BASICS OF PHP

4.1 Introduction to PHP & Features

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 such as Microsoft's ASP.

Example

```html
<html>
<body>
<?php
echo "My first PHP
script!"; ?>
</body>
</html>
```

What You Should Already Know

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

- HTML
- CSS
- JavaScript

What is 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

What is a PHP File?

- 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

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

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

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host with PHP Support

- If your server has activated support for PHP you do not need to do anything.
- Just create some .php files, place them in your web directory, and the server will automatically parse them for you.
- You do not need to compile anything or install any extra tools.
- Because PHP is free, most web hosts offer PHP support.

- Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- install a web server
- install PHP
- install a database, such as MySQL

4.2 PHP Scripts

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**:

```
<?php
// PHP code goes
here ?>
```

Example
```
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```
- PHP statements end with a semicolon (;)

Comments in PHP

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.

Comments can be used to:

- Let others understand what you are doing
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

Example

```
<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>
```

Example

```
<html>

<body>
<?php

ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

In the example below, only the first statement will display the value of the $color variable (this is because $color, $COLOR, and $coLOR are treated as three different variables):

Example

```
<html>

<body>
<?php
$color = "red";
echo "My car is " . $color . "<br>";

echo "My house is " . $COLOR .
"<br>"; echo "My boat is " . $coLOR .
"<br>"; ?>
```

```
</body>
</html>
```

4.3 Data Types

- Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

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

PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

Example

```html
<html>
<body>
<?php
$x = "Hello world!";
$y = 'Hello world!';
echo $x;
echo "<br>";
echo $y;
?>
</body>
</html>
```

OUTPUT:

Hello world!
Hello world!

String Functions

- Get The Length of a String
- The PHP strlen() function returns the length of a string.
- The example below returns the length of the string "Hello world!":

Example

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

OUPUT:

12

Count The Number of Words in a String

The PHP str_word_count() function counts the number of words in a string:

Example

```html
<html>
<body>
<?php
echo str_word_count("Hello world!");
?>
 </body>
</html>
```

OUPUT:

2

Reverse a String

- The PHP strrev() function reverses a string:

Example

```html
<html>
<body>
<?php
```

```
echo strrev("Hello world!");
?>
 </body>
</html>
```

OUTPUT:

!dlrow olleH

Search For a Specific Text Within a String

- The PHP strpos() function searches for a specific text within a string.
- If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
- The example below searches for the text "world" in the string "Hello world!":

Example

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

OUPUT:

6

Replace Text Within a String

- The PHP str_replace() function replaces some characters with some other characters in a string.
- The example below replaces the text "world" with "Dolly":

Example

```
<html>
<body>
<?php
echo str_replace("world", "Dolly", "Hello world!");
?>
 </body>
</html>
```

OUPUT:

Hello Dolly!

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

Example

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

OUTPUT:

int(5985)

PHP Float

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- In the following example $x is a float. The PHP var_dump() function returns the data type and value:

Example

```
<html>
<body>
```

```php
<?php
$x = 10.365;
var_dump($x);
?>
```
```html
</body>
</html>
```

OUTPUT:

float(10.365)

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```php
$x = true;
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Array

- An array stores multiple values in one single variable:

- 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:

```php
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

- The solution is to create an array!

- An array can hold many values under a single name, and you can access the values by referring to an index number.

Example

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

```
</body>
</html>
```

OUTPUT:

I like Volvo, BMW and Toyota.

Create an Array in PHP

- 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

4.4 Variables

- Variables are "containers" for storing information.

- Creating (Declaring) PHP Variables

- In PHP, a variable starts with the $ sign, followed by the name of the variable:

Example

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

echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>

</body>
</html>
```

Output:

Hello world!
5
10.5

After the execution of the statements above, the variable **$txt** will hold the value **Hello world!**, the variable **$x** will hold the value **5**, and the variable **$y** will hold the value **10.5**.

Note: When you assign a text value to a variable, put quotes around the value.

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Rules for PHP variables:

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

- 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)

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:

Example

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

Output:

I love W3Schools.com!

Example

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

Output:

I love W3Schools.com!

Example

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

Output:
9

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

```
<html>
<body>
<?php
$x = 5; // global scope
function myTest() {
  // using x inside this function will generate an error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is:
$x</p>"; ?>
</body>
</html>
```

OUTPUT:

Variable x inside function is:

Variable x outside function is: 5

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Example

```html
<html>
<body>
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate an
error echo "<p>Variable x outside function is:
$x</p>"; ?>
</body>
</html>
```

OUTPUT:

Variable x inside function is: 5
Variable x outside function is:

The global Keyword

- The global keyword is used to access a global variable from within a function.
- To do this, use the global keyword before the variables (inside the function):

Example

```html
<html>
<body>
<?php
$x = 5;
$y = 10;
```

Output:
15

```php
function myTest() {
    global $x, $y; $y
    = $x + $y;
}
myTest(); // run function
echo $y; // output the new value for variable
$y ?>
</body>
</html>
```

The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the static keyword when you first declare the variable:

Example

```php
<html>
<body>
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}
myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
</body>
</html>
```

Output:
0
1
2

echo and print Statements

- In PHP there are two basic ways to get output: echo and print.
- In this tutorial we use echo (and print) in almost every example. So, this chapter contains a little more info about those two output statements.
- echo and print are more or less the same. They are both used to output data to the screen.
- The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

Example

```
<html>
<body>
<?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."; ?>
</body>
</html>
```

OUTPUT:

PHP is Fun!
Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

Display Variables

The following example shows how to output text and variables with the echo statement:

Example

```
<html>
<body>
<?php
$txt1 = "Learn PHP"; $txt2
= "W3Schools.com"; $x =
5;
$y = 4;
echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
```

```php
echo $x + $y;
?>
</body>
</html>
```

OUTPUT:

Learn PHP

Study PHP at
W3Schools.com 9

The PHP print Statement

- The print statement can be used with or without parentheses: print or print().

Display Text

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

Example

```php
<html>
<body>
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";

print "I'm about to learn
PHP!"; ?>
</body>
</html>
```

OUTPUT:

PHP is Fun!
Hello world!
I'm about to learn PHP!

Display Variables

The following example shows how to output text and variables with the print statement:

Example
```php
<html>
<body>
<?php
```

```php
$txt1 = "Learn PHP";

$txt2= "W3Schools.com";

$x = 5;
$y = 4;
print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x +
$y; ?>
</body>
</html>
```

OUTPUT:

Learn PHP

Study PHP at
W3Schools.com 9

PHP Object

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.
- First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example

```php
<html>
<body>
<?php
class Car {
   function Car() { $this-
      >model = "VW";
   }
}
// create an object
$herbie = new Car();

// show object
properties echo
$herbie->model; ?>

</body>
</html>
```

OUPUT:

VW

PHP NULL Value

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL:

Example

```
<html>
<body>
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
</body>
</html>
```

OUTPUT:

NULL

Constants

- Constants are like variables except that once they are defined they cannot be changed or undefined.
- **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).

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

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

The example below creates a constant with a **case-sensitive** name:

Example

```html
<html>
<body>
<?php
// case-sensitive constant name
define("GREETING", "Welcome to W3Schools.com!"); echo GREETING;
?>
</body>
</html>
```

OUTPUT:

Welcome to W3Schools.com!

The example below creates a constant with a **case-insensitive** name:

Example

```html
<html>
<body>
<?php
// case-insensitive constant name
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
</body>
</html>
```

OUPUT:

Welcome to W3Schools.com!

Constants are Global

- Constants are automatically global and can be used across the entire script.
- The example below uses a constant inside a function, even if it is defined outside the function:

Example

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

OUPUT:

Welcome to W3Schools.com!

4.5 PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

PHP 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 (Introduced in PHP 5.6)

PHP 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	Same as...	Description

x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP 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

PHP 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	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP 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

PHP 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	

PHP Array Operators

- The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y

==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

4.6 PHP Conditional Statements

- Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif....else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

The if Statement

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

Syntax

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

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

Example

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

OUTPUT:

Have a good day!

The if...else Statement

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;
}
```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

Example

```html
<html>
<body>
<?php
```

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

OUTPUT:

Have a good day!

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

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

Example

```php
<html>
<body>
<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";
if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
```

```
} else {
    echo "Have a good night!";
}
?>
</body>
</html>
```

OUTPUT:

The hour (of the server) is 01, and will give the following message:

Have a good morning!

The switch Statement

- The switch statement is used to perform different actions based on different conditions.

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;
}
```

This is how it works: 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

```
<html>
<body>
<?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!";
}
?>
</body>
</html>
```

OUTPUT:

Your favorite color is red!

PHP Loops

Often when you write code, 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.

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

The 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;
}

The example below first sets a variable $x to 1 ($x = 1). Then, the while loop will continue to run as long as $x is less than, or equal to 5 ($x <= 5). $x will increase by 1 each time the loop runs ($x++):

Example

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

OUTPUT:

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

The 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);
```

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

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

OUTPUT:

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

Notice that 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.

The example below sets the $x variable to 6, then it runs the loop, **and then the condition is checked**:

Example

```html
<html>
<body>
<?php
$x = 6;
```

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

OUTPUT:

The number is: 6

for Loops

- PHP for loops execute a block of code a specified number of times.

The 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;
}
```

Parameters:

- *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

The example below displays the numbers from 0 to 10:

Example

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

```
</body>
</html>
```

OUTPUT:

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

The 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;
}
```

- For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

- The following example demonstrates a loop that will output the values of the given array ($colors):

Example

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

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

```
}
?>
</body>
</html>
```

OUTPUT:

red
green
blue
yellow

4.7 Working with Arrays

PHP 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";
- The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

Example

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

OUTPUT:

I like Volvo, BMW and Toyota.

Get The Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

Example

```
<html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
</body>
</html>
```

OUTPUT:

3

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

Example

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

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

OUTPUT:

Volvo
BMW
Toyota

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";
```

- The named keys can then be used in a script:

Example

```
<html>
<body>
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
</body>
</html>
```

OUTPUT:

Peter is 35 years old.

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

```
<?php

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $x_value) {

    echo "Key=" . $x . ", Value=" . $x_value;

    echo "<br>";

}

?>
```

Sorting Arrays

The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.

- 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

Sort Array in Ascending Order - sort()

The following example sorts the elements of the $cars array in ascending alphabetical order:

Example

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

$clength = count($cars);
for($x = 0; $x < $clength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
</body>
</html>
```

OUTPUT:
BMW
Toyota
Volvo

The following example sorts the elements of the $numbers array in ascending numerical order:

Example

```
<?php
$numbers = array(4, 6, 2, 22,
11); sort($numbers);
?>
```

OUTPUT:

2
4
6
11
22

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

The following example sorts an associative array in ascending order, according to the value:

Example

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

OUTPUT:

Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43

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

The following example sorts an associative array in ascending order, according to the key:

Example

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

OUTPUT:

Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35

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

The following example sorts an associative array in descending order, according to the value:

Example

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

OUTPUT:

Key=Joe, Value=43
Key=Ben, Value=37
Key=Peter, Value=35

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

The following example sorts an associative array in descending order, according to the key:

Example

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

OUTPUT:

Key=Peter, Value=35
Key=Joe, Value=43
Key=Ben, Value=37

- An array stores multiple values in one single variable.
- In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

Example

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

OUTPUT:

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

4.8 Functions

- The real power of PHP comes from its functions; it has more than 1000 built-in functions.

PHP User Defined Functions

- Besides the built-in PHP functions, we can create our own functions.
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user-defined function declaration starts with the word function:

Syntax

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

- Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:

Example

```html
<html>
<body>
<?php
function writeMsg()
echo "Hello world!";
}
writeMsg();
?>
</body>
</html>
```

OUTPUT:

Hello world!

PHP Function Arguments

- Information can be passed to functions through arguments. An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example

```html
<html>
<body>
```

```php
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}
familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
</body>
</html>
```

OUTPUT:

Jani Refsnes.
HegeRefsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.

The following example has a function with two arguments ($fname and $year):

Example

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

OUTPUT:

Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978
Kai Jim Refsnes. Born in 1983

PHP Default Argument Value

- The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```html
<html>
<body>
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}
setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>
</body>
</html>
```

OUTPUT:

The height is : 350
The height is : 50
The height is : 135
The height is : 80

PHP Functions - Returning values

- To let a function return a value, use the return statement:

Example

```html
<html>
<body>
<?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);
?>
</body>
</html>
```

OUTPUT:

5 + 10 = 15

7 + 13 = 20
2 + 4 = 6

4.9 Working with Files

- File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files

PHP has several functions for creating, reading, uploading, and editing files.

Be careful when manipulating files!

- When you are manipulating files you must be very careful.
- You can do a lot of damage if you do something wrong. Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.

PHP readfile() Function

- The readfile() function reads a file and writes it to the output buffer.
- Assume we have a text file called **"webdictionary.txt",** stored on the server, that looks like this:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

- The PHP code to read the file and write it to the output buffer is as follows (the readfile() function returns the number of bytes read on success):

Example

```
<html>
<body>
<?php
echo readfile("webdictionary.txt");
?>
</body>
</html>
```

OUTPUT:

AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language236

- The readfile() function is useful if all you want to do is open up a file and read its contents.

File Open/Read/Close

PHP Open File - fopen()

- A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.

We will use the text file, **"webdictionary.txt",** during the lessons:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified file:

Example

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

OUTPUT:

AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language

The file may be opened in one of the following modes:

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

PHP Read File - 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.
- The following PHP code reads the "webdictionary.txt" file to the end:

fread($myfile,filesize("webdictionary.txt"));

PHP Close File - fclose()

- The fclose() function is used to close an open file.
- It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!
- The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

```php
<?php
$myfile = fopen("webdictionary.txt",
"r"); // some code to be executed....
fclose($myfile);
?>
```

PHP Read Single Line - fgets()

- The fgets() function is used to read a single line from a file.
- The example below outputs the first line of the "webdictionary.txt" file:

Example

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

OUTPUT:

AJAX = Asynchronous JavaScript and XML

After a call to the fgets() function, the file pointer has moved to the next line.

PHP Check End-Of-File - 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.
- The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

Example

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

OUTPUT:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

PHP Read Single Character - fgetc()

- The fgetc() function is used to read a single character from a file.
- The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

Example

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

OUTPUT:

AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language

- After a call to the fgetc() function, the file pointer moves to the next character.

File Create/Write

PHP Create File - fopen()

- The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
- If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).
- The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Example

```
$myfile = fopen("testfile.txt", "w")
```

PHP File Permissions

If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

PHP Write to File - 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.
- The example below writes a couple of names into a new file called "newfile.txt":

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open
file!"); $txt = "John Doe\n";
fwrite($myfile, $txt);
```

```php
$txt = "Jane Doe\n";
fwrite($myfile,   $txt);
fclose($myfile);
?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose() function.

If we open the "newfile.txt" file it would look like this:

John Doe
Jane Doe

PHP Overwriting

- Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.
- In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

```php
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open
file!"); $txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

OUTPUT:

Mickey Mouse
Minnie Mouse

PHP ftp_fput() Function

Example

- Open local file, and upload it to a file on the FTP server:

```php
<?php
// connect and login to FTP server
$ftp_server = "ftp.example.com";

$ftp_conn = ftp_connect($ftp_server) or die("Could not connect to
$ftp_server"); $login = ftp_login($ftp_conn, $ftp_username, $ftp_userpass);

// open file for reading
$file = "test.txt";
$fp = fopen($file,"r");

// upload file
if (ftp_fput($ftp_conn, "somefile.txt", $fp, FTP_ASCII))
  {
  echo "Successfully uploaded $file.";
  }
else
  {
  echo "Error uploading $file.";
  }

// close this connection and file handler
ftp_close($ftp_conn);
fclose($fp);
?>
```

Definition and Usage

- The ftp_fput() function uploads from an open file and saves it to a file on the FTP server.

Syntax

ftp_fput(*ftp_connection,remote_file,open_file,mode,startpos*);

Parameter	Description
ftp_connection	Required. Specifies the FTP connection to use
remote_file	Required. Specifies the file path to upload to

open_file	Required. Specifies an open local file. Reading stops at end of file
mode	Required. Specifies the transfer mode. Possible values: FTP_ASCII or FTP_BINARY
startpos	Optional. Specifies the position in the remote file to start uploading to

4.10 Working with Databases

PHP MySQL Database

- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers
- Orders

PHP + MySQL Database System

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

PHP Connect to MySQL

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".

Both MySQLi and PDO have their advantages:

- PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
- So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.
- Both are object-oriented, but MySQLi also offers a procedural API.
- Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

MySQL Examples in Both MySQLi and PDO Syntax

In this, and in the following chapters we demonstrate three ways of working with PHP and MySQL:

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

MySQLi Installation

- For Linux and Windows: The MySQLi extension is automatically installed in most cases, when php5 mysql package is installed.
- For installation details, go to: http://php.net/manual/en/mysqli.installation.php

- PDO Installation
- For installation details, go to: http://php.net/manual/en/pdo.installation.php

Open a Connection to MySQL

- Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

```php
<?php
$servername = "localhost";
$username  = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected
successfully"; ?>
```

Example (MySQLi Procedural)

```php
<?php
$servername = "localhost";
$username  = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected
successfully"; ?>
```

Example (PDO)

```php
<?php
$servername = "localhost";
$username  = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB",
$username, $password);
    // set the PDO error mode to exception $conn-
    >setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); echo
    "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Notice that in the PDO example above we have also specified a database (myDB). PDO require a valid database to connect to. If no database is specified, an exception is thrown.

Tip: A great benefit of PDO is that it has an exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

Example (MySQLi Object-Oriented)

$conn->close();

Example (MySQLi Procedural)

mysqli_close($conn);

Example (PDO)

$conn = null;

PHP Create a MySQL Database

- A database consists of one or more tables.
- You will need special CREATE privileges to create or to delete a MySQL database.
- **Create a MySQL Database Using MySQLi**
- The CREATE DATABASE statement is used to create a database in MySQL.
- The following examples create a database named "myDB":

Example (MySQLi Object-oriented)

```php
<?php
$servername = "localhost";
$username  = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username,
$password); // Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
   echo "Database created
successfully"; } else {
   echo "Error creating database: " . $conn->error;
}
$conn-
>close(); ?>
```

Note: When you create a new database, you must only specify the first three arguments to the mysqli object (servername, username and password).

Tip: If you have to use a specific port, add an empty string for the database-name argument, like this: new mysqli("localhost", "username", "password", "", port)

PHP Create MySQL Table

- A database table has its own unique name and consists of columns and rows.

- Create a MySQL Table Using MySQLi
- The CREATE TABLE statement is used to create a table in MySQL.
- We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

CREATE TABLE MyGuests (

id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY, firstname VARCHAR(30) NOT NULL,

lastname VARCHAR(30) NOT NULL, email VARCHAR(50),
reg_date TIMESTAMP
)

Notes on the table above:

- The data type specifies what type of data the column can hold. For a complete reference of all the available data types, go to our [Data Types reference](#).

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

Example (MySQLi Object-oriented)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
```

```php
// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname); // Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY
KEY, firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT
NULL, email VARCHAR(50),
reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created
successfully"; } else {
    echo "Error creating table: " . $conn->error;
}

$conn-
>close(); ?>
```

PHP Insert Data Into MySQL

Insert Data Into MySQL Using MySQLi

- After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Note: If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

The following examples add a new record to the "MyGuests" table:

Example (MySQLi Object-oriented)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname); // Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created
successfully"; } else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn-
>close(); ?>
```

Example:

Student Application using PHP and Mysql

aa.html:

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

```html
<html>
<head>
<title>Student
Table</title> </head>
<body>
<div id="dept"> <h3 align=center>Student table
entry</h3> <form method="POST" action="connect.php">
SName <br><input type="text" name="sname"><br>
Reg.No <br><input type="text" name="regno"><br>
Mark1<br><input type="text" name="m1"><br>
Mark2<br><input type="text" name="m2"><br> <input
type="submit" value="ok">
</form>
</div>
</body>
</html>
```

Conn.php:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "sample";
// Create connection
$conn = new mysqli($servername, $username, $password,$dbname);
// Check connection
```

```php
if ($conn->connect_error)
{
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully"; //connect table
$sql="desc student"; if($conn->query($sql)==TRUE)
{
echo "<br>";
 echo "connected to the table";
}
else
{
echo "error";
}
//Inserting the contents echo "<br>";
//insertion from html
$sname=$_POST['sname'];
$regno=$_POST['regno'];
$m1=$_POST['m1'];
$m2=$_POST['m2'];
$sql11="insert into student values('$sname',$regno,$m1,$m2)";
if($conn->query($sql11)==TRUE)
```

```php
{
echo "inserted";
}
else
{echo "error";}
echo "<br>";
$sql1="select * from student";
$result = $conn->query($sql1);
if ($result->num_rows > 0) {
    // output data of each row
echo "<b>Sname Regno M1 M2</b><br>";
    while($row = $result->fetch_assoc()) {
        echo $row["sname"]." ". $row["regno"]." ".$row["m1"]." ".$row["m2"]."<br>";
    }
} else {
    echo "empty table";
}
$conn->close(); ?>
```

OUTPUT:

Index of /sample

Name	Last modified	Size	Description
Parent Directory		-	
connect.php	04-Oct-2017 22:32	1.2K	
stud.html	04-Oct-2017 21:58	448	

Apache/2.2.9 (Win32) DAV/2 mod_ssl/2.2.9 OpenSSL/0.9.8h mod_autoindex_color PHP/5.2.6 Server at localhost Port 80

Student table entry

SName

Reg.No

Mark1

Mark2

ok

Connected successfully
connected to the table
inserted
Sname Regno M1 M2
aaa 1234 67 90
xxx 123 56 66
zzz 1234567 78 99