8.Redirecting the user

Redirection from one page to another in PHP is commonly achieved using the following two ways:

Using Header Function in PHP:

The header() function is an inbuilt function in PHP which is used to send the raw HTTP (Hyper Text Transfer Protocol) header to the client.

Syntax:

```
header( $header, $replace, $http_response_code )
```

Parameters: This function accepts three parameters as mentioned above and described below:

$header: This parameter is used to hold the header string.

$replace: This parameter is used to hold the replace parameter which indicates the header should replace a previous similar header, or add a second header of the same type. It is optional parameter.

$http_response_code: This parameter hold the HTTP response code.

Below program illustrates the header() function in PHP:

Program

```
<?php

// Redirect browser

header("Location: http://www.geeksforgeeks.org");

exit;
```

```
?>
```

Note: The die() or exit() function after header is mandatory. If die() or exit() is not put after after the header('Location: ….') then script may continue resulting in unexpected behavior. For example, result in content being disclosed that actually wanted to prevent with the redirect (HTTP 301).

Using JavaScript via PHP:

The windows.location object in JavaScript is used to get the current page address(URL) and to redirect the browser to a new page. The window.location object contains the crucial information about a page such as hostname, href, pathname, port etc.

Example:

```
<html>
  <head>
    <title>window.location function</title>
  </head>
  <body>
  <p id="demo"></p>
  <script>
    document.getElementById("demo").innerHTML =
      "URL: " + window.location.href +"</br>";
    document.getElementById("demo").innerHTML =
    document.getElementById("demo").innerHTML +
    "Hostname: " + window.location.hostname + "</br>";
    document.getElementById("demo").innerHTML =
    document.getElementById("demo").innerHTML +
        "Protocal: " + window.location.protocol + "</br>";
    </script>
    </body>
</html>
```

Output:

```
URL: https://ide.geeksforgeeks.org/tryit.php
Hostname: ide.geeksforgeeks.org
Protocal: https:
```

8.8. File upload

- ❖ A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script.
- ❖ Information in the phpinfo.php page describes the temporary directory that is used for file uploads as upload_tmp_dir and the maximum permitted size of files that can be uploaded is stated as upload_max_filesize. These parameters are set into PHP configuration file php.ini
- ❖ The process of uploading a file follows these steps −

- ❖ The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
- ❖ The user clicks the browse button and selects a file to upload from the local PC.
- ❖ The full path to the selected file appears in the text filed then the user clicks the submit button.
- ❖ The selected file is sent to the temporary directory on the server.

- ❖ The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
- ❖ The PHP script confirms the success to the user.
- ❖ As usual when writing files it is necessary for both temporary and final locations to have permissions set that enable file writing. If either is set to be read-only then process will fail.

An uploaded file could be a text file or image file or any document.

Creating an upload form

The following HTM code below creates an uploader form. This form is having method attribute set to post and enctype attribute is set to multipart/form-data

```
<?php
  if(isset($_FILES['image'])){
    $errors= array();
    $file_name = $_FILES['image']['name'];
    $file_size =$_FILES['image']['size'];
    $file_tmp =$_FILES['image']['tmp_name'];
    $file_type=$_FILES['image']['type'];
    $file_ext=strtolower(end(explode('.',$_FILES['image']['name'])));
    $extensions= array("jpeg","jpg","png");
    if(in_array($file_ext,$extensions)=== false){
      $errors[]="extension not allowed, please choose a JPEG or PNG file.";
    }
    if($file_size > 2097152){
      $errors[]='File size must be excately 2 MB';
    }
```

```
      if(empty($errors)==true){
         move_uploaded_file($file_tmp,"images/".$file_name);
         echo "Success";
      }else{
         print_r($errors);
      }
   }
?>
<html>
   <body>
      <form action="" method="POST" enctype="multipart/form-data">
         <input type="file" name="image" />
         <input type="submit"/>
      </form>
   </body>
</html>
```

It will produce the following result −

Upload Form

Creating an upload script

There is one global PHP variable called $_FILES. This variable is an associate double dimension array and keeps all the information related to uploaded file. So if the value assigned to the input's name attribute in uploading form was file, then PHP would create following five variables −

$_FILES['file']['tmp_name'] − the uploaded file in the temporary directory on the web server.

$_FILES['file']['name'] − the actual name of the uploaded file.

$_FILES['file']['size'] − the size in bytes of the uploaded file.

$_FILES['file']['type'] − the MIME type of the uploaded file.

$_FILES['file']['error'] − the error code associated with this file upload.

Example

Below example should allow upload images and gives back result as uploaded file information.

Live Demo

```
<?php
   if(isset($_FILES['image'])){
      $errors= array();
      $file_name = $_FILES['image']['name'];
      $file_size = $_FILES['image']['size'];
      $file_tmp = $_FILES['image']['tmp_name'];
      $file_type = $_FILES['image']['type'];
      $file_ext=strtolower(end(explode('.',$_FILES['image']['name'])));
      $extensions= array("jpeg","jpg","png");

      if(in_array($file_ext,$extensions)=== false){
         $errors[]="extension not allowed, please choose a JPEG or PNG file.";
      }

      if($file_size > 2097152) {
         $errors[]='File size must be excately 2 MB';
      }
      if(empty($errors)==true) {
         move_uploaded_file($file_tmp,"images/".$file_name);
         echo "Success";
      }else{
         print_r($errors);
      }
   }
?>
<html>
   <body>
      <form action = "" method = "POST" enctype = "multipart/form-data">
         <input type = "file" name = "image" />
         <input type = "submit"/>
         <ul>
            <li>Sent file: <?php echo $_FILES['image']['name'];  ?>
            <li>File size: <?php echo $_FILES['image']['size'];  ?>
            <li>File type: <?php echo $_FILES['image']['type'] ?>
         </ul>
      </form
   </body>
</html>
```

It will produce the following result −

Upload Script

9. Validation

9.1 client side and server side validation

Validation means check the input submitted by the user. There are two types of validation are available in PHP. They are as follows

Client-Side Validation − Validation is performed on the client machine web browsers.

Server Side Validation − After submitted by data, The data has sent to a server and perform validation checks in server machine.

Some of Validation rules for field

Field	**Validation Rules**
Name	**Should required letters and white-spaces**
Email	**Should required @ and .**
Website	**Should required a valid URL**
Radio	**Must be selectable at least once**
Check Box	**Must be checkable at least once**
Drop Down menu	**Must be selectable at least once**

Valid URL

Below code shows validation of URL

```
$website = input($_POST["site"]);


if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {

   $websiteErr = "Invalid URL";

}
```

Above syntax will verify whether a given URL is valid or not. It should allow some keywords as https, ftp, www, a-z, 0-9,..etc..

