# PHP(PHP; Hypertext Preprocessor)

Unit 4

Web technologies black book

(chapter 3)

# PHP advantages over other languages

- ASP-active server pages
- Cold fusion
- Perl-Practical extraction and report language
- JSP-java server pages

# PHP advantages over other languages

- ASP: Server side scripting language by Microsoft IIS  unlike php.Slow,less secureand stable unlike php which works with apache so is fast,reliable and secure
- Cold fusion : Runs on Win 32,solaris,linux.UX so targets non developers unlike php. PHP is fast,reliable and efficient.
- Perl: perl was meant for complex task unlike Php which was designed for scripting for the web .Easy to learn compared to Perl .No prior knowledge of programming needed whereas c and shell scripting knowledge needed for perl.Php easy integrates with html.
- JSP:needsjvm .php performs 5 times more than jsp.
- Php V2,3,4,5,6

# PHP:Features

- Access control :built-in web based config screen
- File upload support using MIME
- Http based authentication control
- Supports Variables, numeric arrays , associative arrays- can be send across web pages
- Supports regular expressions
- Conditional statements /loops
- Safe mode support
- Access logging
- Open source
- Supports third party app like databases

# XHTML

- Combination of xml and html
- Designed for mobile phones and wireless devices

# XHTML

- XHTML is html doc following rules like html,head,body tags have to be present,all attribute values should be in quotes,all tags opened must be closed and all tags written in lowercase only
- Php code embedded in xhtml page can be written in 3 ways:

1. Xml style
2. Short
3. script

# XHTML

1.Xml style:

<?php

?>

2.Short

<?

?>

3.Script:<script language=“php”> </script >

# Escape character

\”: print next character as double quote

\’: print next character as double quote

\n: prints newline character

\t: prints tab character

\r: prints carriage return

\$:prints dollar symbol

\\: prints a \

# PHP Servers

- WAMP server: “Windows, Apache, MySQL, and

PHP,”

- **XAMPP server: X (cross plaftorm), Apache, MySQL, PHP, Perl**

# Working with variables and constants(chapter 4)

1.Variables:

Naming rules:

- Variable names must start with a letter of the alphabet or the _ (underscore) character.
- Variable names can contain only the characters a-z, A-Z, 0-9, and _ (underscore).
- Variable names may not contain spaces. If a variable must comprise more than one word, the words should be separated with the _ (underscore) character ($user_name).
- Variable names are case-sensitive. The variable $High_Score is not the same as the variable $high_score

# Working with variables and constants(chapter 4)

```
<?php
   $txt = ’Hello world!’;
   $x = 5;
$y = 10.5;
$x = $y + 1;
echo $txt;
echo $x;
echo $y;
   ?>
```

# Working with variables and constants(chapter 4)

```
<?php
   $txt = ’Hello world!’;
   $alt_txt = &$txt;
$alt_txt = ’hi’;
echo $txt,$alt_txt;//both say hi
echo “$txt is txt value and $alt_txt is alt_txt
   value”;
unset($txt);//destroys variable
   ?>
```

# Working with variables and constants(chapter 4)

Create variable at Runtime:

```
<?php
   $txt = ’kogent’;
   ${$txt} =‘Hello’;
echo $kogent;//Hello
?>
```

# Working with variables and constants(chapter 4)

2.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).

```
<?php
   define("GREETING", "Welcome to W3Schools.com!");
   echo GREETING; //constant with a case sensitive name
   define(“USER_NAME”,hello,true);
    //case insensitive by default its false
   Echo user_name;//hello
?>
```

# Working with variables and constants(chapter 4)

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

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

function myTest() {
    echo GREETING;
}
 myTest();
?>
```

# Working with variables and constants(chapter 4)

3.Data types:

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

# Working with variables and constants(chapter 4)

Data types:

- String:

```
$x = ’Hello world!’;
$y = 'Hello';
echo gettype($x);//string
echo $x;
echo "<br>";
echo $y;
```

# Working with variables and constants(chapter 4)

Data types:

- Integer:

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

# Working with variables and constants(chapter 4)

Data types:

- Float (floating point numbers - also called double)
- <?php
  $x = 10.365;
  var_dump($x);
  ?>

# Working with variables and constants(chapter 4)

Data types:

- Boolean :

  $x = true;
  $y = false;

# Working with variables and constants(chapter 4)

Data types:

- Array:

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

Output:

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

# Working with variables and constants(chapter 4)

Data types:

- Object: An object is a data type which stores data and information on how to process that data.Explicitly declared.
- ```
  <?php
  class Car {
      function Car() {
          $this->model = "VW";
      }
  }
  // create an object
  $herbie = new Car();
  // show object properties
  echo $herbie->model;//VW
  ?>
  ```
  ```

# Working with variables and constants(chapter 4)

Data types:

- NULL: If variable is created without a value, it is automatically assigned a value of NULL.
- $x = "Hello world!";
  $x = null;
  var_dump($x);//null
  ?>

# Working with variables and constants(chapter 4)

Data Types:

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.

# Working with variables and constants(chapter 4)

Specialized functions:

- Is_bool()//check if variable holds a bool value
- is_int
- is_float
- is_numeric
- is_string
- is_null
- is_array //test if variable is an array
- is_object //test if variable is an object

# Working with variables and constants(chapter 4)

Type casting:

$x=(float)$x;

Or

settype($x, “bool”);

# Working with variables and constants(chapter 4)

- Arithmetic operators(+-,*,/,%,-(negation))
- Assignment operators(=)
- Increment/Decrement operators($x++ return value then increment,++$x,$x--,--$x)
- Logical operators(and(&&),or(||),xor)
- Arithmetic Assignment operators(x += y,similarly for -,*,/,% and . For concatenation)
- Operator precedence(!,*/%,+-.,& bitwise ,|,&&,||,Assignment operators ,and ,xor ,or , comma ,)

# Working with variables and constants(chapter 4)

- Comparison operators(== true if first operand equal to second operand , != or <> , <,> , <=, >=,===(true if first operand equals second in both value and type,!===)

# Working with variables and constants(chapter 4)

- String operators(concatenation ‘.’ )

Echo ‘we are’ . ‘ one india’;

Echo ‘we are’ .$txt;

# Controlling program flow:chp 5

1.IF

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

# Controlling program flow:chp 5

2. if...else

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

# Controlling program flow:chp 5

3. if...elseif....else

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

# Controlling program flow:chp 5

**4. switch**

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

# Controlling program flow:chp 5

5. while Loops

```
<?php
   $x = 1;

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

# Controlling program flow:chp 5

6. Do while Loops

```
<?php
   $x = 1;
   do {
      echo "The number is: $x <br>";
      $x++;
   } while ($x <= 5);
   ?>
```

# Controlling program flow:chp 5

7. for Loop

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

8. foreach Loop

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

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

# Controlling program flow:chp 5

- Continue;

If($i==2)

{

- Exit;//stop program execution

}

# Functions,Arrays,Files,Directories(chp 6)

1.User Defined Functions

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

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

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

# Functions,Arrays,Files,Directories(chp 6)

```
<?php
   function familyName($fname, $year) {
      echo "$fname Refsnes. Born in $year <br>";
   }
   familyName("Hege", "1975");
   familyName("Stale", "1978");
   familyName("Kai Jim", "1983");
   ?>
```

# Functions,Arrays,Files,Directories(chp 6)

- Default Argument Value: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,Arrays,Files,Directories(chp 6)

```
function Change()
{
$score=20;
}
$score=40;
echo “score is”.$score;//40
Change();
echo “score is”.$score;//?
```

# Functions,Arrays,Files,Directories(chp 6)

```
function Change()
{
global $score=20;
}
$score=40;
echo “score is”.$score;//40
Change();
echo “score is”.$score;//20
```

# Functions,Arrays,Files,Directories(chp 6)

```
function change(&$score)
{
$score=20;
}
$score=40;
echo “score is”.$score;//40
Change($score);
echo “score is”.$score;//20
```

# Functions,Arrays,Files,Directories(chp 6)

