Web Server

Web server can be referred to either the hardware (the machine) or the software that helps to deliver content that can be accessed on the web.

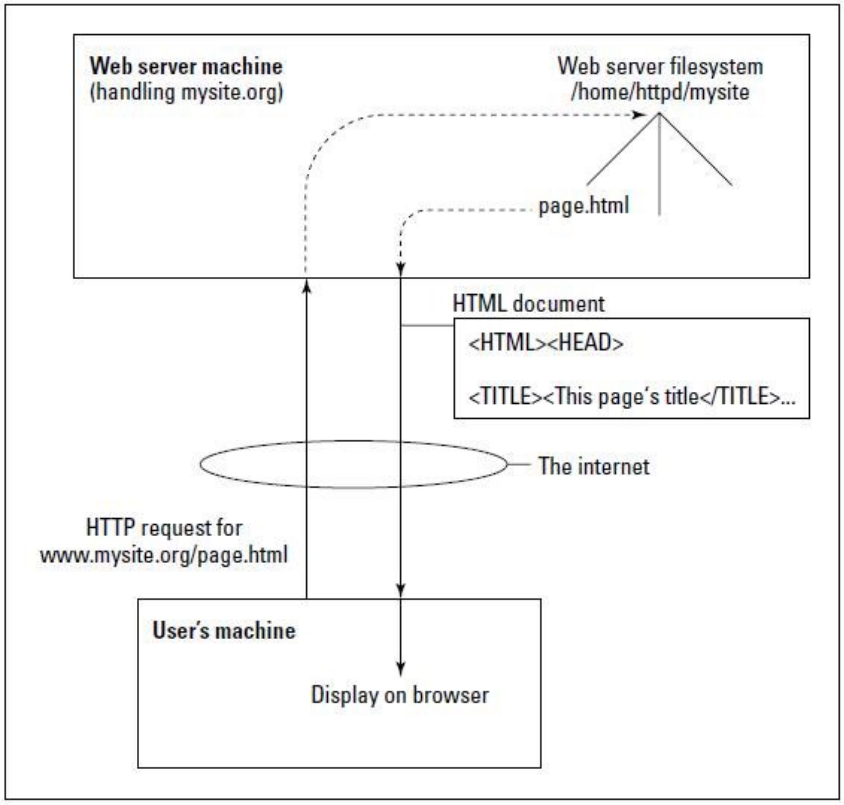
A web server can be a set of machine(s) of any hardware/software combination.

Each web server is identified with a unique IP address and is available round the clock every day.

A software called web server application is installed on these machines.

Web pages we develop are to be uploaded to a web server. Only then it will be accessible to the public thru www.

After a client computer makes an HTTP request for a page from the server machine across the Web, as shown in Figure 1, the server simply passes that file to the client. Figure 1: A simple HTTP request and response



Two leading web server software applications are Apache and Internet Information Server (IIS).

Role of web servers:

- Create websites.
- Configure log file settings.
- Configure website/directory security.
- Create FTP site.
- Configure custom error pages.
- Deliver web pages requested by the client.

Server Side Scripts

- We have already learned that static web pages can only be displayed by the browser at the client side – no modification can be done to the page based on an action taken by the user.
- Dynamic pages bring this functionality. We understood how client side scripting languages like JavaScript makes our pages dynamic.
- But the client-side technologies have few limitations:
    - They depend entirely on the browser.
    - Wide variations exist in the capabilities of each browser and even among versions of the same brand of browser.
    - Individuals can also choose to configure their own browsers by disabling JavaScript for security reasons. Hence for such users the functionalities provided by JavaScript can be availed.
    - Many consumers are very slow to upgrade their browsers for reasons of cost or technical anxiety or both.
    - Client-side technologies cannot do anything that requires connecting to a back end server.
    - Client side script is visible to the public.
- Figure 2 shows a schematic representation of a server-side scripting data flow.
- Server-side Web scripting is mostly about connecting Web sites to back end servers, such as databases. This enables the following types of two-way communication:
    - Server to client: Web pages can be assembled from back end-server output.
    - Client to server: Customer-entered information can be acted upon.

