Unit -1: Introduction to Server-Side Scripting & PHP Programming

Presented By:
Bipin Maharjan

Learning Objective

- Understand server-side scripting and its role in web development
- Learn about PHP and why it is widely used
- Set up a PHP development environment
- Explore basic syntax, data types, variables, constants, and operators in PHP

Table of Content

Introduction to HTTP

- HTTP is a communication standard governing the requests and responses that take place between the browser running on the end user’s computer and the web server.
- The server’s job is to accept a request from the client and attempt to reply to it in a meaningful way, usually by serving up a requested web page that’s why the term server is used.
- The natural counterpart to a server is a client, so that term is applied to the web browser and the computer on which it’s running.

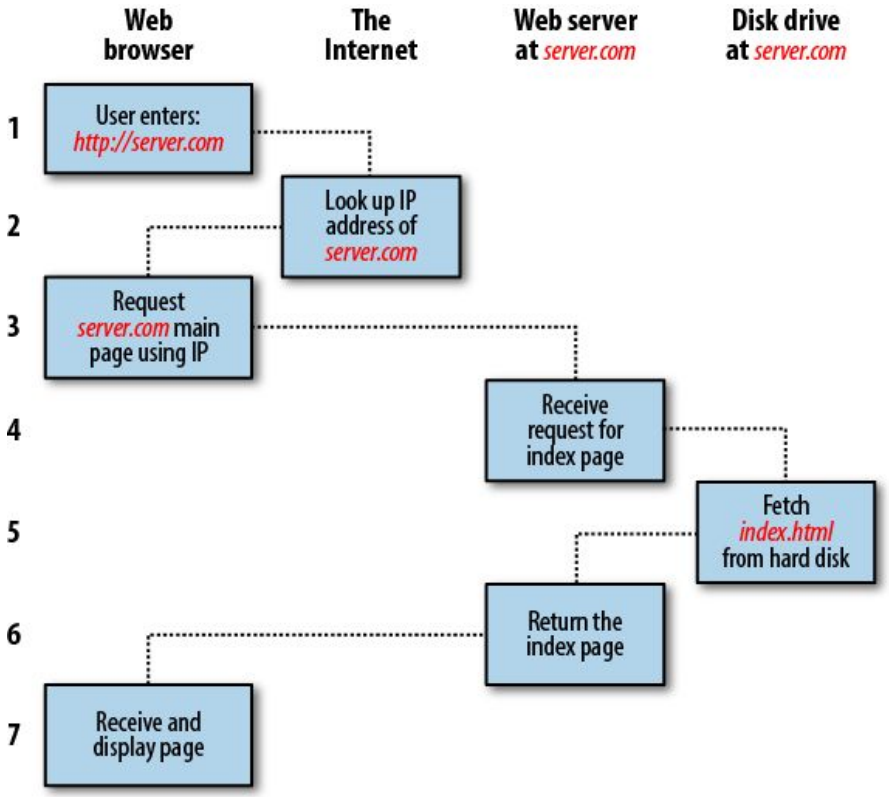
Introduction contd…

- Between the client and the server there can be several other devices, such as routers, proxies, gateways, and so on.
- They serve different roles in ensuring that the requests and responses are correctly transferred between the client and server.
- Typically, they use the Internet to send this information.
- A web server can usually handle multiple simultaneous connections and when not communicating with a client spends its time listening for an incoming connection.
- When one arrives, the server sends back a response to confirm its receipt.

The Request/Response Procedure

- At its most basic level, the request/response process consists of a web browser asking the web server to send it a web page and the server sending back the page.

