UNIT-1

Introduction to PHP

SYLLABUS

- Introduction to PHP
- A Brief History of PHP
- How PHP works?, PHP file structure, PHP start and end tags, Commenting codes (Single line, Multi line)
- Creating and saving a PHP file
- Output statement, echo and print statement
- Installing PHP for (Windows, Wamp server , linux , Lamp server, XXAMP server), Configuring Apache to use PHP, Testing the PHP Installation

- PHP Variable and value types, data types, changing types with settype(), casting
- PHP Operators (Arithmetic, Logical, Bitwise, Assignment, String, Inc/ Decrement, Comparison)
- Operator precedence, constants, predefined constants

- Flow control statements: The simple if statement, the if-else statement, else if clause, switch statement, The ? operator
- Loops: the While statement, do.. While statement, For statement, breaking out with break statement, continue statement, nesting loops.

INTRODUCTION TO PHP

- 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.
- 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.

- 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

WAMP, LAMP & XAMPP Server

- WAMP Server
  - WAMP is Windows, **A**pache, **M**ySQL and **P**HP.
  - WAMP Server is work on Windows Operating System only.
- LAMP Server
  - LAMP is Linux Apache MySQL and PHP
  - LAMP Server is work on Linux Operating System only

- XAMPP
  - xampp stands for x-operating systems, apache, MySQL, php , Perl.
  - X-OS means it can be used for any operating system.
  - XAMPP for major operating system including windows, Mac, Linux.
  - XAMPP come with additional features including support of Perl, filezilla, mercury mail and some scripts.

