Marwadi University

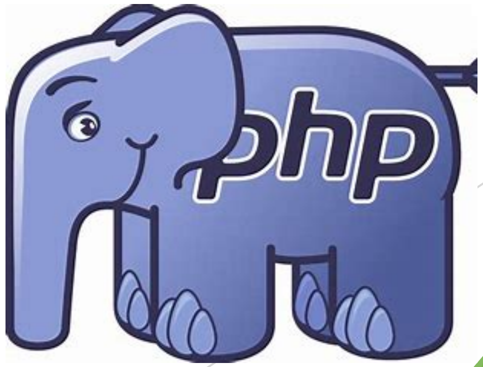
Web Application Development (PHP)

Unit – 1

PHP Installation and Basics

Prof. Hardik Chavda

FOCA,MU



Unit – 1

- **PHP Installation and Configuration:**
  - Current version of Apache, XAMPP Installation.
  - Apache Log file, PHP.INI file, basics of PHP scripts.
- **PHP Overview**
  - Flow Control and building blocks:
  - Variables, Data types
  - Operators and expressions
  - Constants, Switching flow
  - Looping Structure
  - Code blocks and browser Output.
  - Include & require Function

Open Source

- Free Redistribution
- Source Code
- Integrity of The Author's Source Code
- No Discrimination Against Persons or Groups or field
- Distribution of License
- License Must Not Be Specific to a Product
- License Must Not Restrict Other Software
- License Must Be Technology-Neutral

PHP Installation and Configuration:
Current version of Apache, XAMPP Installation

- XAMPP is the most popular software package which is used to set up a PHP development environment for web services by providing all the required software components.
- During the process of software deployment, most of the web servers use almost similar components, so use of XAMPP provides easy transition from local server to live server.
- XAMPP is a AMP stack which stands for Cross platform, Apache, MySQL, PHP, perl with some additional administrative software tools such as PHPMyAdmin (for database access).
- Other commonly known software packages like XAMPP are WAMP, LAMP, and others.

Advantages of XAMPP:

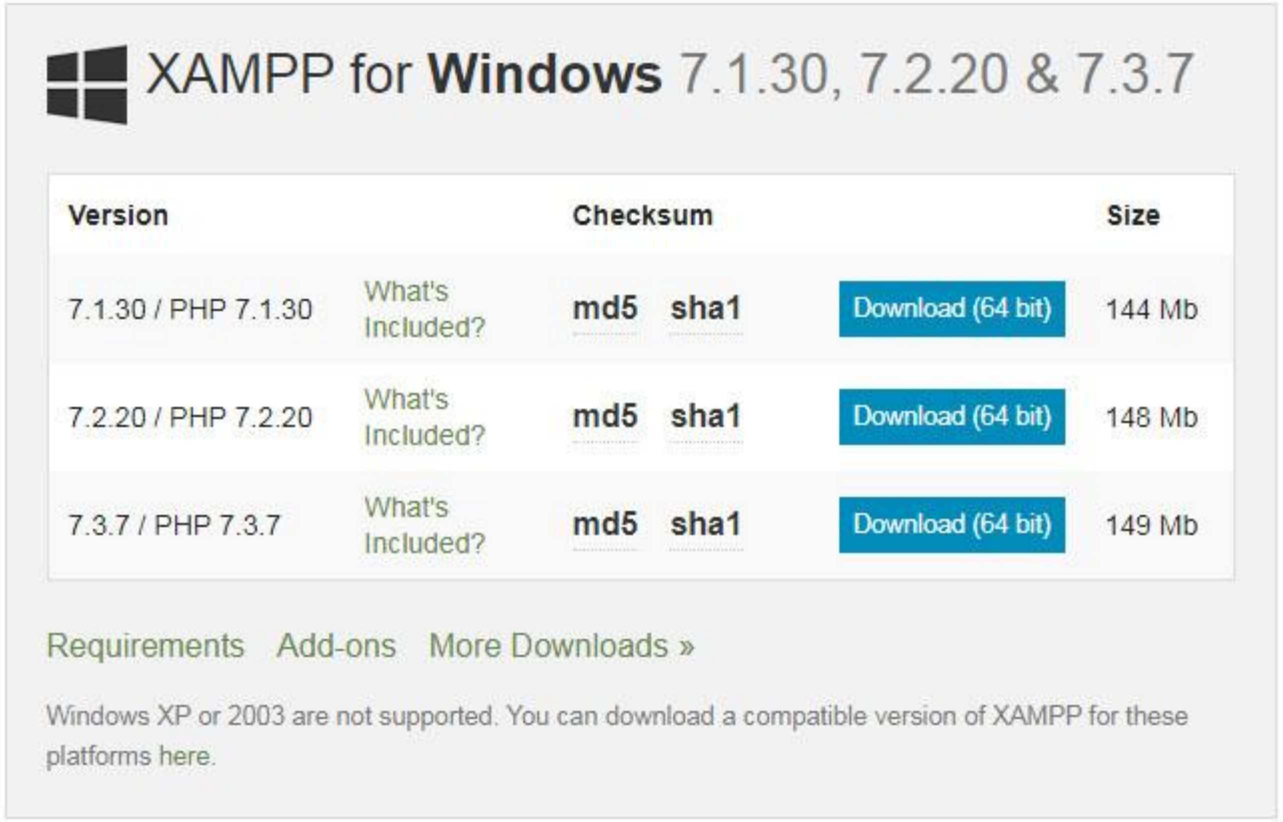
- It is free and easy to use and easily available for Windows, Linux and Mac OS .
- It is a beginners friendly solution package for full stack web development.
- It is a **open source** software package which gives a easy installation experience.
- It is very simple and **lightweight** to create set up for development, testing and deployment.
- It is a time-saver and provides several ways for managing configuration changes.
- It handles many administrative tasks like checking the status and security.