Valid Email

Below code shows validation of Email address

```
$email = input($_POST["email"]);


if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
   $emailErr = "Invalid format and please re-enter valid email";
}
```

Above syntax will verify whether given Email address is well-formed or not.if it is not, it will show an error message.

Example

Example below shows the form with required field validation

```
<html>

   <head>
      <style>
         .error {color: #FF0000;}
      </style>
   </head>

   <body>
      <?php
```

```
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  }else {
    $name = test_input($_POST["name"]);
  }
  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  }else {
    $email = test_input($_POST["email"]);

    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }
  if (empty($_POST["website"])) {
    $website = "";
  }else {
    $website = test_input($_POST["website"]);
```

```
      }


      if (empty($_POST["comment"])) {
         $comment = "";
      }else {
         $comment = test_input($_POST["comment"]);
      }
      if (empty($_POST["gender"])) {
         $genderErr = "Gender is required";
      }else {
         $gender = test_input($_POST["gender"]);
      }
   }
   function test_input($data) {
      $data = trim($data);
      $data = stripslashes($data);
      $data = htmlspecialchars($data);
      return $data;
   }
?>
<h2>Absolute classes registration</h2>
<p><span class = "error">* required field.</span></p>

<form method = "post" action = "<?php
```

```
echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">

<table>

  <tr>

    <td>Name:</td>

    <td><input type = "text" name = "name">

      <span class = "error">* <?php echo $nameErr;?></span>

    </td>

  </tr>

  <tr>

    <td>E-mail: </td>

    <td><input type = "text" name = "email">

      <span class = "error">* <?php echo $emailErr;?></span>

    </td>

  </tr>

  <tr>

    <td>Time:</td>

    <td> <input type = "text" name = "website">

      <span class = "error"><?php echo $websiteErr;?></span>

    </td>

  </tr>

  <tr>

    <td>Classes:</td>

    <td> <textarea name = "comment" rows = "5" cols =
"40"></textarea></td>

  </tr>
```

```
    <tr>
      <td>Gender:</td>
      <td>
        <input type = "radio" name = "gender" value = "female">Female
        <input type = "radio" name = "gender" value = "male">Male
        <span class = "error">* <?php echo $genderErr;?></span>
      </td>
    </tr>
    <td>
      <input type = "submit" name = "submit" value = "Submit">
    </td>
  </table>
</form>
<?php
  echo "<h2>Your given values are as:</h2>";
  echo $name;
  echo "<br>";

  echo $email;
  echo "<br>";
  echo $website;
  echo "<br>"
  echo $comment;
  echo "<br>";
```

```
      echo $gender;
    ?>
  </body>
</html>
```

It will produce the following result

9.2. Working with Regular Expressions

- Regular expressions commonly known as a regex (regexes) are a sequence of characters describing a special search pattern in the form of text string.

❖ They are basically used in programming world algorithms for matching some loosely defined patterns to achieve some relevant tasks

❖ . Some times regexes are understood as a mini programming language with a pattern notation which allows the users to parse text strings.

❖ The exact sequence of characters are unpredictable beforehand, so the regex helps in fetching the required strings based on a pattern definition.

❖ Regular Expression is a compact way of describing a string pattern that matches a particular amount of text.

❖ As you know, PHP is an open-source language commonly used for website creation, it provides regular expression functions as an important tool. Like PHP, many other programming languages have their own implementation of regular expressions.

❖ This is the same with other applications also, which have their own support of regexes having various syntaxes. Many available modern languages and tools apply regexes on very large files and strings.

❖ Let us look into some of the advantages and uses of regular expressions in our applications.

❖ Advantages and uses of Regular expressions:

- In many scenarios, developers face problems whenever data are collected in free text fields as most of the programming deals with data entries. Regular expressions are used almost everywhere in today's application programming.
- Regular expressions help in validation of text strings which are of programmer's interest.
- It offers a powerful tool for analyzing, searching a pattern and modifying the text data.
- It helps in searching specific string pattern and extracting matching results in a flexible manner.
- It helps in parsing text files looking for a defined sequence of characters for further analysis or data manipulation.
- With the help of in-built regexes functions, easy and simple solutions are provided for identifying patterns.
- It effectively saves a lot of development time, which are in search of specific string pattern.
- It helps in important user information validations like email address, phone numbers and IP address.
- It helps in highlighting special keywords in a file based on search result or input.
- It helps in identifying specific template tags and replacing those data with the actual data as per the requirement.
- Regexes are very useful for creation of HTML template system recognizing tags.
- Regexes are mostly used for browser detection, spam filteration, checking password strength and form validations.
- We cannot cover everything under this topic, but let us look into some of the major regular expression concepts. The following table shows some regular expressions and the corresponding string which matches the regular expression patterns
- Regular Expression   Matches

geeks	The string “geeks”
^geeks	The string which starts with “geeks”
geeks$	The string which have “geeks” at the end.
^geeks$	The string where “geeks” is alone on a string.
[abc]	a, b, or c
[a-z]	Any lowercase letter
[^A-Z]	Any letter which is NOT a uppercase letter
(gif|png)	Either “gif” or “png”
[a-z]+	One or more lowercase letters
^[a-zA-Z0-9]{1, }$	Any word with at least one number or one letter
([ax])([by])	ab, ay, xb, xy
[^A-Za-z0-9]	Any symbol other than a letter or other than number
([A-Z]{3}|[0-9]{5})	Matches three letters or five numbers

Note: Complex search patterns can be created by applying some basic regular expression rules. Even many arithmetic operators like +, ^, – are used by regular expressions for creating little complex patterns.

Operators in Regular Expression: Let us look into some of the operators in PHP regular expressions.

Operator	Description
^	It denotes the start of string.
$	It denotes the end of string.
.	It denotes almost any single character.
()	It denotes a group of expressions.
[]	It finds a range of characters for example [xyz] means x, y or z .

[^]	It finds the items which are not in range for example [^abc] means NOT a, b or c.
– (dash)	It finds for character range within the given item range for example [a-z] means a through z.
| (pipe)	It is the logical OR for example x | y means x OR y.
?	It denotes zero or one of preceding character or item range.
*	It denotes zero or more of preceding character or item range.
+	It denotes one or more of preceding character or item range.
{n}	It denotes exactly n times of preceding character or item range for example n{2}.
{n, }	It denotes atleast n times of preceding character or item range for example n{2, }.
{n, m}	It denotes atleast n but not more than m times for example n{2, 4} means 2 to 4 of n.
\	It denotes the escape character.

Special Character Classes in Regular Expressions: Let us look into some of the special characters used in regular expressions

The below listed in-built functions are case-sensitive.