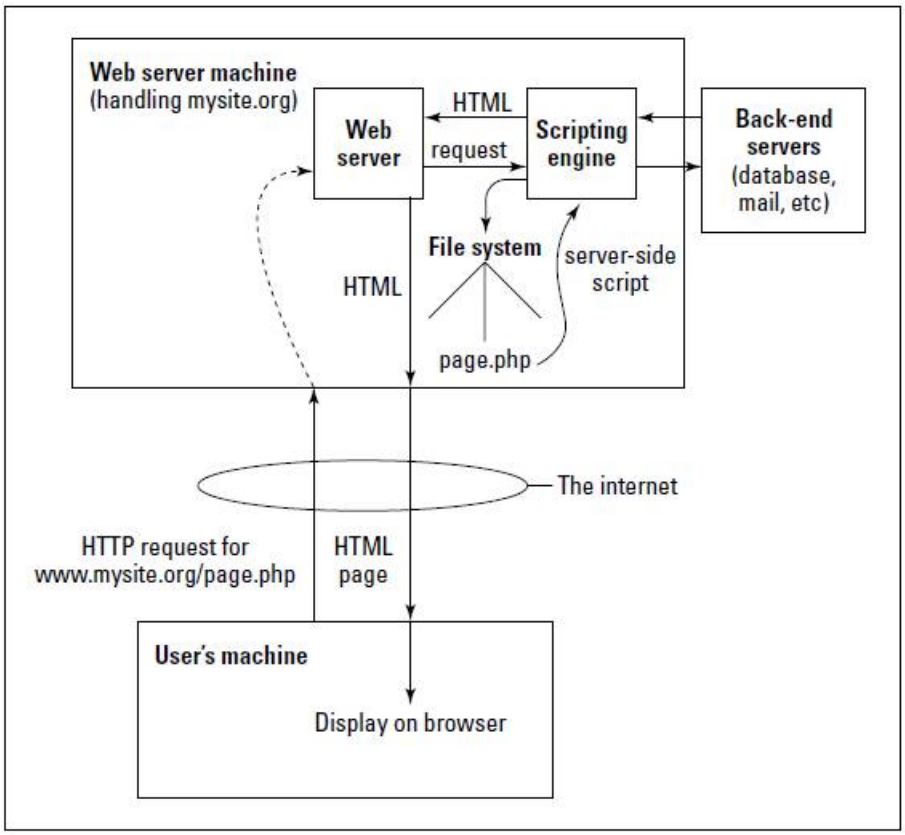


Figure 2 Server Side Tasks

- Server-side scripting products consist of two main parts: the scripting language and the scripting engine (which may or may not be built into the Web server). The engine parses and interprets pages written in the language.
- In server side scripting, the server side program is run on the web server and the result is passed to the client in HTML format.
- User will not see the server side code.
- Server side scripting is handy in cases such as Content sites (both production and display), community features (forums, bulletin boards, and so on), E-mail (Web mail, mail forwarding, and sending mail from a Web application), customer-support and technical-support systems, advertising networks, web-delivered business applications, directories and membership rolls, surveys, polls, and tests and filling out and submitting forms online.

WAMP/LAMP Server

- Many webserver suits are available.
- Apache is probably the Web server most commonly used with PHP and MySQL—so common that the acronym LAMP has emerged to describe precisely this combo (Linux Apache MySQL PHP). In Windows platforms, WAMP is available. We use this software.

- WAMP/LAMP is to be installed in your machine, before you start with PHP.
- You might have noted that HTML pages can be opened by the browsers from any folders.
- But a page containing server side such as PHP, is to be "hosted" to the server first.
- When we use our machine as the web server, it is called "localhost". Now all the PHP pages we developed are to be hosted to the "root" folder of the web server.
- Root folder for WAMP server is D:\WAMP\www, if the software is installed in the D: drive of your machine, in folder WAMP.

PHP

- PHP stands for **PHP: Hypertext Preprocessor**.
- The product was originally named *Personal Home Page* Tools, and many people still think that's what the acronym stands for.
- PHP is a server-side scripting language, which can be embedded in HTML or used as a standalone binary (although the former use is much more common).
- Strictly speaking, PHP has little to do with layout, events, on the fly DOM manipulation, or really anything about what a Web page looks and sounds like.
- In fact, most of what PHP does is invisible to the end user. Someone looking at a PHP page will not necessarily be able to tell that it was not written purely in HTML, because usually the result of PHP *is* HTML.
- PHP is an official module of Apache HTTP Server, the market-leading free Web server that runs about 67 percent of the World Wide Web (according to the widely quoted Netcraft Web server survey).

History

- Created by Rasmus Lerdorf—software engineer, Apache team member for his personal use in late 1994.
- Current stable version is Current Stable PHP 5.6.14.