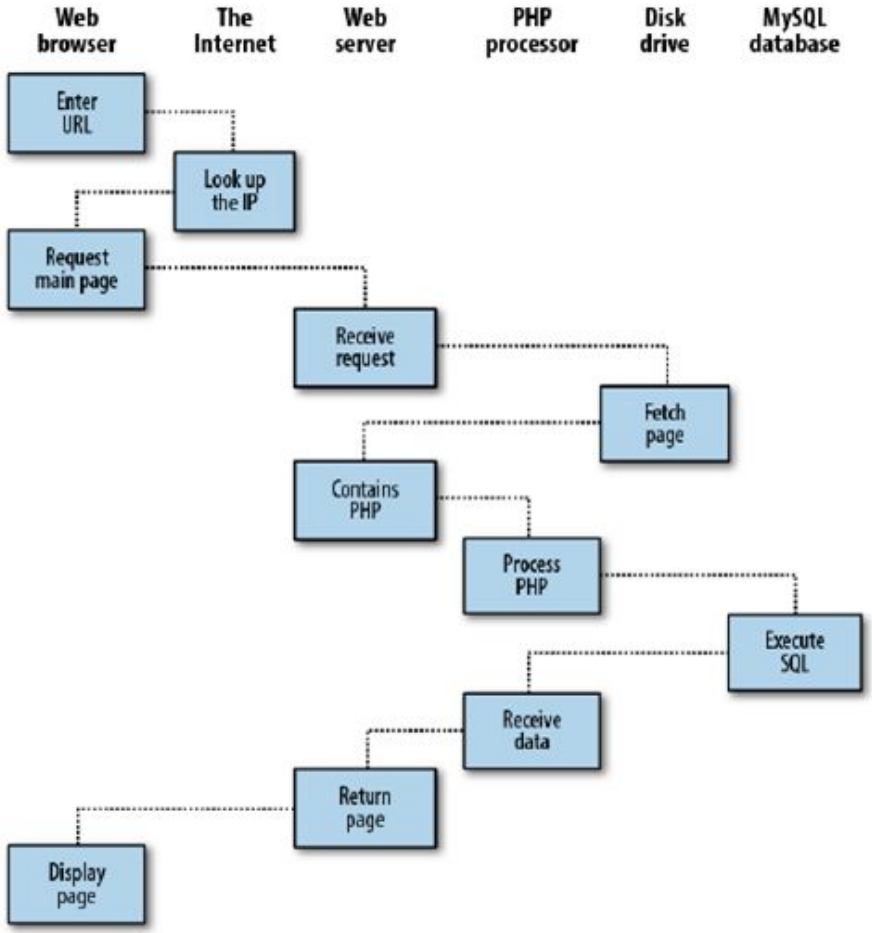
- The browser then takes care of displaying the page.



Each step in the request and response sequence:

1. You enter **http://server.com** into your browser's address bar.
2. Your browser looks up the IP address for server.com.
3. Your browser issues a request for the home page at server.com.
4. The request crosses the Internet and arrives at the server.com web server.
5. The web server, having received the request, looks for the web page on its hard disk.
6. The web page is retrieved by the server and returned to the browser.
7. Your browser displays the web page.

For dynamic web pages, the procedure is a little more involved, because it may bring both PHP and MySQL into the mix.



The steps for a dynamic client/server request/response sequence:

1. You enter http://server.com into your browser's address bar.
2. Your browser looks up the IP address for server.com.
3. Your browser issues a request to that address for the web server's home page.
4. The request crosses the Internet and arrives at the server.com web server.
5. The web server, having received the request, fetches the home page from its hard disk.
6. With the home page now in memory, the web server notices that it is a file incorporating PHP scripting and passes the page to the **PHP interpreter**.

Dynamic Client/Server Request/Response Contd…

7. The PHP interpreter executes the PHP code.

8. Some of the PHP contains MySQL statements, which the PHP interpreter now passes to the MySQL database engine.

9. The MySQL database returns the results of the statements back to the PHP interpreter.

10. The PHP interpreter returns the results of the executed PHP code, along with the results from the MySQL database, to the web server.

11. The web server returns the page to the requesting client, which displays it.

Introduction to server-side scripting

- The server is where the Web page and other content lives.
- The server sends pages to the user/client on request.

The Process is:

- The user requests a Web page from the server the script in the page is interpreted by the server creating or changing the page content to suit the user and the occasion and/or passing data.
- Around the page in its final form is sent to the user and then cannot be changed using server-side scripting.

