CHAPTER

5 INTRODUCTION TO PHP

Chapter Outlines

5.1 INSTALLATION OF PHP

The Windows PHP installer for later versions of PHP is built using MSI technology using the Wix Toolkit (http://wix.sourceforge.net/). It will install and configure PHP and all the built-in and PECL extensions, as well as configure many of the popular web servers such as IIS, Apache, Wamp and Xitami.

First, install your selected HTTP (web) server on your system, and make sure that it works. Then proceed with one of the following install types.

Normal Install : Run the MSI installer and follow the instructions provided by the installation wizard. You will be prompted to select the Web Server you wish to configure first, along with any configuration details needed.

You will then be prompted to select which features and extensions you wish to install and enable. By selecting **"Will be installed on local hard drive"** in the drop-down menu for each item you can trigger whether to install the feature (or) not. By selecting ***"Entire feature will be installed on local hard drive",*** you will be able to install all sub-features of the included feature (for example by selecting this options for the feature "PDO" you will install all PDO Drivers).

The installer then sets up PHP to be used in Windows and the *php.ini* file, and configures certain web servers to use PHP. The installer will currently configure IIS, Apache, Xitami and Sambar Server ; if you are using a different web server you'll need to configure it manually.

Basics of PHP : PHP is a powerful tool for making dynamic and interactive Web pages. PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. PHP code is executed on the server, and the plain HTML result is sent to the browser.

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document. A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code. Below, we have an example of a simple PHP script which sends the text **"Hello World"** to the browser :

```
<html>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

5.2 PHP BASICS AND FEATURES

Introduction : PHP stands for **"hypertext preprocessor"**, which gives you a good idea of its core purpose. It is a free software that builds dynamic, interactive web sites. PHP is a reflective language. As a program, it can observe and modify itself at the same time that it is being executed.

As a general rule, PHP program run on a web server, and serve web pages to visitors on request, and its complete source code is available for free, so users can customize it to their needs. Its focus is server-side scripting and it can be compared to other languages like ASP.NET by Microsoft and JavaServer Pages (JSPs) by Sun Microsystems in its ability to provide users with dynamic content through web servers. In HTML, PHP is embedded between tags. These tags make it possible to jump between the HTML and PHP.

To make PHP work as server-side scripting, three components are needed: a PHP parser, a programming routine that analyzes and breaks down programming language/sentences into more easily processed functions. PHP parsers are usually CGIs (or) server modules. CGIs are the most common way for users to interact with web servers. Server modules are programming modules that provide services to other programs. The other two required components are a web browser and a web server. PHP creates dynamic web content and can send (or) receive cookies. Cookies are text files sent between the web browser and web server used to analyze user behavior and create customized web content.

5.3 EMBEDDING PHP IN HTML

PHP is a powerful tool for making dynamic and interactive Web pages. PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. PHP code is executed on the server, and the plain HTML result is sent to the browser.

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document. A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.Below, we have an example of a simple PHP script which sends the text **"Hello World"** to the browser :

```
<html>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

5.4 VARIOUS DATA TYPES WITH EXAMPLES

Data Types : In PHP, a variable does not need to be declared before adding a value to it .PHP automatically converts the variable to the correct data type, depending on its value. In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

Examples :

1. **$name = 'Robbin Jackman';**

```
$salary = 5000;
```

Here **$name** is a string and **$salary** is a number. You can see that these two are two different data types. In PHP we call them **String** and **Integer** respectively.

2. **$name = 'Mr '.$name;**

```
$salary = $salary+200;
```

For **$name** we could do String_Concatenation using dot operator because **$name** is a string. We could use Addition operator on **$salary** because it's an integer.

5.5 DECLARE VARIABLES AND CONSTANTS

Variables in PHP : Variables are used for storing values, like text strings, numbers (or) arrays. When a variable is declared, it can be used over and over again in your script. All variables in PHP start with a $ sign symbol

Syntax :

```
$var_name = value;
```

ASSIGNMENT OPERATORS :

Operator	Example	Is The Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x/y
.=	x . = y	x = x . y
%=	x % = y	x = x % y

COMPARISON OPERATORS :

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than (or) equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

LOGICAL OPERATORS :

Operator	Description	Example

&&	and	x = 6
		y = 3
		(x < 10 && y > 1) returns true

||	or	x = 6
		y = 3
		(x==5 || y==5) returns false

!	not	x = 6
		y = 3
		!(x==y) returns true

Example :

```
<?php
$txt="Hello World!";
$x=16;
?>
```

Naming Rules for Variables :

- A variable name must start with a letter (or) an underscore "_"
- variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string) (or) with capitalization ($myString).

5.6 VARIOUS OPERATORS

Operators : Operators are used to operate on values. The following are different categories of operators in PHP.

ARITHMETIC OPERATORS :

Operator	Description	Example	Result
+	Addition	x = 2 x + 2	4
–	Subtraction	x = 25 – x	3
*	Multiplication	x = 4 x *5	20
/	Division	15/55/2	32.5
%	Modulus (division remainder)	5% 210% 810% 2	120
++	Increment	x = 5 x ++	x = 6
—	Decrement	x = 5 x —	x = 4

5.7 VARIOUS CONDITIONAL STATEMENTS WITH EXAMPLES

Control Structure : There are two control structures namely conditional statements and looping statements.

Conditional statements are used to perform different actions based on different conditions.

The following are the conditional statements :

- if statement.
- if...else statement.
- if...elseif....else statement.
- switch statement.

if Statement : Use this statement to execute some code only if a specified condition is true.

Syntax :

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

Example :

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!";
?>
</body>
</html>
```

if...else Statement : Use this statement to execute some code if a condition is true and to execute 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 :

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
    echo "Have a nice weekend!";
else
    echo "Have a nice day!";
?>
</body>
</html>
```

if...elseif....else Statement : Use this statement to select one of several blocks of code to be executed

Syntax :

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

Example :

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

Switch Statement : Use this 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 both label1 and
label2;
}
```

Example :

```
<?php
$destination = "Tokyo";
echo "Traveling to $destination<br />";
switch ($destination)
{
    case "Las Vegas":
            echo "Bring an extra $500";
            break;
    case "Amsterdam":
            echo "Bring an open mind";
            break;
    case "Egypt":
            echo "Bring 15 bottles of SPF 50 Sunscreen";
            break;
    case "Tokyo":
            echo "Bring lots of money";
            break;
    case "Caribbean Islands":
            echo "Bring a swimsuit";
            break;
    default:
            echo "invalid destination";
}
?>
```

Output :

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

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

Syntax :

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

Parameters :

- ***init*** **:** Mostly used to initialize the counter variable.
- ***condition*** **:** Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- ***increment*** **:** Mostly used to increment the counter variable.

Example :

```
<?php
for ($i=1; $i<=5; $i++)
 {
 echo "The number is" . $i . "<br />";
}
?>
```

Output :

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

Output :

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

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

Syntax :

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

Example : The example below defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than (or) equal to 5:

```
<html>
<body>

<?php
$i=1;
do
 {
 $i++;
 echo "The number is" . $i . "<br />";
 }
while ($i<=5);
?>

</body>
</html>
```

foreach Loop : The *foreach* loop is used to loop through arrays.

Syntax :

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

Example :

```
<?php
$x=array("one","two","three");
foreach ($x as $value)
 {
 echo $value . "<br />";
 }
?>
```

Output :

```
One
Two
Three
```

5.9 STRINGS AND STRING METHODS

String : It can be defined as a group characters which is terminated by a NULL.

Each character of a string occupies one byte and last character of a string is always the character '\0'.

\0 → Null character and it stands for a character with a value of zero.

The string functions allow you to manipulate strings. Following are the most commonly used string-handling functions which are contained in <string.h>

5.8 VARIOUS LOOP STATEMENTS WITH EXAMPLES

The Following Looping Statements :

- while loop
- do...while loop
- for loop
- foreach loop

The while Loop : The while loop executes a block of code repeatedly while a condition is true.

Syntax :

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

Example : The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, (or) equal to 5. i will increase by 1 each time the loop runs :

```
<html>
<body>
<?php
$i=1;
while($i<=5)
 {
 echo "The number is" . $i . "<br />";
 $i++;
 }
?>
</body>
</html>
```

```
test(arr);
// length:5, 0:a, 1:b, 2:c, 3:d, 4:e
var arr = [undefined, undefined, undefined, undefined,
undefined];
test(arr);
// length:5, 0:undefined, 1:undefined, 2:undefined,
3:undefined, 4:undefined
```

5.10.3 MANIPULATE AN ARRAY

Changing the Length

```
var arr = ['a', 'b', 'c', 'd', 'e', 'f'];
test(arr);
// length:6, 0:a, 1:b, 2:c, 3:d, 4:e, 5:f
arr.length = 5;
test(arr);
// length:5, 0:a, 1:b, 2:c, 3:d, 4:e
var arr = ['a','b','c','d','e','f',,,];
test(arr);
// length:8, 0:a, 1:b, 2:c, 3:d, 4:e, 5:f
arr.length = 7;
test(arr);
// length:7, 0:a, 1:b, 2:c, 3:d, 4:e, 5:f
```

Removing Entries : JavaScript provides three methods pop, shift and splice that can remove entries from the array and which therefore reduce the length of the array. In each case the value (or values) removed are returned by the call.

// pop() removes the last element from an array.

```
var arr = ['a','b','c','d','e','f'];
var el = arr.pop();
test(arr); // length:5, 0:a, 1:b, 2:c, 3:d, 4:e
console.log(el); // f
```

```
// shift() removes the first element from an array
var arr = ['a','b','c','d','e','f'];
var el = arr.shift();
test(arr); // length:5, 0:b, 1:c, 2:d, 3:e, 4:f
console.log(el); // a
// splice() can remove existing elements
var arr1 = ['a','b','c','d','e','f'];
var arr2 = arr1.splice(0,2); // remove 2 elements starting at
index 0
test(arr1); // length:4, 0:c, 1:d, 2:e, 3:f
test(arr2); // length:2, 0:a, 1:b
var arr1 = ['a','b','c','d','e','f',,,'i'];
var arr2 = arr1.splice(6,2); // remove 2 elements starting at
index 6
test(arr1); // length:7, 0:a, 1:b, 2:c, 3:d, 4:e, 5:f, 6:i
test(arr2); // length:2
```

5.10.4 ARRAY METHODS

Adding Entries : JavaScript provides e three methods push, unshift and slice for inserting new entries.

```
// push() adds one or more elements to the end of an array
var arr = ['a','b','c','d','e','f',,,,];
arr.push('j');
test(arr);
// length:10, 0:a, 1:b, 2:c, 3:d, 5:f, 9:j
// unshift() adds one or more elements to the beginning of an
array
var arr = ['a','b','c','d','e','f',,,,];
arr.unshift('x');
test(arr);
// length:10, 0:x, 1:a, 2:b, 3:c, 4:d, 5:e, 6:f
```

```
arr1 = ['a','b','c','d','e','f',,,'i'];

arr2 = arr1.splice(6,0,'g','h'); // removes 0 elements from index
6, and inserts 'g', 'h'

test(arr1); // length:11, 0:a, 1:b, 2:c, 3:d, 5:f, 6:g, 7:h, 10:i

test(arr2); // length:0

<html>

<body>

<p>The push method appends a new element to an array.
</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>

var fruits = ["Banana", "Orange", "Apple", "Mango"];

document.getElementById("demo").innerHTML = fruits;

function myFunction() {

   fruits.push("Lemon")

   document.getElementById("demo").innerHTML = fruits;

}

</script>

</body>

</html>
```

5.10.5 PROGRAMS USING ARRAYS AND ARRAY METHODS

Program below illustrates one dimensional array in javascript

```
<script>

var people = ["joseph","Maria", "Brian", "Susan"];

 document.write(people[0]);

</script>
```

Function	Description
str_repeat()	Repeats a string a specified number of times.
strlen()	Find the length of a string.
strcmp()	Compares two strings (case-sensitive).
strrev()	Reverses a string.
chr()	Returns a character from a specified ASCII value.

5.10 IMPLEMENT ARRAYS

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

Using an array literal is the easiest way to create a JavaScript Array.

5.10.1 SINGLE AND MULTI DIMENSIONAL ARRAYS

Using an array literal is the easiest way to create a JavaScript Array.

Syntax :

```
var array-name = [item1, item2, ...];
```

Example :

```
var cars = ["Saab", "Volvo", "BMW"];
```

5.10.2 DECLARE AN ARRAY

//Creates an array with no numbered entries.

```
var arr = new Array(5);
test(arr);
// length: 5
var arr = [];
arr.length = 5;
test(arr);
// length: 5
var arr = ['a', 'b', 'c', 'd', 'e'];
```

Programs below illustrate multi-dimensional arrays in javascript

```
<script>
var people = [
["joseph", 27, "united states"],
["Maria", 21, "united states" ],
 ["Brian", 35, "united states" ],
 ["Susan", 42, "united states"]
];
 document.write(people[0][0]);//display name joseph
</script>
<script>
var people = [
["joseph", 27, "United states"],
["Maria", 21, "Uraguay" ],
 ["Brian", 35, "United kingdom" ],
 ["Susan", 42, "Australia"]
];
 document.write(people[0][0]);//display name joseph
</script>
<script>
var people = [
["joseph", 27, "United states",["blue","black"]],
["Maria", 21, "Uraguay", ",["brown","green"]  ],
 ["Brian", 35, "United kingdom", ",["green","red"]  ],
 ["Susan", 42, "Australia", ",["blue","blonde"]]
];
 document.write(people[0][3][0]);//display name joseph eye
colour blue
</script>
<script>
var people = [
["joseph", 27, "United states",["blue","black"]],
```