



M.I.E.T. Arts & Science College
(Affiliated Bharathidasan University)

PHP PROGRAMMING

Y.J.M.SHERIF M.Tech(CSE).,

Head of the Department,

DEPARTMENT OF COMPUTER APPLICATIONS

CORE COURSE IX

PROGRAMMING IN PHP

Unit I

Essentials of PHP - Operators and Flow Control - Strings and Arrays.

Unit II

Creating Functions - Reading Data in Web Pages - PHP Browser - Handling Power.

Unit III

Object-Oriented Programming –Advanced Object-Oriented Programming .

Unit IV

File Handling –Working with Databases – Sessions, Cookies, and FTP

Unit V

Ajax – Advanced Ajax – Drawing Images on the Server.

UNIT I

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor" and its original name is "Personal Home Page".

- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use What is a PHP File?
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain

HTML

- PHP files have extension ".php" What Can PHP Do?
- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data Why PHP?

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

Unique Features OR Features of PHP

- **Performance** -Scripts written in PHP execute faster than those written in other scripting languages.
- **Portability**- PHP is available for UNIX, Microsoft Windows, Mac OS, and OS/2, and PHP programs are portable between platforms. This ability to easily undertake cross-platform development which is a valuable one.
- **Ease of Use**- PHP is an extremely sophisticated programming language. Its syntax is clear and consistent, so it is easily used by both learning programmers and experienced programmers.
- **Open Source** - PHP is an open-source project—the language is developed by a worldwide team ,make its source code freely available on the Web, and it may be used without payment of licensing fee.
- **Community Support**- It is community-supported language and it offers to the creativity and imagination of hundreds of developers across the world.
- **Third-Party Application Support**- One of PHP's strengths has historically been its support for a wide range of different databases, including MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.
- **PHP's extensible architecture** allows developers to read and write the GIF, JPEG, and PNG image formats; send and receive e-mail using the SMTP, IMAP, and POP3 protocols; interface with Web services using the SOAP and REST protocols; validate input using Perl regular expressions; and create and manipulate PDF documents.

Getting into PHP

It should be clear that to get started building PHP applications, your development environment must contain at least three components: • **A base operating system (OS) and server environment (usually Linux)** • **A Web server** (usually Apache on Linux or IIS on Windows) to intercept HTTP requests and either serve them directly or pass them on to the PHP interpreter for execution

- **A PHP interpreter** to parse and execute PHP code, and return the results to the Web server .

There's also often a fourth optional but very useful component:

- **A database engine** (such as MySQL) that holds application data, accepts connections from the PHP layer, and modifies or retrieves data from the database.

Basic PHP Syntax or Creating Your First PHP Script or Creating Your First PHP page

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

A PHP script can be placed anywhere in the document.

A PHP script starts with **<?php** and ends with **?>**

<?php // PHP code goes here ?>

The default file extension for PHP files is ".php". [Example](#)

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

Output

My first PHP page Hello
World!

Comments in PHP

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

PHP supports several ways of commenting:

```
<html>
<body>
<?php
// This is a single-line comment

# This is also a single-line comment

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

Output

10

PHP Case Sensitivity

In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.

In the example below, all three echo statements below are legal:

```
<html>
<body>
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
</body>
</html>
```

Output
Hello World!
Hello World!
Hello World!

PHP **whitespace insensitive** means that it almost never matters how many whitespace characters you have in a row. one whitespace character is the same as many such characters.

However all variable names are case-sensitive.

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

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

Output

My car is red
My house is
My boat is

Statements are expressions terminated by semicolons

A *statement* in PHP is any expression that is followed by a semicolon (;).

Here is a typical statement in PHP, which in this case assigns a string of characters to a variable called \$greeting

```
$greeting = "Welcome to PHP!";
```

Expressions are combinations of tokens

The smallest building blocks of PHP are the individual tokens, such as numbers, strings, variables (\$two), constants (TRUE), and the special words that make up the syntax of PHP itself like if, else, while, for and etc.

Running PHP Script from Command Prompt

We can run our PHP script on our command prompt. Assuming you have following content in test.php file

```
<?php
echo "Hello PHP!!!!!!";
?>
```

Now run this script as command prompt as follows –

```
$ php test.php
```

It will produce the following result –

```
Hello PHP!!!!!!
```

Escaping Special Characters

If you simply enclose one set of quotation marks within another, PHP will get confused about which quotation marks are to be printed literally, and which ones are simply used to enclose the string value, and will generate a parser error. Therefore, to handle these situations, PHP allows you to escape certain characters by preceding them with a backslash (\). There so-called escape sequences include

Sequence	What It Represents
\n	a line feed character
\t	a tab
\r	a carriage return
\"	a double quotation mark
\'	a single quotation mark
\\	a single \
\\$	a single \$ sign

PHP echo and print Statements

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

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: **echo has no return value while print has a return value of 1 so it can be used in expressions.**

echo can take multiple parameters while print can take one argument. echo is marginally faster than print.

The PHP echo Statement

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

Display Text

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

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>"; echo "I'm
about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?> Output
```

PHP is Fun!

Hello world!

I'm about to learn PHP!

This string was made with multiple parameters.

The PHP print Statement

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

Display Text

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

```
<?php print "<h2>PHP is
Fun!</h2>"; print "Hello
world!<br>"; print "I'm about
to learn PHP!"; ?>
```

Output**PHP is Fun!**

Hello world!

I'm about to learn PHP!

PHP echo() Function

The echo() function outputs one or more strings.

Parameter	Description
strings	Required. One or more strings to be sent to the output

Using PHP “Here” Documents

In fact, there’s another way of displaying text that you should be aware of, and that’s using PHP “here” documents. A *here* document is just some text inserted directly in a PHP page, between two instances of the same token; that token is a word, such as `END`. Then you can display the text in a here document by using the syntax `echo <<<TOKEN`, where *TOKEN* is the word that begins and ends the here document.

Here’s an example, `phphere.php`:

```
<html>
  <head>
    <title>
      Displaying text from PHP
    </title>
  </head>

  <body>
    <h1>
      Displaying text from PHP
    </h1>
    Here's what PHP has to say:
    <br>
    <br>
    <?php
echo <<<END
This example uses
"here document" syntax to display all

the text until the ending token is reached.
END;
    ?>
  </body>
</html>
```

And you can see what this produces in a browser in Figure 1-10. As the figure shows, the text in the here document was displayed in the browser.

Using the token `END` is by no means necessary; here’s the same here document with the token `FINISH`:

```
echo <<<FINISH
This example uses
"here document" syntax to display all
the text until the ending token is reached.
FINISH;
```

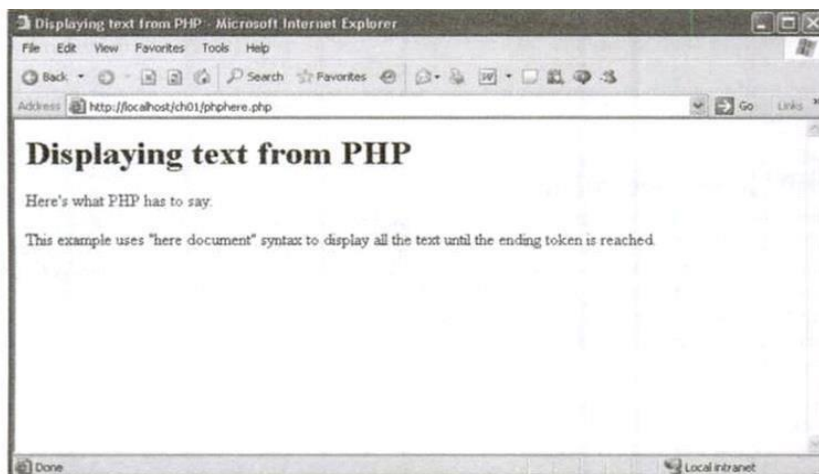


FIGURE 1-10 Displaying a here document in PHP

PHP Variables

Variables are “containers” for storing information.

Creating (Declaring) PHP Variables

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

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

```
Hello world!
5
10.5
```

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

Rules for PHP variables:

- A variable starts with the \$ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (\$age and \$AGE are two different variables)
Remember that PHP variable names are case-sensitive!

Output Variables

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

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

```
<?php
$txt = "Ice cream"; echo
"I like $txt!";
?>
```

```
I like Ice cream
```

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

```
<?php
$txt = "Ice cream";
echo "I like " . $txt . "!";
?>
```

```
I like Ice cream
```

The following example will output the sum of two variables:

```
<?php
$x = 5; $y =
4; echo $x +
$y;
?> 9
```

PHP Variables Scope

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

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

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

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

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

function myTest() { echo "<p>Variable x inside
    function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
```

OUTPUT

Variable x inside function is:

Variable x outside function is: 5

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

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

OUTPUT

Variable x inside function is: 5 Variable
x outside function is:

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared .

PHP The global Keyword

The global keyword is used to access a global variable from within a function. To do this, use the global keyword before the variables (inside the function):

Example

```
<?php
$x = 5; $y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;
```

```

}

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

```

OUTPUT 15

PHP also stores all global variables in an array called `$GLOBALS[index]`. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this:

```

<?php
$x = 5;
$y = 10;
function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}
myTest();
echo $y; // outputs 15
?>

```

PHP The static Keyword

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

To do this, use the **static** keyword when you first declare the variable:

```

<?php    function
myTest() { static $x
= 0; echo $x;
    $x++;
}

```

```

myTest();
myTest();
myTest(); ?>

```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

PHP Constants

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

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

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

Create a PHP Constant

To create a constant, use the `define()` function. [Syntax](#)

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

Parameters:

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

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

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

Welcome to MIET!

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

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

Welcome to MIET!

Differences between constants and variables are

- There is no need to write a dollar sign (\$) before a constant, where as in Variable one has to write a dollar sign.
- Constants cannot be defined by simple assignment, they may only be defined using the `define()` function.
- Constants may be defined and accessed anywhere without regard to variable scoping rules.
- Once the Constants have been set, may not be redefined or undefined.

Valid and invalid constant names

```
// Valid constant names
define("ONE", "first thing");
define("TWO2", "second thing");
define("THREE_3", "third thing");
```

```
// Invalid constant names
define("2TWO", "second thing");
define(".THREE", "third value");
```

PHP Magic constants

PHP provides a large number of predefined constants to any script which it runs.

There are five magical constants that change depending on where they are used.

These special constants or “magical” PHP constants are case-insensitive and are as follows –

Sr.No	Name & Description
-------	--------------------

1	__LINE__ The current line number of the file.
2	__FILE__ The full path and filename of the file. If used inside an include, the name of the included file is returned. Since PHP 4.0.2, __FILE__ always contains an absolute path whereas in older versions it contained relative path under some circumstances.
3	__FUNCTION__ The function name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the function name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
4	__CLASS__ The class name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the class name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
5	__METHOD__ The class method name. (Added in PHP 5.0.0) The method name is returned as it was declared (case-sensitive).

PHP

Data Types

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

PHP supports the following data types:

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

PHP String

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

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

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

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

Hello world!
Hello world!

PHP Integer

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

Rules for integers:

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

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

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

PHP

```
?>
```

```
int(5985)
```

Float

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

In the following example \$x is a float.