Introduction to server-side scripting contd…

- Server-side scripting tends to be used for allowing users to have individual accounts and providing data from databases. It allows a level of privacy, personalization and provision of information that is very powerful.
- E-commerce and social networking sites all rely heavily on server-side scripting.
- Server-side scripts are never seen by the user.
- They run on the server and generate results which are sent to the user.

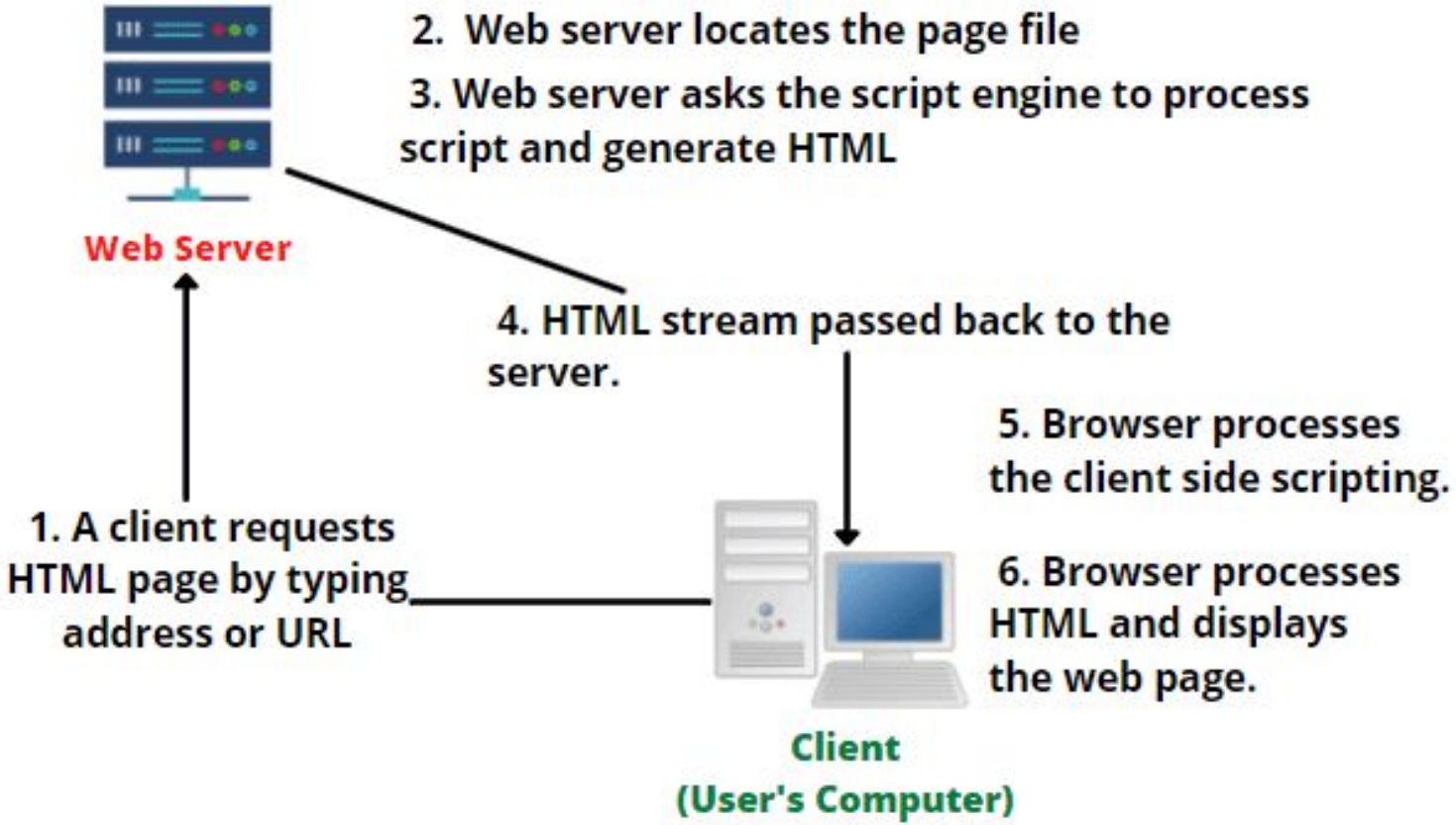


Figure: Server-side and Client-side scripting

Examples of Server-side Scripting Language

- ASP (*.asp)
- ASP.NET (*.aspx)
- JavaServer Pages
- PHP (*.php)
- Python (*.py)
- Ruby (*.rb, *.rbw)

