UNIT-1

Introduction to PHP: Declaring variables, data types, arrays, strings, operators, expressions, control structures, functions, Reading data from web from controls like text boxes, radio buttons, lists etc., Handling File Uploads, Connecting to database (MySQL as reference), executing simple queries, handling results, Handling sessions and cookies.

File Handling in PHP: File operations like opening, closing, reading, writing, appending, deleting etc. on text and binary files, listing directories.

Introduction to PHP

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

•PHP is a recursive acronym for "PHP: Hypertext Preprocessor".

•PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire ecommerce sites. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

•PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

•PHP Syntax is C-Like.

Introduction to PHP

Common Uses of PHP:

PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.

The other uses of PHP are:

•PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.

•You add, delete, modify elements within your database thru PHP.

•Access cookies variables and set cookies.

•Using PHP, you can restrict users to access some pages of your website.

•It can encrypt data.

Introduction to PHP

Characteristics of PHP:

Five important characteristics make PHP's practical nature possible:

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

Introduction to PHP

A PHP script is executed on the server.

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>

```
<?php
// PHP code goes here
?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- "echo" is the built-in PHP Function to output the text on a web page.

Introduction to PHP

```
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

Note:

PHP statements end with a semicolon (;).

Introduction to PHP

Comments in PHP

A comment in PHP code is a line that is not read/executed as part of the program.

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>
```

Introduction to PHP

PHP Case Sensitivity

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

However; all variable names are case-sensitive.

Introduction to PHP

```
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

Hello World!
Hello World!
Hello World!

```
<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>
```

My car is red
My house is
My boat is

Declaring Variables

Variables are "containers" for storing information.

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

Example:

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

Declaring Variables

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

Rules for PHP variables:

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

Declaring Variables

Output Variables

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

```
<?php
$txt = “PHP";
echo “Welcome to $txt!";
?>
```

```
<?php
$txt = “PHP";
echo “Welcome to " . $txt . "!";
?>
```

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

Declaring Variables

PHP is a Loosely Typed Language:

•We did not have to tell PHP which data type the variable is.

•PHP automatically converts the variable to the correct data type, depending on its value.

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

Declaring Variables

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

Declaring Variables

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

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

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

Declaring Variables

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();

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

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

Declaring Variables

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):

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

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

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

15

Declaring Variables

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.

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

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

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

15

Declaring Variables

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();
?>
```

```
0
1
2
```

Data types

PHP Data Types:

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

PHP supports the following data types:

1. String
2. Integer
3. Float (floating point numbers - also called double)
4. Boolean
5. Array
6. Object
7. NULL

Data types

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;
?>
```

Data types

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

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

•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:

Data types

PHP Float

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

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

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

Data types

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

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

Data types

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);
?>

OUTPUT:

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

Data types

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 = "VW";
    }
}

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

// show object properties
echo $herbie->model;                    //Output: VM
```

Data types

PHP NULL Value

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

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

•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);          //Output: NULL
?>
```

Arrays

An array is a collection of similar type of elements. In PHP we can have the elements of mixed type together. Each element has two parts key and value.

The key represents the index at which the value of the element can be stored. The keys are positive integers that are in ascending order.

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

```
array();
```

Arrays

Array Creation:

There two ways to create an array in PHP.

1. $mylist = array(10,20,30,40,50);
2. $mylist[0] = 10; //Assign value directly to the array

$mylist = array(); //creates an empty array

$mylist = array("Sowmya" => "Delhi","Swetha" => 89.93); //mixed type of elements

Arrays

Functions for Dealing with Arrays:

- **unset** function is used to remove particular element from the array.

```
<?php
$mylist=array(10,20,30,40,50);
unset($mylist[1]);
for($i=0;$i<=4;$i++)
{
        print $mylist[$i];
        print " ";
}
?>
```

Arrays

Functions for Dealing with Arrays:

- **array_keys** and **array_values** are used to return the array keys and the values at corresponding key.

```
<?php
        $a=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander");
        print_r(array_keys($a));
        $a=array("Name"=>"Peter","Age"=>"41","Country"=>"USA");
        print_r(array_values($a));
?>
```

Arrays

Functions for Dealing with Arrays:

- **implode** and **explode** functions are used to break the word into strings or vice versa.

```
<?php
        $arr = array('Hello','World!','Beautiful','Day!');
        echo implode(" ",$arr);
        $str = "Hello world. It's a beautiful day.";
        print_r (explode(" ",$str));
?>
```

```
Hello World! Beautiful Day!
Array ( [0] => Hello [1] => world. [2] => It's [3] => a [4] => beautiful [5] => day. )
```

Arrays

Sequential Access to Array Elements

The array element reference start at the first element and every array maintains an internal pointer using which the next element can be easily accessible.

the pointer **current** is used to point to the current element in the array. Using **next** function the next subsequent element can be accessed.