Software components of XAMPP:

- **Apache** plays the role of processing the HTTP request. It is the actual default web server application. It is the most popular web servers maintained by Apache Software Foundation.
- **MySQL** The role of database management system in XAMPP is played by MySQL. It helps to store and manage collected data very efficiently. It is an open-source and most popular.
- **PHP** is the server-side scripting language which stand for Hypertext Preprocessor. It is embedded with HTML code which interacts with the webserver. It is an open-source and work well with MySQL and has become a common choice for web developers.
- **Perl** is the high-level programming language designed for text editing which serves purpose like web development and network programming.

Steps to install XAMPP on Windows:

- Steps to install XAMPP on Windows:

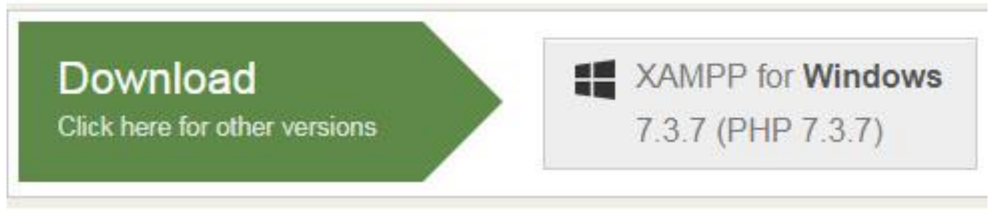


XAMPP for **Windows** 7.1.30, 7.2.20 & 7.3.7

Version		Checksum			Size
7.1.30 / PHP 7.1.30	What's Included?	md5	sha1	Download (64 bit)	144 Mb
7.2.20 / PHP 7.2.20	What's Included?	md5	sha1	Download (64 bit)	148 Mb
7.3.7 / PHP 7.3.7	What's Included?	md5	sha1	Download (64 bit)	149 Mb