Introduction to PHP

Introduction to PHP

- PHP stands for PHP: Hypertext Preprocessor
- It is a widely-used open source general-purpose server-side scripting language that is especially suited for web development and can be embedded into HTML
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Oracle, PostgreSQL, etc.)
- PHP is open source software
- PHP is free to download and use

Introduction to PHP contd…

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, Nginx, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side
- Instead of lots of commands to output HTML, PHP pages contain HTML with embedded code that does “something”.
- The PHP code is enclosed in special start and end processing instructions
- <?php and ?> that allow you to jump into and out of “PHP mode.”

PHP: Interpreted Language

- Code is executed line by line at runtime. No need to compile before running.
- Errors are caught during execution.
- PHP scripts are not compiled into machine code.
- The PHP engine (Zend Engine) interprets the code on the server.
- The result (usually HTML) is sent to the client’s browser.

Advantages of Interpreted Languages

- Easy to test and debug (no compilation needed).
- Cross-platform (runs anywhere PHP interpreter exists).
- Faster development cycle.

Disadvantage of Interpreted Languages

- Slower than compiled languages (e.g., C, Java).

First PHP Program

```
<!DOCTYPE html>
<html>
<head>
   <title>Basic PHP Page</title>
</head>
<body>
   <p>This is standard HTML.</p>
   <?php
      echo "Hello, World!";
   ?>
</body>
</html>
```

PHP is an HTML-embedded scripting language, meaning that you can intermingle PHP and HTML code within the same file.

Introduction to PHP Contd…

- PHP code is executed on the server, generating HTML which is then sent to the client.
- The client would receive the results of running that script, but would not know what the underlying code was.

Use cases of PHP

- Web Applications
  - Build custom web apps (e.g., Student Management Systems, E-commerce).
  - Example: Facebook (initially built with PHP).
- Content Management Systems (CMS)
  - Many popular CMS platforms are built on PHP.
  - Example: WordPress, Joomla, Drupal.
- E-commerce Platforms
  - Online shopping systems & order management.
  - Example: Magento, OpenCart, WooCommerce (WordPress plugin).
- Blog & News Portals
  - Easy content publishing and management.
  - Example: WordPress Blogs, MediaWiki (Wikipedia).
- REST APIs & Backend Services
  - Build APIs that interact with mobile apps or other systems.
  - Example: A PHP API for a mobile banking app.
- File & Data Processing
  - Handling uploads (images, PDFs, documents).
  - Generating reports (CSV, Excel, PDF).

Why PHP is still relevant?

- Open source & cost effective.
- Runs on almost every hosting provider.
- Large developer community & resources.
- Modern frameworks (Laravel, Symfony, CodeIgniter) for scalable apps.

Development Environment

- For Windows Users
  - Install XAMPP
    - Php 8.2>
    - Apache
    - MySQL
- For Macbook Users
  - Install MAMP
    - Php 8.2>
    - Apache/Nginx
    - MySQL

Hello World Program

```
<html>
    <head>
        <title>PHP Test</title>
    </head>
    <body>
        <?php echo ‘<p>Hello World</p>’; ?>
    </body>
</html>
```

Above is the PHP Source code

Hello World Program

```
<html>
    <head>
        <title>PHP Test</title>
    </head>
    <body>
        <p>Hello World</p>
    </body>
</html>
```

Rendered as HTML

Hello World Program

- This program is extremely simple and you really did not need to use PHP to create a page like this.
- All it does is display: Hello World using the PHP echo() statement.
- Think of this as a normal HTML file which happens to have a set of special tags available to you that do a lot of interesting things.