Function	Definition
preg_match()	This function searches for a specific pattern against some string. It returns true if pattern exists and false otherwise.
preg_match_all()	This function searches for all the occurrences of string pattern against the string. This function is very useful for search and replace.
ereg_replace()	This function searches for specific string pattern and replace the original string with the replacement string, if found.
eregi_replace()	The function behaves like ereg_replace() provided the search for pattern is not case sensitive.

preg_replace()	This function behaves like ereg_replace() function provided the regular expressions can be used in the pattern and replacement strings.

preg_split()	The function behaves like the PHP split() function. It splits the string by regular expressions as its paramaters.

preg_grep()	This function searches all elements which matches the regular expression pattern and returns the output array.

preg_quote()	This function takes string and quotes in front of every character which matches the regular expression.

ereg()	This function searches for a string which is specified by a pattern and returns true if found, otherwise returns false.

eregi()	This function behaves like ereg() function provided the search is not case sensitive.

Example 1:

```
<?php

// Declare a regular expression

$regex = '/^[a-zA-Z ]*$/';

// Declare a string

$nameString = 'Sharukh khan';

// Use preg_match() function to

// search string pattern

if(preg_match($regex, $nameString)) {

    echo("Name string matching with"

        . " regular expression");

}

else {

    echo("Only letters and white space   . " allowed in name string");
}

?>
```

Output:

```
Name string matching with regular expression
```

UNIT - IV

CLASSES AND OBJECTS

1. Introduction of object oriented programming define a class
2. Creating an object:
3. Object properties and methods
4. Object constructor and Destructor
5. Class Constants and access modifier
6. Class inheritance
7. Abstract Classes and methods
8. Object serialization
9. Checking for class and method existence
10. What is Exception?
11. Introduction to data base and sql
12. Connecting to My sql
13. Database creation, selection

14. Database table creation updates, delete

15. Insert table, update,delete, data to the table

16. Fetch data from table acquiring the values, Joins query

<u>17. COOKIES</u>

17.1 Anatomy of a Cookies

17.2 .Setting Cookies with PHP

17.3 Deleting Cookie with PHP

17.4 Creating cookies

. 17.5Creating Session Cookies

17.6. Query string

1.INTRODUCTION OF OBJECT ORIENTED PROGRAMMING

DEFINE A CLASS

Object-oriented programming is a programming model organized around **Object rather than the actions** and **data rather than logic**.

Class:

- A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object**.**
- In PHP, declare a class using the class keyword, followed by the name of the class and a set of curly braces ({}).

Syntax :

```
<? Php

Class MyClass
  {
    // Class properties and methods go here
  }
?>
```

2. Creating an Object:

- A class defines an individual instance of the data structure. We define a class once and then make many objects that belong to it.
- Objects are also known as an instance.
- An object is something that can perform a set of related activities.

Syntax:

```
<?Php
Class MyClass
{
	// Class properties and methods go here
}
$obj = new MyClass;
var_dump($obj);
?>
```

Example of class and object:

```
<?php
class demo
{
	private $a= "hello javatpoint";
	public function display()
	{
	echo $this->a;
	}
}
$obj = new demo();
$obj->display();
?>
```

3. Object properties and methods

- We call **properties** to the variables inside a class. Properties can accept values like strings, integers, and booleans (true/false values), like any other variable. Let's add some properties to the Car class.

Example:

```
Class Car {

  public $comp;
  public $color = 'beige';
  public $hasSunRoof = true;

}
```

- We put the **public** keyword in front of a class property.
- The naming convention is to start the property name with a lower case letter.

- If the name contains more than one word, all of the words, except for the first word, start with an upper case letter. For example, **$color** or **$hasSunRoof**.
- A property can have a default value. For example, **$color = 'beige'**.
- We can also create a property without a default value. See the property **$comp** in the above example.

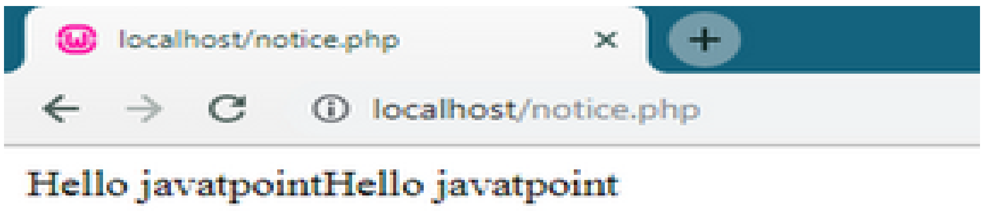
4. Object constructor and Destructor

- **PHP** 5 allows developers to declare **constructor methods for classes**.
- Constructor is suitable for any **initialization that the object may need before it is used**.
- We can design constructor using **"__construct" or same name as class name**.
- Parent constructors are not called implicitly if the child class defines a constructor. In order to run a parent constructor, a call to **parent::__construct()**.

Example

```
  <?Php
class Example
{
  public function __construct()
  {
    echo "Hello javatpoint";
  }
 }
$obj = new Example();
$obj = new Example();
?>
```

Output:



```
Hello javatpointHello javatpoint
```

DESTRUCTOR

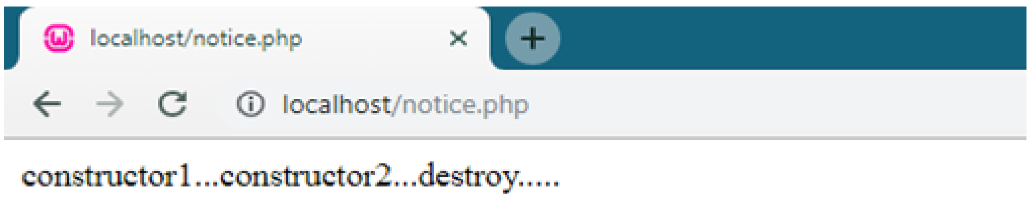
- **PHP 5** introduces a destructor concept similar to that of other object-oriented languages, such as C++.
- The destructor method will be called as soon as all references to a particular object are removed or when the object is **explicitly destroyed in any order in shutdown sequence.**
- We create destructor by using **"__destruct" function.**

Example

```
<?php
  class demo
  {
    public function demo()
    {
      echo "constructor1...";
    }
  }

  class demo1 extends demo
  {
    public function __construct()
    {
      echo parent::demo();
      echo "constructor2...";
    }
    public function __destruct()
    {
      echo "destroy.....";
    }
  }
  $obj= new demo1();
?>
```

Output:



```
constructor1...constructor2...destroy.....
```

5. Class Constants and Access modifiers

- A constant is somewhat like a variable, in that it holds a value, but is really more like a function because a constant is immutable.
- Once you declare a constant, it does not change.
- Declaring one constant is easy, as is done in this version of MyClass

```
class MyClass {
  const requiredMargin = 1.7;

  function __construct($incomingValue) {
    // Statements here run every time
    // an instance of the class
    // is created.
  }
}
```

Access modifier

To set the access rights for class methods and variables we use access modifiers which are nothing but PHP keywords.