Basic PHP Syntax

- A PHP script always starts with <?php and ends with ?>. A PHP script can be placed anywhere in the document.
- It also possible to use the <SCRIPT LANGUAGE= "PHP"> and </SCRIPT> tags for the same purpose.
- A PHP file must have a .php extension.

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

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

NB:- This file is to be kept in the www folder which will be inside the wamp installation folder.

- Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.
- PHP is whitespace insensitive - it means that it almost never matters how many whitespace characters you have in a row—one whitespace character is the same as many such characters.
- There are two basic statements to output text with PHP: echo and print.
- Comments in PHP
- In PHP, we use // to make a one-line comment or /* and */ to make a comment block.

A Sample Script using PHP – to print a multiplication table

```
<html>
    <body>
        <h2> Multiplication Table </h2>
        <table border = "1">
            <?php
            $myInteger = 13;
            $i = 0;
            $iNo = 0;
            for ($i=1; $i<=10; $i++) {
                $iNo = $i*$myInteger;
                print("<tr>");
                print("<td> $i $iNo </td>");
                print("<td> x </td>");
                print("<td> $myInteger </td>");
                print("<td> = </td>");
                print("<td> $iNo </td> </tr>\n");
            }
        ?>
    </table>
</body>
</html>
```

Variables

Rules for PHP variables:

- 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)
- Variable names are case-sensitive!

Datatypes

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

PHP is a Loosely Typed Language

- Data type of a variable need not be declared while initialization.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

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.

- PHP does some preprocessing of doubly quoted strings (strings with quotes like “this”) before constructing the string value itself. For one thing, variables are replaced by their values.

Example:-

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

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

Integer

An integer is a whole number (without decimals). It is a number between -2,147,483,648 and +2,147,483,647.

Rules for integers:

- An integer must have at least one digit (0-9)
- An integer cannot contain comma or blanks
- 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)

Example:-

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

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

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

Example

```
<?php

        $x = 10.365;
        var_dump($x);
?>
```

Boolean

A Boolean represents two possible states: TRUE or FALSE.
$x = true;
$y = false;

Booleans are often used in conditional testing.

Interpreting other types as Booleans

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

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is a compound type (an array or an object), it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).

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

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

Echo and print

- The simplest use of **echo** is to print a string as argument.

Example:

echo "This will print in the user's browser window.";
Or equivalently:
echo("This will print in the user's browser window.");

- You can also give multiple arguments to the unparenthesized version of echo, separated by commas.
- The command **print** is very similar to **echo**, with two important differences:
    - Unlike **echo**, **print** can accept only one argument.
    - Unlike **echo**, **print** returns a value, which represents whether the **print** statement succeeded. The value returned by print will be 1 if the printing was successful and 0 if unsuccessful.
- Both **echo** and **print** are usually used with string arguments, but due to PHP's type flexibility, it is possible to use any type of argument at them without causing an error – PHP will convert it to string before printing.

Operators

Type	Operators	Remarks
Arithmetic	**+, -, *, /, %, ++, --**	• All arithmetic expressions are evaluated to an arithmetic value. • + is used to add numbers and

		concatenate strings. Hence it is also called as **String** operator.
Assignment	**=, +=, -=, *=, /=, %=,**	• a += b ➔ a = a + b
Comparison	== (Equal to) != (Not Equal to) === (Equal value and equal type) !=== (Not Equal value and equal type) >, <, >=, <=	• All comparison expressions are evaluated to either true or false. • === tests for equality of value and data type.
Logical	&& (AND), || (OR), ! (NOT)	•
Conditional	?:	• ? : (Conditional) • If Condition is true? Then value X : Otherwise value Y
typeof		• Returns the datatype of the operand. • typeof b

Operator Precedence

- Multiplication (*) and division (/) have higher precedence than addition (+) and subtraction (-).
- The precedence can be changed by using parentheses.
- The following table lists JavaScript **arithmetic operators**, ordered from highest to lowest precedence:

Operator	Precedence
( )	Expression grouping
++ --	Increment and decrement
* / %	Multiplication, division, and modulo division
+ -	Addition and subtraction

Conditional Statements

- **if** statements, which are used when you want the script to execute if a condition is true.

- **if...else** statements, which are used when you want to execute one set of code if a condition is true and another if it is false.
- **switch** statements, which are used when you want to select one block of code from many depending on a situation.

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

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

```
<?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!";
}
?>
```

Conditional (or Ternary) Operator

variablename=(condition)?value1:value2

Looping

- A while loop runs the same block of code while or until a condition is true.
- A do while loop runs once before the condition is checked. If the condition is true, it will continue to run until the condition is false. (The difference between the do and do while loops is that do while runs once whether or not the condition is met.)
- A for loop runs the same block of code a specified number of times.

while (condition)
{
code to be executed
}
	do
	{
	code to be executed
	}
	while (condition)
		for (a; b; c)
		{
		code to be executed
		}
		
		❑**a** evaluates before the loop is run, and is only evaluated once. It is ideal for assigning a value to a variable.
		❑ **b** should be a condition that indicates whether the loop should be run again; if it returns true the loop runs again.
		❑ **c** is evaluated after the loop has run and can contain multiple expressions separated by a comma (for example i++, j++;).

- **break** statement is used to terminate the current execution and come out of a loop.
- **continue** statements suspends the current execution and tries to do the next iteration of the loop.

PHP Programs

Program 1

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

Program 2

```
<html>
    <body>
        <h2> Multiplication Table </h2>
        <table border = "1">
            <?php
            $myInteger = 13;
            $i = 0;
            $iNo = 0;
            for ($i=1; $i<=10; $i++) {
                $iNo = $i*$myInteger;
                print("<tr>");
```

```
                print("<td> $i $iNo </td>");
                print("<td> x </td>");
                print("<td> $myInteger </td>");
                print("<td> = </td>");
                 print("<td> $iNo </td> </tr>\n");
            }
        ?>
    </table>
  </body>
</html>
```

Program 3

Factorial Numbers

Program 4

```
<html>
  <body>

    <?php
       // Seed the random number generator.
       //
      srand(124); // add some seed value
      $num = rand(1, 4);

      switch( $num )
      {
        case 1: $image_file = "/php/images/logo.png";
          break;

        case 2: $image_file = "/php/images/php.jpg";
          break;

        case 3: $image_file = "/php/images/logo.png";
          break;

        case 4: $image_file = "/php/images/php.jpg";
          break;
      }
```

```
    echo "Random Image : <img src=$image_file />";
  ?>

 </body>
</html>
```

Arrays

- Simple and multi-dimensional arrays are supported.

- 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

Create an Array in PHP

- The array() function is used to create an array: array();

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

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 **count()** function is used to return the length (the number of elements) of an array.

Loop Through an Indexed Array

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

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

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:

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

To declare an associative array:

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

Now array elements can be accessed, as shown below:

```
echo $age[“peter”];

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

        foreach($age as $x => $x_value) {
                echo "Key=" . $x . ", Value=" . $x_value;
                echo "<br>";
        }
?>
```

list() Function: Assign variables as if they were an array:

```
<?php
        $my_array = array("Dog","Cat","Horse");
        list($a, $b, $c) = $my_array;
        echo "I have several animals, a $a, a $b and a $c.";
?>
```

Functions

There are more than 1000 built-in functions in PHP. In addition to this, we can create user defined functions.

- There are three parts to creating or defining a function:
  - Define a name for it.
  - Indicate any values that might be required as arguments.
  - Add statements.

Example:-

```
<?php
        function calculateArea($width, $height) {
                $area = $width * $height;
                return $area;
        }
<?
```

- Function names can start with an alphabet or underscore.
- Function names are not case sensitive.
- There is no need to explicitly specify the data types for the arguments.
- **We** can set default values in the function definition for arguments.

```
<?php
    function calculateArea($width = 20, $height=30) {
            $area = $width * $height;
            return $area;
    }
<?
```

- If no values are given for the arguments while calling the function, default values specified in the function definition, if any, will be used.

```
<?php
    echo calculateArea(); // returns 600
?>
```

- Call by reference is possible by prefixing the names of the argument variables by &.

```
<?php
    function swap(&$width, &$height) {
            $temp = $width
            $width = $height;
            $height = $temp;
            return
    }
    <?
```

- When multiple values are returned, the statement **return array (list of comma separated values)** can be used.

```
<?php
    function sqAreaPeri($side) {
    $area = $side*$side;
    $peri=4*$side
    return array($area, $peri);
    }
<?
```

- A variable declared within a function will have local scope within that function. Variables declared outside a function have global scope within that function.
- All variable declared with a **global** keyword are global variables. A variable can be declared as global within a function also.