```
<?php
```

```
$x = 10.365; var_dump($x);
```

```
?>
```

```
float(10.365)
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE. Booleans are often used in conditional testing

```
$x = true;
```

```
$y = false;
```

PHP Array

An array stores multiple values in one single variable.

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

```
<?php
```

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

```
?>
```

```
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }
```

PHP Object

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

In PHP, an object must be explicitly declared.

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

```
<?php
```

```
class Car {
```

```
    function Car() {
```

```
        $this->model = "BMW";
```

```
    }
```

```
}
```

```
// create an object
```

```
$herbie = new Car(); //
```

```
show object properties
```

```
echo $herbie->model;
```

PHP

```
?>
```

OUTPUT: BMW

NULL Value

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

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

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

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

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

NULL

PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP. A common example of using the resource data type is a database call.

PHP String

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

PHP String Functions

Some commonly used functions to manipulate strings are

Get The Length of a String

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

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

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

The output of the code above will be: 12.

Count The Number of Words in a String

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

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

The output of the code above will be: 2.

PHP

Reverse a String

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

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

The output of the code above will be: !dlrowolleH.

Search For a Specific Text Within a String

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

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

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

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

The output of the code above will be: 6.

Replace Text Within a String

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

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

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

ARRAYS

An array stores multiple values in one single variable:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

OUTPUT:

I like Volvo, BMW and Toyota.

What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP

In PHP, the `array()` function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

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

PHP Indexed Arrays

There are two ways to create indexed arrays:

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

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

the index can be assigned manually:

```
$cars[0] = "Volvo";
```

```
$cars[1] = "BMW";
```

```
$cars[2] = "Toyota";
```

The following example creates an indexed array named `$cars`, assigns three elements to it, and then prints a text containing the array values:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Get The Length of an Array - The `count()` Function

The `count()` function is used to return the length (the number of elements) of an array:

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

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

</body> </html>
OUTPUT:
3
```

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a `for` loop, like this:

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

```

<body>

<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x= 0; $x < $arrlength; $x++) {
    echo          $cars[$x];
    echo "<br>";
}
?>
</body>
</html>

```

OUTPUT:

Volvo
BMW
Toyota

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
or:
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";

```

The named keys can then be used in a script:

```

<!DOCTYPE html>
<html>
<body>

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>

</body> </html>

```

OUTPUT:

Key=Peter, Value=35
 Key=Ben, Value=37
 Key=Joe, Value=43

PHP - Multidimensional Arrays

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

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

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

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

PHP - Two-dimensional Arrays

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

First, take a look at the following table:

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

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

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

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

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

Example <?php

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

```
?>
```

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

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

Row number 0

- Volvo
- 22 □ 18

Row number 1

- BMW
- 15 □ 13

Row number 2

- Saab
- 5
- 2

Row number 3

- Land Rover
- 17
- 15

Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array. This function is explained in function reference.

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

```
<?php
/* First method to create array. */
$numbers = array(1,2,3,4,5);
```

```
foreach( $numbers as $value ){
    echo "Value is $value <br />";
}
```

```
/* Second method to create array. */
```

```
$numbers[0]="one";
$numbers[1]="two";
$numbers[2]="three";
$numbers[3]="four";
$numbers[4]="five";

foreach( $numbers as $value ){
echo"Value is $value <br />";
}
?>
</body>
</html>
```

This will produce the following result – Value
is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
Value is five

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

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

PHP Arithmetic Operators

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

	Operator	Name	Example	Result	
III E	+	Addition	$\$x + \y	Sum of $\$x$ and $\$y$	ning in PHP
	-	Subtraction	$\$x - \y	Difference of $\$x$ and $\$y$	b
	*	Multiplication	$\$x * \y	Product of $\$x$ and $\$y$	
	/	Division	$\$x / \y	Quotient of $\$x$ and $\$y$	
	%	Modulus	$\$x \% \y	Remainder of $\$x$ divided	PHP Assignment Operators
	**	Exponentiation	$\$x ** \y	Result of raising $\$x$ to the (Introduced in PHP 5.6)	
	$x = y$	$x = y$	The left operand gets set to the value of the expression on the right		

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

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

Assignment	Same as...	Description

PHP Comparison Operators

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

<code>x += y</code>	<code>x = x + y</code>	Addition
<code>x -= y</code>	<code>x = x - y</code>	Subtraction
<code>x *= y</code>	<code>x = x * y</code>	Multiplication
<code>x /= y</code>	<code>x = x / y</code>	Division
<code>x %= y</code>	<code>x = x % y</code>	Modulus

Operator	Name	Example	Result
<code>==</code>	Equal	<code>\$x == \$y</code>	Returns true if \$x is equal to \$
<code>===</code>	Identical	<code>\$x === \$y</code>	Returns true if \$x is equal to \$ same type
<code>!=</code>	Not equal	<code>\$x != \$y</code>	Returns true if \$x is not equal
<code><></code>	Not equal	<code>\$x <> \$y</code>	Returns true if \$x is not equal
<code>!==</code>	Not identical	<code>\$x !== \$y</code>	Returns true if \$x is not equal of the same type
<code>></code>	Greater than	<code>\$x > \$y</code>	Returns true if \$x is greater than
<code><</code>	Less than	<code>\$x < \$y</code>	Returns true if \$x is less than
<code>>=</code>	Greater than or equal to	<code>\$x >= \$y</code>	Returns true if \$x is greater than or equal to
<code><=</code>	Less than or equal to	<code>\$x <= \$y</code>	Returns true if \$x is less than or equal to

PHP Increment / Decrement Operators

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

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

Operator	Name	Description
----------	------	-------------

operators are used to decrement a variable's value.

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

++\$x	Pre-increment	Increments \$x by one, then returns \$x	
\$x++	Post-increment	Returns \$x, then increments \$x by one	
--\$x	Pre-decrement	Decrements \$x by one, then returns \$x	
\$x--	Post-decrement	Returns \$x, then decrements \$x by one	
Operator	Name	Example	Result
And	And	\$x and \$y	True if both \$x and \$y
Or	Or	\$x or \$y	True if either \$x or \$y
Xor	Xor	\$x xor \$y	True if either \$x or \$y
&&	And	\$x && \$y	True if both \$x and \$y
	Or	\$x \$y	True if either \$x or \$y
!	Not	!\$x	True if \$x is not true

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator

.	Concatenation	\$txt1 . \$txt2	Concatenation of \$txt1 and \$txt2
.=	Concatenation assignment	\$txt1 .= \$txt2	Appends \$txt2 to \$txt1

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	\$x + \$y	Union of \$x and \$y

==	Equality	\$x == \$y	Returns true if \$x and \$y have the same key/value pairs
===	Identity	\$x === \$y	Returns true if \$x and \$y have the same key/value pairs in same order and of the same types
!=	Inequality	\$x != \$y	Returns true if \$x is not equal to \$y
<>	Inequality	\$x <> \$y	Returns true if \$x is not equal to \$y
!==	Non-identity	\$x !== \$y	Returns true if \$x is not identical to \$y

PHP-Decision Making OR Flow Control

The if, elseif ...else and switch statements are used to take decision based on the different condition. You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements or conditional statements:

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

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

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

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

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

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

Output

Have a good day!

PHP - The if...else Statement

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

```

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

```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```

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

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

```

Output

Have a good day!

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

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

Syntax

```

if (condition) { code to be executed if this
    condition is true;
} elseif (condition) { code to be executed if
    this condition is true;
} else { code to be executed if all conditions are
    false;
}

```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```

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

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") { echo
    "Have a good day!";
} else { echo "Have a good
    night!";
}

```

?>

Output

The hour (of the server) is 13, and will give the following message:

Have a good day!

Switch Statement

The switch statement is used to perform different actions based on different conditions. Use the switch statement to **select one of many blocks of code to be executed**.

Syntax switch (n) { case 1:
code to be executed if n=1;
break;
case 2: code to be executed
if n=2; break;
case 3: code to be executed
if n=3;
break;
... default:
code to be executed if n is different from all labels; }

If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The **default** statement is used if no match is found.

<?php

\$favcolor = "red";

```
switch ($favcolor) {
    case "red": echo "Your favorite
        color is red!"; break;
    case "blue": echo "Your favorite
        color is blue!"; break;
    case "green":
        echo "Your favorite color is green!"; break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
```

?>

Output:

Your favorite color is red!

PHP Loops

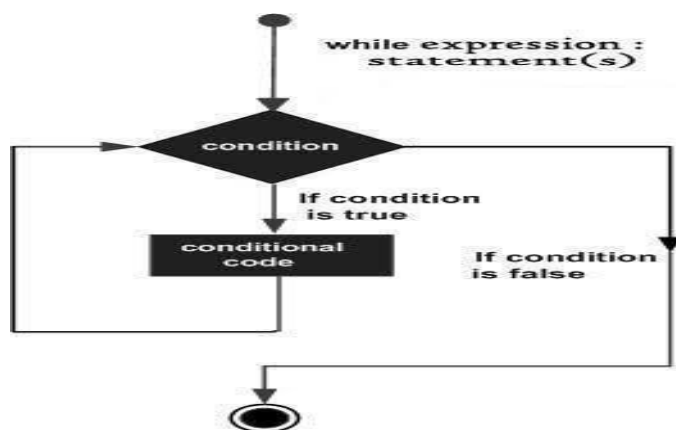
Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

The while Loop

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



Syntax

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

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

```
<?php
$x = 1;

while($x <= 3) {
    echo "The number is: $x <br>";
    $x++;
}
```

?> Output:

```
The number is: 1
The number is: 2
The number is: 3
```

The do...while Loop

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

Syntax

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

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

```
<?php
$x = 1;

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

Output:

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

The for Loop

The for loops execute a block of code a specified number of times.

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

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

Parameters:

- *init* : Initialize the loop counter value
- *test* : Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment* : Increases the loop counter value The example below displays the numbers from 0 to 5:

```
<html>
<body>
<?php
for ($x = 0; $x <= 5; $x++) {
    echo "The number is: $x <br>";
}
?>
</body>
```

```
</html>
```

Output

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

The foreach Loop

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

Syntax

```
foreach ($array as $value) {
    code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element.

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

```
<html>
```

```
<body>
```

```
<?php
```

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

```
foreach ($colors as $value) {
```

```
    echo "$value <br>";
```

```
}
```

```
?>
```

```
</body>
```

```
</html>
```

Output:

```
red green
blue
yellow
```


UNIT-II

PHP User Defined Functions

What is a Function?

- A function is a piece or block of code that performs a specific task whenever it is called.
- A function can be reused many times. It can take input as argument list and return value.
- A function doesn't execute when it's defined, it executes when it is called.