We can even assign some of these access modifiers to the class itself to make the class behave in a special way.

Following are the PHP **keywords** which are used as access modifiers along with their meaning:

1. public: When we define class members as public, then they are accessible from anywhere, even from outside of the class scope.
2. private: When we define class members as private, they can only be accessed from within the class itself.

3. protected: This is same as private, with one exception, the class members defined as protected can still be accessed from its subclass(We will learn about subclasses when we will learn about Inheritance).
4. abstract: This keyword is only used for PHP classes and its member functions.
5. final: The class methods defined as final, can not be changed or overriden by any subclass.
6. **Class inheritance**

Inheritance in OOP = When a class derives from another class.

The child class will inherit all the public and protected properties and methods from the parent class.

In addition, it can have its own properties and methods.

An inherited class is defined by using the extends keyword.

Example:

```
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  public function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}
```

```
// Strawberry is inherited from Fruit
class Strawberry extends Fruit {
 public function message() {
  echo "Am I a fruit or a berry? ";
 }
}
$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
```

7.Abstract Classes and methods

- ❖ An abstract class is one that cannot be instantiated, only inherited. You declare an abstract class with the keyword **abstract**, like this
- ❖ When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child; additionally, these methods must be defined with the same visibility.

Syntax:

```
abstract class MyAbstractClass {
  abstract function myAbstractFunction() {
  }
}
```

8.Object serialization

- ➢ Serialization is used to convert an object to a byte-stream representation and vice-versa.
- ➢ This is useful when we need to pass object data in the form of string of text between scripts and applications.
- ➢ There are various situations in which we need to pass objects in the form of string such as:

- Converting object to string and storing it in a text file or a database table.
- Converting and passing objects in URL query string.

- Carrying objects between webpages in the session, etc.
- To convert a object to a string and then back to object when required we can use

Following functions:

- serialize(): the serialize() function takes the object and converts it to a string
- representation containing class name and its properties.

- unserialize(): the unserialize() function takes the serialized string or string that is obtained from object and converts it again to the object.

- When the object is serialized, the content is placed with some type of a specifier followed by a colon, then followed with the actual data followed by a semi-colon.

- When we serialize an object it stores the class name and all its properties. It doesn't store methods of the class. Hence to unserialize the serialized object the class should be present in the script before unserializing it.

- So we have to unserialize the object in the script where the class is defined or we can include the class in the script where we want to unserialize the object with the include statement as shown below:

```
<?php
	include("worker.inc");
	//some code here
	unserialize($work);
?>
```

Now let us see how the object is serialized and unserialized using the function serialize() and unserialize() in the following code shown below:

```
<?php
class Worker
{
	private $id;
	protected $name;
	protected $dept;

	public function __construct($id,$name,$dept)
	{
		$this->id=$id;
		$this->name=$name;
		$this->dept=$dept;
	}

	public function display()
	{
		echo "ID : ".$this->id."<br>Name : ".$this->name."<br>
Department : ".$this->dept;
	}
}

//creating an instance of Worker class
$work=new Worker(1,'James Das','Development');

//serializing the object $work using serialize() function
$serialized=serialize($work);

//displaying the serialized string
echo "*Serialized object in string format:<br>".$serialized."<br><br>";
```

```
//unserializing the string $serialized to the object back
$work1=unserialize($serialized);

//calling the function display() of the Worker class using object $work1
echo "*Object Unserialised:<br>";
$work1->display();
?>
```

- In the above code, a class Worker is defined and its object $work is created and values are passed in the constructor.

- Then the object is serialized using the serialize() function, the object $work is passed as a parameter in the funciton .

- You can think that this data is to be stored somewhere, may in a database table or in a text file.
- The serialized data is stored in a variable $serialized.

- The statement used for it is as shown below:

- $serialized=serialize($work);

- The string is displayed using the echo statement.

- After this the string variable $serialized is passed in the unserialize() function to unserialize the object using the statement shown below:

- $work1=unserialize($serialized);

- Then to show that the object has been unserialized, its display() method is called using the $work1 object.

Output of the code is shown below:

serialize_unserialize_output

9. Checking for class and method existence

The method exists () function checks if the class method exists.

Syntax

method_exists (object, name_of_method)

Parameters

object − The object instance or class name
name_of_method − The method name
Return
The method_exists() function returns TRUE if the method given by method_name has been defined for the given object, FALSE otherwise.
Example
The following is an example

```
<?php
$directory = new Directory('.');
var_dump(method_exists($directory,'anything'));
?>
```

Output

```
bool(false)
```

Example

```
<?php
var_dump(method_exists('Directory','read'));
?>
```

Output

```
bool(true)
```

10. Exception

- An exception is an unexcepted outcome of a program, which can be handled by the program itself.
- Basically, an exception disrupts the normal flow of the program.
- But it is different from an error because an exception can be handled, whereas an error cannot be handled by the program itself.

- In other words, - "An unexpected result of a program is an exception, which can be handled by the program itself."
- Exceptions can be thrown and caught in php.
- Try block
- The try block contains the code that may have an exception or where an exception can arise.
- When an exception occurs inside the try block during runtime of code, it is caught and resolved in catch block.
- The try block must be followed by catch or finally block. A try block can be followed by minimum one and maximum any number of catch blocks.

catch -

The catch block contains the code that executes when a specified exception is thrown.

It is always used with a try block, not alone. When an exception occurs, PHP finds the matching catch block.

throw -

It is a keyword used to throw an exception. It also helps to list all the exceptions that a function throws but does not handle itself.

finally -

The finally block contains a code, which is used for clean-up activity in PHP. Basically, it executes the essential code of the program.

Example 1

Let's take an example to explain the common flow of throw and try-catch block:

```
<?php
//user-defined function with an exception
function checkNumber($num) {
  if($num>=1) {
    //throw an exception
    throw new Exception("Value must be less than 1");
  }
  return true;
}
//trigger an exception in a "try" block
try {
  checkNumber(5);
  //If the exception throws, below text will not be display
  echo 'If you see this text, the passed value is less than 1';
}
//catch exception
catch (Exception $e) {
  echo 'Exception Message: ' .$e->getMessage();
}
?>
```

Output:

```
Exception Message: Value must be less than 1
```

11. INTRODUCTION TO DATABASE

A database is a systematic collection of data.

They support electronic storage and manipulation of data.

Databases make data management easy.

