CHAPTER-5-PHP

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 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 is 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

PHP **Installation- https://youtu.be/1lxzfBQ0NPo**

https://codingcampus.net/how-to-run-php-in-visual-studio-code/

What Do I Need?

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:

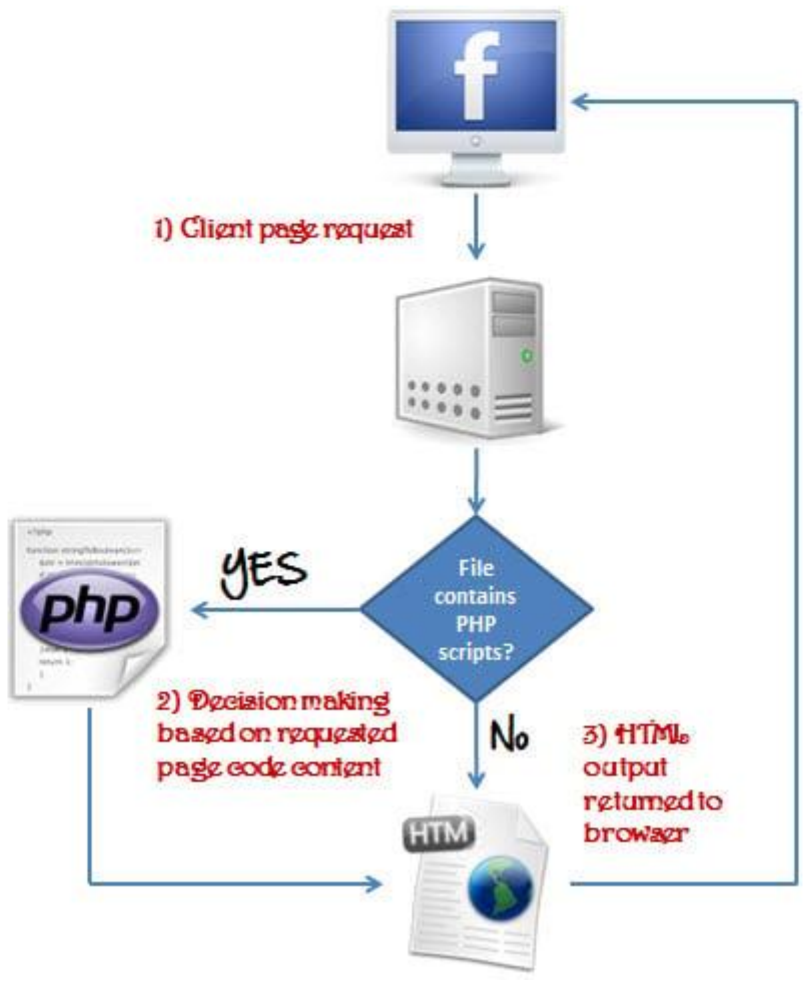
- install a web server
- install PHP
- install a database, such as MySQL

The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

A PHP file can also contain tags such as HTML and client side scripts such as JavaScript.

- **HTML is an added advantage** when learning PHP Language. You can even learn PHP without knowing HTML but it's recommended you at least know the basics of HTML.
- **Database management systems** DBMS for database powered applications.
- For more advanced topics such as interactive applications and web services, you will need **JavaScript and XML**.

The flowchart diagram shown below illustrates the basic architecture of a PHP web application and how the server handles the requests.


1) Client page request
File contains PHP scripts?
YES
2) Decision making based on requested page code content
No
3) HTML output returned to browser

What is XAMPP?

XAMPP is an open-source web server solution package. It is mainly used for web application testing on a local host webserver.

XAMPP stands for:

X = Cross-platform

A = Apache Server

M = MariaDB

P = PHP

P = Perl

Now that you have a better understanding of the XAMPP software, let us move on to its installation.

Why Do You Need XAMPP?

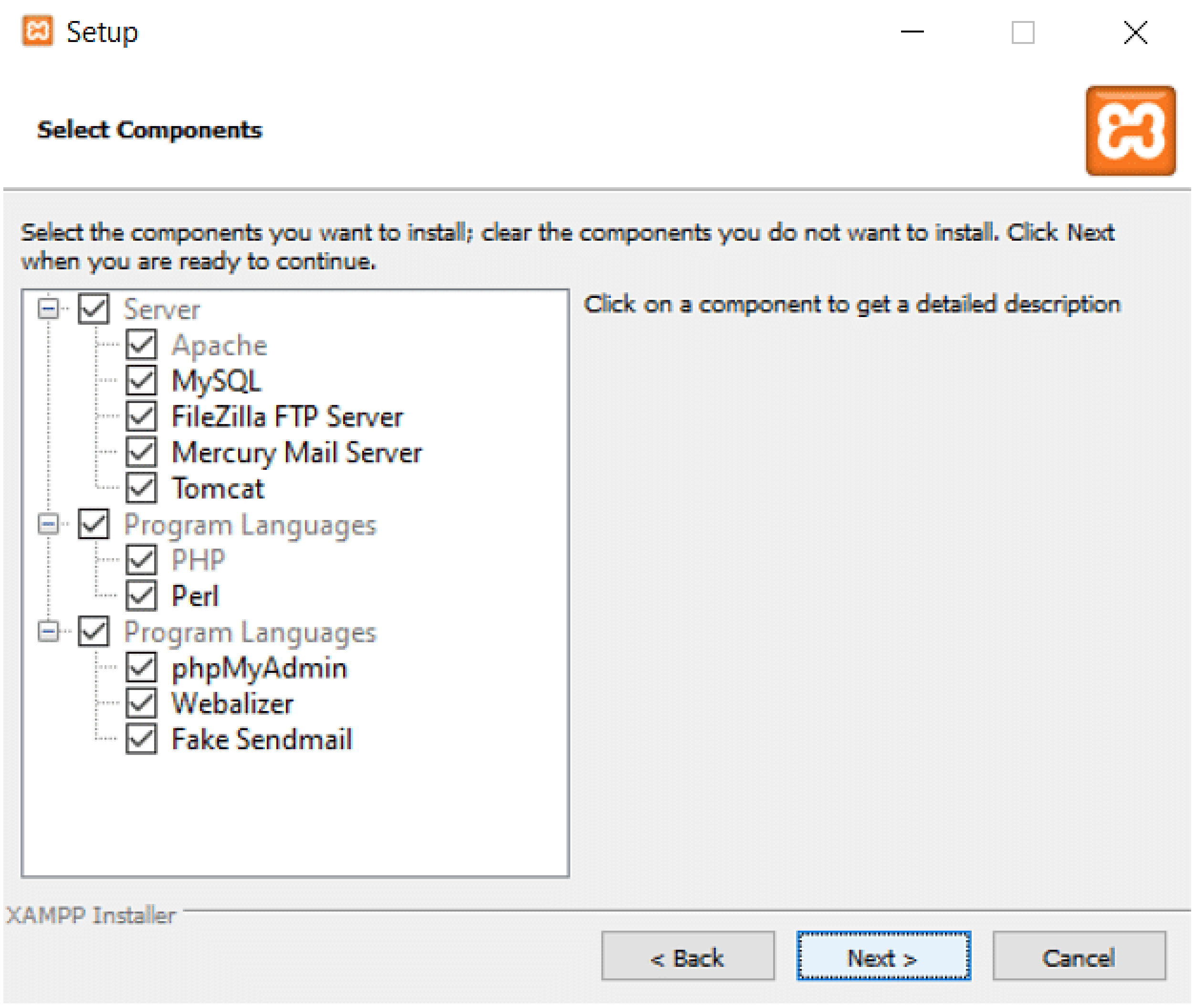
To run any PHP program, you will need Apache or MYSQL databases, both supported by XAMPP. XAMPP helps you to run your program smoothly.

How to Install XAMPP?

Apache friends developed XAMPP, and it is available for everyone free of cost.

You can download XAMPP through the official website, https://www.apachefriends.org/download.html.

On completing the download of the setup file, begin the installation process and, in the "Select Components" section, select all the required components.


Setup

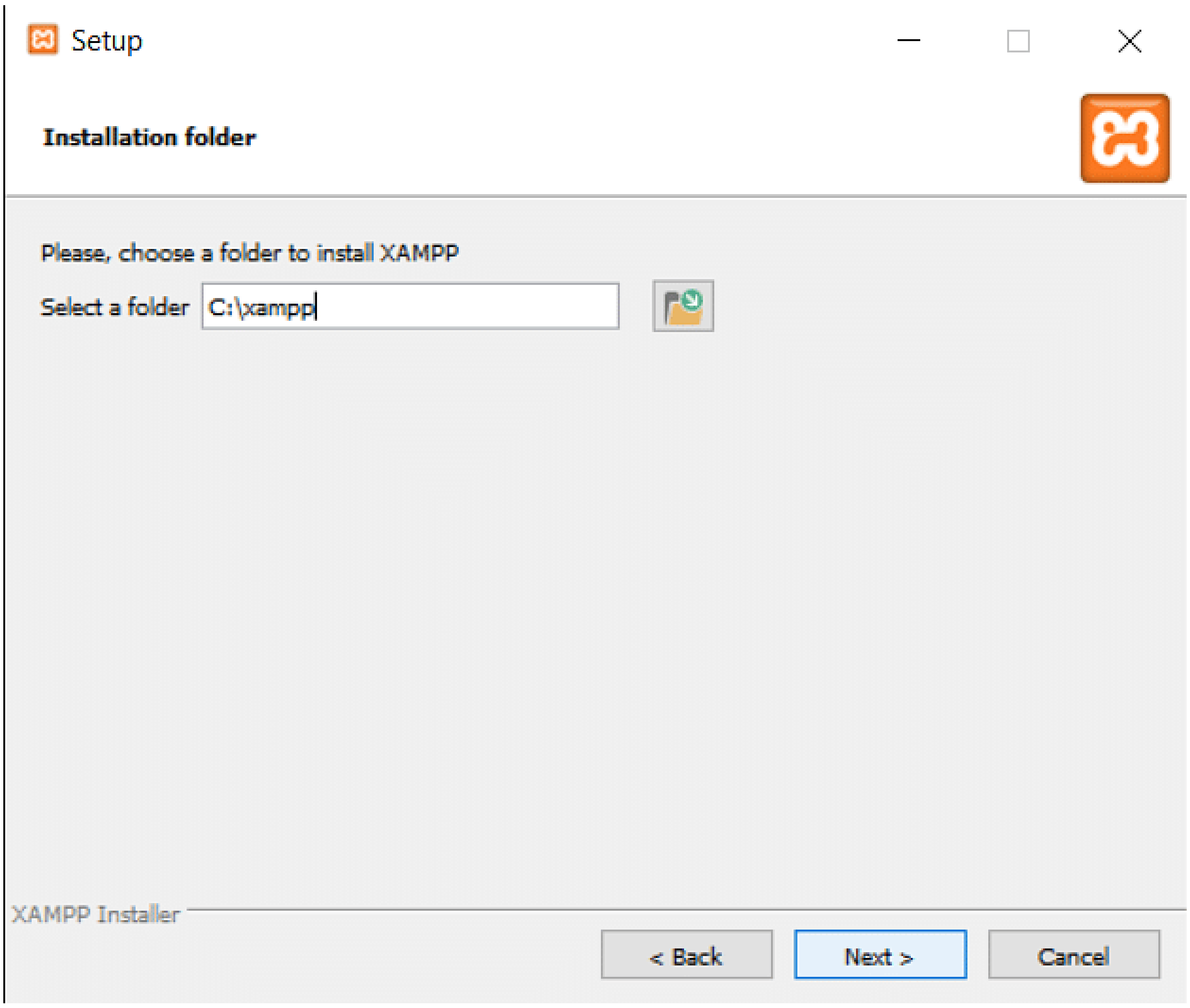
Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

Click on a component to get a detailed description

Next, select the directory where you want the software to be installed. It is recommended that you keep the default directory "C:\xampp" and click on "next" to complete the installation.



Please, choose a folder to install XAMPP

Select a folder C:\xampp

Now that the installation is complete, let's move ahead to see how to run a PHP script using the same.

How to Start a New PHP Project in XAMPP?

- Before you run or start writing any program in PHP, you should start Apache and MYSQL.
- After starting both servers, you have to write a program in Notepad.
- After writing it, save that file as "program.php".
- Then copy that file program.php to C:/Program Files/XAMPP/htdocs.
- Open the browser and type http://localhost.
- Now run your code in that browser.

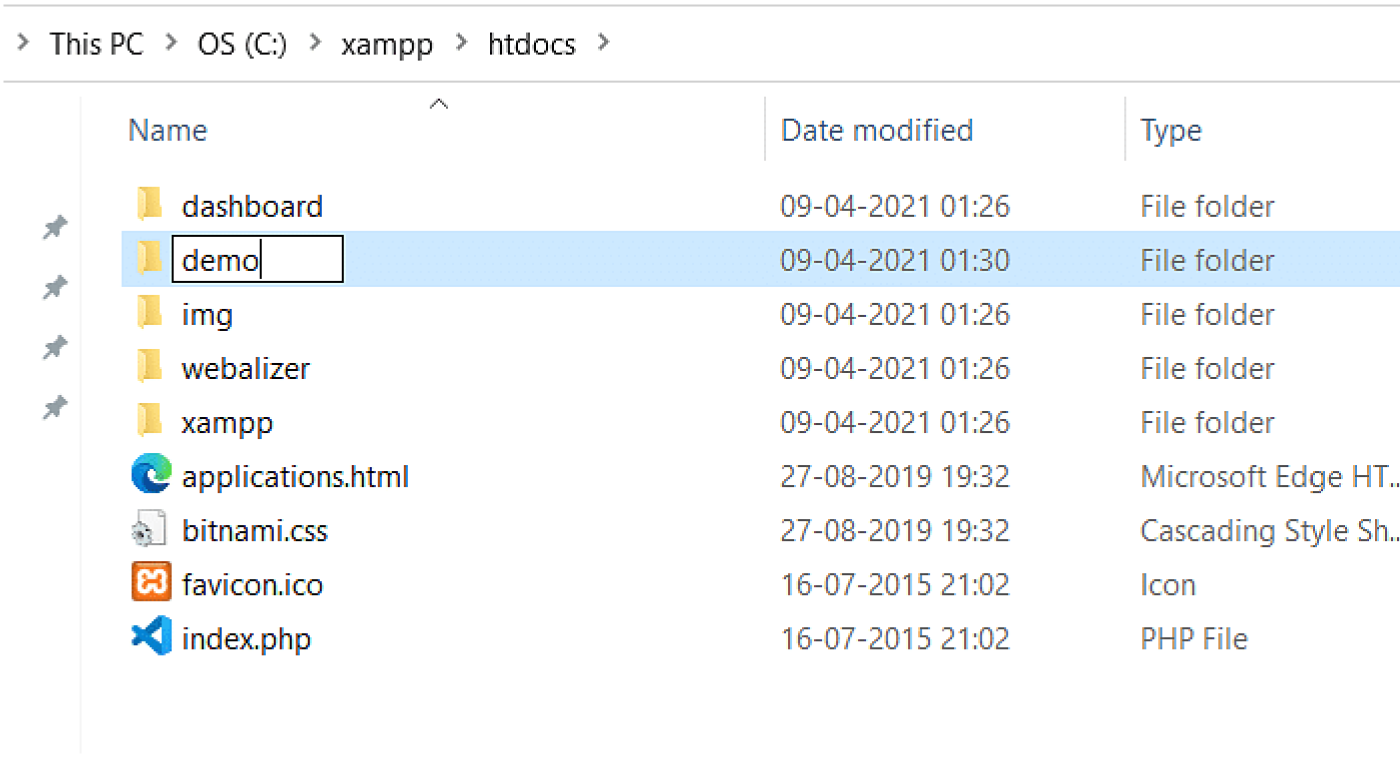
How to Run a PHP Code Using XAMPP?

Before running a PHP script, you must know where to write it.

In the XAMPP directory, there exists a folder called “htdocs”. This is where all the programs for the web pages will be stored.

Now, to run a PHP script:

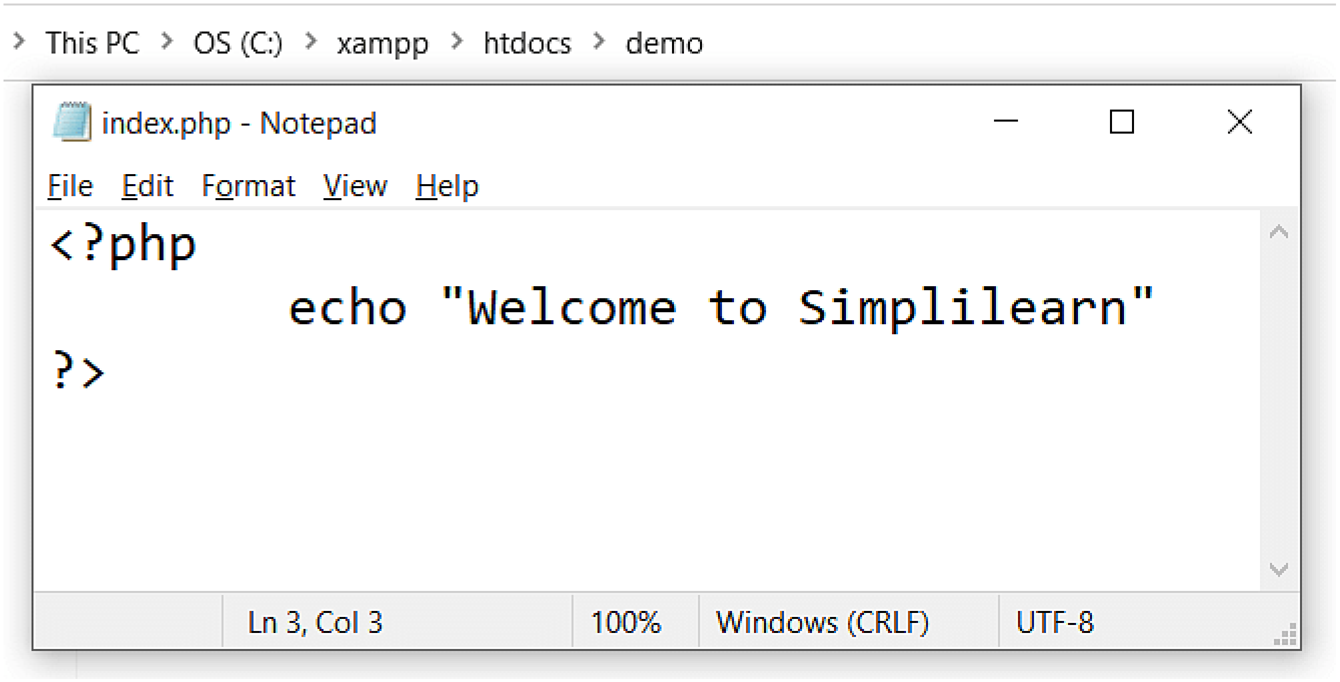
1. Go to “C:\xampp\htdocs” and inside it, create a folder. Let’s call it “demo”. It’s considered good practice to create a new folder for every project you work on.



› This PC › OS (C:) › xampp › htdocs ›

Name	Date modified	Type
dashboard	09-04-2021 01:26	File folder
demo	09-04-2021 01:30	File folder
img	09-04-2021 01:26	File folder
webalizer	09-04-2021 01:26	File folder
xampp	09-04-2021 01:26	File folder
applications.html	27-08-2019 19:32	Microsoft Edge HT..
bitnami.css	27-08-2019 19:32	Cascading Style Sh..
favicon.ico	16-07-2015 21:02	Icon
index.php	16-07-2015 21:02	PHP File

2. Inside the demo folder, create a new text file and name it “index.php” and write the following script.



› This PC › OS (C:) › xampp › htdocs › demo

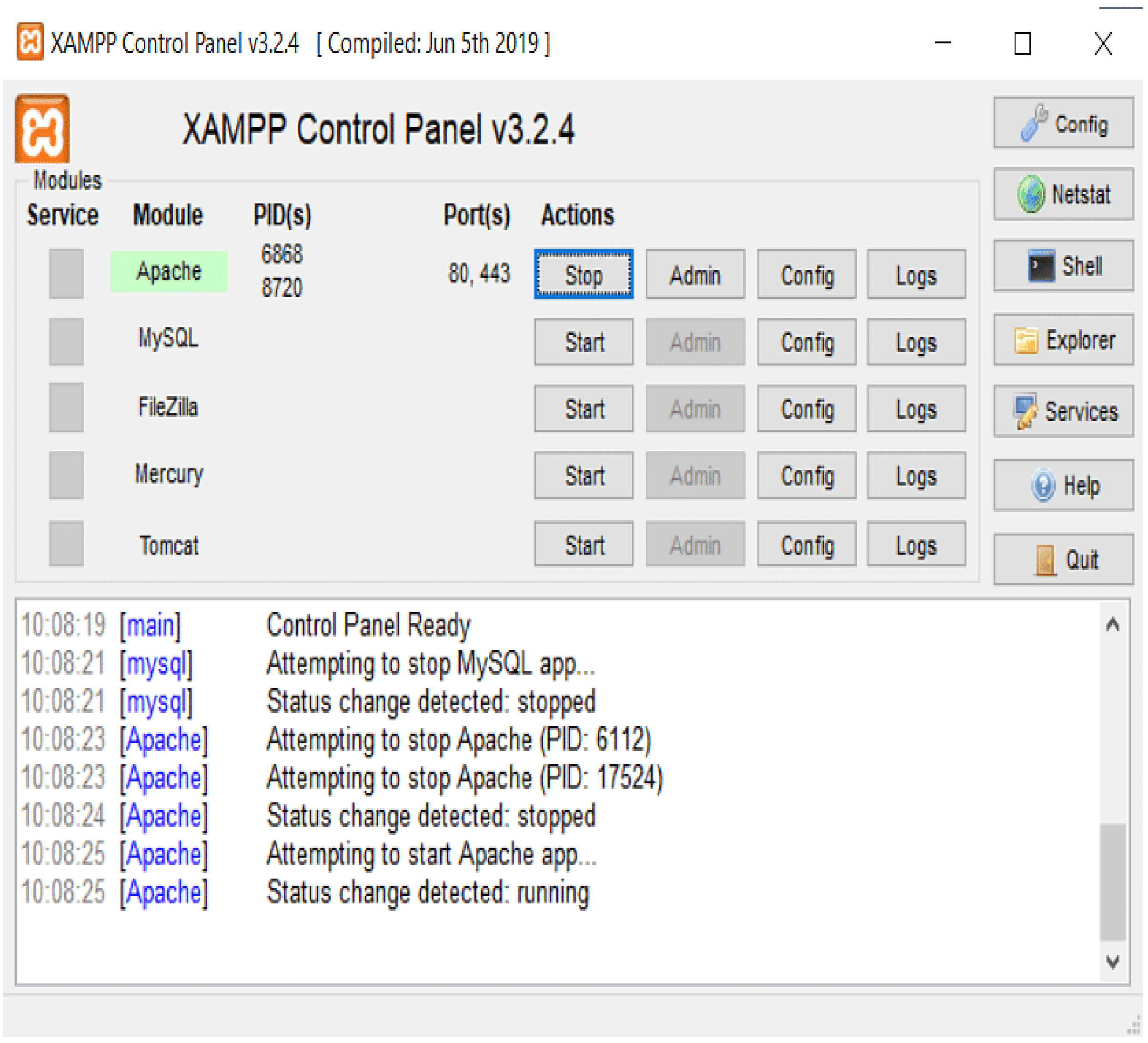
index.php - Notepad

File Edit Format View Help

```php
<?php
        echo "Welcome to Simplilearn"
?>
```

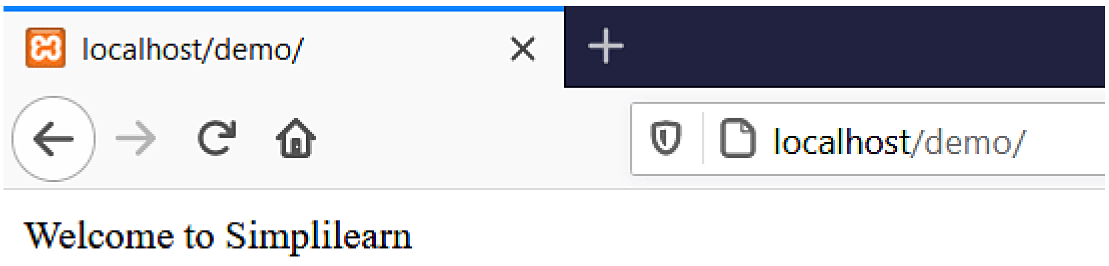
Ln 3, Col 3 | 100% | Windows (CRLF) | UTF-8

3. Now, to see the script output, open the XAMPP control panel and start Apache to host the local webserver, where our script will be running.



```
10:08:19 [main]     Control Panel Ready
10:08:21 [mysql]    Attempting to stop MySQL app...
10:08:21 [mysql]    Status change detected: stopped
10:08:23 [Apache]   Attempting to stop Apache (PID: 6112)
10:08:23 [Apache]   Attempting to stop Apache (PID: 17524)
10:08:24 [Apache]   Status change detected: stopped
10:08:25 [Apache]   Attempting to start Apache app...
10:08:25 [Apache]   Status change detected: running
```

4. Now navigate to your browser and type in “localhost/demo/” in the address bar to view the output.



Welcome to Simplilearn

PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

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
?>
```

The default file extension for PHP files is "`.php`".

A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "`echo`" to output the text "Hello World!" on a web page:

ExampleGet your own PHP Server

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

Try it Yourself »

Note: PHP statements end with a semicolon (`;`).

PHP Case Sensitivity

In PHP, keywords (e.g. `if`, `else`, `while`, `echo`, etc.), classes, functions, and user-defined functions are not case-sensitive.

In the example below, all three echo statements below are equal and legal:

Example

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

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

</body>
</html>
```

Look at 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

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

PHP 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:

<!DOCTYPE html>

```
<html>

<body>


<?php

$txt = "Hello world!";

$x = 5;

$y = 10.5;

echo $txt;

echo "<br>";

echo $x;

echo "<br>";

echo $y;

?>

</body>

</html>
```

```
o/p-Hello world!
5
10.5
```

PHP Variables

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

Rules for PHP variables:

- A variable starts with the `$` sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (`$age` and `$AGE` are two different variables)

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:

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

<?php
$txt = "FY-MCA-I";
echo "Good morning $txt!";
?>

</body>
</html>
```

O/P-Good morning FY-MCA-I!

The following example will output the sum of two variables:

```
<!DOCTYPE html>

<html>

<body>

<?php

$x = 5;

$y = 4;

echo $x + $y;

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

o/p-9

PHP echo and print Statements

With PHP, there are two basic ways to get output: `echo` and `print`.

In this tutorial we use `echo` or `print` in almost every example. So, this chapter contains a little more info about those two output statements.

PHP echo and print 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`.

The PHP 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):

```
<!DOCTYPE html>

<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>
```

o/p-

```
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:

```
<html>

<body>



<?php

$txt1 = "Learn PHP";

$txt2 = "FY-MCXA-II";

$x = 5;

$y = 4;



echo "<h2>" . $txt1 . "</h2>";

echo "Study PHP at " . $txt2 . "<br>";

echo $x + $y;

?>



</body>

</html>
```

O/P-

```
Learn PHP

Study PHP at FY-MCA-II
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):

```
<!DOCTYPE html>

<html>

<body>


<?php

print "<h2>PHP is Fun!</h2>";

print "Hello world!<br>";

print "I'm about to learn PHP!";

?>


</body>

</html>
```

O/P-

```
PHP is Fun!

Hello world!
I'm about to learn PHP!
```

PHP 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:

```
<!DOCTYPE html>

<html>

<body>


<?php

$x = "Hello world!";

$y = 'Hello world!';


echo $x;

echo "<br>";

echo $y;

?>


</body>

</html>
```

O/P-

```
Hello world!

Hello world!
```

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: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<!DOCTYPE html>

<html>

<body>



<?php

$x = 5985;

var_dump($x);

?>



</body>

</html>
```

o/p-

```
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:

```
<!DOCTYPE html>

<html>

<body>


<?php

$x = 10.365;

var_dump($x);

?>


</body>

</html>
```

o/p-

```
float(10.365)
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$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.

In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```
<!DOCTYPE html>

<html>

<body>
```

```
<?php

$cars = array("Volvo","BMW","Toyota");

var_dump($cars);

?>


</body>

</html>
```

O/P-

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

PHP Strings

A string is a sequence of characters, like "Hello world!".

PHP String Functions

In this chapter we will look at some commonly used functions to manipulate strings.

strlen() - Return the Length of a String

The PHP `strlen()` function returns the length of a string.

```
<!DOCTYPE html>

<html>
```

```
<body>


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


</body>
</html>
```

O/P-

```
12
```

str_word_count() - Count Words in a String

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

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


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


</body>
</html>
```

O/P-

```
2
```

strrev() - Reverse a String

The PHP `strrev()` function reverses a string.

```
<!DOCTYPE html>

<html>

<body>


<?php

echo strrev("Hello world!");

?>


</body>

</html>
```

O/P-

```
!dlrow olleH
```

strpos() - Search For a 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 first character position in a string is 0 (not 1).

```
<!DOCTYPE html>

<html>

<body>


<?php

echo strpos("Hello world!", "world");

?>
```

```
</body>

</html>
```

O/P-6

str_replace() - Replace Text Within a String

The PHP `str_replace()` function replaces some characters with some other characters in a string.

```
<!DOCTYPE html>

<html>

<body>


<?php

echo str_replace("world", "Dolly", "Hello world!");

?>


</body>

</html>
```

O/P-

```
Hello Dolly!
```

PHP 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).

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

```
<!DOCTYPE html>

<html>

<body>

<?php

// case-sensitive constant name

define("GREETING", "Welcome to FY-MCA-I!");

echo GREETING;

?>

</body>

</html>
```

O/P-

```
Welcome to FY-MCA-I!
```

```
<!DOCTYPE html>

<html>

<body>


<?php

// case-insensitive constant name
```

```
define("GREETING", "Welcome to FY-MCA-II!", true);

echo greeting;

?>


</body>

</html>
```

O/P-

```
Welcome to FY-MCA-I!
```

Constants are Global

Constants are automatically global and can be used across the entire script.

Example

This example uses a constant inside a function, even if it is defined outside the function:

```
<!DOCTYPE html>

<html>

<body>


<?php

define("GREETING", "Welcome to FY-MCA-II!");

function myTest() {

  echo GREETING;

}


myTest();

?>
```

```
</body>

</html>
```

O/P-

Welcome to FY-MCA-II!!

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
- Conditional assignment 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**	**Show it**
+	Addition	$x + $y	Sum of $x and $y	Try it »
-	Subtraction	$x - $y	Difference of $x and $y	Try it »

*	Multiplication	$x * $y	Product of $x and $y	Try it »
/	Division	$x / $y	Quotient of $x and $y	Try it »
%	Modulus	$x % $y	Remainder of $x divided by $y	Try it »
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power	Try it »

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

<?php
$x = 10;
$y = 6;

echo $x + $y;
?>

</body>
</html>
```

O/P-16

PHP Assignment Operators

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

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result	Show

				it
==	Equal	$x == $y	Returns true if $x is equal to $y	Try it »
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type	Try it »
!=	Not equal	$x != $y	Returns true if $x is not equal to $y	Try it »
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y	Try it »
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type	Try it »
>	Greater than	$x > $y	Returns true if $x is greater than $y	Try it »
<	Less than	$x < $y	Returns true if $x is less than $y	Try it »
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y	Try it »
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y	Try it »
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.	Try it »

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	Show it
++$x	Pre-increment	Increments $x by one, then returns $x	Try it »
$x++	Post-increment	Returns $x, then increments $x by one	Try it »
--$x	Pre-decrement	Decrements $x by one, then returns $x	Try it »
$x--	Post-decrement	Returns $x, then decrements $x by one	Try it »

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result	Show it

and	And	$x and $y	True if both $x and $y are true	Try it »
or	Or	$x or $y	True if either $x or $y is true	Try it »
xor	Xor	$x xor $y	True if either $x or $y is true, but not both	Try it »
&&	And	$x && $y	True if both $x and $y are true	Try it »
||	Or	$x || $y	True if either $x or $y is true	Try it »
!	Not	!$x	True if $x is not true	Try it »

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result	Show it
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2	Try it »
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1	Try it »

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result	Show it
+	Union	$x + $y	Union of $x and $y	Try it »
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs	Try it »
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types	Try it »
!=	Inequality	$x != $y	Returns true if $x is not equal to $y	Try it »
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y	Try it »
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y	Try it »

PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	Result	Show it

?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE	Try it »
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7	Try it »

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

PHP - 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;
}
```

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

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

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

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

</body>
</html>
```

O/P-

```
Have a good day!
```

PHP - 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;
}
```

Example

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<!DOCTYPE html>

<html>

<body>


<?php

$t = date("H");


if ($t < "20") {

  echo "Have a good day!";

} else {

  echo "Have a good night!";

}

?>


</body>

</html>
```

O/P- Have a good day!

PHP - The if...elseif...else Statement

The `if...elseif...else` statement executes different codes for more than two conditions.

Syntax

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

Example

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!":

```
<!DOCTYPE html>

<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>
```

O/P- The hour (of the server) is 10, and will give the following message:
Have a good day!

PHP switch Statement

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

The PHP switch Statement

Use the `switch` statement to **select one of many blocks of code to be executed**.

Syntax

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

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.

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

O/P- Your favorite color is red!

PHP Loops

In the following chapters you will learn how to repeat code by using loops in PHP.

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- `while` - loops through a block of code as long as the specified condition is true
- `do...while` - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- `for` - loops through a block of code a specified number of times
- `foreach` - loops through a block of code for each element in an array

The following chapters will explain and give examples of each loop type.

PHP while Loop

The `while` loop - Loops through a block of code as long as the specified condition is true.

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;
}
```

Examples

The example below displays the numbers from 1 to 5:

```
<!DOCTYPE html>
```

```
<html>

<body>


<?php

$x = 1;


while($x <= 5) {

  echo "The number is: $x <br>";

  $x++;

}

?>


</body>

</html>
```

O/p-
```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

Example Explained

- `$x = 1;` - Initialize the loop counter ($x), and set the start value to 1
- `$x <= 5` - Continue the loop as long as $x is less than or equal to 5
- `$x++;` - Increase the loop counter value by 1 for each iteration

PHP do while Loop

The `do...while` loop - Loops through a block of code once, and then repeats the loop as long as the specified condition is true.

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);
```

Examples

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:

```
<!DOCTYPE html>

<html>

<body>



<?php

$x = 1;



do {

  echo "The number is: $x <br>";

  $x++;

} while ($x <= 5);

?>



</body>

</html>
```

O/P- The number is: 1
The number is: 2
The number is: 3

```
The number is: 4
The number is: 5
```

PHP for Loop

The `for` loop - Loops through 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 for each iteration;
}
```

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

Examples

The example below displays the numbers from 0 to 10:

```
<!DOCTYPE html>

<html>

<body>

<?php

for ($x = 0; $x <= 10; $x++) {

  echo "The number is: $x <br>";

}

?>

</body>

</html>
```

```
O/P- 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
```

Example Explained

- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 10; - Continue the loop as long as $x is less than or equal to 10
- $x++ - Increase the loop counter value by 1 for each iteration

PHP foreach Loop

The `foreach` loop - Loops through a block of code for each element in an array.

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.

Examples

The following example will output the values of the given array ($colors):

```
<!DOCTYPE html>

<html>

<body>
```

```
<?php

$colors = array("red", "green", "blue", "yellow");


foreach ($colors as $value) {

  echo "$value <br>";

}

?>


</body>

</html>
```

O/P- red
green
blue
yellow

PHP Break and Continue

PHP Break

You have already seen the `break` statement used in an earlier chapter of this tutorial. It was used to "jump out" of a `switch` statement.

The `break` statement can also be used to jump out of a loop.

This example jumps out of the loop when **x** is equal to **4**:

```
<!DOCTYPE html>

<html>

<body>


<?php

for ($x = 0; $x < 10; $x++) {

  if ($x == 4) {
    break;

  }

  echo "The number is: $x <br>";

}

?>



</body>

</html>
```

O/P- The number is: 0

```
The number is: 1
The number is: 2
The number is: 3
```

PHP Continue

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of **4**:

```
<!DOCTYPE html>

<html>

<body>



<?php

for ($x = 0; $x < 10; $x++) {

  if ($x == 4) {

    continue;

  }

  echo "The number is: $x <br>";

}

?>
```

```
</body>

</html>
```

O/P- The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9

PHP Functions

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

Please check out our PHP reference for a complete overview of the PHP built-in functions.

PHP User Defined Functions

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

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

Create a User Defined Function in PHP

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

Syntax

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

Note: A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

```
<!DOCTYPE html>

<html>

<body>



<?php

function writeMsg() {

  echo "Hello world!";

}



writeMsg();

?>



</body>

</html>
```

O/P-

```
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:

```
<!DOCTYPE html>

<html>

<body>
```

```
<?php

function familyName($fname) {

  echo "$fname ss.<br>";

}


familyName("Jani");

familyName("Hege");

familyName("Stale");

familyName("Kai Jim");

familyName("Borge");

?>


</body>

</html>
```

O/P-
```
Jani Refsnes.
Hege Refsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.
```

PHP Return Type Declarations

PHP 7 also supports Type Declarations for the `return` statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

To declare a type for the function return, add a colon (`:`) and the type right before the opening curly (`{`)bracket when declaring the function.

In the following example we specify the return type for the function:

```
<?php declare(strict_types=1); // strict requirement

function addNumbers(float $a, float $b) : float {

  return $a + $b;

}
```

```
echo addNumbers(1.2, 5.2);

?>

O/P-6.4
```

Passing Arguments by Reference

In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.

When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the & operator is used:

```
<!DOCTYPE html>

<html>

<body>


<?php

function add_five(&$value) {

  $value += 5;

}


$num = 2;

add_five($num);

echo $num;

?>


</body>

</html>

O/P-7
```

PHP File Handling

PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

PHP Open File - fopen()

The PHP fopen() function is used to open a file.

Syntax

1. resource fopen (string $filename , string $mode [, bool $use_include_path = false [, resource $context]])

Example

PHP Open File Mode

Mode	Description
R	Opens file in **read-only** mode. It places the file pointer at the beginning of the file.
r+	Opens file in **read-write** mode. It places the file pointer at the beginning of the file.
w	Opens file in **write-only** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
w+	Opens file in **read-write** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
A	Opens file in **write-only** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.

a+	Opens file in **read-write** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
X	Creates and opens file in **write-only** mode. It places the file pointer at the beginning of the file. If file is found, fopen() function returns FALSE.
x+	It is same as x but it creates and opens file in **read-write** mode.
C	Opens file in **write-only** mode. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to 'w'), nor the call to this function fails (as is the case with 'x'). The file pointer is positioned on the beginning of the file
c+	It is same as c but it opens file in **read-write** mode.

PHP Open File Example

```
<?php
$handle = fopen("c:\\folder\\file.txt", "r");
?>
```

PHP Close File - fclose()

The PHP fclose() function is used to close an open file pointer.

Syntax

```
fclose ( resource $handle )
```

Example

```
<?php
fclose($handle);
?>
```

PHP Read File - fread()

The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

Syntax

1. string fread (resource $handle , int $length)

Example

```
<?php
$filename = "c:\\myfile.txt";
$handle = fopen($filename, "r");//open file in read mode

$contents = fread($handle, filesize($filename));//read file

echo $contents;//printing data of file
fclose($handle);//close file
?>
```

Output

```
hello php file
```

PHP Write File - fwrite()

The PHP fwrite() function is used to write content of the string into file.

Syntax

1. int fwrite (resource $handle , string $string [, int $length])

Example

```
<?php
$fp = fopen('data.txt', 'w');//open file in write mode
fwrite($fp, 'hello ');
fwrite($fp, 'php file');
fclose($fp);

echo "File written successfully";
?>
```

Output

```
File written successfully
```

PHP Delete File - unlink()

The PHP unlink() function is used to delete file.

Syntax

1. bool unlink (string $filename [, resource $context])

Example

```
<?php
```

```
unlink('data.txt');

echo "File deleted successfully";
?>
```

PHP File Upload

PHP allows you to upload single and multiple files through few lines of code only.

PHP file upload features allow you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

PHP $_FILES

The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

Here, we are assuming that file name is *filename*.

$_FILES['filename']['name']

returns file name.

$_FILES['filename']['type']

returns MIME type of the file.

$_FILES['filename']['size']

returns size of the file (in bytes).

$_FILES['filename']['tmp_name']

returns temporary file name of the file which was stored on the server.

$_FILES['filename']['error']

returns error code associated with this file.

move_uploaded_file() function

The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.

Syntax

bool move_uploaded_file (string $filename , string $destination)

PHP File Upload Example

File: uploadform.html

```
<form action="uploader.php" method="post" enctype="multipart/form-data">
    Select File:
    <input type="file" name="fileToUpload"/>
    <input type="submit" value="Upload Image" name="submit"/>
</form>
```

File: uploader.php

```
<?php
$target_path = "e:/";
$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);
  if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)) {
    echo "File uploaded successfully!";
} else{
    echo "Sorry, file not uploaded, please try again!";
}
?>
```

https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/HTML5-PHP-File-Upload-Example-Apache-Server

PHP Download File

PHP enables you to download file easily using built-in readfile() function. The readfile() function reads a file and writes it to the output buffer.

PHP readfile() function

Syntax

1. int readfile (string $filename [, bool $use_include_path = false [, resource $context]])

$filename: represents the file name

$use_include_path: it is the optional parameter. It is by default false. You can set it to true to the search the file in the included_path.

$context: represents the context stream resource.

int: it returns the number of bytes read from the file.

PHP Download File Example: Text File

File: download1.php

```
<?php

$file_url = 'http://www.javatpoint.com/f.txt';

header('Content-Type: application/octet-stream');

header("Content-Transfer-Encoding: utf-8");

header("Content-disposition: attachment; filename=\"" .
basename($file_url) . "\"");

readfile($file_url);

?>
```

PHP Download File Example: Binary File

File: download2.php

```
<?php

$file_url = 'http://www.myremoteserver.com/file.exe';

header('Content-Type: application/octet-stream');

header("Content-Transfer-Encoding: Binary");

header("Content-disposition: attachment; filename=\"" .
basename($file_url) . "\"");

readfile($file_url);

?>
```

PHP Cookies

What are Cookies?

A cookie is a small file with the maximum size of 4KB that the web server stores on the client computer. They are typically used to keeping track of information such as a username that the site can retrieve to personalize the page when the user visits the website next time. A cookie can only be read from the domain that it has been issued from. Cookies are usually set in an HTTP header but JavaScript can also set a cookie directly on a browser.

Setting Cookie In PHP: To set a cookie in PHP,the **setcookie()** function is used.The setcookie() function needs to be called prior to any output generated by the script otherwise the cookie will not be set.
Syntax :
setcookie(name, value, expire, path, domain, security);

Parameters: The setcookie() function requires six arguments in general which are:
1. **Name:** It is used to set the name of the cookie.
2. **Value:** It is used to set the value of the cookie.
3. **Expire:** It is used to set the expiry timestamp of the cookie after which the cookie can't be accessed.
4. **Path:** It is used to specify the path on the server for which the cookie will be available.
5. **Domain:** It is used to specify the domain for which the cookie is available.
6. **Security:** It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.

Below are some operations that can be performed on Cookies in PHP:

- **Creating Cookies**: Creating a cookie named Auction_Item and assigning the value Luxury Car to it.The cookie will expire after 2 days(2 days * 24 hours * 60 mins * 60 seconds).

```
<?php

setcookie("Auction_Item", "Luxury Car", time()+2*24*60*60);

?>
```

Checking Whether a Cookie Is Set Or Not: It is always advisable to check whether a cookie is set or not before accessing its value.Therefore to check whether a cookie is set or not, the PHP isset() function is used.
To check whether a cookie "Auction_Item" is set or not,the isset() function is executed as follows:

```
<?php

if(isset($_COOKIE["Auction_Item"])){

        echo "Auction Item is a " . $_COOKIE["Auction_Item"];

} else{

        echo "No items for auction.";

}

?>
```

Output:
Auction Item is a Luxury Car.

Accessing Cookie Values: For accessing a cookie value, the PHP $_COOKIE superglobal variable is used.It is an associative array that contains a record of all the cookies values sent by the browser in the current request.The records are stored as a list where cookie name is used as the key.
To access a cookie named “Auction_Item”,the following code can be executed:

```
<?php

echo "Auction Item is a " . $_COOKIE["Auction_Item"];

?>
```

Output:

```
Auction Item is a Luxury Car.
```

Deleting Cookies: The setcookie() function can be used to delete a cookie.For deleting a cookie, the setcookie() function is called by passing the cookie name and other arguments or empty strings but however this time, the expiration date is required to be set in the past.
To delete a cookie named “Auction_Item”,the following code can be executed:

```
<?php

setcookie("Auction_Item", "", time()-60);



?>
```

PHP | Sessions

What is a session?

In general, session refers to a frame of communication between two medium. A PHP session is used to store data on a server rather than the computer of the user. Session identifiers or SID is a unique number which is used to identify every user in a session based environment. The SID is used to link the user with his information on the server like posts, emails etc.

How are sessions better than cookies?

Although cookies are also used for storing user related data, they have serious security issues because cookies are stored on the user’s computer and thus they are open to attackers to easily modify the content of the cookie.Addition of harmful data by the attackers in the cookie may result in the breakdown of the application.
Apart from that cookies affect the performance of a site since cookies send the user data each time the user views a page.Every time the browser requests a URL to the server, all the cookie data for that website is automatically sent to the server within the request.

Below are different steps involved in PHP sessions:

Starting a PHP Session: The first step is to start up a session. After a session is started, session variables can be created to store information. The

PHP **session_start()** function is used to begin a new session.It als creates a new session ID for the user.

Below is the PHP code to start a new session:

```
<?php

session_start();

?>
```

Storing Session Data: Session data in key-value pairs using the **$_SESSION[]** superglobal array.The stored data can be accessed during lifetime of a session.
Below is the PHP code to store a session with two session variables Rollnumber and Name:

```
<?php

session_start();

$_SESSION["Rollnumber"] = "11";

$_SESSION["Name"] = "Ajay";

?>
```

Accessing Session Data: Data stored in sessions can be easily accessed by firstly calling **session_start()** and then by passing the corresponding key to the **$_SESSION** associative array.
The PHP code to access a session data with two session variables Rollnumber and Name is shown below:

```
<?php

session_start();

echo 'The Name of the student is :' .
$_SESSION["Name"] . '<br>';

echo 'The Roll number of the student is :' .
$_SESSION["Rollnumber"] . '<br>';

?>
```

Output:

```
The Name of the student is :Ajay

The Roll number of the student is :11
```

Destroying Certain Session Data: To delete only a certain session data,the unset feature can be used with the corresponding session variable in the **$_SESSION** associative array.
The PHP code to unset only the “Rollnumber” session variable from the associative session array:

```
<?php

session_start();

if(isset($_SESSION["Name"])){

    unset($_SESSION["Rollnumber"]);

}

?>
```

Destroying Complete Session: The **session_destroy()** function is used to completely destroy a session. The session_destroy() function does not require any argument.

```
<?php

  session_start();

session_destroy();

  ?>
```

PHP Form Handling

The PHP superglobals $_GET and $_POST are used to collect form-data.

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

Example

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

```
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

The output could be something like this:

```
Welcome ss
Your email address is: ss@gmail.com
```

PHP Form Validation

The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

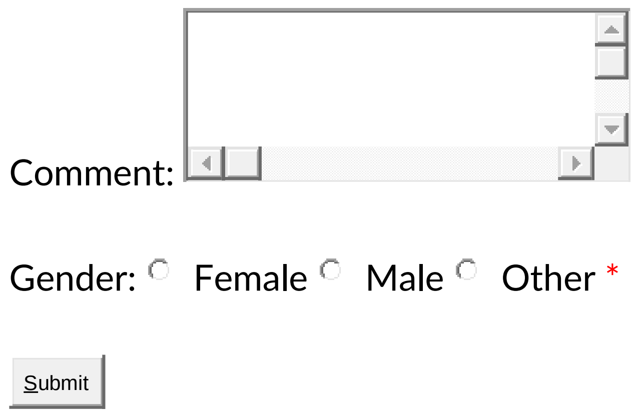
PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:


Comment:

Gender: ○ Female ○ Male ○ Other *

Submit

Your Input:

The validation rules for the form above are as follows:

Field	**Validation Rules**
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"
>
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Big Note on PHP Form Security

The $_SERVER["PHP_SELF"] variable can be used by hackers!

If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

Validate Form Data With PHP

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

Example

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
```

```
?>

<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"
>
  Name: <input type="text" name="name">
  <br><br>
  E-mail: <input type="text" name="email">
  <br><br>
  Website: <input type="text" name="website">
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>
```

Notice that at the start of the script, we check whether the form has been submitted using $_SERVER["REQUEST_METHOD"]. If the REQUEST_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.

However, in the example above, all input fields are optional. The script works fine even if the user does not enter any data.

The next step is to make input fields required and create error messages if needed.

PHP Forms - Required Fields

PHP - Required Fields

From the validation rules table on the previous page, we see that the "Name", "E-mail", and "Gender" fields are required. These fields cannot be empty and must be filled out in the HTML form.

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

In the previous chapter, all input fields were optional.

In the following code we have added some new variables: $nameErr, $emailErr, $genderErr, and $websiteErr. These error variables will hold error messages for the required fields. We have also added an if else statement for each $_POST variable. This checks if the $_POST variable is empty (with the PHP empty() function). If it is empty, an error message is stored in the different error variables, and if it is not empty, it sends the user input data through the test_input() function:

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }
  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

PHP - Display The Error Messages

Then in the HTML form, we add a little script after each required field, which generates the correct error message if needed (that is if the user tries to submit the form without filling out the required fields):

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }
  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"
>
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

The next step is to validate the input data, that is "Does the Name field contain only letters and whitespace?", and "Does the E-mail field contain a valid e-mail address syntax?", and if filled out, "Does the Website field contain a valid URL?".

PHP Forms - Validate E-mail and URL

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters, dashes, apostrophes and whitespaces. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

The preg_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter_var() function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

PHP - Validate Name, E-mail, and URL

Now, the script looks like this:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"
>
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

PHP Complete Form Example

PHP - Keep The Values in The Form

To show the values in the input fields after the user hits the submit button, we add a little PHP script inside the value attribute of the following input fields: name, email, and website. In the comment textarea field, we put the script between the <textarea> and </textarea> tags. The little script outputs the value of the $name, $email, $website, and $comment variables.

Then, we also need to show which radio button that was checked. For this, we must manipulate the checked attribute (not the value attribute for radio buttons):

```
Name: <input type="text" name="name" value="<?php echo $name;?>">

E-mail: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></t
extarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="other") echo "checked";?>
value="other">Other
```

PHP - Complete Form Example

Here is the complete code for the PHP Form Validation Example:

Example

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
      if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $emailErr = "Invalid email format";
      }
    }

    if (empty($_POST["website"])) {
      $website = "";
    } else {
      $website = test_input($_POST["website"]);
      // check if URL address syntax is valid (this regular expression
also allows dashes in the URL)
      if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
        $websiteErr = "Invalid URL";
      }
    }

    if (empty($_POST["comment"])) {
      $comment = "";
    } else {
      $comment = test_input($_POST["comment"]);
    }

    if (empty($_POST["gender"])) {
```

```
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_S
ELF"]);?>">
  Name: <input type="text" name="name" value="<?php echo $name;?>">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email" value="<?php echo $email;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website" value="<?php echo $website
;?>">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comm
ent;?></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="female") echo "checked";?> value="female">Female
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="male") echo "checked";?> value="male">Male
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="other") echo "checked";?> value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```