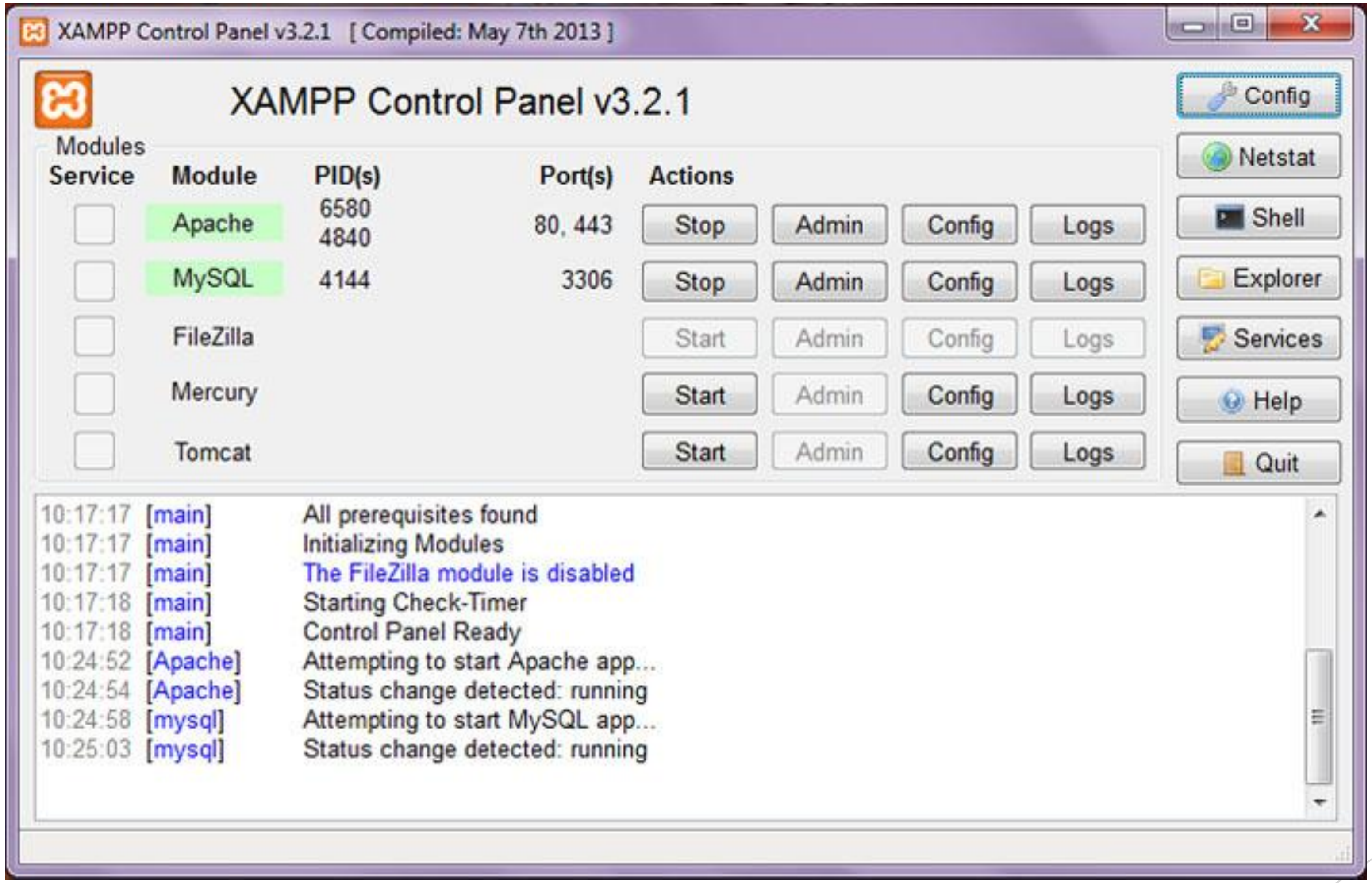
Requirements Add-ons More Downloads »

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms here.



- During the installation process, select the required components like MySQL, FileZilla ftp server, PHP, phpMyAdmin or leave the default options and click the **Next** button.
- Uncheck the Learn more about bitnami option and click Next button.
- Choose the root directory path to set up the htdocs folder for our applications. For example ‘C:\xampp’.
- Click the Allow access button to allow the XAMPP modules from the Windows firewall.
- After the installation process, click the Finish button of the XAMPP Setup wizard.
- Now the XAMPP icon is clearly visible on the right side of start menu. Show or Hide can be set by using the control panel by clicking on the icon.

- To start Apache and MySql, just click on the Start button on the control panel.



XAMPP Control Panel v3.2.1 [Compiled: May 7th 2013]
XAMPP Control Panel v3.2.1
Config
Netstat
Shell
Explorer
Services
Help
Quit

Modules Service	Module	PID(s)	Port(s)	Actions			
	Apache	6580 4840	80, 443	Stop	Admin	Config	Logs
	MySQL	4144	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
10:17:17  [main]     All prerequisites found
10:17:17  [main]     Initializing Modules
10:17:17  [main]     The FileZilla module is disabled
10:17:18  [main]     Starting Check-Timer
10:17:18  [main]     Control Panel Ready
10:24:52  [Apache]   Attempting to start Apache app...
10:24:54  [Apache]   Status change detected: running
10:24:58  [mysql]    Attempting to start MySQL app...
10:25:03  [mysql]    Status change detected: running
```

PHP Installation and Configuration:

Apache Log file, PHP.INI file, basics of PHP scripts

- To see all the settings made by xampp for current version of apache and php we can use below method to check any functionalities provided by xampp environment.