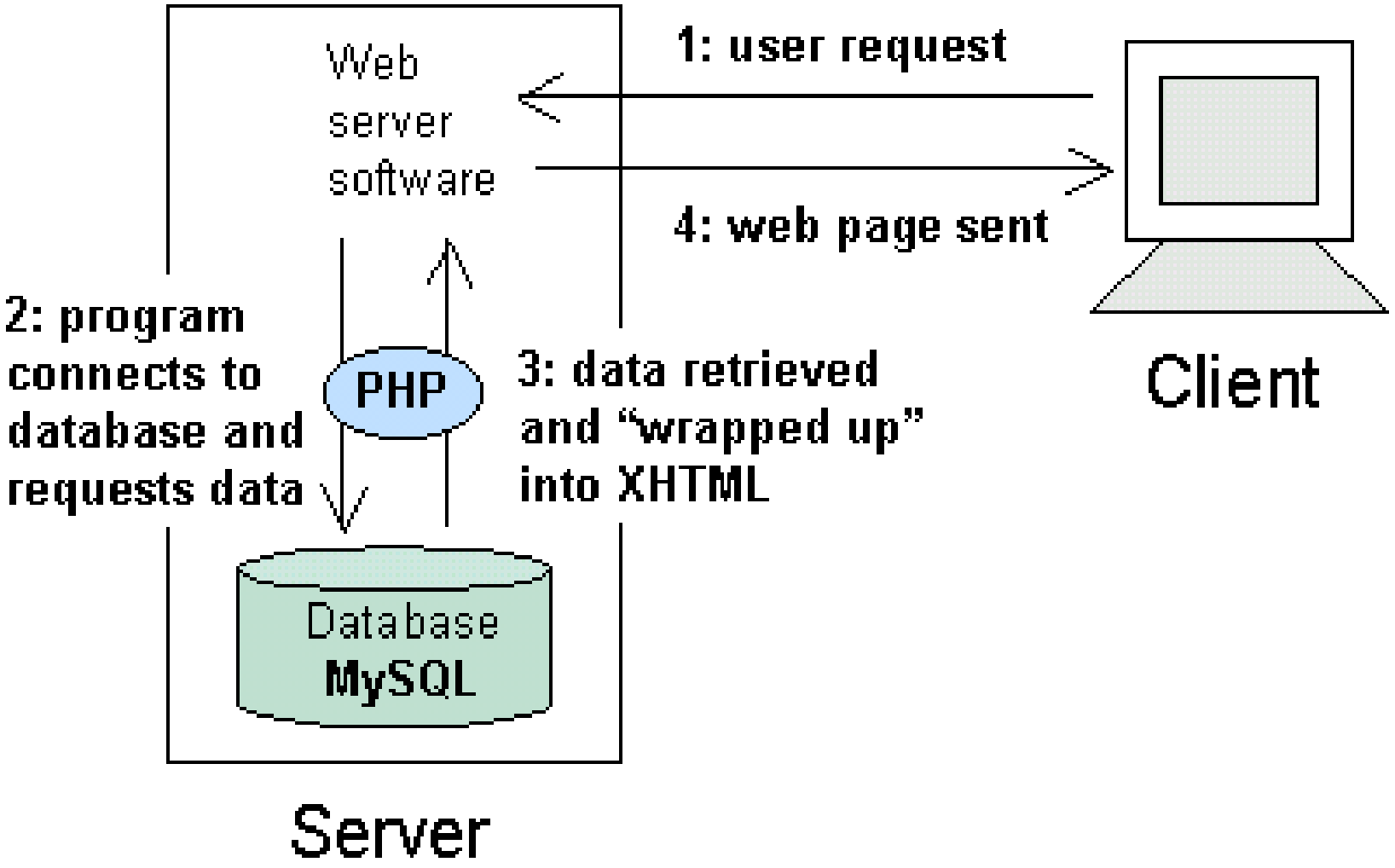
A Brief History of PHP

- PHP Created in **1994 by Rasmus Lerdorf,** the very first incarnation of PHP was a written in the C programming language. Originally used for tracking visits to his online resume, he named the suite of scripts "Personal Home Page Tools,"
- In June of 1995, the source code for PHP Tools to the public, which allowed developers to use it as they saw fit. This also permitted - and encouraged - users to provide fixes for bugs in the code, and to generally improve upon it& its give the name of PHP1.0.
- Zeev Suraski and Andi Gutmans rewrote the parser in 1997 and formed the base of PHP 3, changing the language's name to the recursive acronym PHP: Hypertext Preprocessor. Afterwards, public testing of PHP 3 began, and the official launch came in June 1998.

- Suraski and Gutmans then started a new rewrite of PHP's core it release Version PHP 2.0.In year 1998 they release new version of PHP & it is known as PHP version 3.0.
- After they release of PHP Version 3.0 Zeev Suraski and Andi Gutmans add various feature to the exiting version of PHP such as session handling,output buffering etc..& release a PHP version 4.0
- PHP version 4.0 does not support full Object Oriented Programming features.Finally the PHP Version 5.0 was release in 2004 with Object Oriented Programming Concept.

How PHP works?, PHP file structure, PHP start and end tags, Commenting codes (Single line, Multi line)

- The best way to explain how PHP works is by comparing it with standard HTML. Imagine you type the address of an HTML document (e.g. **http://www.mysite.com/page.htm**) in the address line of the browser.
- This way you **request** an HTML page. It could be illustrated like this: As you can see, the server simply sends an HTML file to the client.
- But if you instead type **http://www.mysite.com/page.php** - and thus request an **PHP page** - the server is put to work.


Web
server
software
1: user request
4: web page sent
2: program
connects to
database and
requests data
PHP
3: data retrieved
and "wrapped up"
into XHTML
Client
Database
MySQL
Server

PHP START AND END TAGS

- To execute the php script it must be written between php start tag and eng tag.
- If it is not written between start tag and end tag then server will not execute the php script.
- PHP support four types of start and end tag.

Sr. No.	Tag Type	Start Tag	End Tag
01	Standard Tag	<?php	?>
02	Short Tag	<?	?>
03	Script Tag	<script language="PHP">	</script>
04	ASP style Tag	<%	%>

- The standard tag and script tag are supported by all servers. And the Short tag and ASP style tag are not supported by all the servers.
- To enable the short tag in our server we have to set our php.ini file as given below:

short_open_tag = on;

- To enable the ASP style tag in our server we have to set our php.ini file as given below:

asp_tags=on;

Commenting codes (Single line, Multi line)

- A comment in PHP code is a line that is not read/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:
  - Single line comment
    - Using // or #
  - Multi line comment
    - Using /* and */

Example of Comment in PHP

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

CREATING AND SAVING A PHP FILE

- 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".

- A PHP file normally contains HTML tags, and some PHP scripting code.
- We have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:
- Example of PHP Script

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

NOTE

- PHP statements end with a semicolon (;).
- 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.