PHP provides us with two major types of functions:

- In PHP there are thousands of **built-in functions** which we can directly use in our program.
Example: var_dump, fopen(), print_r(), gettype() and so on.
- PHP also supports **user defined functions**, where we can define our own functions.

Advantage of PHP Functions

Code Reusability: PHP functions are defined only once and can be invoked many times.

Less Code: It saves a lot of code because you don't need to write the logic many times.

Easy to understand: PHP functions separate the programming logic. So it is easier to understand the flow of the application.

Create a User Defined Function in PHP

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

Syntax function
functionName()

```
{ code to be  
  executed;  
}
```

Program:

```
<html>  
<body>  
<?php  
function hello() // called function  
  
{ echo "Hello  
  world!";  
}  
hello(); // calling function  
?>  
</body>  
</html>
```

Output:

Hello world!

PHP Function Arguments or Parameters

- Information can be passed to functions through arguments. An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

Parameter passing to Functions

There are two different ways to pass parameters

1. Passing arguments by Value or call by value

- On passing arguments using pass by value, the value of the argument gets changed within a function, but the original value outside the function remains unchanged.
- That means a duplicate of the original value is passed as an argument.

```
<?php    function
    sum($a)
    {
        $a+=1; echo "sum inside function=" .
        $a."<br/>"; }
    $a=5;
    calculate($a);
    echo "sum outside function=" . $a;
```

?> *Output*

```
sum    inside    function=6
sum outside function=5
```

2. Passing arguments by reference or call by reference

- On passing arguments as pass by reference, the original value is passed. Therefore, the original value gets altered.
- In pass by reference we actually pass the address of the value, where it is stored using ampersand sign (&).

```
<?php    function
    sum(&$a)
    {
        $a+=1; echo "sum inside function=" .
        $a."<br/>";
    }
    $a=5;
    sum($a);
    echo "sum outside function=" . $a;
```

?> *Output*

```
sum inside function=6
sum outside function=6
```

Functions returning value

A function can return a value using the **return** statement in conjunction. Return sends the value back to the calling code.

```
<?php function add($a,
    $b)
    {
        $c=$a+$b;
        return $c;
    }
    $sum= add(10, 20); echo
        "sum=" . $sum;
?>
```

Output:

Sum=30

```
<?php function add(
    $a,$b)
    { return $a +
        $b;
    }
    echo add(20,30);
?> Output:
Sum=50
```

PHP Function: Default Argument Value

While calling PHP function if you don't specify any argument, it will take the default argument. <?php

```
function Hello($name="JACK")
{
    echo "Hello $name<br/>";
}
Hello("Rajesh");
Hello(); //passing no value
```

```
Hello("John");  
?>
```

OUTPUT

Hello Rajesh
Hello JACK
Hello John

```
<?php  
function Hello($name="JACK")  
{  
echo "Hello $name<br/>";  
}  
Hello(""); //passing no value  
?>
```

OUTPUT

Hello JACK

PHP Recursive Functions

A function which calls itself again and again is Recursive function.

```
<?php  
function fact($n)  
{  
if ($n === 0)  
{  
return 1;  
}  
else  
{  
return $n * fact($n-1); // <--calling itself.  
}  
}  
echo fact(5);  
?>
```

OUTPUT:
120

Reading Data from web pages

PHP Form Handling

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

HTML Form Syntax

```
<form action="server url" method="get or post">  
//input controls e.g. textfield, textarea, radiobutton, button  
</form>
```

1. **action**- the attribute used to enter server url or filename where form to be submitted.
2. **Method**- specifies how to send form data.(get or post)

- We can create and use forms in PHP.
- The form request may be get or post.
- To get form data, we need to use PHP superglobals \$_GET and \$_POST.

PHP Get Method

- 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.□

PHP Post Method

- 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.□

PHP - A Simple HTML Form

eg.html

```

<html>
<body>
<form action="welcome.php" method="get">
Name:      <input      type="text"
name="name"><br> age: <input type="text"
name="age"><br>
<input type="submit" value="submit">
</form>
</body>
</html>

```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP GET method.

Welcome.php

```

<?php
$name=$_GET["name");//receiving name field value in $name variable
$age=$_GET["age"]; echo "Welcome
$name"."<br>".$age";
?> Output:

```

Handling Textbox, Text areas, Submit buttons, checkbox, radio button

The **<input>** tag specifies an input field where the user can enter data. The

<input> element is the most important form element.

The **<input>** element can be displayed in several ways, depending on the type attribute. The different input types are as follows:

text	Defines a one-line text input field
textarea	Defines a multi-line text input field

password	Defines a one-line password input field
submit	Defines a submit button to submit the form to server
reset	Defines a reset button to reset all values in the form.
radio	Defines a radio button which allows select one option.
checkbox	Defines checkboxes which allow select multiple options form.
button	Defines a simple push button, which can be programmed to perform a task on an event.
file	Defines to select the file from device storage.
image	Defines a graphical submit button.
list box	The list box is a graphical control element in the HTML document that allows a user to select one or more options from the list of options.
hidden	A hidden field let web developers include data that cannot be seen or modified by users when a form is submitted.

PHP FORM CONTROLS

File: webpage.html

```
<html>
<head><title>
<h1><center>Php web form</center></h1>
</title>
</head>
<body>
<form method="post" action="webpage.php"> Name:
<input type="text" name="Name"><br> Email ID: <input
type="text" name="Email"><br>
Password: <input type="password" name="Password"><br>Address:
<textarea name="address" cols="50" rows="5">
</textarea><br>
Gender: <input name="radio1" type="radio" value="Male"> Male
<input name="radio1" type="radio" value="Female"> Female
<br><br>
```



```

Class: <input type="checkbox" name="check" value="I">I
<input type="checkbox" name="check" value="II">II
<input type="checkbox" name="check" value="III">III
<br><br>
Select your favourite subject :
<select name="subject">
<option></option>
<option> C</option>
<option> C++</option>
<option> Java</option>
<option> DataBase</option>
<option> No answer</option>
</select>
<br><br>
<input type="submit" value="submit">
<br><br>
<input type="reset" value="reset">
</form>
</body>
</html>


```

File: webpage.php

```

<html>
<body> <?php echo "Welcome!". $_POST
["Name"];echo
"<br>"; echo " You have entered these
values";echo
"<br>"; echo "Email:". $_POST
["Email"];echo
"<br>"; echo "Address:". $_POST
["address"];echo
"<br>"; echo "Gender:". $_POST
["radio1"];echo
"<br>"; echo "Class:". $_POST
["check"];echo
"<br>"; echo "Subject interested:". $_POST
["subject"];echo
"<br>";
?>
</body>
</html>

```

Output:


http://localhost:8080/arp/webpage.html

<h1><center>Php web for... x

File Edit View Favorites Tools Help

Name:

Email ID:

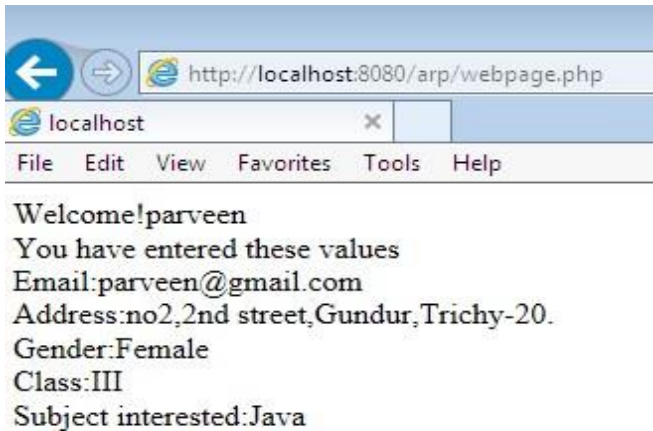
Password:

Address:

gender: ☐ Male ☒ Female

Class: ☐ I ☐ II ☒ III

Select your favourite subject : v



http://localhost:8080/arp/webpage.php

localhost x

File Edit View Favorites Tools Help

Welcome!parveen

You have entered these values

Email:parveen@gmail.com

Address:no2,2nd street,Gundur,Trichy-20.

Gender:Female

Class:III

Subject interested:Java

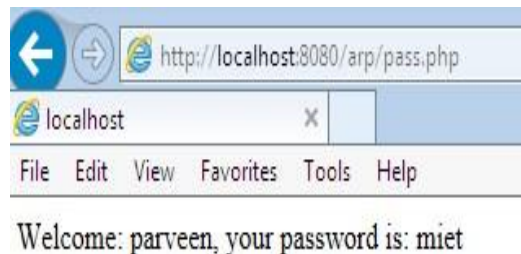
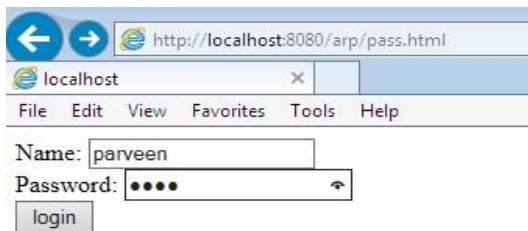
PASSWORD CONTROLS**File: pass.html**

```
<form action="pass.php" method="post">
Name: <input type="text" name="name"/> <br>
Password: <input type="password" name="password"/><br> <tr><td
colspan="2"><input type="submit" value="login"/> </form>
```

File: pass.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:

**Super Global Variables in PHP**

- PHP super global variable is used to access global variables from anywhere in the PHP script.
- PHP Super global variables is accessible inside the same page that defines it, as well as outside the page.
- While local variable's scope is within the page that defines it.

