Chapter 1: PHP Introduction

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML
- CSS
- JavaScript

What is PHP?

- PHP stands for PHP: Hypertext Preprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

What type of scripting is PHP client or a server side?

PHP is a server side scripting language; server side scripting language is one, which has the capability of executing the script on the server and serving the output as a HTML File, server side script has the main advantage of interacting with the databases and to perform all types of server manipulations directly. Server side scripting language is responsible for manipulating the data which is filled in the entire web forms, anywhere in the net.

But this is not the case of a client side script. Some of the client side scripting languages is the JavaScript and VB script, but there are exceptions where, client side script can also be used for server side scripting, which is limited to some extent.

Client side scripting	Server side scripting
1. It executes on client.	1. It executes on server
2. Examples : DHTML, JavaScript, VBScript, Flash Animation, css, Java Applets	2. Online forms with some dropdown lists That assembles on server.
3. It is glamorous, eye catching part of web Development	3. It is invisible to user.
4. It reduces the round trip so quickly response to user input.	4. It has round trip.
5. They depend entirely on Browsers. Wide variations exist in capability of each browsers, some people disable JavaScript for security reasons.	5. They do not depend entirely on browsers.

Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support

PHP Variable Types

The main way to store information in the middle of a PHP program is by using a variable. Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables:

- **Integers:** are whole numbers, without a decimal point, like 4195.
- **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
- **Booleans:** have only two possible values either true or false.
- **NULL:** is a special type that only has one value: NULL.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'
- **Arrays:** are named and indexed collections of other values.
- **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

We will explain only simile data type in this chapters. Array and Objects will be explained separately.

Integers:

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so:

```
$int_var = 12345;
$another_int = -12345 + 12345;
```

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.

For most common platforms, the largest integer is (2**31 . 1) (or 2,147,483,647), and the smallest (most negative) integer is . (2**31 . 1) (or .2,147,483,647).

Doubles:

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code:

```
$many = 2.2888800;
$many_2 = 2.2111200;
$few = $many + $many_2;
print(.$many + $many_2 = $few<br>.);
```

It produces the following browser output:

```
2.28888 + 2.21112 = 4.5
```

Boolean:

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so:

```
if (TRUE)
   print("This will always print<br>");
else
   print("This will never print<br>");
```

Interpreting other types as Booleans:

Here are the rules for determine the "truth" of any value not already of the Boolean type:

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
$true_num = 3 + 0.14159;
$true_str = "Tried and true"
$true_array[49] = "An array element";
$false_array = array();
$false_null = NULL;
$false_num = 999 - 999;
$false_str = "";
```

NULL:

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this:

```
$my_var = NULL;
```

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed:

```
$my_var = null;
```

A variable that has been assigned NULL has the following properties:

- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with IsSet() function.

Strings:

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = "This is a somewhat longer, singly quoted string";
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?
$variable = "name";
$literally = 'My $variable will not print!\\n';
print($literally);
$literally = "My $variable will print!\\n";
print($literally);
?>
```

This will produce following result:

```
My $variable will not print!\n
My name will print
```

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP:

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

PHP Operator Types

What is Operator? Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.

- Arithmetic Operators
- Comparision Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Let’s have a look on all operators one by one.

Arithmetic Operators:

There are following arithmatic operators supported by PHP language:

Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denumerator	B / A will give 2
%	Modulus Operator and remainder of	B % A will give 0

	after an integer division	
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Try following example to understand all the arithmatic operators. Copy and paste following PHP program in test.php file ando keep it in your PHP Server's document root and browse it using any browser.

```
<html>
<head><title>Arithmetical Operators</title><head>
<body>
<?php
    $a = 42;
    $b = 20;

    $c = $a + $b;
    echo "Addtion Operation Result: $c <br/>";
    $c = $a - $b;
    echo "Substraction Operation Result: $c <br/>";
    $c = $a * $b;
    echo "Multiplication Operation Result: $c <br/>";
    $c = $a / $b;
    echo "Division Operation Result: $c <br/>";
    $c = $a % $b;
    echo "Modulus Operation Result: $c <br/>";
    $c = $a++;
    echo "Increment Operation Result: $c <br/>";
    $c = $a--;
    echo "Decrement Operation Result: $c <br/>";
?>
</body>
</html>
```

This will produce following result

```
Addtion Operation Result: 62
Substraction Operation Result: 22
Multiplication Operation Result: 840
Division Operation Result: 2.1
Modulus Operation Result: 2
Increment Operation Result: 42
Decrement Operation Result: 43
```

Comparison Operators:

There are following comparison operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Try following example to understand all the comparision operators. Copy and paste following PHP program in test.php file ando keep it in your PHP Server's document root and browse it using any browser.

```
<html>
<head><title>Comparision Operators</title><head>
<body>
<?php
    $a = 42;
    $b = 20;

    if( $a == $b ){
       echo "TEST1 : a is equal to b<br/>";
    }else{
       echo "TEST1 : a is not equal to b<br/>";
    }

    if( $a > $b ){
       echo "TEST2 : a is greater than  b<br/>";
```

```
    }else{
       echo "TEST2 : a is not greater than b<br/>";
    }
    if( $a < $b ){
       echo "TEST3 : a is less than  b<br/>";
    }else{
       echo "TEST3 : a is not less than b<br/>";
    }
    if( $a != $b ){
       echo "TEST4 : a is not equal to b<br/>";
    }else{
       echo "TEST4 : a is equal to b<br/>";
    }
    if( $a >= $b ){
       echo "TEST5 : a is either grater than or equal to b<br/>";
    }else{
       echo "TEST5 : a is nieghter greater than nor equal to b<br/>";
    }
    if( $a <= $b ){
       echo "TEST6 : a is either less than or equal to b<br/>";
    }else{
       echo "TEST6 : a is nieghter less than nor equal to b<br/>";
    }
?>
</body>
</html>
```

This will produce following result

```
TEST1 : a is not equal to b
TEST2 : a is greater than b
TEST3 : a is not less than b
TEST4 : a is not equal to b
TEST5 : a is either grater than or equal to b
TEST6 : a is nieghter less than nor equal to b
```

Logical Operators:

There are following logical operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Try following example to understand all the logical operators. Copy and paste following PHP program in test.php file ando keep it in your PHP Server's document root and browse it using any browser.

```
<html>
<head><title>Logical Operators</title><head>
<body>
<?php
    $a = 42;
    $b = 0;

    if( $a && $b ){
       echo "TEST1 : Both a and b are true<br/>";
    }else{
       echo "TEST1 : Either a or b is false<br/>";
    }
    if( $a and $b ){
       echo "TEST2 : Both a and b are true<br/>";
    }else{
       echo "TEST2 : Either a or b is false<br/>";
    }
    if( $a || $b ){
       echo "TEST3 : Either a or b is true<br/>";
    }else{
       echo "TEST3 : Both a and b are false<br/>";
    }
    if( $a or $b ){
       echo "TEST4 : Either a or b is true<br/>";
```

```
    }else{
       echo "TEST4 : Both a and b are false<br/>";
    }
    $a = 10;
    $b = 20;
    if( $a ){
       echo "TEST5 : a is true <br/>";
    }else{
       echo "TEST5 : a  is false<br/>";
    }
    if( $b ){
       echo "TEST6 : b is true <br/>";
    }else{
       echo "TEST6 : b  is false<br/>";
    }
    if( !$a ){
       echo "TEST7 : a is true <br/>";
    }else{
       echo "TEST7 : a  is false<br/>";
    }
    if( !$b ){
       echo "TEST8 : b is true <br/>";
    }else{
       echo "TEST8 : b  is false<br/>";
    }
?>
</body>
</html>
```

This will produce following result

```
TEST1 : Either a or b is false
TEST2 : Either a or b is false
TEST3 : Either a or b is true
TEST4 : Either a or b is true
TEST5 : a is true
TEST6 : b is true
TEST7 : a is false
TEST8 : b is false
```

Assignment Operators:

There are following assignment operators supported by PHP language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Try following example to understand all the assignment operators. Copy and paste following PHP program in test.php file ando keep it in your PHP Server's document root and browse it using any browser.

```
<html>
<head><title>Assignment Operators</title><head>
<body>
<?php
    $a = 42;
    $b = 20;

    $c = $a + $b;   /* Assignment operator */
    echo "Addtion Operation Result: $c <br/>";
    $c += $a;  /* c value was 42 + 20 = 62 */
    echo "Add AND Assigment Operation Result: $c <br/>";
    $c -= $a; /* c value was 42 + 20 + 42 = 104 */
    echo "Subtract AND Assignment Operation Result: $c <br/>";
    $c *= $a; /* c value was 104 - 42 = 62 */
    echo "Multiply AND Assignment Operation Result: $c <br/>";
    $c /= $a;  /* c value was 62 * 42 = 2604 */
    echo "Division AND Assignment Operation Result: $c <br/>";
    $c %= $a; /* c value was 2604/42 = 62*/
    echo "Modulus AND Assignment Operation Result: $c <br/>";
?>
</body>
</html>
```

This will produce following result

```
Addtion Operation Result: 62
Add AND Assigment Operation Result: 104
Subtract AND Assignment Operation Result: 62
Multiply AND Assignment Operation Result: 2604
Division AND Assignment Operation Result: 62
Modulus AND Assignment Operation Result: 20
```

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax:

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Try following example to understand the conditional operator. Copy and paste following PHP program in test.php file ando keep it in your PHP Server's document root and browse it using any browser.

```
<html>
<head><title>Arithmetical Operators</title><head>
<body>
<?php
    $a = 10;
```

```
    $b = 20;

    /* If condition is true then assign a to result otheriwse b */
    $result = ($a > $b ) ? $a :$b;
    echo "TEST1 : Value of result is $result<br/>";
    /* If condition is true then assign a to result otheriwse b */
    $result = ($a < $b ) ? $a :$b;
    echo "TEST2 : Value of result is $result<br/>";
?>
</body>
</html>
```

This will produce following result

```
TEST1 : Value of result is 20
TEST2 : Value of result is 10
```

PHP Arrays

PHP Arrays

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

- **Numeric array** - An array with a numeric index. Values are stored and accessed in linear fashion
- **Associative array** - An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- **Multidimensional array** - An array containing one or more arrays and values are accessed using multiple indices

Numeric Array

These arrays can store numbers, strings and any object but their index will be prepresented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
<body>
<?php
/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);
foreach( $numbers as $value )
{
  echo "Value is $value <br />";
}
/* Second method to create array. */
$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";

foreach( $numbers as $value )
{
  echo "Value is $value <br />";
}
?>
</body>
</html>
```

This will produce following result:

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
Value is five
```

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

NOTE: Don't keep associative array inside double quote while printing otherwise it would not return any value.

Example

```
<html>
<body>
<?php
/* First method to associate create array. */
$salaries = array(
                  "mohammad" => 2000,
                  "qadir" => 1000,
                  "zara" => 500
                 );

echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
echo "Salary of zara is ".  $salaries['zara']. "<br />";

/* Second method to create array. */
$salaries['mohammad'] = "high";
$salaries['qadir'] = "medium";
$salaries['zara'] = "low";

echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
echo "Salary of zara is ".  $salaries['zara']. "<br />";
?>
</body>
</html>
```

This will produce following result:

```
Salary of mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low
```

Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example

In this example we create a two dimensional array to store marks of three students in three subjects:

This example is an associative array, you can create numeric array in the same fashion.

```
<html>
<body>
<?php
   $marks = array(
               "mohammad" => array
               (
               "physics" => 35,
               "maths" => 30,
               "chemistry" => 39
               ),
               "qadir" => array
                (
                "physics" => 30,
                "maths" => 32,
                "chemistry" => 29
                ),
                "zara" => array
                (
                "physics" => 31,
                "maths" => 22,
                "chemistry" => 39
                )
            );
            /* Accessing multi-dimensional array values */
            echo "Marks for mohammad in physics : " ;
            echo $marks['mohammad']['physics'] . "<br />";
            echo "Marks for qadir in maths : ";
            echo $marks['qadir']['maths'] . "<br />";
            echo "Marks for zara in chemistry : " ;
            echo $marks['zara']['chemistry'] . "<br />";
?>
</body>
</html>
```

This will produce following result:

```
Marks for mohammad in physics : 35
Marks for qadir in maths : 32
Marks for zara in chemistry : 39
```

PHP Strings

They are sequences of characters, like "PHP supports string operations".

Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = "This is a somewhat longer, singly quoted string";
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?
$variable = "name";
$literally = 'My $variable will not print!\\n';
print($literally);
$literally = "My $variable will print!\\n";
print($literally);
?>
```

This will produce following result:

```
My $variable will not print!\n
My name will print
```

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP:

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

String Concatenation Operator

To concatenate two string variables together, use the dot (.) operator:

```
<?php
$string1="Hello World";
$string2="1234";
echo $string1 . " " . $string2;
?>
```

This will produce following result:

```
Hello World 1234
```

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string.

Between the two string variables we added a string with a single character, an empty space, to separate the two variables.

Using the strlen() function

The strlen() function is used to find the length of a string.

Let's find the length of our string "Hello world!":

```
<?php
echo strlen("Hello world!");
?>
```

This will produce following result:

```
12
```

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string)

Using the strpos() function

The strpos() function is used to search for a string or character within a string.

If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string:

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

This will produce following result:

```
6
```

NOTE: For Built-in string functions ref Chapter 3