```
<?php
phpinfo();
?>
```

Introduction to PHP

- **What is PHP?**
  - PHP stands for PHP: **Hypertext Preprocessor**
  - PHP is a server-side scripting language, like ASP
  - PHP scripts are executed on the server
  - PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
  - PHP is an open source software (OSS)
  - PHP is free to download and use

- **What is a PHP File?**
  - PHP files may contain text, HTML tags and scripts
  - PHP files are returned to the browser as plain HTML
  - PHP files have a file extension of ".php", ".php3", or ".phtml“
- **PHP + MySQL**
  - PHP combined with MySQL are cross-platform (means that you can develop in Windows and serve on a Unix platform)
- **Why PHP?**
  - PHP runs on different platforms (Windows, Linux, Unix, etc.)
  - PHP is compatible with almost all servers used today (Apache, IIS, 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

Variables in PHP

- 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)
- PHP automatically associates a data type to the variable, depending on its value.

Examples

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

```
string is: hello string
integer is: 200
float is: 44.6
```

PHP Variable: case sensitive

- In PHP, variable names are case sensitive.
- So variable name "color" is different from Color, COLOR, COLor etc

```
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

```
My car is red
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4
My house is
Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5
My boat is
```

PHP Variables Scope

- PHP has three different variable scopes:
  - Local
    - A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function
  - Global
    - A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function
  - Static
    - 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

Local Variable Example

```
<?php
    function mytest()
    {
        $lang = "PHP";
        echo "Web development language: " .$lang;
    }
    mytest();
    //using $lang (local variable) outside the function will generate an error
    echo $lang;
?>
```

```
Web development language: PHP
Notice: Undefined variable: lang in D:\xampp\htdocs\program\p3.php on line 28
```

Globals

- The global variables are the variables that are declared outside the function.
- These variables can be accessed anywhere in the program.
- To access the global variable within a function, use the GLOBAL keyword before the variable.
- However, these variables can be directly accessed or used outside the function without any keyword.
- Therefore there is no need to use any keyword to access a global variable outside the function.

Global Variable Example

```
<?php
    $name = “Hardik Chavda";        //Global Variable
    function global_var()
    {
        global $name;
        echo "Variable inside the function: ". $name;
        echo "</br>";
    }
    global_var();
    echo "Variable outside the function: ". $name;
?>
```

```
Variable inside the function: Hardik Chavda
Variable outside the function: Hardik Chavda
```

Using $GLOBALS instead of global

```
<?php
    $num1 = 5;      //global variable
    $num2 = 13;     //global variable
    function global_var()
    {
            $sum = $GLOBALS['num1'] + $GLOBALS['num2'];
            echo "Sum of global variables is: " .$sum;
    }
    global_var();
?>
```

```
Sum of global variables is: 18
```

Static variable

- It is a feature of PHP to delete the variable, once it completes its execution and memory is freed.
- Sometimes we need to store a variable even after completion of function execution.
- Therefore, another important feature of variable scoping is static variable.
- We use the static keyword before the variable to define a variable, and this variable is called as **static variable**.
- Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope.

Example

```
<?php
    function static_var()
    {
        static $num1 = 3;       //static variable
        $num2 = 6;          //Non-static variable
        //increment in non-static variable
        $num1++;
        //increment in static variable
        $num2++;
        echo "Static: " .$num1 ."</br>";
        echo "Non-static: " .$num2 ."</br>";
    }
//first function call
    static_var();
//second function call
    static_var();
?>
```

```
Static: 4
Non-static: 7
Static: 5
Non-static: 7
```

Constants

- Constants are like variables except that once they are defined they cannot be changed or undefined.
- 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.
- To create a constant, use the define() function.
- Syntax
  - define(*name*, *value*, *case-insensitive*)
  - 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

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)
  - Boolean
  - Array
  - NULL