The PHP superglobal variables are:

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

PHP

1. \$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:

```
<?php
$x = 75; $y = 25;
function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
addition();
echo $z;
?>
```

Output:

100

2. PHP \$_REQUEST

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

eg.html

```
<html>
<body>
<form action="welcome.php" method="get">
Name: <input type="text" name="name"><br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Welcome.php

```
<?php
$name=$_REQUEST["name");//receiving name field value in $name variable
echo "Welcome $name"."<br>".$age;"?>
```

3. \$_POST

PHP

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.

eg.html

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Welcome.php

```
<?php
$name=$_POST["name");//receiving name field value in $name variable
echo "Welcome $name".<br>".$age";
?>
```

4. 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.

eg.html

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Welcome.php

```
<?php
$name=$_POST["name");//receiving name field value in $name variable
echo "Welcome $name".<br>".$age";?>
```

5. \$_SERVER

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

PHP

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['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)
\$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
\$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
\$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
\$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
\$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
\$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
\$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by (such as 8080)
\$_SERVER['SCRIPT_NAME']	Returns the path of the current script
\$_SERVER['SCRIPT_URI']	Returns the URI of the current page

PHP Browser - Handling Power

- \$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.
- With the help of \$_SERVER variable we will able to the user browser detail.

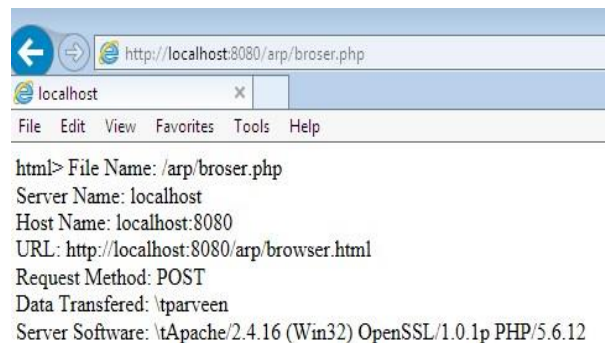
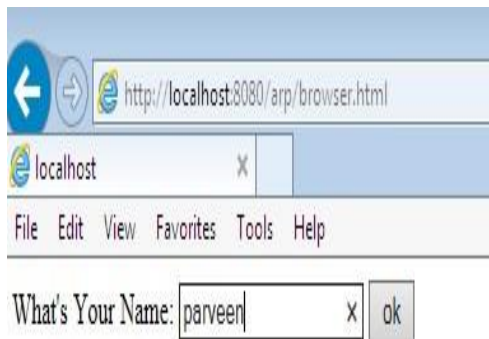
File: test.html

```
<html>
<body>
<form method="post" action="phpbrowser.php"> What's
Your Name:
<input type="text" name="stud">
<input type="submit" value="ok">
```

```
</form>
</body> </html>
```

File: *phpbrowser.php*

```
<html>
<body><?php echo "File Name: ".
$_SERVER['PHP_SELF'];echo
"<br>"; echo "Server Name: ".
$_SERVER['SERVER_NAME'];echo
"<br>";
echo "Host Name: ". $_SERVER['HTTP_HOST']; echo "<br>";
echo "URL: ". $_SERVER['HTTP_REFERER']; echo "<br>"; echo
"Request Method: ". $_SERVER['REQUEST_METHOD'];echo
"<br>"; echo "Data Transferred: ".\t'.
$_REQUEST['stud'];echo
"<br>";
echo "Server Software: ".\t'. $_SERVER['SERVER_SOFTWARE'];
?>
</body>
</html>
```

***Redirect Webpage in PHP or Redirecting web browser using HTTPHeader***

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

File: `redirect.html`

```
<html>
<body>
<form action="redirect.php" method="post">
```

Hello Welcome

```
<input type="submit" value="click"> here
</body> </html>
```

Redirect.php

Use the PHP `header()` function to redirect a user to a different page.

Here when click submit button the page will be redirected to `redirect1.php` file.

```
<html>
```

```
<body> <?php
```

```
echo "welcome";
```

```
header("Location: http://localhost:8080/arp/redirect1.php"); /* Redirect browser */ ?>
```

```
</body>
```

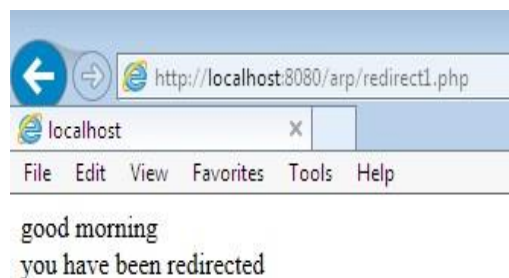
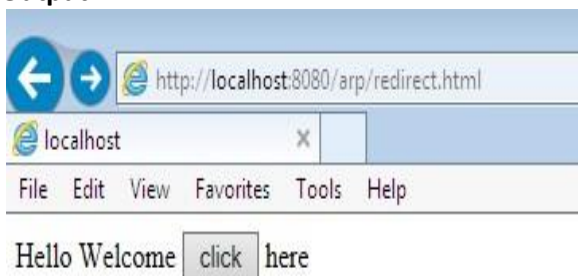
```
</html> redirect1.php <?php echo
```

```
"good morning","<br>"; echo " you
```

```
have been redirected";
```

```
?>
```

Output:



To Find Browser Name in PHP

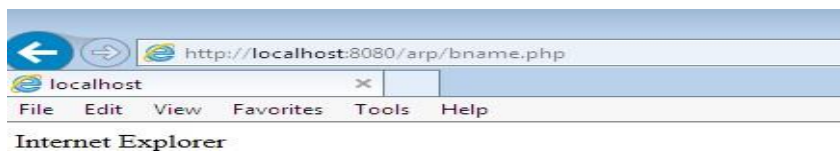
- A Browser is application software used to open web pages from servers over the internet.
- A browser is application software which displays content from the WorldWide Web,
- Today there are many browsers available. Some browsers are as follows:

1. Internet Explorer
2. Firefox
3. Google Chrome
4. Safari etc...

The following PHP code can be used to know the user's browser name. The `$_SERVER["HTTP_USER_AGENT"]` gives the user browser detail.


```
<?php
if(strpos($_SERVER["HTTP_USER_AGENT"],"MSIE"))
{
    echo "Internet Explorer";
}
else if(strpos($_SERVER["HTTP_USER_AGENT"],"GOOGLE CHROME"))
{
    echo "Google Chrome";
}
else if(strpos($_SERVER["HTTP_USER_AGENT"],"MOZILLA"))
{
    echo "FIREFOX";
} else
{
    echo "OTHER";
} ?
```

OUTPUT:



UNIT- III

Object Oriented Programming in PHP

The PHP Object-Oriented Programming concepts are:

- **Class** – A class is a user-defined data type, which includes local methods and local variables.

A class may be a template for making many instances of the same kind of object.

- **Object** – An object is an instance of class. This means that an object is created from the definition of the class and it is loaded in memory.

You define a class once and then make many objects that belong to it.

- **Inheritance** – When the properties and the methods of a class are accessed by another class is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Polymorphism** – This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it take different number of arguments and can do different task.
- **Constructor** – refers to a special type of function which will be called automatically whenever there is an object formation from a class.
- **Destructor** – refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.

Class and 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 classname
{
    Data members;
    Methods;
}
?>
```

Objects of a class is created using the **new** keyword and object name should start with \$ symbol. Her “new” is an operator which is responsible for creating new instance (space for object) of a class.

Syntax:

```
$objectname = new classname();
```

Example:

```
<?php class
demo
{
    $a= "helb"; function
    display()
    {
        echo $this->a;
    }
}
$obj = new demo(); //instatination of object
$obj->display();
?>
```

Output

Helb

Here demo is class which contains \$a-data member and one method display() .

The **\$this** keyword indicates that we use the class's own methods and properties, and allows us to have access to them within the class's scope.

\$this ->data member;

\$this ->methodName();

\$obj is an object created for class demo and call data member and methods.

PHP ACCESS SPECIFIERS

Access Specifiers in a [class](#) are used to specify the accessibility to the class members. That is, it sets some restrictions on the class members not to get directly accessed by the outside functions.

There are 3 types of Access Specifiers in PHP, Public, Private and Protected.

Public : Means "Accessible to All"

Private - Means "Accessible to Same Class"

Protected - Means "Accessible to Same Class and Derived class"

Public Access

Class members with this access specifier will be publicly accessible from anywhere, even from outside the scope of the class.

```
<?php class
demo
{
    public $name="Ajeet";
    function display()
```

```

{
    echo $this->name;
}
}
class child extends demo
{
    function show()
    {
        echo $this->name;
    }
}
$obj= new child;
echo      $obj-
>name;    $obj-
>display();
$obj->show();
?>

```

Output:

Ajeet

Ajeet

Ajeet Here member and function are all given with public access specifier. So they can be easily accessible from inside and outside world.

Private Access

Class members with this keyword will be accessed within the class itself. It protects members from outside class access with the reference of the class instance.

```

<?php
Class Person
{
    Private $name="Sam"; private
    function show()
    {
        echo "This is private method of parent class";
    }
}
class Friend extends Person
{
    function show1()
    {
        echo $this->name;
    }
}

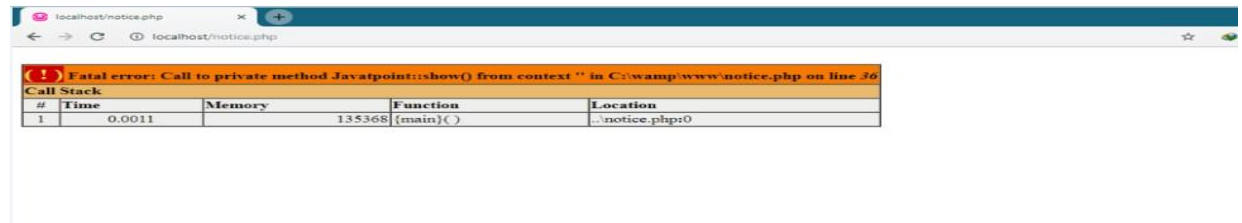
```

```

$obj= new Friend;
$obj->show();
$obj->show1();
?>

```

Output:



Here both member and function are given private specifier, we are not able to access them through derived class. So we get fatal error.

Protected Access

It is same as private, except by allowing subclasses to access protected superclass members.

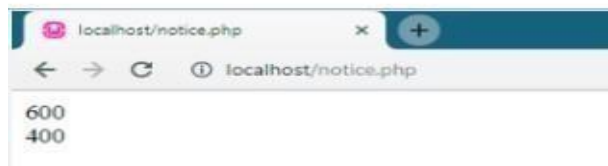
```

<?php
Class Person
{
    protected $x=500;
    protected $y=100;
    function add()
    {
        echo $sum=$this->x+$this->y."<br/>";
    }
}
class Friend extends Person
{
    function sub()
    {
        echo $sub=$this->x-$this->y."<br/>";
    }
}
$obj= new Friend;
$obj->add();
$obj->sub();

?>

```

Output:



Constructor

- A constructor is a special function which will be called automatically whenever object is created .
- Constructors have a special name in PHP using "**__construct**" or **same name as class name**.

Destructor

- The construct function starts with two underscores (_).
- ✓ A destructor is a special function which will be called automatically whenever an object is deleted or goes out of scope.
- ✓ We create destructor by using "**__destruct**" **function**.
- ✓ Destructor should not have any argument.

```
<?php class
```

```
add
```

```
{
```

```
    function __construct ($a,$b) //constructor
```

```
    {
```

```
        $this->a=$a;
```

```
        $this->b=$b;
```

```
    echo ($a+$b);
```

```
    }
```

```
    function __destruct() //destructor to destroy object
```

```
    {
```

```
    echo "destroy the object";
```

```
    } }
```

```
$obj = new add(4,5); //passing value directly when object created ?>
```

OUTPUT:

9 destroy the object

Inheritance

- ✓ It is a concept of accessing the features of one class from another class.