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");

echo current($people) . "<br>";
echo next($people);
?>
```

```
Peter
Joe
```

Arrays

each and **foreach**

Using each function we can iterate through the array elements.

```
<?php
        $people = array("Peter", "Joe", "Glenn", "Cleveland");
        print_r (each($people));
?>

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

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

Arrays

Sorting Arrays

sort() - sort arrays in ascending order

```
<?php
        $numbers = array(4, 6, 2, 22, 11);
        sort($numbers);

        $arrlength = count($numbers);
        for($x = 0; $x < $arrlength; $x++) {
                echo $numbers[$x];
                echo "<br>";
        }
?>
```

1. rsort() - sort arrays in descending order
2. asort() - sort arrays in ascending order, according to the value
3. ksort() - sort arrays in ascending order, according to the key
4. arsort() - sort arrays in descending order, according to the value
5. krsort() - sort arrays in descending order, according to the key

Strings

A string is a sequence of characters .

PHP String Functions

Get The Length of a String

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

```
<?php

echo strlen("Hello world!"); // outputs 12

?>
```

Strings

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
?>
```

Reverse a String

The PHP strrev() function reverses a string:

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

Strings

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.

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

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

Strings

Replace Text Within a String

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

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

Strings

String Compare

The PHP strcmp(str1,str2) function compares two strings.

```
<?php
echo strcmp(“PHP“,”PHP”); // outputs 0
?>
```

String to Lowercase

The PHP strtolower(str1) function converts characters to lower case.

```
<?php
echo strtolower(“PHP”) // outputs php
?>
```

Strings

String to Uppercase

The PHP strtoupper(str1) function converts characters to upper case.

```
<?php
echo strtoupper(“php”) // outputs PHP
?>
```

The PHP trim(str1) function eliminates the white space from both ends of the string.

```
<?php
echo trim(“  php  ”) // outputs php
?>
```

Strings

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

•A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
•A valid constant name starts with a letter or underscore (no $ sign before the constant name).

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

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

Syntax:

define(name, value, case-insensitive)

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

Strings

```
<?php

define("GREETING", "Welcome to PHP!");

echo GREETING;

?>
```

```
<?php

define("GREETING", "Welcome to PHP!", true);

echo greeting;

?>
```

Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Increment/Decrement operators
5. Logical operators
6. String operators
7. Array operators

Operators

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

Operators

Assignment Operators

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

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

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Operators

Comparison Operators

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

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

Operators

Comparison Operators

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

Operators

Increment / Decrement Operators

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

Operator	Name	Description
++$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

Operators

Logical Operators

The PHP logical operators are used to combine conditional statements.

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

Operators

String Operators

PHP has two operators that are specially designed for strings.

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

Operators

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 the 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

Expressions

An expression is a bit of PHP that can be evaluated to produce a value. The simplest expressions are literal values and variables. A literal value evaluates to itself, while a variable evaluates to the value stored in the variable. More complex expressions can be formed using simple expressions and operators.

Control Structures

Conditional Statements

In PHP we have the following conditional statements:

•if statement - executes some code if one condition is true

•if...else statement - executes some code if a condition is true and another code if that condition is false

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

•switch statement - selects one of many blocks of code to be executed

Control Structures

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;
}
```

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

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

Control Structures

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;
}
```

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

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

Control Structures

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;
}
```

```php
<?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!";
}
?>
```

Control Structures

switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

Syntax

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

Control Structures

```php
<?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!";
}
?>
```

Control Structures

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:

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

Control Structures

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;
}

```
<?php
$x = 1;

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

Control Structures

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);
```

```
<?php
$x = 1;

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

Control Structures

for Loop

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

Syntax

for (*init counter; test counter; increment counter*) {

    code to be executed;

}

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

Parameters:

init counter: Initialize the loop counter value

test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment counter: Increases the loop counter value

Control Structures

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;
}
```

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

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

Functions

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

User Defined Functions

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

1. A function is a block of statements that can be used repeatedly in a program.
2. A function will not execute immediately when a page loads.
3. A function will be executed by a call to the function.

Functions

Create a User Defined Function in PHP

A user defined function declaration starts with the word "function":

Syntax

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

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

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

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

Functions

Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.
Arguments are specified after the function name, inside the parentheses.

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

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

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

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

Functions

Default Argument Value

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

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

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

Functions

Functions - Returning values

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

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

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

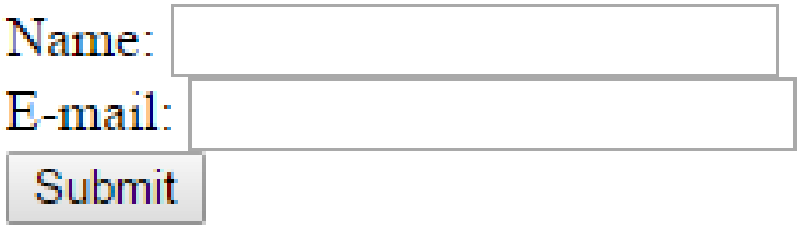
Reading data from web form controls
(like textboxes, radio buttons, lists etc..)

A Simple HTML Form

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```



Reading data from web form controls
(like textboxes, radio buttons, lists etc..)

A Simple HTML Form

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 POST** method.

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

Reading data from web form controls
(like textboxes, radio buttons, lists etc..)

A Simple HTML Form

The same result could also be achieved using the **HTTP GET** method:

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

Reading data from web form controls
(like textboxes, radio buttons, lists etc..)

GET vs. POST

Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

Reading data from web form controls
(like textboxes, radio buttons, lists etc..)

GET

Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send.

1. The GET method is restricted to send up to 1024 characters only.
2. Never use GET method if you have password or other sensitive information to be sent to the server.
3. GET can't be used to send binary data, like images or word documents, to the server.
4. the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

Reading data from web form controls
(like textboxes, radio buttons, lists etc..)

POST

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

1. The POST method does not have any restriction on data size to be sent.

2. The POST method can be used to send ASCII as well as binary data.

3. The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.

4. The PHP provides $_POST associative array to access all the sent information using POST method.

Handling File Uploads

PHP helps in uploading a file. For uploading file configure php.ini

open your php.ini file and search for following keys and make some changes as below :

1 . file_uploads = On

2 . upload_max_filesize = 50M

upload_max_filesize helps you to maximize file uploading size, by default it is 2, you can maximize as your need.

Handling File Uploads

```
<?php
if(isset($_POST['btn-upload']))
{
     $pic = rand(1000,100000)."-".$_FILES['pic']['name'];
    $pic_loc = $_FILES['pic']['tmp_name'];
     $folder="uploaded_files/";
     if(move_uploaded_file($pic_loc,$folder.$pic))
     {
        ?><script>alert('successfully
uploaded');</script><?php
     }
     else
     {
        ?><script>alert('error while uploading
file');</script><?php
     }
}
?>
```

```
<html>
<head>
<title>File Uploading With PHP and
MySql</title>
</head>
<body>
<form action="" method="post"
enctype="multipart/form-data">
<input type="file" name="pic" />
<button type="submit" name="btn-
upload">upload</button>
</form>
</body>
</html>
```

Handling File Uploads

1 . enctype="multipart/form-data" : it's specify content-type to be uploaded.

2. $folder : Directory where files are uploaded.

3. move_uploaded_files : php function that moves selected file to the specified folder.

4 . rand() : this is the awesome function that enables you to upload same files multiple times.

5. $_FILES['pic']['name'] represents the original name of the file on the client machine.

6. $_FILES['pic'][‘tmp_name'] represents the temporary filename of the file in which the uploaded file was stored on the server.

Connecting to Database

MySQL is the most popular database system used with PHP.

What is MySQL?

1. A database system used on the web
2. A database system that runs on a server
3. It is ideal for both small and large applications
4. It is very fast, reliable, and easy to use
5. It uses standard SQL
6. It compiles on a number of platforms
7. It is free to download and use
8. It is developed, distributed, and supported by Oracle Corporation
9. MySQL is named after co-founder Monty Widenius's daughter: My

Connecting to Database

MYSQL is a open source database product and can be downloaded from the web site http://dev.mysql.com/downloads/mysql.

MYSQL is a kind of database in which the records are stored in an entity called tables. We can query a database to retrieve particular information. Query is a request or a question for the database.

Benefits of using PHP and MYSQL

1. PHP is a server side scripting language and it has an ability to create dynamic pages with customized features. Using PHP-MySQL user friendly and interactive web sites can be created.
2. Both PHP and MySQL are open-source technologies that work hand-in-hand to create rich internet applications.
3. Due to availability of these technologies as free of cost, the cost effective web solutions can be created.
4. PHP-MySQL are stable technologies and have cross platform compatibility. Hence the web applications developed using them become portable.

Connecting to Database

5. Since HTML can be embedded within the PHP, there is no need to write separate code for web-scripting.

6. The most popular web sites being developed using PHP and MySQL technologies are-

a) Facebook b) Wordpress c)Wikipedia 4. Yahoo

HANDLING MySQL QUERIES: (mysql - - user = root)

1. Connecting database:

create database database_name;

2. Displaying all databases:

show databases;

Connecting to Database

3. Selecting particular database:

use database_name;

4. Creating table:

create table table_name(id INT(4),name VARCHAR(20));

5. Displaying a table:

show tables;

6. Displaying the table fields:

describe table_name;

7.Insert values into the table:

insert into table_name values(1,'VIKRAM');

Connecting to Database

8. displaying the contents of the table

select * from table_name;

9. Updating the record:

update table_name set name='VIRAT' where id=1;

10. Deleting record

delete from tabl_name where id=1;

11. Deleting table:

drop table_name;

Connecting to Database

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

1. MySQLi extension (the "i" stands for improved)

2. PDO (PHP Data Objects)

Connecting to Server

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

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

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

Connecting to Database

Creating Database

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

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

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

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

Connecting to Database

Creating Table

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

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

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

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
```

Connecting to Database

Inserting Data

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

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

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

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

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

Connecting to Database

Selecting Data

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

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

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
```

Connecting to Database

Delete Data

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

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

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

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

Connecting to Database

Update Data

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

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

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

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

Handling Sessions

- When you open some application, use it for some time and then close it. This entire scenario is named as **session**. Web pages hold only temporary data, unless you specifically store your data on the server. One way of storing data on the server is to use **sessions**.
- Sometimes the information about the session is required by the server. This information can be collected during the session. This process is called **session tracking**.
- There exists a session array which often stores the unique session ID for the session.
- PHP keeps track of session by using a function called session_start(). Due to the call to this function the session ID is created and recorder.
- To store data in the session, use the $_SESSION array. We can access the data again by using $_SESSION again.

Handling Sessions

//Storing data in sessions

```
<?php
	// Start the session
	session_start();
?>

<html>
	<body>

	<?php
		// Set session variables
		$_SESSION["favcolor"] = "green";
		$_SESSION["favanimal"] = "cat";
		echo "Session variables are set.";
	?>

	</body>
</html>
```

Handling Sessions

//Retrieving data from sessions

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

<?php
        // Echo session variables that were set on previous page
        echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
        echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Handling Sessions

```
//Hit counter using sessions
<?php
   session_start();

   if( isset( $_SESSION['counter'] ) ) {
      $_SESSION['counter'] += 1;
   }else {
      $_SESSION['counter'] = 1;
   }

   $msg = "You have visited this page ".  $_SESSION['counter'];
   $msg .= "in this session.";
?>

<html>

   <head>
      <title>Setting up a PHP session</title>
   </head>

   <body>
      <?php  echo ( $msg ); ?>
   </body>

</html>
```

Handling Cookies

- Cookies are text files stored on the client computer and they are kept of use tracking purpose.
- Cookies are used to identify the users.
- A cookie consists of a name and textual value.
- In every HTTP communication between browser and server a header is included. The header part of http contains the cookies.
- PHP can be used create and retrieve the cookies.
- **Syntax:**

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

name-name of the cookie

value-value of the cookie stored on the clients computer.

expire-The time the cookie expires.

path-specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.

domain-the domain for which the cookie is available.

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

Handling Cookies

```
<?php
  setcookie("name", "John Watkin", time()+60*60*24*30);
  setcookie("age", "36", time()+60*60*24*30);
?>
<html>

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

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

</html>
```

Handling Cookies

Accessing Cookies:

```
<html>

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

  <body>

    <?php
      echo $_COOKIE["name"]. "<br />";

      echo $_COOKIE["age"] . "<br />";

    ?>

  </body>
</html>
```

Handling Cookies

Deleting Cookies:

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

```
<?php
   setcookie( "name", "", time()-3600);
   setcookie( "age", "", time()-3600);
?>
<html>

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

   <body>
      <?php echo "Deleted Cookies" ?>
      <a href="cookie2.php">cookie2.php</a>
   </body>

</html>
```

File Handling in PHP

File handling is an important part of any web application. You often need to open and process a file for different tasks. PHP has several functions for creating, reading, uploading, and editing files.

Opening and Closing File

The first step in file handling is opening of the file.
It takes two parameters – i) name of the file ii)file mode

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

File Handling in PHP

Example: "webdictionary.txt“
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));  //maximum no of bytes to be read
fclose($myfile);
?>
```

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

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

File Handling in PHP

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.

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

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

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

File Handling in PHP

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

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

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

Listing Directories

PHP provides a support for directory functions that allow us to retrieve information about directories and its contents. Various directory functions are as given below-

Function	Description
chdir()	Changes the current directory
chroot()	Changes the root directory
closedir()	Closes a directory handle
dir()	Returns an instance of the Directory class
getcwd()	returns the current working directory
opendir()	Opens a directory handle
readdir()	Returns an entry from a directory handle
rewinddir()	Returns a directory handle
scandir()	Returns an array of files and directories of a specified directory