Operators and expressions

- 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

Arithmetic Operator

Operator	Name	Example	Explanation
+	Addition	$a + $b	Sum of operands
-	Subtraction	$a - $b	Difference of operands
*	Multiplication	$a * $b	Product of operands
/	Division	$a / $b	Quotient of operands
%	Modulus	$a % $b	Remainder of operands
**	Exponentiation	$a ** $b	$a raised to the power $b

- Assignment Operator

Operator	Name	Example	Explanation
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a += $b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b
/=	Divide then Assign (quotient)	$a /= $b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Find remainder same as $a = $a % $b

- Comparison operators

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b

- Increment/Decrement operators

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
		$a++	Return $a, then increment the value of $a by one
--	decrement	--$a	Decrement the value of $a by one, then return $a
		$a--	Return $a, then decrement the value of $a by one

- Logical operators

Operator	Name	Example	Explanation
and	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

- String operators

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenate both $a and $b
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

- Conditional assignment operators

Operator	Name	Example	Explanation
?:	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

- Array operators

Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order
!==	Non-Identity	$a !== $b	Return TRUE if $a is not identical to $b
<>	Inequality	$a <> $b	Return TRUE if $a is not equal to $b

Switching flow

- PHP if else statement is used to test condition. There are various ways to use if statement in PHP.
  - if
  - if-else
  - if-else-if
  - nested if
  - Switch

if

- PHP if statement allows conditional execution of code. It is executed if condition is true.
- If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.
- **Syntax**

```
if(condition){
//code to be executed
}
```

- **Example**

```
<?php
    $num=12;
    if($num<100){
    echo "$num is less than 100";
    }
?>
```

Output:

```
12 is less than 100
```

If else

- PHP if-else statement is executed whether condition is true or false.
- If-else statement is slightly different from if statement. It executes one block of code if the specified condition is **true** and another block of code if the condition is **false**.
- **Syntax**

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

- Example

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

Output:

```
12 is even number
```

If else if

- PHP If-else-if Statement
- The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.
- **Syntax**

```
if (condition1){
//code to be executed if condition1 is true
} elseif (condition2){
//code to be executed if condition2 is true
} elseif (condition3){
//code to be executed if condition3 is true
....
}  else{
//code to be executed if all given conditions are false
}
```

- Example

```
<?php
    $marks=69;
    if ($marks<33){
        echo "fail";
    }
else if ($marks>=50 && $marks<65) {
       echo "C grade";
    }
    else if ($marks>=65 && $marks<80) {
        echo "B grade";
    }
    else if ($marks>=80 && $marks<90) {
        echo "A grade";
    }
else {
        echo "Invalid input";
    }
?>
```

Output:

```
B Grade
```

PHP nested if Statement

- The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is **true**.
- **Syntax**

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

- Example

```
<?php
    $age = 23;
    $nationality = "Indian";
    //applying conditions on nationality and age
    if ($nationality == "Indian")
    {
        if ($age >= 18) {
            echo "Eligible to give vote";
        }
        else {
            echo "Not eligible to give vote";
        }
    }
?>
```

- Output:
- Eligible to give vote

Switch

- PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.
- **Syntax**

```
switch(expression){
case value1:
 //code to be executed
 break;
case value2:
 //code to be executed
 break;
......
default:
 code to be executed if all cases are not matched;
}
```

- Important points to be noticed about switch case:
  - The **default** is an optional statement. Even it is not important, that default must always be the last statement.
  - There can be only one **default** in a switch statement. More than one default may lead to a **Fatal** error.
  - Each case can have a **break** statement, which is used to terminate the sequence of statement.
  - The **break** statement is optional to use in switch. If break is not used, all the statements will execute after finding matched case value.
  - PHP allows you to use number, character, string, as well as functions in switch expression.
  - Nesting of switch statements is allowed, but it makes the program more complex and less readable.
  - You can use semicolon (;) instead of colon (:). It will not generate any error.

```
<?php
	$num=20;
	switch($num){
	case 10:
	echo("number is equals to 10");
	break;
	case 20:
	echo("number is equal to 20");
	break;
	case 30:
	echo("number is equal to 30");
	break;
	default:
	echo("number is not equal to 10, 20 or 30");
	}
?>
```

Output:

```
number is equal to 20
```

Loops

- for
- foreach
- while
- dowhile
- break
- continue

For loop

- PHP for loop can be used to traverse set of code for the specified number of times.
- It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.
- It allows users to put all the loop related statements in one place. See in the syntax given below:

  Syntax

  ```
  for(initialization; condition; increment/decrement){
   //code to be executed
  }
  ```
- Parameters
  - **initialization** - Initialize the loop counter value. The initial value of the for loop is done only once. This parameter is optional.
  - **condition** - Evaluate each iteration value. The loop continuously executes until the condition is false. If TRUE, the loop execution continues, otherwise the execution of the loop ends.
  - **Increment/decrement** - It increments or decrements the value of the variable.