- ✓ If we inherit the class features into another class, we can access both class properties. We can extend the features of a class by using 'extends' keyword.
- ✓ The base class is original class and derived class is another class which uses the property of another class.
- ✓ PHP supports only single inheritance, where only one class can be derived from single parent class.
- ✓ We can simulate multiple inheritance by using interfaces.

```
<?php class
a
{
    function show()
    {
        echo "welcome";
    }
}
class b extends a
{
    function display()
    {
        echo "<br>";
        echo "Trichy";
    }
}
$obj = new b();
$obj->show();
$obj->display();
?>
```

OUTPUT:

welcome
Trichy

- ✓ Here **a** is base class and **b** is derived class.
- ✓ **Object only created for derived class b.**
- ✓ Both methods are called by single object

\$obj. Constructor and Inheritance

PHP allows constructor usage along with Inheritance.

```
<?php class demo
```

```

{
public function demo() //constructor because same name as class
{
echo "constructor1...";
}
}

class demo1 extends demo
{
public function _construct()
{
echo parent::demo(); //base class constructor called in derived class
echo "constructor2...";
}
}
$obj= new demo1();
?>

```

OUTPUT:

constructor1...constructor2...

The base class constructor called in derived class by
parent::demo or parent::_construct.

Method Overriding or Overriding Methods

- In PHP, as in other OOPs language, we can Override methods. □
- That means we can redefine a base class method in a derived class. □
- ***Method Overriding or Overriding Methods is a process of redefining a function of a base class with same name, signature in derived class.*** □

```

<?php class
a
{
function show()
{
echo "welcome";
}
}
class b extends a
{
function show()
{
echo "<br>";
echo "Trichy";
}
}

```



```
}  
$obj= new b();  
$obj->show();  
?>
```

OUTPUT:**Trichy**

- ✓ Here both a (base class) and b (derived class) have same function show() with same signature.
- ✓ Whenever the object is created, the method in derived class will be called and executed. The show() function in base class will not be called. This is called method overriding.
- ✓ Therefore Welcome won't be printed.

The final Keyword

The **final** keyword can be used to prevent class inheritance or to prevent method overriding.

The following example shows how to prevent class inheritance:

```
<?php  
class a  
{  
    final function show()  
    {  
        echo "welcome";  
    }  
}  
class b extends a  
{  
    Function show()  
    {  
        echo "<br>";  
        echo "Trichy";  
    }  
}  
$obj= new b();  
$obj->show();
```

```
?>
```

OUTPUT:

```
PHP Fatal error: Cannot override final method a::show() in  
/home/bAkPCG/prog.php on line 20
```

- ✓ Since final keyword is used in base class function show() we cant override methods . so fatal error is thrown.

Method Overloading or Overloading method

- ✓ Overriding a method means redefining it, but overloading means creating an alternative version with a different argument list.
- ✓ In Simple, method overloading means in a class two methods have same name but different argument list.

```
<?php
```

```
Class shape
```

```
{
function __call($functionname,$arguments) //__call is magic function which accepts
                                           function name and arguments
{
    if($functionname == 'area')           // It will match the function name
    {
        switch(count($arguments))        // If there is only one
                                           argument then area of
                                           circle is calculated
        {
            case 1: return 3.14 * $arguments[0] *
                $arguments[0];

            case 2: // If two arguments then area is rectangle return
                $arguments[0]*$arguments[1];
        }
    }
}
```

```
$s = new Shape;
```

```
echo($s->area(2)); of // Function call for area
circle echo "\n";
echo ($s->area(4,2)); // Function call for area of rectangle
```

```
?>
```

OUTPUT:

12.56 8

So here when we pass one argument area of circle is calculated. When two arguments passed area of rectangle is calculated.

UNIT -IV

COOKIES

A cookie is a **small amount of information** sent from the server to client, and then sent back to the server each time it is accessed by the client.

Creating cookies

In PHP cookies are created using `setcookie()` function:

Setcookie (name,value,expire, path, domain, secure)

name: The name of your cookie.

value: The value that is stored in your cookie

life time of cookie given in seconds **path :** Server path

in which the cookie will be available **domain :** To

which domain the cookie is available.

secure : If set true, the cookie is available over secure connection only

Reading cookies

To access a cookie received from a client, PHP uses super **global array** `$_COOKIE` **[]**

You can use `isset()` function to check if a cookie is already set or not.

Example:

```
<?php
$college = "MIET"; setcookie("collegename",
$college, time()+3600);if
(isset($_COOKIE['collegename'])) echo
"Welcme to ".$_COOKIE['collegename']; else
echo "Sorry...";
?>
```

Output: Welcme
to MIET

DELETING COOKIES

To delete a cookie, you simply set the value of the cookie to null, and also set the expiring time of the cookie in the past.

Here's some code to delete cookies set in the above example:

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

SESSIONS

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session creates unique user id (UID) for each browser to recognize the user and avoid conflict between multiple browsers.

The UID is either stored in a cookie or is propagated in the URL. **Starting a PHP session:**

Before you can store user information in your PHP session, you must first start up the session. Sessions in PHP are started by using the `session_start()` function.

```
<?php session_start();
?>
```

- This tells PHP that a session is requested.
- A session ID is then allocated at the server end.
- session ID looks like:

sess_f1234781237468123768asjkhfa7891234g

STORING A SESSION VARIABLE

When you want to store user data in a session use the `$_SESSION[]` SUPERGLOBAL VARIABLE.. This is where you both store and retrieve session data. Example:

```
<?php session_start();
echo "Usage of Session Variables";
$_SESSION['favcolor'] = 'green'; //store session data
$_SESSION['animal'] = 'cat'; echo "Favcolor = " . $_SESSION['favcolor'] . "<br>";
//retrieve session data echo "Pet animal = " . $_SESSION['animal'];
?>
```

Output:

Usage of Session Variables

Favcolor = green

Pet animal = cat

DESTROYING A SESSION

If you wish to delete some session data, you can use the `unset()` or the `session_destroy()` function.

```
<?php session_start();
session_destroy();
?>
```

File Handling in PHP

- A file is simply a resource for storing information on a computer.
- File handling is an important part of any web application.

- PHP File handling allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

Opening and Closing Files

The fopen function is used to open files. It has the following syntax `fopen($filename,$mode,)`

HERE,

- “fopen” is the PHP open file function
- “\$filename” is the name of the file to be opened
- “\$mode” is the mode in which the file should be opened

File Modes

The following table shows the different modes the file may be opened in.

Mode	Description
r	Read Only mode, with the file pointer at the start of the file. r+
	Read/Write mode, with the file pointer at the start of the file.
w	Open the file for writing only. Any existing content will be lost. If the file does not exist, PHP attempts to create it.
w+	Open the file for WRITE/READ. Any existing content will be lost. If the file does not exist, PHP attempts to create it.
a	Append mode, with the file pointer at the end of the file. If the file doesn't exist, fopen will attempt to create the file.
a+	Read/Append, with the file pointer at the end of the file. If the file doesn't exist, fopen will attempt to create the file

Closing a File

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

`fclose($fp);`

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

OUTPUT

T:

Welcome to miet

Reading files

1. fread() function

The fread() function can be used to read a **string of characters** from a file. It takes two arguments: a file handle and the number of characters to read.

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

```
<?php
$myfile = fopen( "miet.txt", "r" ) or die("Unable to open file!");
$data = fread( $myfile, 10 );
echo $data; fclose($myfile);
?>
```

This code reads the first ten characters from miet.txt and assigns them to \$data as a string.

2. PHP Read Single Line - fgets()

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

below outputs the first line of the "miet.txt" file:

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

OUTPU

T:

Welcome to Miet

PHP Check End-Of-File - feof()

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

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

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

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

OUTPU

T:

Welcome to

Miet.Have a nice day.

3. Reading from files using fgetc() function

If you want to read only one character at a time you can use the fgetc() function. The

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

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

OUTPUT:

Welcome to Miet.

Have a nice day.

Writing to Files

1. fwrite() function

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

The fwrite() function can be used to write a string of characters into 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"

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

Notice that we wrote to the file "newfile.txt" twice.

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

Joh
n
Do
ra

2. Writing Single Line - fputs()

The fputs() function can be used to write **single line into a file**.

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

Output:

Welcome to trichy

3. fputc() function

If you want to write only one character at a time you can use the fputc() function. The example below writes the "newfile.txt" file character by character,

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

Output:

Welcome to trichy

PHP File_exists Function

This function is used to determine whether a file exists or not. The file_exists function has the following syntax.

```
<?php
file_exists($filename);
?>
```

HER

E,

"file_exists()" is the PHP function that returns true if the file exists and false if it does not exist.

"\$file_name" is the path and name of the file to be checked

The code below uses file_exists function to determine if the file my_settings.txt exists.

```
<?php
```

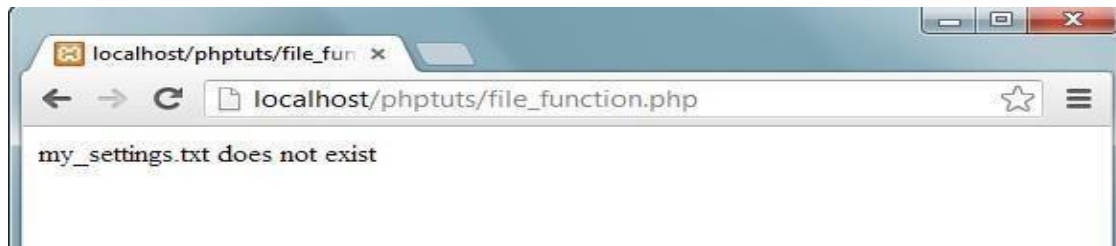
```

    if (file_exists('my_settings.txt'))
    { echo 'file found!';
    } el
    se
    {   echo 'my_settings.txt does not exist';

    } ?
    >

```

IF file not found then,



PHP Delete File- unlink()

In PHP, we can delete any file using **unlink()** function. The unlink() function accepts one argument only: file name. It is similar to UNIX C unlink() function.

PHP unlink() generates E_WARNING level error if file is not deleted. It returns TRUE if file is deleted successfully otherwise FALSE.

Syntax

```

<?php
$status=unlink('data.txt'); if($status){
echo "File deleted successfully";
} else
{
echo "Sorry!";
}
?>

```

Output

File deleted successfully

PHP Copy Function

The PHP copy function is used to copy files. It has the following basic syntax.

```
copy($file,$copied_file);
```

HERE,

“\$file” specifies the file path and name of the file to be copied.
 “copied_file” specified the path and name of the copied file The
 code below illustrates the implementation

```
<?php
copy('miet.txt', 'trichy.txt') or die("Could not copy file"); echo
"File successfully copied to 'trichy.txt'";
```

?> **fseek()** **function**

This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.

Syntax

`fseek(file, offset)`

parameters are

file Required. Specifies the open file to seek in

offset Required. Specifies the new position (measured in bytes from the beginning of the file)

In this example Read first line from the open file, then move the file pointer back to the beginning of the file.

```
<?php
$file = fopen("test.txt", "r");
// Read first
lineecho fgets($file);
// Move back to beginning of file
fseek($file, 0); fclose($file);
?>
```

Hello, this is a test file.

touch() function

The touch() function sets the access and modification time of the specified file.

Note: If the specified file does not exist, it will be created.