INTROCTION TO SQL :

PHP Connect to MySQL

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

MySQL Examples in Both MySQLi and PDO Syntax

In this and in the following chapters we demonstrate three ways of working with PHP and MySQL:

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

12. Connecting to my sql

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

MySQLi Object-Oriented:

```
$conn->close();
```

MySQLi Procedural:

```
mysqli_close($conn);
```

PDO:

```
$conn = null;
```

13. Database creation, selection

- PHP Create a MySQL Database
- A database consists of one or more tables.
- You will need special CREATE privileges to create or to delete a MySQL database
- Create a MySQL Database Using MySQLi and PDO
- The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
  $conn = new PDO("mysql:host=$servername", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "CREATE DATABASE myDBPDO";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Database created successfully<br>";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

14. Database table creation updates, delete

PHP MySQL Create Table

A database table has its own unique name and consists of columns and rows.

Create a MySQL Table Using MySQLi and PDO

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)
```

Notes on the table above:

The data type specifies what type of data the column can hold.

For a complete reference of all the available data types, go to our Data Types reference.

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

Example (MySQLi Object-oriented)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . mysqli_error($conn);
}
```

```
mysqli_close($conn);
?>
```

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // sql to create table
  $sql = "CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
  lastname VARCHAR(30) NOT NULL,
  email VARCHAR(50),
  reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
  )";

  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Table MyGuests created successfully";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}
```

```
$conn = null;
?>
```

15. Insert table, update, delete, data to the table

PHP MySQL Insert Data

Insert Data Into MySQL Using MySQLi and PDO

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

The following examples add a new record to the "MyGuests" table:

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('John', 'Doe', 'john@example.com')";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "New record created successfully";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

16. Fetch data from table acquiring the values, Joins query

Using Joins at Command Prompt

Suppose we have two tables tutorials_bks and tutorials_inf, in TUTORIALS. A complete listing is given below

Example

Try out the following examples –

```
root@host# mysql -u root -p password;

Enter password:*******


mysql> use TUTORIALSTUTORIAL

Database changed


mysql> SELECT * FROM tcount_bks;

+----+---------+

| id | book    |

+----+---------+

|  1 | java    |

|  2 | java    |

|  3 | html    |

|  4 | c++     |

|  5 | Android |

+----+---------+
5 rows in set (0.00 sec)


mysql> SELECT * from tutorials_inf;
+----+-------+
| id | name  |
+----+-------+
|  1 | sai   |
|  2 | johar |
|  3 | raghu |
|  4 | ram   |
+----+-------+
4 rows in set (0.00 sec)


mysql>
```

Now we can write a SQL query to join these two tables. This query will select all the names from table tutorials_inf and will pickup corresponding number of tutorials fromtutorials_bks

```
mysql> SELECT a.id, a.name,b.id FROM tutorials_inf a,tutorials_bks b WHERE
a.id = b.id;
+----+-------+----+
| id | name  | id |
+----+-------+----+
|  1 | sai   |  1 |
|  2 | johar |  2 |
|  3 | raghu |  3 |
|  4 | ram   |  4 |
+----+-------+----+
4 rows in set (0.00 sec)
mysql>
```

In tutorials_bks table, we have 5 records but in the above example it filters and gives only 4 id records as per query

.Using Joins in PHP Script

You can use any of the above-mentioned SQL query in PHP script. You only need to pass SQL query into PHP function mysqli_query() and then you will fetch results in usual way.

Example

Try out the following example –

```
<?php
   $dbhost = 'localhost:3306';
   $dbuser = 'root';
   $dbpass = '';
   $dbname = 'TUTORIALS';
   $conn = mysqli_connect($dbhost, $dbuser, $dbpass,$dbname);

   if(! $conn ) {
      die('Could not connect: ' . mysqli_error());
```

```
  }
  echo 'Connected successfully</br>';
  $sql = 'SELECT a.id, a.name,b.id FROM tutorials_inf a,tutorials_bks b WHERE
a.id = b.id';
  if($result = mysqli_query($conn, $sql)) {
    if(mysqli_num_rows($result) > 0) {
      echo "<table>";
      echo "<tr>";
      echo "<th>id</th>";
      echo "<th>name</th>";
      echo "<th>id</th>";
      echo "</tr>";
      while($row = mysqli_fetch_array($result)){
        echo "<tr>";
        echo "<td>" . $row['id'] . "</td>";
        echo "<td>" . $row['name'] . "</td>";
                                          echo "<td>" . $row['id'] . "</td>";
        echo "</tr>";
      }
      echo "</table>";
      mysqli_free_result($result);
    } else {
      echo "No records matching your query were found.";
      }
   } else {
      echo "ERROR: Could not able to execute $sql. " . mysqli_error($conn);
   }
   mysqli_close($conn);
?>
```

The sample output should be like this –

```
Connected successfully
id      name id
1       sai     1
2       johar 2
3       raghu 3
4       ram    4
```

MySQL LEFT JOIN

A MySQLi left join is different from a simple join. A MySQLi LEFT JOIN gives extra consideration to the table that is on the left.

If I do a LEFT JOIN, I get all the records that match in the same way and IN ADDITION I get an extra record for each unmatched record in the left table of the join - thus ensuring (in my example) that every name gets a mention –

<u>Example</u>

Try out the following example to understand LEFT JOIN –

```
root@host# mysql -u root -p password;
Enter password:*******
```

```
mysql> use TUTORIALS;

Database change
```

17.COOKIES :

17.1The Anatomy of a Cookie

Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser).

A PHP script that sets a cookie might send headers that look something like this −

```
HTTP/1.1 200 OK

Date: Fri, 04 Feb 2000 21:03:38 GMT

Server: Apache/1.3.9 (UNIX) PHP/4.0b3

Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;

              path=/; domain=tutorialspoint.com

Connection: close

Content-Type: text/html
```

As you can see, the Set-Cookie header contains a name value pair, a GMT date, a path and a domain.

The name and value will be URL encoded. The expires field is an instruction to the browser to "forget" the cookie after the given time and date.

If the browser is configured to store cookies, it will then keep this information until the expiry date.

If the user points the browser at any page that matches the path and domain of the cookie, it will resend the cookie to the server.

The browser's headers might look something like this –

```
GET / HTTP/1.0

Connection: Keep-Alive

User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)

Host: zink.demon.co.uk:1126

Accept: image/gif, */*

Accept-Encoding: gzip

Accept-Language: en

Accept-Charset: iso-8859-1,*,utf-8

Cookie: name=xyz
```

A PHP script will then have access to the cookie in the environmental variables $_COOKIE or $HTTP_COOKIE_VARS[] which holds all cookie names and values.

Above cookie can be accessed using $HTTP_COOKIE_VARS["name"].

17.2.Setting Cookies with PHP

- PHP provided **setcookie()** function to set a cookie.
- This function requires upto six arguments and should be called before <html> tag.
- For each cookie this function has to be called separately.

setcookie(name, value, expire, path, domain, security);

<u>Here is the detail of all the arguments</u> –

Name – This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.

Value – This sets the value of the named variable and is the content that you actually want to store.

Expiry – This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.

Path – This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.

Domain – This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.

Security – This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Following example will create two cookies **name** and **age** these cookies will be expired after one hour.