Send Data to the Web Browser

The only way that your embedded PHP code will display anything in a user's browser program is either by means of statements that print something to output or by calling functions that, in turn, call print statements.

Echo and print

The two most basic constructs for printing to output are echo and print. Their language status is somewhat confusing, because they are basic constructs of the PHP language, rather than being functions.

As a result, they can be used either with parentheses or without them.

echo

- echo is not actually a function (it is a language construct), so you are not required to use parentheses with it.
- echo does not behave like a function, so it cannot always be used in the context of a function.
- Additionally, if you want to pass more than one parameter to echo, the parameters must not be enclosed within parentheses.
- The simplest use of echo is to print a string as argument, for example:
- **echo “This will print in the user’s browser window.”;**

Echo Contd…

- Or equivalently: **echo(“This will print in the user’s browser window.”);**
- Both of these statements will cause the given sentence to be displayed, without displaying the quote signs.
- You can also give multiple arguments to the unparenthesized version of echo, separated by commas, as in:
- **echo “This will print in the”, “user’s browser window.”;**
- The parenthesized version, however, will not accept multiple arguments:

  echo (“This will produce a”, “PARSE ERROR!”);

Print

- Print is not actually a real function (it is a language construct) so you are not required to use parentheses with its argument list.
- The only difference to echo is that print only accepts a single argument.
- Unlike echo, print returns a value, which represents weather or not the print statement succeeded.
- The value returned by print is always 1.
- Both echo and print are usually used with string arguments, but PHP’s type flexibility means that you can throw pretty much any type of argument at them without causing an error.

Print Contd…

- For example, the following two lines will print exactly the same thing:
- print(“3.14159”); //print a string
- Print(3.1459); //print a number
- Technically, what is happening in the second line is that, because print expects a string argument, the floating-point version of the number is converted to a string value before print gets hold of it.
- However, the effect is that both print and echo will reliably print out numbers as well as string arguments.

Comments

- In PHP, we use **//** or **#** to make a single-line comment
- **/*.....*/** to make a large comment block

Data Types

- PHP data types are used to hold different types of data or values.
- PHP supports 8 primitive data types that can be categorized further in 3 types:
  - Scalar Types
  - Compound Types
  - Special Types

Scalar Types

There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float (double)
4. string

Compound Types

There are 2 compound data types in PHP.

1. array
2. object

Special Types

There are 2 special data types in PHP.

1. Resource: is a special variable, holding a reference to an external resource.
2. NULL: The special NULL value represents a variable with no values. NULL is the only possible value of type null.

A variable is considered to be null if:

- It has been assigned the constant NULL.
- It has not been set to any value yet.
- There is only one value of type null, and that is the case-insensitive constant NULL.

Variables

- Variables are used for storing values, like text strings, numbers, booleans, or arrays
- When a variable is declared, it can be used over and over again in your script.
- All variables in PHP start with a $ sign symbol.
- The correct way of declaring a variable in PHP:

$var_name = value;

Variable Contd…

```
<?php
    $txt = ”Hello World”;
    $x = 16;
?>
```

- In PHP, a variable does not need to be declared before adding a value to it.
- In the example above, you see that you do not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value

Variable Contd…

- A variable name must start with a letter or an underscore “_” – not a number
- A variable name can only contain alpha-numeric characters, underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces.
- If a variable name is more than one word, it should be separated with an underscore ($my_string) or with capitalization($myString)

Variable 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 Variable Scope

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

```
$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>";
```

Local Variable Scope

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

```
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>";
```

Static Variable Scope

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:

```
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}
myTest();
myTest();
myTest();
```

Super Global Variables

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Constants

- In addition to variables, which may be reassigned, PHP offers constants, which have a single value throughout their lifetime.
- Constants do not have a $ before their names, and by convention the names of constants usually are in uppercase letters.

Option-1: by using built-in function define()

The name of a constant follows the same rules as any label in PHP. A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

A constant is case-sensitive by default. By convention, constant identifiers are always uppercase.