Syntax touch(*filename*,
time, *atime*)

Example

Sets the access and modification time of the specified file:

```
<?php
filename = "test.txt"; if
(touch($filename)) {
echo $filename . " modification time has been changed to present time";
} else { echo "Sorry, could not change modification time of " .
$filename;
}
?>
```

DIRECTORY FUNCTION IN PHP

1. mkdir() - Creates a directory

We create a directory through the PHP mkdir() function.

This function is very simple. All you need to know is that it takes one parameter, which is the name of the directory that you want to create. The code to create a directory called miet is shown below.

```
<?php
mkdir("miet");
?>
```

2. opendir() - Open a directory

The opendir() function in PHP is an inbuilt function which is used to open a directory.

Syntax:

```
opendir($path, $context)
```

The opendir() function in PHP accepts two parameters.

- **\$path** : It is a mandatory parameter which specifies the path of the directory to be opened.
 - **\$context** : It is an optional parameter which specifies the behavior of the stream.
- Return Value:** It returns a directory handle resource on success, or FALSE on failure.

3. readdir()

The readdir() function is used to read a file or directory name from a directory.

Syntax readdir(*dir*)

dir Optional. Specifies the directory handle resource previously opened with opendir()

4. **closedir()** the **closedir()** function used to close a directory

Example:

```
<?php
$dh = opendir("/images/"); // Opening a directory if(is_dir($dh))
{
    echo("Directory Opened Successfully.");

    while (($file = readdir($dh)) !== false) //read the file name
    { echo "filename:" . $file . "<br>";

    }
    closedir($dh); // closing the directory else
    {
        echo("Directory Cannot Be Opened.");
    }
?>
```

Output:

Directory Opened Successfully

```
filename: cat.gif
filename: dog.gif
filename:
horse.gif
```

5. **scandir()** Read a directory(Return result in array format)

The scandir() function returns an array of files and directories of the specified directory.

`scandir(directory, order)`

The Parameter used in scandir() function are

directory	Required. Specifies the directory to be scanned
order	Optional. Specifies the sorting order. Default sort order is alphabetical in ascending order (0). Set to SCANDIR_SORT_DESCENDING or 1 to sort in alphabetical descending order

```
<?php
$dir = "/images/";
$a = scandir($dir); // Sort in ascending order - this is default
```

```
$b = scandir($dir,1);           // Sort in descending order
print_r($a);
print_r($b);
?>
```

Output: Array

```
(
[0] => .
[1] => ..
[2] => cat.gif
[3] => dog.gif
[4] => horse.gif
[5] => myimages
)
Array (
[0] => myimages
[1] => horse.gif [2] => dog.gif [3] => cat.gif [4] => ..
[5] => .
)
```

6. rmdir() Removes a directory

The rmdir() function removes an empty directory.

rmdir(*dir*) *dir* **Required.** Specifies the path to the directory
to be removed

WORKING WITH DATABASE

PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.

Opening Database Connection

PHP provides mysql_connect function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success, or FALSE on failure.

Syntax

```
connection mysql_connect(server,user,password)
```

The parameters are

1. server

Optional – The host name running database server. If not specified then default value is localhost:3306.

2.user

Optional – The username accessing the database. If not specified then default is the name of the user that owns the server process.

3.password

Optional – The password of the user accessing the database. If not specified then default is an empty password.

Closing Database Connection

Its simplest function `mysql_close` PHP provides to close a database connection. This function takes connection resource returned by `mysql_connect` function. It returns `TRUE` on success or `FALSE` on failure.

Syntax

```
bool mysql_close ( resource $link_identifier );
```

If a resource is not specified then last opened database is closed. Example

```
<?php
```

```
$host = 'localhost:8080';
```

```
$user = 'guest';
```

```
$pass = 'guest123';
```

```
$conn = mysql_connect($host,$user,$pass);if(!
```

```
    $conn) {
```

```
    die('Could not connect: ' . mysql_error());
```

```
}
```

```
    echo 'Connected successfully';
```

```
mysql_close($conn);
```

```
?>
```

CREATING A DATABASE USING PHP

The `CREATE DATABASE` statement is used to create a database table in MySQL. We must add the `CREATE DATABASE` statement to the `mysql_query()` function to execute the command. Let us try with this simple example for creating a database.

```

<?php
$host = "localhost";
$user = "user02";
$pass = "jasmine";
$con=mysql_connect($host,$user,$pass);if
(!$con)

{
echo "Failed to connect to MySQL: " . mysql_connect_error();
}
$dos=mysql_query('create database smb');if($dos)
{
print("Database created successfully");
} else
e
{
print("error creating database:".mysql_error());
}
mysql_close($con);
?>

```

Output

Database created successfully

SELECTING MYSQL DATABASE USING PHP

PHP provides function `mysql_select_db` to select a database. It returns TRUE on success or FALSE on failure. The generic form of `mysql_select_db` is:

```
mysql_select_db( db_name, connection );
```

Where `db_name` is database name and it is mandatory, `connection` is the link identified and it is optional.

Here is an example that illustrates you how to select a database:

```

<?php
$host = "localhost";
$user = "user02";
$pass = "jasmine";
$con=mysql_connect($host,$user,$pass);if
(!$con)
{
echo "Failed to connect to MySQL: " . mysql_connect_error();
}

echo 'Connected successfully:<br>';
mysql_select_db('smb');          echo

```



```
'Database selected successfully';
mysql_close($con);
?>
```

Output:

Connected successfully Database selected
successfully

CREATING A TABLE USING PHP

Creating tables in the new database is very similar to creating the database. First create the SQL query to create the tables then execute the query using `mysql_query()` function. Following is an example to create a table:

```
<?php
$host = "localhost";
$user = "user02";
$pass = "jasmine";
$con=mysql_connect($host,$user,$pass);if
(!$con)
{
echo "Failed to connect to MySQL: " . mysql_connect_error();
}
echo 'connected successfully'<br>;
$sql='create table student(Rollno smallint(3) not null, Name varchar(25) not null, Age int(3)
not null, primary key(Rollno))';
mysql_select_db('smb');
$val=mysql_query($sql,$con);
if(!$val)
{
die('could not create table: ' . mysql_error());
}
echo 'Table student created successfully';
mysql_close($con);
?>
```

Output:

Connected successfully

Table student created successfully

INSERT DATA INTO MYSQL USING PHP

Data can be entered into MySQL tables by executing SQL INSERT statement through PHP function `mysql_query`. Given below is a simple example to insert a record into student table:

```
<?php
$host = "localhost";
$user = "user02";
$pass = "jasmine";
$con=mysql_connect($host,$user,$pass);if
(!$con)
```

```

{
echo "Failed to connect to MySQL: " . mysql_connect_error();
}
echo 'connected successfully'<br>;
$sql='insert into student(Rollno,Name,Age)values(123,"jenifer",20)';
mysql_select_db('smb'); $val=mysql_query($sql,$con); if(!$val)
{
die('could not enter data:' . mysql_error());
}
echo 'Entered data successfully'; mysql_close($con);
?>

```

GETTING DATA FROM MYSQL DATABASE

Data can be fetched from MySQL tables by executing SQL SELECT statement through PHP function `mysql_query`. The most frequently used option is to use function `mysql_fetch_array()`. This function returns row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows.

Given below is a simple example to fetch records from student table:

```

<?php
$host = "localhost";
$user = "user02";
$pass = "jasmine";
$con=mysql_connect($host,$user,$pass);if
(!$con)
{
echo "Failed to connect to MySQL: " . mysql_connect_error();
}
$sql='select Rollno,Name, Age from student';
mysql_select_db('smb'); $val=mysql_query($sql,$con);
echo "<table border='1-><tr><th>Rollno</th><th>Name</th><th>Age</th> </tr>"; if(!$val)
{
die('could not get data:' . mysql_error());
}
while($row=mysql_fetch_array($val,MYSQL_ASSOC))
{
echo "<tr>";
echo "<td>". $row['Rollno']. "</td>";
echo "<td>". $row['Name']. "</td>"; echo
"<td>". $row['Age']. "</td>";
}
echo "Fetched data successfully"; mysql_close($con);
?>

```

Output

Fetch data successfully

Rollno	Name	Age
123	jenifer	20
124	Jaysudha	22
125	Mary	25

UPDATING DATA INTO MYSQL DATABASE

The UPDATE statement is used to update existing records in a table. Data can be updated into MySQL tables by executing SQL UPDATE statement through PHP function mysql_query. A simple example to update records into student table is given as follows:

```
<?php
$host = "localhost";
$user = "user02";
$pass = "jasmine";
$con=mysql_connect($host,$user,$pass);if
(!$con)
{
echo "Failed to connect to MySQL: " . mysql_connect_error();
}
$sql='update student set Age=27 where Rollno=124-;
mysql_select_db('smb');
$val=mysql_query($sql,$con); if(!$val)
{
die('could not update data:' . mysql_error());
}
echo "updated data successfully"; mysql_close($con);
?>
```

Output:**updated data successfully**

Before the update operation, the student table looks like this:

mysql> select * from student;

Rollno	Name	Age
123	Jennifer	20
124	Jaysudha	23
125	Mary	25

3 rows in set (0.00 sec)

After the script given above is executed successfully, the student table looks like this:

mysql> select * from student;

Rollno	Name	Age
123	Jennifer	20
124	Jaysudha	27
125	Mary	25

3 rows in set (0.00 sec)

DELETING DATA IN A DATABASE

The DELETE FROM statement is used to delete records from a database table. Data can be deleted from MySQL tables by executing SQL DELETE statement through PHP function mysql_query. Given below is a simple example to delete records from a student table.

```
<?php
$host = "localhost";
$user = "user02";
$pass = "jasmine";
$con=mysql_connect($host,$user,$pass);if
(!$con)
{
echo "Failed to connect to MySQL: " . mysql_connect_error();
}
echo 'connected successfully'<br>;
$sql='delete from student where Rollno=123-;
mysql_select_db('smb');
$val=mysql_query($sql,$con); if(!$val)
{
die('could not delete data:' . mysql_error());
}
```

B. SASI KUMAR

```
echo 'Deleted data successfully';mysql_close($con);
```

?> Output:

Connected successfully Deleted data
successfully

DELETING A DATABASE

If a database is no longer required then it can be deleted forever. You can pass a SQL command to `mysql_query` to delete a database. Given below is a simple example to delete database from MySQL server.

```
<?php
$host = "localhost";
$user = "user02";
$pass = "jasmine";
$con=mysql_connect($host,$user,$pass);if (!$con)
{
echo "Failed to connect to MySQL: " . mysql_connect_error();
}
echo 'connected successfully'<br>;
$sql='drop database smb';
$val=mysql_query($sql,$con);if (!$val)
{
die('could not delete database smb:' . mysql_error());
}
echo 'Deleted database successfully';mysql_close($con); ?>
Output:
Connected successfully Deleted database successfully
```

UNIT-5

AJAX

AJAX = Asynchronous JavaScript and XML.