```
<?php
  setcookie("name", "John Watkin", time()+3600, "/","", 0);
  setcookie("age", "36", time()+3600, "/", "",  0);
?>
<html>

  <head>
    <title>Setting Cookies with PHP</title>
  </head>

  <body>
    <?php echo "Set Cookies"?>
  </body>

</html>
```

17.3.Deleting Cookie with PHP

Officially, to delete a cookie you should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired –

```
<?php
  setcookie( "name", "", time()- 60, "/","", 0);
  setcookie( "age", "", time()- 60, "/","", 0);
?>
<html>

  <head>
    <title>Deleting Cookies with PHP</title>
  </head>
  <body>
    <?php echo "Deleted Cookies" ?>
  </body>


</html>
```

17.4 Creating a Cookie:

As mentioned earlier, we can set cookies by using the function **setcookie()**.

Example:

```
<!DOCTYPE html>
<?php
$cookie_name = "gfg";
$cookie_value = "GeeksforGeeks";
 // 86400 = 1 day
setcookie($cookie_name, $cookie_value, time() + (86400 * 15), "/");
?>
<html>
<body>
   <?php
   if(!isset($_COOKIE[$cookie_name])) {
        echo "Cookie named '" . $cookie_name . "' is not set!";
```

```
  }
  else
          {
      echo "Cookie '" . $cookie_name . "' is set!<br>";
      echo "Value is: " . $_COOKIE[$cookie_name];
    }
  ?>
</body>
</html>
```

Output:

```
Cookie 'gfg' is set!
Value is: GeeksforGeeks
```

Deleting Cookie:

- There is no special dedicated function provided in PHP to delete a cookie.
- All we have to do is to update the **expire-time** value of the cookie by setting it to a past time using the **setcookie()** function.
- A very simple way of doing this is to deduct a few seconds from the current time.

Syntax:

```
setcookie(name, time() - 3600);
```

Example:

```
<!DOCTYPE html>
<?php
// Set the expiration date to one hour ago
setcookie("gfg", "", time() - 3600);
?>
<html>
 <body>
   <?php
    echo "Cookie 'gfg' is deleted.";
    ?>
</body>
</html>
```

Output:

```
Cookie 'gfg' is deleted.
```

17.4.CREATING SESSION COOKIE

- session_set_cookie_params (int $lifetime [, string $path [, string $domain [, bool $secure = FALSE [, bool $httponly = FALSE]]]]) : bool
- session_set_cookie_params (array $options) : bool
- Set cookie parameters defined in the php.ini file. The effect of this function only lasts for the duration of the script. Thus, you need to call session_set_cookie_params() for every request and before session_start() is called.

❖ This function updates the runtime ini values of the corresponding PHP ini configuration keys which can be retrieved with the ini_get().

Parameters :

lifetime

Lifetime of the session cookie, defined in seconds.

path

Path on the domain where the cookie will work. Use a single slash ('/') for all paths on the domain.

domain

Cookie domain, for example 'www.php.net'. To make cookies visible on all subdomains then the domain must be prefixed with a dot like '.php.net'.

secure

If TRUE cookie will only be sent over secure connections.

http only.

If set to TRUE then PHP will attempt to send the httponly flag when setting the session cookie.

options

✓ An associative array which may have any of the keys lifetime, path, domain, secure, httponly and samesite.
✓ The values have the same meaning as described for the parameters with the same name.
✓ The value of the samesite element should be either Lax or Strict.
✓ If any of the allowed options are not given, their default values are the same as the default values of the explicit parameters.
✓ If the samesite element is omitted, no SameSite cookie attribute is set.

Return Values

Returns TRUE on success or FALSE on failure.

17.6.CREATING QUERY STRING

The information can be sent across the web pages. This information is called query string.

- This query string can be passed from one page to another by appending it to the address of the page.
- You can pass more than one query string by inserting the & sign between the query strings.
- A query string can contain two things: the query string ID and its value. The query string passed across the web pages is stored in $_REQUEST, $_GET, or $_POST variable.
- Whether you passed the query string by using GET or POST method, it is stored in $_REQUEST variable. If you want to access the query string you can use these variables.
- You should note that whether the query string is passed by the GET or POST method it can be accessed by using the $_REQUEST variable.
- If you want to use $_GET variable to access the query string the form method need to be GET. Also you can use $_POST variable to get the query string if the form method is POST.

  In the following example, the query strings username and email(the names of texboxes) are passed from a page called login.php to another page called welcome.php when you click the submit button.

Login.php:

```
<html>
<head>
<title>Login form</title>
</head>
<body>
<form action="welcome.php" method=”get”>
<table>
<tr>
<td>User name:</td><td> <input type="text" name="username" ></td>
</tr>
<tr>
<td>E-mail: </td><td><input type="text" name="email" ></td>
</tr>
<tr>
<td><input type="submit" name="sub" value="submit"></td>
</tr>
</table>
</form>
</body>
</html>
```

welcome.php:

```
<?php
echo "<strong>Welcome ".$_GET['username']. "!</strong><br/>";
echo "Please remember this e-mail: ".$_GET['email']. " for later use.";
?>
```

CREATING QUERY STRING:

[9:47 pm, 31/10/2020] Sathya M.sc: php

Search

PHP 7.4.12 Released!

Change language:

English

Edit Report a Bug

http_build_query

(PHP 5, PHP 7)

http_build_query — Generate URL-encoded query string

Description ¶

http_build_query (mixed $query_data [, string $numeric_prefix [, string $arg_separator [, int $enc_type = PHP_QUERY_RFC1738]]]) : string

Generates a URL-encoded query string from the associative (or indexed) array provided.

Parameters ¶

query_data

May be an array or object containing properties.

If query_data is an array, it may be a simple one-dimensional structure, or an array of arrays (which in turn may contain other arrays).

If query_data is an object, then only public properties will be incorporated into the result.

numeric_prefix

If numeric indices are used in the base array and this parameter is provided, it will be prepended to the numeric index for elements in the base array only.

This is meant to allow for legal variable names when the data is decoded by PHP or another CGI application later on.

arg_separator

arg_separator.output is used to separate arguments but may be overridden by specifying this parameter.

enc_type

By default, PHP_QUERY_RFC1738.

If enc_type is PHP_QUERY_RFC1738, then encoding is performed per » RFC 1738 and the application/x-www-form-urlencoded media type, which implies that spaces are encoded as plus (+) signs.

If enc_type is PHP_QUERY_RFC3986, then encoding is performed according to » RFC 3986, and spaces will be percent encoded (%20).

Return Values ¶

Returns a URL-encoded string.

Examples ¶

Example #1 Simple usage of http_build_query()

```
<?php
$data = array(
   'foo' => 'bar',
   'baz' => 'boom',
   'cow' => 'milk',
   'php' => 'hypertext processor'
);

echo http_build_query($data) . "\n";
echo http_build_query($data, '', '&amp;');

?>
```

The above example will output:

```
foo=bar&baz=boom&cow=milk&php=hypertext+processor
```

```
foo=bar&amp;baz=boom&amp;cow=milk&amp;php=hypertext+process
or
```

Example #2 http_build_query() with numerically index elements.