Constants contd…

Like superglobals, the scope of a constant is global. You can access constants anywhere in your script without regard to scope.

```
<?php
//Valid constant names
define(“FOO”, “something”);
define(“FOO2”, “something else”);
define(“FOO_BAR”, “something more”);
//Invalid constant names
define(“2FO”, “something”);
Echo FOO;
?>
```

PHP Strings

A string is merely a quoted chunk of characters: letters, numbers, spaces, punctuation, and so forth.

These are all strings:

- 'Tobias'
- "In watermelon sugar"
- '100'
- 'August 2, 2017'

To make a string variable, assign a string

value to a valid variable name:

$first_name = 'Tobias';
$today = 'August 2, 2011';

String Concatenation

Concatenation is like addition for strings, whereby characters are added to the end of the string. It is performed using the concatenation operator, which is the period(.):

```
$city= 'Seattle';
$state = 'Washington';
$address = $city . $state;
```

The $address variable now has the value SeattleWashington, which almost achieves the desired result (Seattle, Washington).To improve upon this, you could write

```
$address = $city . ', ' . $state;
```

so that a comma and a space are concatenated to the variables as well.

String Operations

PHP has a set of built-in functions that you can use to modify strings.

- strlen("Hello") → 5 (length of string)
- strtoupper("hello") → HELLO
- strtolower("HELLO") → hello
- ucfirst("php") → Php
- substr("Hello World", 0, 5) → Hello
- str_replace("world", "PHP", "Hello world") → Hello PHP
- strpos("Hello PHP", "PHP") → 6 (position of substring)

PHP Numbers

PHP has both integer and floating-point (decimal) number types. Valid numbers in PHP can be anything like

- 8
- 3.14
- 10980843985
- –4.2398508
- 4.4e2

Number Operations

PHP has a set of built-in functions that you can use to modify strings.

- abs(-5) → 5 (absolute value)
- pow(2, 3) → 8 (power)
- sqrt(16) → 4 (square root)
- round(3.6) → 4 (round to nearest)
- ceil(3.2) → 4 (round up)
- floor(3.9) → 3 (round down)
- rand(1, 100) → random number between 1–100
- round(3.141592, 3); // 3.142
- number_format(20943); // 20,943
- number_format(20943, 2); //20,943.00

PHP Operators

Operators are used to perform operations on variables and values. PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Logical operators

Arithmetic operators

The PHP arithmetic operators are used with numerical values to perform common arithmetical operations such as addition, subtraction, multiplication, etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum the operands
-	Subtraction	$x - $y	Differences in the operands
*	Multiplication	$x * $y	Product of the operands
/	Division	$x / $y	The quotient of the operands
**	Exponentiation	$x ** $y	$x raised to the power $y
%	Modulus	$x % $y	The remainder of the operands

Assignment operators

The PHP assignment operators are used with numeric values to write a value to a variable. The basic assignment operator in PHP is “=”. It means that the left operand gets set to the value of the assignment expression on the right.

Operator	Name	Example	Result
=	Assign	$x = $y	Operand on the left obtains the value of the operand on the right
+=	Add then Assign	$x += $y	Simple Addition same as $x = $x + $y
-=	Subtract then Assign	$x -= $y	Simple subtraction same as $x = $x - $y
*=	Multiply then Assign	$x *= $y	Simple product same as $x = $x * $y
/=	Divide, then assign (quotient)	$x /= $y	Simple division same as $x = $x/$y
%=	Divide, then assign (remainder)	$x %= $y	Simple division same as $x = $x%$y

Logical operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	Logical AND	$x and $y	True if both the operands are true else false
or	Logical OR	$x or $y	True if either of the operands is true otherwise, it is false
xor	Logical XOR	$x xor $y	True if either of the operands is true and false if both are true
&&	Logical AND	$x && $y	True if either of the operands is true otherwise, it is false
||	Logical OR	$x || $y	True if either of the operands is true otherwise, it is false
!	Logical NOT	!$x	True if $x is false

Any Questions?