AJAX is a technique for creating fast and dynamic web pages.

AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.

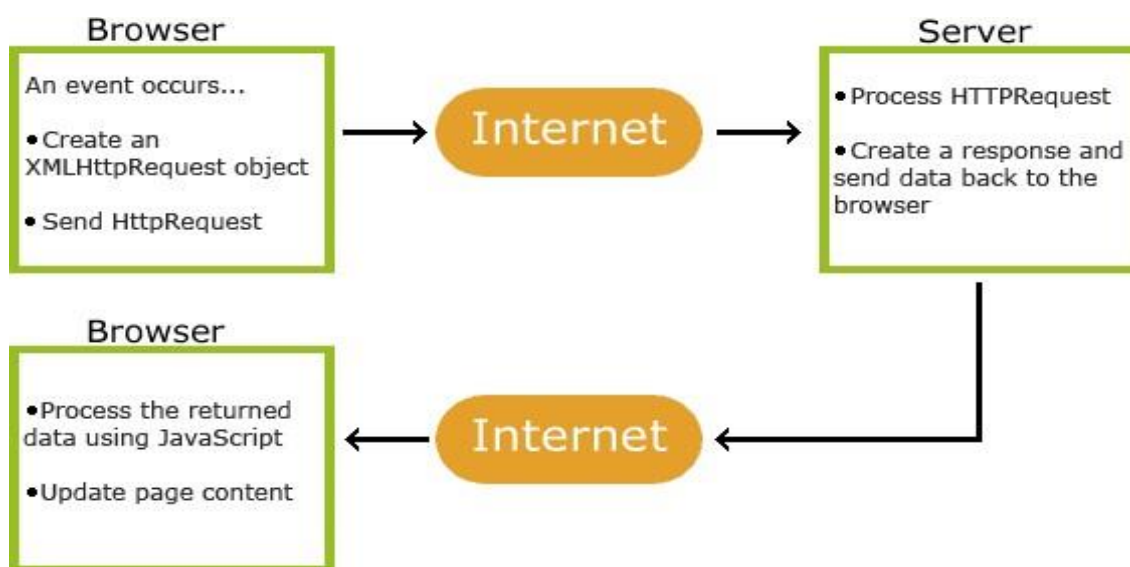
Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

ADVANTAGES OF AJAX

- Ajax is an efficient way for a web application to handle user interactions with a web page
- Ajax interactions are handled asynchronously in the background. As this happens, a user can continue working with the page.
- Ajax interactions are initiated by JavaScript code. When the Ajax interaction is complete, JavaScript updates the HTML source of the page. The changes are made immediately without requiring a page refresh.
- Ajax interactions can be used to do things such as validate form entries (while the user is entering them) using server-side logic, retrieve detailed data from the server, dynamically update data on a page, and submit partial forms from the page.

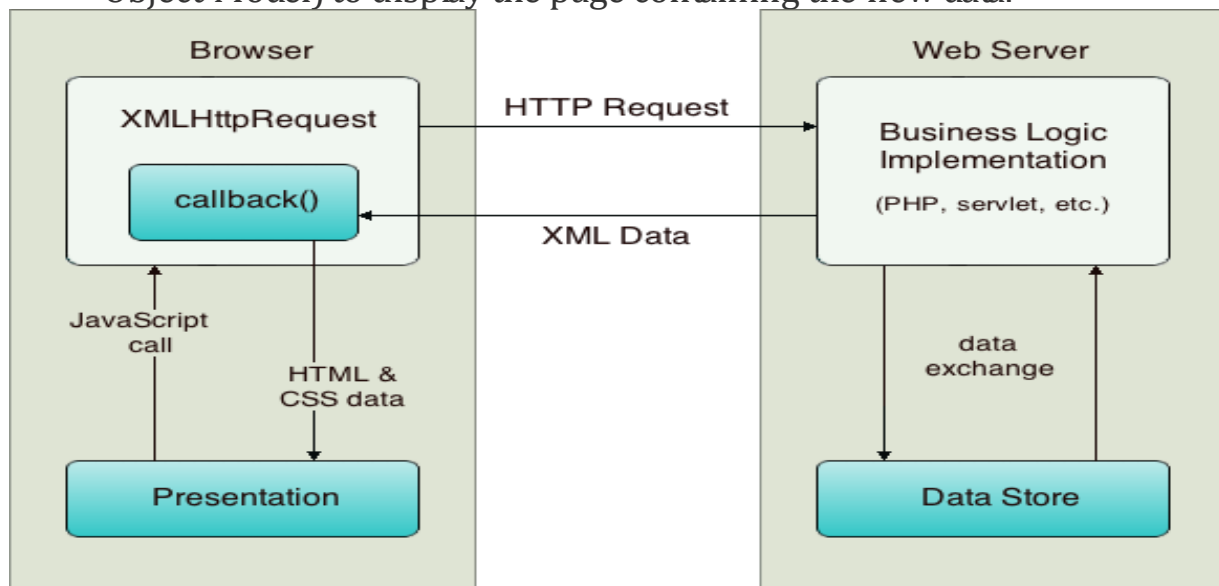
How AJAX Works?

- Implementing auto-completion in a search field is something that can be performed using Ajax.
- Ajax works by employing an XMLHttpRequest object to pass requests and responses asynchronously between the client and server.
- The following diagram illustrates the process flow of the communication that takes place between the client and server.



The process flow of the diagram can be described by the following steps:

1. The user triggers an event, for example by releasing a key when typing in a name. This results in a JavaScript call to a function that initializes an XMLHttpRequest object.
2. The XMLHttpRequest object is configured with a request parameter that includes the ID of the component that triggers the event, and any value that the user entered. The XMLHttpRequest object then makes an asynchronous request to the web server.
3. On the web server, an object such as a servlet or listener handles the request. Data is retrieved from the data store, and a response is prepared containing the data in the form of an XML document.
4. Finally, the XMLHttpRequest object receives the XML data using a callback function, processes it, and updates the HTML DOM (Document Object Model) to display the page containing the new data.



AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to exchange data asynchronously with a server)
- JavaScript/DOM (to display/interact with the information)
- CSS (to style the data)
- XML (often used as the format for transferring data)

B. SASI KUMAR

- AJAX applications are browser- and platform-independent!

AJAX PHP Example

The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:

```
<!DOCTYPE html>
<html>
<head> <script> function
showHint(str) {if
    (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";return;
    } else {
        var xmlhttp = new XMLHttpRequest(); xmlhttp.onreadystatechange = function() { if (xmlhttp.readyState == 4
        && xmlhttp.status == 200) {document.getElementById("txtHint").innerHTML =
xmlhttp.responseText;
        }
        }
        xmlhttp.open("GET", "gethint.php?q="+str, true); xmlhttp.send();
    }
}
</script>
</head>
<body>
<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

INPUT:

Start typing a name in the input field below:

First name:

Suggestions:

Code explanation:

B. SASI KUMAR

First, check if the input field is empty (`str.length == 0`). If it is, clear the content of the `txtHint` placeholder and exit the function.

However, if the input field is not empty, do the following:

- Create an `XMLHttpRequest` object
- Create the function to be executed when the server response is ready
- Send the request off to a PHP file (`gethint.php`) on the server
- Notice that `q` parameter is added to the url (`gethint.php?q="+str`)
- And the `str` variable holds the content of the input field

OUTPUT:

Start typing a name in the input field below:

First name:

Suggestions: Tove

The XMLHttpRequest Object

All modern browsers support 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.

Syntax for creating an `XMLHttpRequest` object:

```
variable = new XMLHttpRequest();
```

XMLHttpRequest Object Methods

Method	Description
<code>new XMLHttpRequest()</code>	Creates a new <code>XMLHttpRequest</code> object
<code>abort()</code>	Cancels the current request
<code>getAllResponseHeaders()</code>	Returns header information
<code>getResponseHeader()</code>	Returns specific header information
	Specifies the request
	<i>method</i> : the request type GET or POST

	<i>url</i> : the file location
<code>open(<i>method,url,async,user,psw</i>)</code>	<i>async</i> : true (asynchronous) or false (synchronous) <i>user</i> : optional user name <i>psw</i> : optional password
	Sends the request to the server
<code>send()</code>	Used for GET requests
	Sends the request to the server. <code>send(<i>string</i>)</code>
	Used for POST requests
	Adds a label/value pair to the header to be
<code>setRequestHeader()</code>	sent

XMLHttpRequest Object Properties

Property	Description
<code>onreadystatechange</code>	Defines a function to be called when the <code>readyState</code> property changes
<code>readyState</code>	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
<code>responseText</code>	Returns the response data as a string
<code>responseXML</code>	Returns the response data as XML data
<code>status</code>	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
<code>statusText</code>	Returns the status-text (e.g. "OK" or "Not Found")

PHP FTP- FILE TRANSFER PROTOCOL

Introduction

The FTP functions give client access to file servers through the **File Transfer Protocol (FTP)**.

- The FTP functions are used to open, login and close connections, as well as upload, download, rename, delete, and get information on files from file servers.
- Not all of the FTP functions will work with every server or return the same results.
- The FTP functions became available with PHP 3.
- If you only wish to read from or write to a file on an FTP server, consider using the **ftp:// wrapper** with the Filesystem functions which provide a simpler and more intuitive interface.

Installation

For these functions to work, you have to compile PHP with --enable-ftp. The Windows version of PHP has built-in support for this extension.

PHP 5 FTP Functions

Function	Description
ftp_alloc()	Allocates space for a file to be uploaded to the FTP server
ftp_cdup()	Changes to the parent directory on the FTP server
ftp_chdir()	Changes the current directory on the FTP server
ftp_chmod()	Sets permissions on a file via FTP

<code>ftp_close()</code>	Closes an FTP connection
<code>ftp_connect()</code>	Opens an FTP connection
<code>ftp_delete()</code>	Deletes a file on the FTP server
<code>ftp_exec()</code>	Executes a command on the FTP server
<code>ftp_fget()</code>	Downloads a file from the FTP server and saves it into an open local file
<code>ftp_fput()</code>	Uploads from an open file and saves it to a file on the FTP server
<code>ftp_get_option())</code>	Returns runtime options of the FTP connection
<code>ftp_get()</code>	Downloads a file from the FTP server
<code>ftp_login()</code>	Logs in to the FTP connection
<code>ftp_mdtm()</code>	Returns the last modified time of a specified file
<code>ftp_mkdir()</code>	Creates a new directory on the FTP server
<code>ftp_nb_continue()</code>	Continues retrieving/sending a file (non-blocking)
<code>ftp_nb_fget()</code>	Downloads a file from the FTP server and saves it into an open file (non-
<code>ftp_nb_fput()</code>	Uploads from an open file and saves it to a file on the FTP server (non-bl
<code>ftp_nb_get()</code>	Downloads a file from the FTP server (non-blocking)
<code>ftp_nb_put()</code>	Uploads a file to the FTP server (non-blocking)

<u>ftp_nlist()</u>	Returns a list of files in the specified directory on the FTP server
<u>ftp_pasv()</u>	Turns passive mode on or off
<u>ftp_put()</u>	Uploads a file to the FTP server
<u>ftp_pwd()</u>	Returns the current directory name
<u>ftp_quit()</u>	An alias of <u>ftp_close()</u>

<code>ftp_raw()</code>	Sends a raw command to the FTP server
<code>ftp_rawlist()</code>	Returns a list of files with file information from a specified directory
<code>ftp_rename()</code>	Renames a file or directory on the FTP server
<code>ftp_rmdir()</code>	Deletes an empty directory on the FTP server
<code>ftp_set_option()</code>	Sets runtime options for the FTP connection
<code>ftp_site()</code>	Sends an FTP SITE command to the FTP server
<code>ftp_size()</code>	Returns the size of the specified file
<code>ftp_ssl_connect()</code>	Opens a secure SSL-FTP connection
<code>ftp_systype()</code>	Returns the system type identifier of the FTP server

PHP-FileUploading

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 field, 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 field 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.

DRAWING IMAGES ON SERVER

The Graphic Functions available in php are listed below:

- [gd_info](#) — Retrieve information about the currently installed GD library
- [getimagesize](#) — Get the size of an image
- [getimagesizefromstring](#) — Get the size of an image from a string
- [image_type_to_extension](#) — Get file extension for image type
- [image_type_to_mime_type](#) — Get Mime-Type for image-type returned by getimagesize, exif_read_data, exif_thumbnail, exif_imagetype
- [image2wbmp](#) — Output image to browser or file
- [imagearc](#) — Draws an arc
- [imagebmp](#) — Output a BMP image to browser or file
- [imagechar](#) — Draw a character horizontally
- [imagecharup](#) — Draw a character vertically
- [imagecolorallocate](#) — Allocate a color for an image
- [imagecolorexactalpha](#) — Get the index of the specified color + alpha
- [imagecolormatch](#) — Makes the colors of the palette version of an image more closely match the true color version
- [imagecolorresolve](#) — Get the index of the specified color or its closest possible alternative
- [imagecolorsforindex](#) — Get the colors for an index
- [imagecolorstotal](#) — Find out the number of colors in an image's palette
- [imagecolortransparent](#) — Define a color as transparent
- [imageconvolution](#) — Apply a 3x3 convolution matrix, using coefficient and offset
- [imagecopy](#) — Copy part of an image
- [imagecopymerge](#) — Copy and merge part of an image
- [imagecopymergegray](#) — Copy and merge part of an image with grayscale
- [imagecopyresampled](#) — Copy and resize part of an image with resampling
- [imagecopyresized](#) — Copy and resize part of an image
- [imagecreate](#) — Create a new palette based image
- [imagecreatefrombmp](#) — Create a new image from file or URL
- [imagecreatefromgd2](#) — Create a new image from GD2 file or URL
- [imagecreatefromgd2part](#) — Create a new image from a given part of GD2 file or URL

- [imagecrop](#) — Crop an image to the given rectangle
- [imagecropauto](#) — Crop an image automatically using one of the available modes
- [imagedashedline](#) — Draw a dashed line
- [imagedestroy](#) — Destroy an image
- [imageellipse](#) — Draw an ellipse
- [imagefill](#) — Flood fill
- [imagefilledarc](#) — Draw a partial arc and fill it
- [imagefilledellipse](#) — Draw a filled ellipse
- [imagefilledpolygon](#) — Draw a filled polygon
- [imagefilledrectangle](#) — Draw a filled rectangle
- [imagefilltoborder](#) — Flood fill to specific color
- [imagefilter](#) — Applies a filter to an image
- [imageflip](#) — Flips an image using a given mode
- [imagefontheight](#) — Get font height
- [imagefontwidth](#) — Get font width
- [imageftbbox](#) — Give the bounding box of a text using fonts via freetype2
- [imagefttext](#) — Write text to the image using fonts using FreeType 2
- [imagegammacorrect](#) — Apply a gamma correction to a GD image
- [imagegd2](#) — Output GD2 image to browser or file
- [imagegd](#) — Output GD image to browser or file
- [imagegrabwindow](#) — Captures a window
- [imageinterlace](#) — Enable or disable interlace
- [imagejpeg](#) — Output image to browser or file
- [imageline](#) — Draw a line
- [imageloadfont](#) — Load a new font
- [imageopenpolygon](#) — Draws an open polygon
- [imagepalettecopy](#) — Copy the palette from one image to another
- [imagepalettetotruecolor](#) — Converts a palette based image to true color
- [imagepng](#) — Output a PNG image to either the browser or a file
- [imagepolygon](#) — Draws a polygon
- [imagepsbbox](#) — Give the bounding box of a text rectangle using PostScript Type1 fonts
- [imagepsencodefont](#) — Change the character encoding vector of a font
- [imagepsextendfont](#) — Extend or condense a font
- [imagepsfreefont](#) — Free memory used by a PostScript Type 1 font □
- [imagepsloadfont](#) — Load a PostScript Type 1 font from

Creating an Image

- The **imagecreate()** function is an inbuilt function in PHP which is used to create a new image.
- This function returns the blank image of given size. In general **imagecreatetruecolor()** function is used instead of **imagecreate()** function because **imagecreatetruecolor()** function creates high quality images.

Syntax:

`imagecreate($width, $height)`

- **\$width:** It is mandatory parameter which is used to specify the imagewidth.
- **\$height:** It is mandatory parameter which is used to specify the imageheight.

Example . A black square on a white background (black.php)

```
<?php
$im = ImageCreate(200,200);
$white = ImageColorAllocate($im,0xFF,0xFF,0xFF);
$black = ImageColorAllocate($im,0x00,0x00,0x00);
ImageFilledRectangle($im,50,50,150,150,$black);
header('Content-Type: image/png'); ImagePNG($im);
?>
```

ImageFilledRectangle(), in which you specify the dimensions of the rectangle by passing the coordinates of the top-left and bottom-right corners:

`ImageFilledRectangle(image, tlx, tly, brx, bry, color);`

- The next step is to send a Content-Type header to the browser with the appropriate content type for the kind of image being created.

Once that is done, we call the appropriate output function.

The `ImageJPEG()`, `ImagePNG()`, and `ImageWBMP()` functions create JPEG, PNG, and WBMP files from the image, respectively:

```
ImageJPEG(image [, filename [, quality ]]); ImagePNG(image [, filename]); ImageWBMP(image [, filename]);
```

Reading an Existing File

If you want to start with an existing image and then modify it, use either `ImageCreateFromJPEG()` or `ImageCreateFromPNG()`:

```
$image = ImageCreateFromJPEG(filename);  
$image = ImageCreateFromPNG(filename);
```

Basic Drawing Functions

GD has functions for drawing Basic points, lines, arcs, rectangles, and polygons.

- The most basic function is `ImageSetPixel()`, which sets the color of a specified pixel:

`ImageSetPixel(image, x, y, color);`

- There are two functions for drawing lines, `ImageLine()` and `ImageDashedLine()`:

`ImageLine(image, start_x, start_y, end_x, end_y, color); ImageDashedLine(image, start_x, start_y, end_x, end_y, color);`

- There are two functions for drawing rectangles, one that simply draws the outline and one that fills the rectangle with the specified color:

`ImageRectangle(image, x1, y1, x2, y2, color); ImageFilledRectangle(image, x1, y1, x2, y2, color);`

- Specify the location and size of the rectangle by passing the coordinates of the top-left and bottom-right corners.
- You can draw arbitrary polygons with the `ImagePolygon()` and `ImageFilledPolygon()` functions:

`ImagePolygon(image, points, number, color); ImageFilledPolygon(image, points, number, color);`

Both functions take an array of points. This array has two integers (the x and y coordinates) for each vertex on the polygon.

The number argument is the number of vertices in the array (typically `count($points)/2`).

The `ImageArc()` function draws an arc (a portion of an ellipse):

`ImageArc(image, center_x, center_y, width, height, start, end, color);`

- The `ImageFill()` function performs a flood fill, changing the color of the pixels starting at the given location.
- Any change in pixel color marks the limits of the fill.
- The `ImageFillToBorder()` function lets you pass the particular color of the limits of the fill:

BSC CS

```
ImageFill(image, x, y, color);
```

```
ImageFillToBorder(image, x, y, border_color, color);
```

To Draw Rectangle

```
<?php
create_image(); print "<img
src=image.png?".date("U").">";function
create_image()
{
    $im = @imagecreate(200, 200) or die("Cannot Initialize new GD imagestream");
    $background_color = imagecolorallocate($im, 255, 255, 0); // yellow

    $red = imagecolorallocate($im, 255, 0, 0); // red $blue =
    imagecolorallocate($im, 0, 0, 255); // blue
    imagerectangle($im, 5, 10, 195, 50, $red); imagefilledrectangle($im, 5, 100, 195,
    140, $blue);imagepng($im,"image.png"); imagedestroy($im);
}
?>
```



To Draw Ellipse

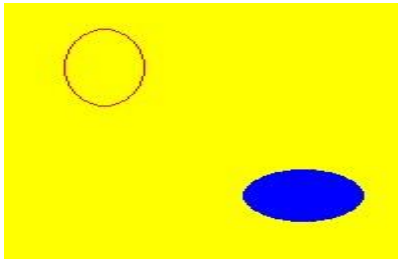
```
<?php create_image();
print "<img src=image.png?".date("U").">";
function create_image()
{
    $im = @imagecreate(200, 200) or die("Cannot Initialize new GD image
stream");
    $background_color= imagecolorallocate($im, 255, 255, 0); // yellow

    $red = imagecolorallocate($im, 255, 0, 0); // red $blue =
    imagecolorallocate($im, 0, 0, 255); // blue imageellipse($im, 50, 50, 40,
    60, $red); imagefilledellipse($im, 150, 150, 60, 40, $blue);
    imagepng($im,"image.png"); imagedestroy($im);
}
```

BSC CS

15

?>



The following script is an example for drawing human face using imagearc()function
<?php

```
$img = imagecreatetruecolor(200, 200);
```

```
$white = imagecolorallocate($img, 255, 255, 255);
```

```
$red = imagecolorallocate($img, 255, 0, 0);
```

```
$green = imagecolorallocate($img, 0, 255, 0);
```

```
$blue = imagecolorallocate($img, 0, 0, 255);
```

```
imagearc($img, 100, 100, 200, 200, 0, 360, $white);
```

```
imagearc($img, 100, 100, 150, 150, 25, 155, $red);
```

```
imagearc($img, 60, 75, 50, 50, 0, 360, $green); imageline($img,100,80,100,150,$green);
```

```
imagearc($img,
```

```
140, 75, 50, 50, 0, 360, $blue);header("Content-type:
```

```
image/png"); imagepng($img);
```

```
imagedestroy($img);
```

?>

OUTPUT:

BSC CS

