Server-side scripting:PHP

Unit -1

Introduction to PHP

- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

- PHP is a widely-used, free, and efficient alternative to competitors

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

What are PHP Files?

- ➢PHP files may contain text, HTML tags and scripts
- ➢After PHP files are processed, their output is returned to browser in plain HTML.
- ➢PHP files usually have a file extension of .php, .php3 or phmtl

Why PHP?

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

Versions of PHP

- PHP/FI 2.0
- PHP 3
- PHP 4
- PHP 5

PHP Basic syntax

```
<?php
?>
          Or
<?
?>
Eg.
<?php
     echo    ”Hello World”;
?>
```

PHP Basic

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

Basic features of PHP

- Escaping from HMTL
- Instruction separation
- Comments

Data types

- Int
- Double
- String
- Array
- Object
- Resouce

Declaring PHP

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

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

PHP Variable

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)

Variable Scope

- ➢Local variable
- ➢Function parameter
- ➢Global variable
- ➢Static variable

Local Variable

➢A variable declared in a function is considered local; that is, it can be referenced solely in that function.

```
<?php
$x = 4;
function assignx ()
        { $x = 0;
        print "\$x inside function is $x. <br />";
        }
        assignx();
        print "\$x outside of function is $x. <br />";
?>
```

Global Variables

➢ A global variable must be explicitly declared to be global in the function in which it is to be modified

```
<?php
$somevar = 15;
function addit()
        { GLOBAL $somevar;
        $somevar++;
        print "Somevar is $somevar"; }
        addit();
?>
```

Function parameter

- Function parameters are declared after the function name and inside parentheses.

```
<?php
function multiply ($value)
      { $value = $value * 10;
      return $value; }
      $retval = multiply (10);
      Print "Return value is $retval\n";
?>
```

Static Variables

➢A static variable will not lose its value when the function exits and will still hold that value should the function be called again.

```
<?php
function keep_track()
        { STATIC $count = 0;
        $count++;
        print $count; print "<br />";
        } keep_track(); keep_track(); keep_track();
?>
```

Types of operators

- Arithmetic operators[+,-,*,/,%,++,--]
- Comparison operator[==,!=,<,>,<=,=>]
- Logical/ Relational operator [&&,||,!,or,and]
- Assignment operator[=]
- Conditional/ ternary operator[?:]

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).

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

Example

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

PHP Constant Arrays

```
<?php
define("cars", ["Alfa Romeo", "BMW", "Toyota"]);
echo cars[0];
?>
```

Constants are Global

```
<?php
define("GREETING", "Welcome to
W3Schools.com!");
function myTest() {
   echo GREETING;
}

myTest();
?>
```

String functions

- strlen()
- str_word_count()
- strrev()
- strpost()
- str_replace()
- addslashes()
- str_repeat()
- str_split()
- strcasecmp()
- strcmp()
- substr_count()

Strfun_b2 demo

Echo & Print

- ➢Both are used to output data to the screen
- ➢Echo has no return value while print has a return value of 1 so it can be used in expression
- ➢Echo can take multiple parameters while print can take on argument.
- ➢Echo is marginally faster than print.

Decision making

- if
- if…else
- elseif…
- switch

If….

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

Syntax

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

If example

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

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

If... else...

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;
}
```

If…else… Example

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

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

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 first condition is false and
this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

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!";
}
?>
```

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;
}
```

Switch Example

```
➢<?php
$favcolor = "red";

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

PHP Loops

In PHP, we have the following loop types:

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

PHP while loop

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

Syntax

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

PHP while loop example

```
<?php
$x = 1;

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

PHP do…while loop

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

Syntax

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

PHP do...while loop example

```
<?php
$x = 1;

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

PHP for loop

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

Syntax

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

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

PHP for loop example

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

PHP foreach loop

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

Syntax

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

PHP foreach loop example

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

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

PHP Arrays

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";
```

PHP Arrays

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

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";
```

Indexed Array Example

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

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

PHP Associative arrays

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

There are two ways to create an associative array:

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

or:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

PHP Associative arrays

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

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

PHP Multidimensional

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

PHP supports 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 2-Multidimensional

```
<?php
$cars = array
  (array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

PHP - Sort Functions For Arrays

sort() - sort arrays in ascending order

rsort() - sort arrays in descending order

asort() - sort associative arrays in ascending order, according to the value

ksort() - sort associative arrays in ascending order, according to the key

arsort() - sort associative arrays in descending order, according to the value

krsort() - sort associative arrays in descending order, according to the key

PHP Global Variables - Superglobals

The PHP superglobal variables are:

$GLOBALS
$_SERVER
$_REQUEST
$_POST
$_GET
$_FILES
$_ENV
$_COOKIE
$_SESSION

PHP User defined Function

Besides the more than 10000 built-in PHP functions, it is possible to create your own functions.

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

➢A function will not execute automatically when a page loads.

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

➢Function arguments can be pass by value or pass by reference

Syntax

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

Fun_in_PHP_b2 demo

Function arguments : Pass by value

Pass by value: here the variables are sent as an argument to a defined function.

Example:

```
Function addFour($num){
$num+=4;
Echo $num;
{
$originalnum = 40;
addFour($originalnum);
}
```

Function arguments : Pass by reference

Pass by reference: It creates a new indicator but indicates to same varibles. The & sign is used while passing by reference.

Example:

```
Function test($num){
$num = $num +10;
Echo $num;
{
$val = 10;
$ ref = &$val;
test($ref);
}
```

PHP User defined Function example

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

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

PHP Build-in Function

- List of Build-in Functions

PHP: Working with Forms

- The PHP superglobals $_GET and $_POST are used to collect form-data.
- **Both GET and POST create an array (e.g. array(key1 => value1, 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.**

When to use 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. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page.
- ➢GET may be used for sending non-sensitive data.

When to use POST?

- Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

PHP Form Validation

Validation Rules

Name: Required. + Must only contain letters and whitespace

E-mail: Required. + Must contain a valid email address (with @ and .)

Website : Optional. If present, it must contain a valid URL

Comment : Optional. Multi-line input field (textarea)

Gender : Required. Must select one

Form_Val_b2

Form_val

PHP: Regular Expressions

- Regular expressions are nothing more than a sequence or pattern of characters itself. They provide the foundation for pattern-matching functionality.

- preg_match(): The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.
Syntax: int preg_match (string pattern, string string [, array pattern_array], [, int $flags [, int $offset]]]);
- preg_match_all(): The preg_match_all() function matches all occurrences of pattern in string.
Syntax:int preg_match_all (string pattern, string string, array pattern_array [, int order]);