```
<?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);
   ?>
```

# Functions,Arrays,Files,Directories(chp 6)

**2.Built-in functions:String**

- Strlen($txt);//for hello it is 5
- Explode-splitting string into array

$element=explode(“ ”, $txt);//txt=“hello world”
echo $element[0].”<br/>”;
echo $element[1];

- Implode:join array elements into single string

$joinstring=array(‘kogent’,’is’,’good’);
Echo implode(“ ”, $joinstring);

- Strpos

Echo “position of wo is ”.strpos($txt , “wo”);//6

- Str_repeat()

Echo str_repeat($txt, 5);

- Strrev($txt);

Refer page 205 for other string functions

# Functions,Arrays,Files,Directories(chp 6)

**2.Built-in functions:mathematical**

1.Rand();//3142

Rand(10,100);//41

2.Log(1)

3.Round(3.4)//3

4.Ceil(4.3)//5

5.Pi()

6.Max()//of 2 numbers

7.Pow()

8.Sqrt()

Refer 210 page for more functions

# Functions,Arrays,Files,Directories(chp 6)

**2.Built-in functions: Date and time**

date():

d (1-31)

D (text rep of a day 3 letters)

F text rep of a day january

t no. days in given month

g 1-12 hour

L returns 1 if leap year else 0

l full textual rep of a day

h 01-12 hour

i minutes 00-59

s seconds 00-59

Y 4 digit year rep

a lowercase am or pm

S english suffix for the day of the month (st,nd,rd,th)

# Functions,Arrays,Files,Directories(chp 6)

echo (date ("l dS \of F Y h: i: s a"), " ");

output: thursday 16th of july 2009 11:37:03 pm

var_dump(checkdate(12,31,2009));

output : bool(true)

mktime()-returns time in unix systems

strtotime("now");

strtotime(“tomorrow");

strtotime(“+5 days");

# Functions,Arrays,Files,Directories(chp 6)

**callback functions**

```
function Factorial($number){
    if($number <= 1){
        return 1;
    }
    else{
        return $number * Factorial($number - 1);
    }
}
 // Driver Code
$number = 10;
$fact = Factorial($number);
echo "Factorial = $fact";
```

# Functions,Arrays,Files,Directories(chp 6)

```
function myfunc($msg){
echo $msg;
}
$fun_name='myfunc';
$fun_name("this is variable function calling");
```

# Functions,Arrays,Files,Directories(chp 6)

Callback function:function is passed to another function so that it calls it:

```
Function doit($callback){
$data="This is my data";
$callback($data);
}

function myfunc($data)
{
echo $data;
}
doit('myfunc');
```

# Arrays
# (chp 6)

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

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

- In PHP, There are three types of arrays:

**Indexed arrays** ,**Associative arrays** and **Multidimensional arrays** .

# Arrays

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

# Indexed Arrays

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

or

- $cars[0] = "Volvo";
  $cars[1] = "BMW";
  $cars[2] = "Toyota";

# Indexed Arrays

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

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

# Arrays
# (chp 6)

2.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";

  echo "Peter is " . $age['Peter'] . " years old.";

# Arrays
# (chp 6)

- Loop Through an Associative Array

```
$age = array (“ Peter“ =>"35“ ,"Ben"=>"37“ , "Joe“ => "43");
foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
```

# Arrays
# (chp 6)

- Multidimensional Arrays:

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

Code for printing the values is shown next .

# Arrays
# (chp 6)

```
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>";
```

# Arrays
# (chp 6)

```
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>";
  }
```

# Sort Indexed Arrays in Ascending Order - sort()

- $cars = array("Volvo", "BMW", "Toyota");
  sort($cars);

Or

- $numbers = array(4, 6, 2, 22, 11);
  sort($numbers);

# Sort Indexed Arrays in Descending Order - sort()

- Sort Array in Descending Order - rsort()

# Sort Association Array (Ascending Order), According to Value - asort()

- $age
  = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"
  43");
  asort($age);

# Sort Association Array (Ascending Order), According to Key - ksort()

- $age
  = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"
  43");
  ksort($age);

# Sort Array (Descending Order), According to Value - arsort()

- $age
  = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"
  43");
  arsort($age);

# Sort Array (Descending Order), According to Key - krsort()

- $age
  = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"
  43");
  krsort($age);

# Date and Time (chp 6)

date() function is used to format a date and/or a time.

- Here are some characters that are commonly used for dates:
- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week
- Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.
- The example below formats today's date in three different ways:

# Date and Time
# (chp 6)

```
<?php
   echo "Today is " . date("Y/m/d") . "<br>";
   echo "Today is " . date("Y.m.d") . "<br>";
   echo "Today is " . date("Y-m-d") . "<br>";
   echo "Today is " . date("l");
   ?>
```

# Date and Time (chp 6)

- Here are some characters that are commonly used for times:
- H - 24-hour format of an hour (00 to 23)
- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)
- The example below outputs the current time in the specified format:

# Date and Time (chp 6)

- echo "The time is " . date("h:i:sa");
- The time is 02:17:15am

# Date and Time
# (chp 6)

- <?php
  $d=strtotime("10:30pm April 15 2014");
  echo "Created date is " . date("Y-m-d h:i:sa", $d);
  ?>

- Created date is 2014-04-15 10:30:00pm

# Date and Time (chp 6)

- $d=strtotime("tomorrow");
  echo date("Y-m-d h:i:sa", $d) . "<br>";

  $d=strtotime("next Saturday");
  echo date("Y-m-d h:i:sa", $d) . "<br>";

  $d=strtotime("+3 Months");
  echo date("Y-m-d h:i:sa", $d) . "<br>";

- 2019-03-06 12:00:00am
  2019-03-09 12:00:00am
  2019-06-05 02:21:04am

# File handling

Unit 4 chapter 6

# PHP 5 File Open/Read/Close

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

# PHP 5 File Open/Read/Close

| 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 |

# fread()

- The fread() function reads from an open file.
- The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
- fread($myfile,filesize("webdictionary.txt"));

# fclose()

- The fclose() requires the name of the file (or a variable that holds the filename) we want to close
- fclose($myfile);

# fgets()

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);
?>
```

# 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.

```
<?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);
?>
```

# fgetc()

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);
?>
```

# fwrite()

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

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

```
<?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);
?>
```

# file()

The file() reads a file into an array.

Each array element contains a line from the file, with newline still attached.

**file(path)**

```
<?php
print_r(file("test.txt"));
?>
```

Output :

```
Array
(
[0] => Hello World. Testing testing!
[1] => Another day, another line.
[2] => If the array picks up this line,
[3] => then is it a pickup line?
)
```

# file()

```
$lines=file('test.txt');
Foreach($lines as $line_num => $line)
{
Print "Line ".$line_num. " : " . $line . "<br/>\n";
}
```

# File_get_contents()

Returns string instead of array

$file= File_get_contents(‘test.txt’);

Echo $file;

# Readfile()

Displays file contents on standard buffer

Echo readfile(“test.txt”);

# fputs

fputs():

- The fputs() writes to an open file.
- The function will stop at the end of the file or when it reaches the specified length, whichever comes first.
- This function returns the number of bytes written on success, or FALSE on failure.

fputs(file,string,length*optional)

# Fputs

Fputs():

```
<?php
$file = fopen("test.txt","w");
echo fputs($file,"Hello World. Testing!");
fclose($file);
?>
```

Output :21

# Retrieving file status

```
<?php
$filename = '/path/to/foo.txt';

if (file_exists($filename)) {
    echo "The file $filename exists";
} else {
    echo "The file $filename does not exist";
}
?>
```

# Directories

```
$res=mkdir(“c:\\test”)
Echo “created”;

$res=Rmdir(path);

$res=getCwd();

Chdir (“C:\\dir”);
```

# Directories

```
<?php
$dir    = '/tmp';
$files1 = scandir($dir);
print_r($files1);
?>
```

# Directories

```
if ($handle = opendir('/path/to/files')) {
    echo "Directory handle: $handle\n";
    echo "Entries:\n";

    /* This is the correct way to loop over the dire
ctory. */
    while (false !== ($entry = readdir($handle))) {
        echo "$entry\n";
    } closedir($handle);
}
```

# Retrieving file status

is_executable ($filename) ;

Is_file(name)

Is_link(name);

Is_readable(path)

Is_writable(path)

Flock(filename,mode)//lock_sh/EX/UN/NB---sharedlock,exclusive,unlock,prevents other users access locked file