```
<?php

$data = array('foo', 'bar', 'baz', 'boom', 'cow' => 'milk', 'php' => 'hypertext
processor');


echo http_build_query($data) . "\n";

echo http_build_query($data, 'myvar_');

?>
```

The above example will output:

```
0=foo&1=bar&2=baz&3=boom&cow=milk&php=hypertext+processor

myvar_0=foo&myvar_1=bar&myvar_2=baz&myvar_3=boom&cow=milk
&php=hypertext+processor
```

Unit -V

SESSION

1. What Is a Session

- A session is a mechanism to persist information across the different web pages to identify users as they navigate a site or app.
- The HTTP protocol is a stateless protocol, which means that there's no way a server can remember a specific user between multiple requests.
- For example, when you access a web page, the server is just responsible for providing the contents of the requested page.
- So when you access other pages of the same website, the web server interprets each and every request separately, as if they were unrelated to one another.
- There's no way for the server to know that each request originated from the same user.

2. Starting a Session

- Whenever you want to deal with session variables, you need to make sure that a session is already started.
- There are a couple of ways you can start a session in PHP.
- Use the session_start Function
- This is the method that you'll see most often, where a session is started by the session_start function.

```
<?php
// start a session
session_start();
 // manipulate session variables
?>
```

- The important thing is that the session_start function must be called at the beginning of the script, before any output is sent to the browser.
- Otherwise, you’ll encounter the infamous Headers are already sent error.
- Automatically Start a Session
- If there’s a need to use sessions throughout your application, you can also opt in to starting a session automatically without using the session_start function.
- There’s a configuration option in the php.ini file which allows you to start a session automatically for every request—session.auto_start.
- By default, it’s set to 0, and you can set it to 1 to enable the auto startup functionality.
- session.auto_start = 1
- On the other hand, if you don’t have access to the php.ini file, and you're using the Apache web server, you could also set this variable using the .htaccess file.

4. Working with Session Variables

Once a session is started, the $_SESSION super-global array is initialized with the corresponding session information.

By default, it’s initialized with a blank array, and you can store more information by using a key-value pair.

```
<?php

// start a session

session_start();

// initialize session variables

$_SESSION['logged_in_user_id'] = '1';

$_SESSION['logged_in_user_name'] = 'Tutsplus';


// access session variables

echo $_SESSION['logged_in_user_id'];

echo $_SESSION['logged_in_user_name'];

?>
```

- We’ve started a session at the beginning of the script using the session_start function.
- Following that, we’ve initialized a couple of session variables.
- Finally, we’ve accessed those variables using the $_SESSION super-global.
- When you store the data in a session using the $_SESSION super-global, it’s eventually stored in a corresponding session file on the server which was created when the session was started.
- In this way, the session data is shared across multiple requests.

5. Destroying a Session

The session_destroy function deletes everything that’s stored in the current session.

Having said that, it doesn't unset global variables associated with session or unset the session cookie.

So if you're using the session_destroy function to log out a user, you must unset the $_SESSION variable and unset the session cookie as well .

```
<?php

// start a session

session_start();

// destroy everything in this session

unset($_SESSION);

if (ini_get("session.use_cookies")) {

   $params = session_get_cookie_params();

   setcookie(session_name(), '', time() - 42000, $params["path"],
$params["domain"], $params["secure"],$params["httponly"]);

}

session_destroy();

?>
```

5. Passing Session Id

The server creates a unique number for every new session.

If you want to get a session id, you can use the session_id function, as shown in the following snippet.

```
<?php

session_start();

echo session_id();

?>
```

That should give you the current session id.

The session_id function is interesting in that it can also take one argument—a session id.

If you want to replace the system-generated session id with your own, you can supply it to the first argument of the session_id function.

```
<?php

session_id(YOUR_SESSION_ID);

session_start();

?>
```

6. Encoding and Decoding Session variables

- ❖ In PHP, session encodes and decode operations are automatically performed while storing session data into memory and reading stored session, respectively.
- ❖ While encoding, the $_SESSION array is converted into serialized string format and decoding reverts serialized string back to its original form.
- ❖ This serialization will not return the same format like PHP serialize()
- ❖ The encoded session data contains all $_SESSION elements separated by the semicolon.

- ❖ Each element contains three parts: session index, session length, and session value

<u>For example,</u>

index1|s:length1:"value1";index2|s:length2:"value2";...

session data while invoking session_decode.

<u>Example:</u>

```
<?php

session_start();

$_SESSION["product_code"] = "2222";

$_SESSION["logged_in"] = "yes";

$enc_session = session_encode();

print "<b>Encoded Session Data:<br/></b>";

print $enc_session . "<br/><br/>";

// Changing session values

$_SESSION['product_code'] = "2000";

$_SESSION["logged_in"] = "no";

// printing $_SESSION

print "<b>SESSION Array:<br/></b>";

print "<pre>";

print_r($_SESSION);

print "</pre>";

session_decode($enc_session);
```

```
// printing Reloaded $_SESSION
print "<b>Reloaded SESSION Array:<br/></b>";
print "<pre>";
print_r($_SESSION);
print "</pre>";
?>
```

Output:

```
Encoded Session Data:
product_code|s:4:"2222";logged_in|s:3:"yes";
Changed SESSION values:
Array (
   [product_code] => 2000
   [logged_in] => no
)Reloaded SESSION Array:
Array(
   [product_code] => 2222
   [logged_in] => yes
)
```

7. Disk Access,I/O

PHP | disk_free_space( ) Function

- The disk_free_space() function in PHP is an inbuilt function which is used to return the amount of free space in a specified directory.

- The disk_free_space() function denotes the free space in bytes
- It returns the available space on a filesystem or on a disk partition.
- The disk_free_space() function returns the number of bytes available on the corresponding filesystem or disk partition for a specified directory inputted as a string.

Syntax:

float disk_free_space ($directory)

- Parameters: The disk_free_space() function in PHP accepts one parameter which is $directory. This parameter specifies the directory which has to be checked.
- Return Value: It returns the available space on a filesystem or on a disk partition.
- Errors And Exception:
- The disk_free_space() function in PHP may give improper results if a file name is given as parameter instead of a directory.
- The disk_free_space() function in PHP doesn’t works for remote files.It only works on files which are accessible by the server’s filesystem.

Examples:

Input : disk_free_space("D:");

Output : 10969328844

Input : disk_free_space("C:");

Output : 10969327231

8. E- mail?

- PHP mail is the built in PHP function that is used to send emails from PHP scripts.
- The mail function accepts the following parameters;
- Email address
- Subject
- Message
- CC or BC email addresses
- It's a cost effective way of notifying users on important events.
- Let users contact you via email by providing a contact us form on the website that emails the provided content.
- Developers can use it to receive system errors by email
- You can use it to email your newsletter subscribers.
- You can use it to send password reset links to users who forget their passwords
- You can use it to email activation/confirmation links. This is useful when registering users and verifying their email addresses

Sending mail using PHP

The PHP mail function has the following basic syntax

```
<?php

mail($to_email_address,$subject,$message,[$headers],[$parameters]);

?>
```

HERE,

"$to_email_address" is the email address of the mail recipient

“$subject” is the email subject

“$message” is the message to be sent.

“[$headers]” is optional, it can be used to include information such as CC, BCC

CC is the acronym for carbon copy.

It’s used when you want to send a copy to an interested person i.e. a complaint email sent to a company can also be sent as CC to the complaints board.

BCC is the acronym for blind carbon copy.

It is similar to CC. The email addresses included in the BCC section will not be shown to the other recipients.

9. PHP File Upload

- ✓ PHP allows you to upload single and multiple files through few lines of code only.
- ✓ A PHP file upload feature allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

<u>PHP File Upload Example</u>

File: uploadform.html

```
<form action="uploader.php" method="post" enctype="multipart/form-data">
   Select File:
   <input type="file" name="fileToUpload"/>
   <input type="submit" value="Upload Image" name="submit"/>
</form>
```

File: uploader.php

```
<?php
$target_path = "e:/";
$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);
if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)) {
   echo "File uploaded successfully!";
} else{
   echo "Sorry, file not uploaded, please try again!";
}
?>
```

10. File Download in PHP

- Using PHP you can create web page to download file easily using built-in readfile() function.
- The readfile() function reads a file and writes it to the output buffer.

Example

```
<?php
$file_url = 'http://www.your_remote_server.com/f.txt';
header('Content-Type: application/octet-stream');
header("Content-Transfer-Encoding: utf-8");
header("Content-disposition: attachment; filename=\"" . basename($file_url) .
"\"");
readfile($file_url);
?>
readfile($file_url); ?>
```

11.Environment variables in Php

- $_ENV is another superglobal associative array in PHP.
- It stores environment variables available to current script.
- $HTTP_ENV_VARS also contains the same information, but is not a superglobal, and now been deprecated.
- Environment variables are imported into global namespace.
- Most of these variables are provided by the shell under which PHP parser is running.
- Hence, list of environment variables may be different on different platforms.
- This array also includes CGI variables in case whether PHP is running as a server module orCGI processor.
- PHP library has getenv()function to retrieve list of all environment variables or value of a specific environment variable

getenv

Following script displays values of all available environment variables.

```
<?php

$arr=getenv();

foreach ($arr as $key=>$val)

echo "$key=>$val";

?>
```

12. Random number

The rand() is an inbuilt-function in PHP used to generate a random number.

Syntax:

rand()

The rand() function is use to generate a random integer.

To generate a random integer in some range:

Syntax

rand(min,max)

min specifies the lowest value that will be returned.

max specifies the highest value to be returned.

This function will generate a random value in the range [min,max]

Note

If the min and max value is not specified, default is 0 and getrandmax() respectively.

Example

rand() will return a random integer between 0 and getrandmax().

rand(15,35) will return a random integer in the range [15,35].

```
<?php
   $randomNumber = rand();
   print_r($randomNumber);
   print_r("\n");
   $randomNumber = rand(15,35);
```

```
    print_r($randomNumber);
?>
```

13. Introduction to Ajax?

AJAX full form is Asynchronous JavaScript & XML.

- It is a technology that reduces the interactions between the server and client.
- It does this by updating only part of a web page rather than the whole page.
- The asynchronous interactions are initiated by JavaScript.The purpose of AJAX is to exchange small amounts of data with server without page refresh.
- JavaScript is a client side scripting language.
- It is executed on the client side by the web browsers that support JavaScript.JavaScript code only works in browsers that have JavaScript enabled.
- XML is the acronym for Extensible Markup Language.
- It is used to encode messages in both human and machine readable formats. It's like HTML but allows you to create your custom tags.
- For more details on XML, see the article on XML

14. Introduction to The XMLHttpRequest Object

The XMLHttpRequest object can be used to exchange data with a server behind the scenes.

This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.

<u>Syntax for creating an XMLHttpRequest object</u>:

variable = new XMLHttpRequest();

Example

var xhttp = new XMLHttpRequest();

Properties of XMLHttpRequest object

Property   Description

onReadyStateChange

It is called whenever readystate attribute changes. It must not be used with synchronous requests.

readyState   represents the state of the request. It ranges from 0 to 4.

0 UNOPENED open() is not called.

1 OPENED open is called but send() is not called.

2 HEADERS_RECEIVED send() is called, and headers and status are available.

3 LOADING Downloading data; responseText holds the data.

4 DONE The operation is completed fully.

reponseText	returns response as text.
responseXML	returns response as text

15. Method and Properties of XMLHttpRequest Object

XMLHttpRequest Methods

- abort()

  Cancels the current request.

- getAllResponseHeaders()

  Returns the complete set of HTTP headers as a string.

- getResponseHeader(headerName)

  Returns the value of the specified HTTP header.

- open(method, URL)
- open(method, URL, async)
- open(method, URL, async, userName)
- open(method, URL, async, userName, password)

  Specifies the method, URL, and other optional attributes of a request.

  The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.

  The "async" parameter specifies whether the request should be handled asynchronously or not.

  "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.

- send(content)

  Sends the request.

- setRequestHeader(label, value)

  Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties

- onreadystatechange

An event handler for an event that fires at every state change.

- readyState

  The readyState property defines the current state of the XMLHttpRequest object.

State	Description
0	The request is not initialized.
1	The request has been set up.
2	The request has been sent.
3	The request is in process.
4	The request is completed.

The following table provides a list of the possible values for the readyState property –

<u>16. Application of AJAX</u>

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit

submit, and get directed to a new page with new information from the server.

- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven.