Website

Website is a collection of related web pages that may contain text, images, audio and video. The first page of a website is called home page. Each website has specific internet address (URL) that you need to enter in your browser to access a website.

Website is hosted on one or more servers and can be accessed by visiting its homepage using a computer network. A website is managed by its owner that can be an individual, company or an organization.

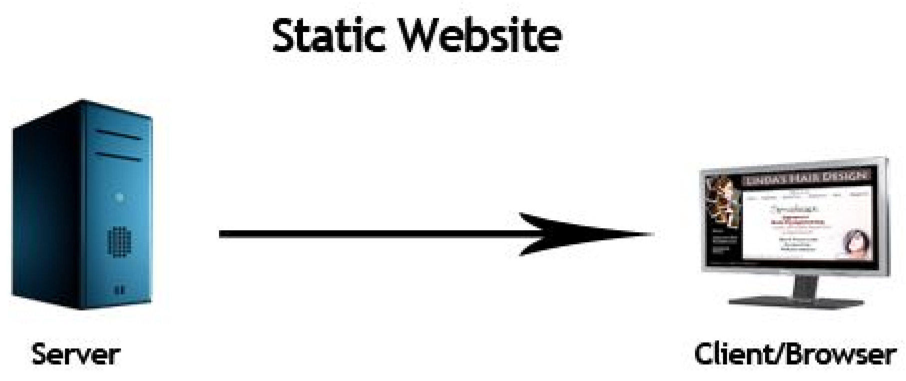
A website can be of two types:

- Static Website
- Dynamic Website

Static website

Static website is the basic type of website that is easy to create. You don't need the knowledge of web programming and database design to create a static website. Its web pages are coded in HTML.

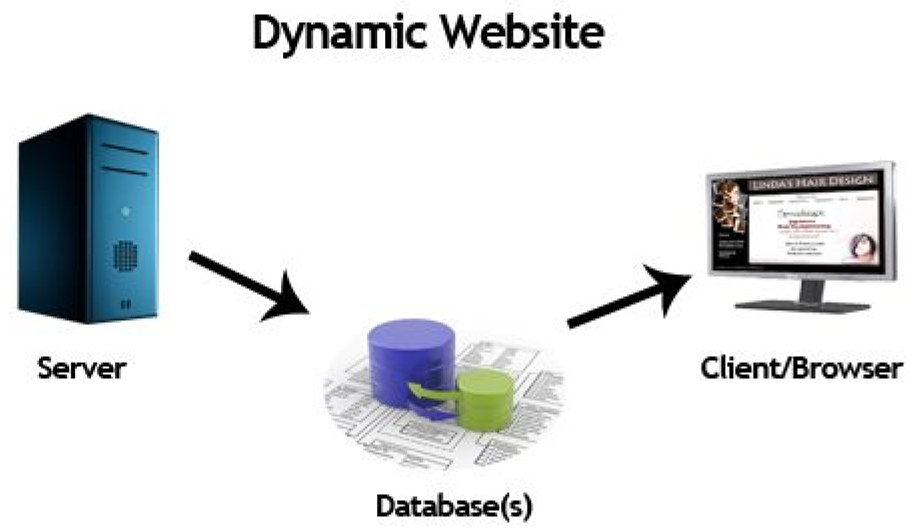
The codes are fixed for each page so the information contained in the page does not change and it looks like a printed page



Dynamic website

Dynamic website is a collection of dynamic web pages whose content changes dynamically. It accesses content from a database or Content Management System (CMS). Therefore,

when you alter or update the content of the database, the content of the website is also altered or updated.
Dynamic website uses client-side scripting or server-side scripting, or both to generate dynamic content.
Client side scripting generates content at the client computer on the basis of user input. The web browser downloads the web page from the server and processes the code within the page to render information to the user.
In server side scripting, the software runs on the server and processing is completed in the server then plain pages are sent to the user.



Static vs Dynamic website

Static Website	Dynamic Website
Prebuilt content is same every time the page is loaded.	Content is generated quickly and changes regularly.
It uses the **HTML** code for developing a website.	It uses the server side languages such as **PHP,SERVLET, JSP, and ASP.NET**etc. for developing a website.
It sends exactly the same response for every request.	It may generate different HTML for each of the request.

The content is only changed when someone publishes and updates the file (sends it to the web server).	The page contains "server-side" code which allows the server to generate the unique content when the page is loaded.
Flexibility is the main advantage of static website.	Content Management System (CMS) is the main advantage of dynamic website.

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:

Example

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

Note: 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

PHP supports several ways of commenting:

Example

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

<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
```

PHP Case Sensitivity

In PHP, NO keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are case-sensitive.
In the example below, all three echo statements below are legal (and equal):

Example

```
<!DOCTYPE html>
```

```
<html>

<body>

<?php

ECHO "Hello World!<br>";

echo "Hello World!<br>";

EcHo "Hello World!<br>";

?>

</body>

</html>
```

```
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>
```

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

```
<!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 5 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
$txt = "Hello world!";
$x = 5;
$y = 10.5;
echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>
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.

Think of variables as containers for storing data.

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)

Remember that PHP variable names are case-sensitive!

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

```
<?php

$txt = "W3Schools.com";

echo "I love $txt!";

?>
```

Output:-

```
I love W3Schools.com!
```

The following example will produce the same output as the example above:

Example

```
<?php

$txt = "W3Schools.com";

echo "I love " . $txt . "!";

?>
```

Output:-

```
I love W3Schools.com!
```

The following example will output the sum of two variables:

Example

```
<?php

$x = 5;
```

Output:- 9

```
$y = 4;

echo $x + $y;

?>
```

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

```
<?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>";

?>
```

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

```
<?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>";

?>
```

Output :-

```
Variable x inside function is: 5

Variable x outside function is:
```

PHP 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

```
<?php

$x = 5;

$y = 10;

function myTest() {

    global $x, $y;

    $y = $x + $y;

}

myTest();

echo $y;                                             // outputs 15

?>
```

PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index*holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this:

Example

```
<?php

$x = 5;

$y = 10;

function myTest() {

    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];

}

myTest();

echo $y;                                       // outputs 15

?>
```

PHP 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

function myTest() {

    static $x = 0;
    echo $x;

     $x++;

}

myTest();

echo "<br>";

myTest();

echo "<br>";

myTest();

?>

Output :-

0

1

2
```

PHP echo and print Statements

`echo` and `print` are more or less the same. They are both used to output data to the screen.

ECHO

1. echo is a statement i.e used to display the output. it can be used with parentheses echo or without parentheses echo.
2. echo can pass multiple string separated as (,)
3. echo doesn't return any value
4. echo is faster then print

PRINT

1. Print is also a statement i.e used to display the output. it can be used with parentheses print( ) or without parentheses print.
2. using print can doesn’t pass multiple argument
3. print always return 1
4. it is slower than echo

The PHP echo Statement

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

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

Example

```
<?php

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

echo "Hello world!<br>";

echo "I'm about to learn PHP!<br>";

echo "This ", "string ", "was ", "made ", "with multiple parameters.";

?>

Output:-
```

PHP is Fun!

```
Hello world!

I'm about to learn PHP!

This string was made with multiple parameters.
```

The PHP print Statement

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

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

Example

```
<?php

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

print "Hello world!<br>";

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

?>
```

Output:-

```
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
- NULL

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

```
<?php

$x = "Hello world!";

$y = 'Hello world!';

echo $x;

echo "<br>";

echo $y;
```

```
?>
```

```
Output:-
```

```
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 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:

```
<html>

<body>

<?php

$x = 10.365;

var_dump($x);

?>

</body>

</html>
```

Output :-

```
float(10.365)
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
```

```
$y = false;
```

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:

```
<html>

<body>

<?php

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

var_dump($cars);       //This function displays structured information about one or more expressions that includes its type and value.

?>

</body>

</html>
```

Output:-

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

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.

Tip: 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:

```
<html>
```

```
<body>

<?php

$x = "Hello world!";

$x = null;

var_dump($x);

?>

</body>

</html>
```

Output :-

```
NULL
```

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:

```
<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:

```
<html>

<body>

<?php

// case-insensitive constant name

define("GREETING", "Welcome to W3Schools.com!", true);

echo greeting;

?>

</body>

</html>
```

Output :-

```
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:

```
<html>

<body>

<?php

define("GREETING", "Welcome to W3Schools.com!");
//defines a named constant

function myTest() {

    echo GREETING;

}

myTest();

?>

</body>

</html>
```

Output :-

```
Welcome to W3Schools.com!
```

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

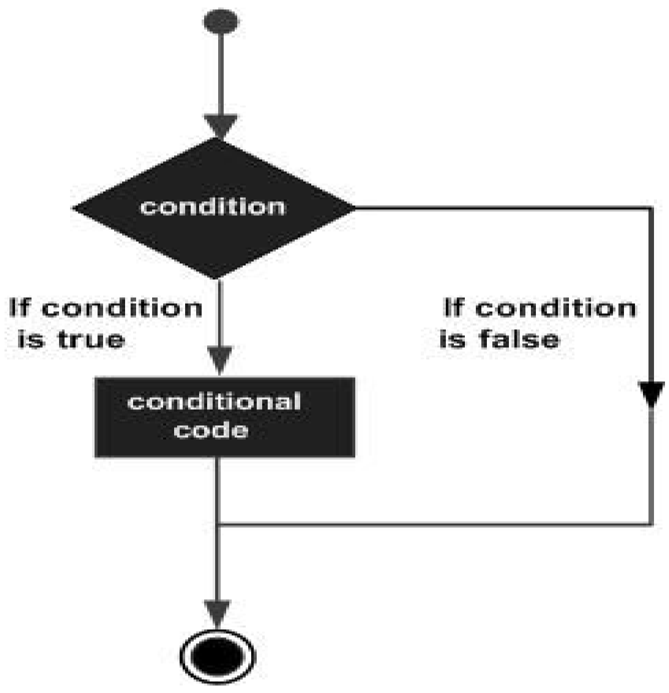
PHP Conditional Statements

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

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:-

```
<html>

<body>

<?php

$t = date("H");

if ($t < "20") {

    echo "Have a good day!";

}

?>

</body>

</html>
```

Output:-

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

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 :-

```
<html>

<body>

<?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!
```

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 this condition is true;

}

else

{

    code to be executed if all conditions are false;

}
```

Example :-

```
<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 02, and will give the following message:

Have a good morning!
```

PHP 5 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.

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!
```