Output statement, echo and print statement

- In PHP output statements are used for displaying any string message or value of variables.
- PHP supports two types of output statements
  - echo statement
  - print statement

ECHO STATEMENT

- The echo statement basically used to display string or value of variable.
- It can be used as a statement or as a function.
- It accepts one or more strings as an argument and display one or more strings on the browser.
- The echo statement does not return any value.
- It is faster compared to print statement.
- **Syntax:** echo string;

- **Example**

```
<?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.";
?>
```

PRINT STATEMENT

- The print statement is basically used to display string or value of variable.
- It can be used as a statement or as a function.
- As a statement
  - **print “Welcome to PHP”;**
- As a function
  - **print (“Welcome to PHP”);**
- The print statement accepts only one string as an argument and display it on browser.
- The print statement always return 1.
- It is slower compared to echo statement.

- **Example**

```
<?php
print "<h2>PHP is Fun!</h2>”;
print "Hello world! “.“ Good Morning";
print ( “<br>I'm about to learn PHP!“);
?>
```

2.6 Installing PHP for (Windows, Wamp server , linux , Lamp server, XXAMP server), Configuring Apache to use PHP, Testing the PHP Installation

- **If you decide to download PHP and install it manually the step is below.**

1. Download PHP and the WinCache extension.
2. Install PHP and WinCache.
3. Add the PHP installation folder to the Path environment variable.
4. Set up a handler mapping for PHP.
5. Add default document entries for PHP.
6. Test your PHP installation.

- **Installing PHP for wamp server**

1. Download the WAMP Server
2. Start Setup Wizard
3. Select Destination and Icon
4. Installing and Complate
5. Run WAMP Server.

- **Installing PHP for Lamp server**

1. A "LAMP" stack is a group of open source software that is typically installed together to enable a server to host dynamic websites and web apps. This term is actually an acronym which represents the **L**inux operating system, with the **A**pache web server. The site data is stored in a **M**ySQL database, and dynamic content is processed by **P**HP.

2. The LAMP stack can be installed automatically on your Droplet by adding this script to its User Data when launching it. Check out this tutorial to learn more about Droplet User Data.

- **Installing PHP for XXAMP server**

1.In your web browser, go to

https://www.apachefriends.org/index.html

2. Click on the download link for **XAMPP.**

3. When prompted for the download, click "Save" and wait for your download to finish.

4.Open CD or DVD drive from My Computer. Install the program, and click on "Run.“

5.Accept the default settings. A command will open and offer an initial installation prompt. Just hit the Enter key, and accept the default settings. To simplify installation, just hit ENTER when prompted on the command line. You can always change settings, by editing the configuration files later.

- 6.When your installation is complete, exit the command window by typing x on the command line.
- 7. Start the XAMPP Control Panel.
- 8. Start the Apache and MySQL components. You can also start the other components, if you plan to use them.
- 9.Verify the Apache install, by clicking on the Apache administrative link in the Control Panel.
- 10. Verify the MySQL installation, by clicking on the MySQL administrative link in the XAMPP Control Panel.

Configuring Apache to use PHP, Testing the PHP Installation

- Now that we have configured PHP to work as we want it, lets go to Apache and do the same.
- Open httpd.conf. and in the "Dynamic Shared Object (DSO) Support" section add the following directives (if you have located your PHP folder differently do make corresponding change for php5apache2_2.dll below):
- LoadModule php5_module "C:/Program Files/PHP/php5apache2_2.dll" AddType application/x-httpd-php .php In the DirectoryIndex add index.php and index.htm as possible files to serve when directory is requested as follows DirectoryIndex index.html index.htm index.php .
- At the end of the file add the following line which will point out where the php.ini file is locatedPHPIniDir "C:/Program Files/PHP"

- **Restart and test PHP**
- As soon as you make any change to any of php.ini or httpd.conf or any other configuration files you need to restart Apache to see the actual effect of the changes. So lets now restart Apache by using the Apache Monitor tool you can find in your Windows status bar. Hopefully you are not prompted with any dialogues and the Apache Monitor continues to run green.
- We will now make a test that PHP is working. Go to your web servers document root (in the default case C:\Program Files\Apache Software Foundation\Apache2.2\htdocs) and add a file called phpinfo.php with the following content:
- <?php phpinfo(); phpinfo(INFO_MODULES); ?> This will render a page containing information about your PHP setup and about the various modules/extensions that are currently loaded.

USE PHP VARIABLES, DATA TYPES AND OPERATORS.

PHP variables

- A variable is memory location which is used to store value.
- 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)
- Example: $name = “GPP”;

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 = "W3Schools.com";
echo "I love $txt!";
?>
```

OR

```
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

- Example: Sum of two variables

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

DATATYPE IN PHP

- Datatype is used to specify which type of value can be stored in the variable and how much memory required by a variable.

- In PHP, the datatype of variable is not required at variable declaration. Because it will automatically determine its type based on the value that is stored in it.

Datatype	Use
Integer	To store number without decimal point.
Float or Double	To store number with decimal point.
String	To store collection of characters enclosed between double quotation mark.
Boolean	To store True or False value.
Array	To store multiple values in the form of key and value pair.
Object	To represent an instance of class.
Null	To represent an initializes variable.

2.7 CHANGING TYPES WITH SETTYPE(), CASTING

- **Gettype( ) function**
- In PHP, gettype( ) function is used to get the datatype of variable.
- **Syntax:**

```
gettype(variable name);
```

- **Example:**

```
<? php
$a=10;
echo gettype($a);
?>
```

- **Output:**

```
Integer
```

SETTYPE( ) FUNCTION

- In PHP, settype( ) function is used to set the datatype of variable.
- It returns Boolean value.
- If variable is set successfully to specific type then it returns 1 otherwise 0.
- **Syntax:**

  settype(variable name, datatype);

- **Example:**

```
<?php
$a;
echo settype($a,”Integer”);
echo gettype($a);
echo $a;
?>
```

- **Output:**

```
Integer
0
```

TYPE CASTING

- Type casting is used to change the datatype of variable but the difference is that casting produces a copy of existing variable and changes datatype of newly created variable.
- The datatype of existing variable remains as it is.

- **Syntax:**

```
$NewVariable = (Datatype) $OldVariable;
```

- Example:

```
<?php
        $a=10;
        $b = (float) $a;
        echo gettype($a);
        echo gettype($b);
?>
```

- Output

```
Integer
Double
```

2.8 Describe PHP 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

1. 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

2. 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	Equal to…	Meaning
x = y	$x + $y	Sum of $x and $y
x += y	$x - $y	Difference of $x and $y
*	$x * $y	Product of $x and $y
/	$x / $y	Quotient of $x and $y
%	$x % $y	Remainder of $x divided by $y
**	$x ** $y	Result of raising $x to the $y'th power

3. 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
>	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

4. 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

5. 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

6. 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	Meaning
++$a	Pre-increment	Increments $a by one, then returns $a
$a++	Post-increment	Returns $a, then increments $a by one
--$a	Pre-Decrement	Decrements $a by one, then returns $a
$a--	Post-decrement	Returns $a, then decrements $a by one

2.9 Operator precedence, constants, predefined constants

- The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression *1 + 5 * 3*, the answer is *16* and not *18* because the multiplication ("*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance: *(1 + 5) * 3* evaluates to *18*.
- Generally in PHP expression are evaluated from left to right & each operator is assigned precedence.

CONSTANT

- 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*)

  - ***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.

- Constants are automatically global and can be used across the entire script.
- **Example:**

```
<?php
        $r=10;
        define(“PI", 3.14);
        echo “Area = ” . PI*$r*$r;
?>
```

2D. APPLY CONTROL STRUCTURES IN PROGRAMMING

2.10 FLOW CONTROL STATEMENTS: THE SIMPLE IF STATEMENT, THE IF-ELSE STATEMENT, ELSE IF CLAUSE, SWITCH STATEMENT, THE ? OPERATOR

- In PHP we have the following conditional statements:
- **if statement** - executes some code only if a specified condition is true
- **if...else statement** - executes some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - specifies a new condition to test, if the first condition is false
- **switch statement** - selects one of many blocks of code to be executed

IF STATEMENT

- The if statement is used to execute some code **only if a specified condition is true**.
- **Syntax**

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

- **Example**

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

The if...else Statement

- Use the if....else statement to execute some code **if a condition is true and another code if the condition is false**.
- **Syntax**

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

- Example

```
<?php
        $a = 10;
        if ($a%2 == 0)
        {
            echo “a is even number";
        }
        else
        {
            echo “a is odd number";
        }
?>
```

The if...elseif....else Statement

- Use the if....else if...else statement to **specify a new condition to test, if the first condition is false.**
- **Syntax:**

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

- Example

```
<?php
$a = 10;
if ($a < 0)
{
    echo “a is negative number";
}
else if ($a >0)
{
    echo " a is positive number ";
}
else
{
    echo " a is zero";
}
?>
```

The 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;
   ...
    default:
        code to be executed if n is different from all labels;
}
```

- First we have a single expression *n* (most often a variable), that is evaluated once.
- The value of the expression is then compared with the values for each case in the structure.
- 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.

- **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!";
}
?>
```

2.11 Loops: the While statement, do.. While statement, For statement, breaking out with break statement, continue statement, nesting loops.

- **PHP Loops**
- If 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.

PHP Loops

- 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

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;
}
```

- **Example**

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

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);
```

- In a do while loop the condition is tested AFTER executing the statements within the loop.
- This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

- Example:

```
<?php
        $x = 6;
        do
        {
                echo "The number is: $x <br>";
                $x++;
        } while ($x<=5);
?>
```

For Loop

- PHP 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 counter; test counter; increment counter*)

  {

  code to be executed;

  }
- *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

FOR LOOP

- Example

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

BREAKING OUT WITH BREAK STATEMENT

- break statement is used inside loop to transfer control of script immediately after the loop either conditionally or unconditionally.
- If we write break statement inside the nested loop then it will transfer control of the script outside loop in which it exits.
- **Example**

```
<?php
   for($i;$i<=10;$i++)
   {
         if ($i%5==0)
         break;
         else
         echo $i."<br>";
   }
?>
```

CONTINUE STATEMENT

- It is required to skip the execution of some statements inside the loop.
- Continue statement allows to skip the statement that appear immediately after it & transfer the control of the script to the next iteration in the loop.
- It is used to inside for,while or do…while loop.
- **Example**

```
<?php
   for($i;$i<=10;$i++)
   {
         if ($i%2==0)
         continue;
         else
         echo $i.”<br>”;
}
?>
```

NESTING LOOPS

- When one loop is contained another loop then it is called as Nested Loop.
- Generally nested loops are used with 2-dimensional Array & Matrix Representation.
- It is also used to display the information of Tablear form in row & column wise data display.
- **Example**

```
<?php
    for($i=1;$i<=5;$i++)
    {
            for($j=1;$j<=5;$j++)
            {
                    echo  $i * $j;
                    echo " ";
            }
    echo "<br>";
    }
?>
```

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;
}
```

Example:

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

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

Value	Constant	Description
1	E_ERROR	Fatal run-time errors. Errors that cannot be recovered from. Execution of the script is halted
2	E_WARNING	Run-time warnings (non-fatal errors). Execution of the script is not halted
4	E_PARSE	Compile-time parse errors. Parse errors should only be generated by the parser
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
16	E_CORE_ERROR	Fatal errors at PHP startup. This is like E_ERROR, except it is generated by the core of PHP
32	E_CORE_WARNING	Non-fatal errors at PHP startup. This is like E_WARNING, except it is generated by the core of PHP
64	E_COMPILE_ERROR	Fatal compile-time errors. This is like E_ERROR, except it is generated by the Zend Scripting Engine
128	E_COMPILE_WARNING	Non-fatal compile-time errors. This is like E_WARNING, except it is generated by the Zend Scripting Engine
256	E_USER_ERROR	Fatal user-generated error. This is like E_ERROR, except it is generated in PHP code by using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like E_WARNING, except it is generated in PHP code by using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like E_NOTICE, except it is generated in PHP code by using the PHP function trigger_error()
2048	E_STRICT	Enable to have PHP suggest changes to your code which will ensure the best interoperability and forward compatibility of yourcode (Since PHP 5 but not included in E_ALL until PHP 5.4)
4096	E_RECOVERABLE_ERROR	Catchable fatal error. Indicates that a probably dangerous error occurred, but did not leave the Engine in an unstable state.If the error is not caught by a user defined handle, the application aborts as it was an E_ERROR (Since PHP 5.2)
8192	E_DEPRECATED	Run-time notices. Enable this to receive warnings about code that will not work in future versions (Since PHP 5.3)
16384	E_USER_DEPRECATED	User-generated warning message. This is like E_DEPRECATED, except it is generated in PHP code by using the PHP function trigger_error() (Since PHP 5.3)
32767	E_ALL	Enable all PHP errors and warnings (except E_STRICT in versions < 5.4)