- Example

```
<?php
for($n=1;$n<=10;$n++){
echo "$n<br/>";
}
?>
```

- Output

```
1
2
3
4
5
6
7
8
9
10
```

While Loop

- PHP while loop can be used to traverse set of code like for loop. The while loop executes a block of code repeatedly until the condition is FALSE. Once the condition gets FALSE, it exits from the body of loop.
- It should be used if the number of iterations is not known.
- The while loop is also called an **Entry control loop** because the condition is checked before entering the loop body. This means that first the condition is checked. If the condition is true, the block of code will be executed.
- **Syntax**

```
while(condition){
//code to be executed
}
```

- **Alternative Syntax**

```
while(condition):
//code to be executed


endwhile;
```

- **Normal Example**

```
<?php
$n=1;
while($n<=10){
echo "$n<br/>";
$n++;
}
?>
```

- **Output:**

```
1
2
3
4
5
6
7
8
9
10
```

- **Alternative Example**

```
<?php
$n=1;
while($n<=10):
echo "$n<br/>";
$n++;
endwhile;
?>
```

- **Output:**

```
1
2
3
4
5
6
7
8
9
10
```

PHP do-while loop

- PHP do-while loop can be used to traverse set of code like php while loop. The PHP do-while loop is guaranteed to run at least once.
- The PHP do-while loop is used to execute a set of code of the program several times. If you have to execute the loop at least once and the number of iterations is not even fixed, it is recommended to use the **do-while** loop.
- It executes the code at least one time always because the condition is checked after executing the code.
- The do-while loop is very much similar to the while loop except the condition check. The main difference between both loops is that while loop checks the condition at the beginning, whereas do-while loop checks the condition at the end of the loop.
- Syntax

```
do{
//code to be executed
}while(condition);
```

- Example

```
<?php
$n=1;
do{
echo "$n<br/>";
$n++;
}while($n<=10);
?>
```

- Output:

```
1
2
3
4
5
6
7
8
9
10
```

Difference between while and do-while loop

while Loop	do-while loop
The while loop is also named as **entry control loop**.	The do-while loop is also named as **exit control loop**.
The body of the loop does not execute if the condition is false.	The body of the loop executes at least once, even if the condition is false.
Condition checks first, and then block of statements executes.	Block of statements executes first and then condition checks.
This loop does not use a semicolon to terminate the loop.	Do-while loop use semicolon to terminate the loop.

Break

- PHP break statement breaks the execution of the current for, while, do-while, switch, and for-each loop. If you use break inside inner loop, it breaks the execution of inner loop only.
- The **break** keyword immediately ends the execution of the loop or switch structure. It breaks the current flow of the program at the specified condition and program control resumes at the next statements outside the loop.
- The break statement can be used in all types of loops such as while, do-while, for, foreach loop, and also with switch case.
- Syntax
- jump statement;
- **break**;

- PHP Break: inside loop
- Let's see a simple example to break the execution of for loop if value of i is equal to 5.

```
<?php
for($i=1;$i<=10;$i++){
echo "$i <br/>";
if($i==5){
break;
}
}
?>
```

- Output:

```
1
2
3
4
5
```

Continue

- The PHP continue statement is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
- The continue statement is used within looping and switch control structure when you immediately jump to the next iteration.
- The continue statement can be used with all types of loops such as - for, while, do-while, and foreach loop. The continue statement allows the user to skip the execution of the code for the specified condition.
- Syntax
- jump-statement;
- **continue**;

- PHP Continue Example with for loop
- **Example**
- In the following example, we will print only those values of i and j that are same and skip others.

```
<?php
    //outer loop
    for ($i =1; $i<=3; $i++) {
        //inner loop
        for ($j=1; $j<=3; $j++) {
            if (!($i == $j) ) {
                continue;       //skip when i and j does not have same values
            }
            echo $i.$j;
            echo "</br>";
        }
    }
?>
```

Output:

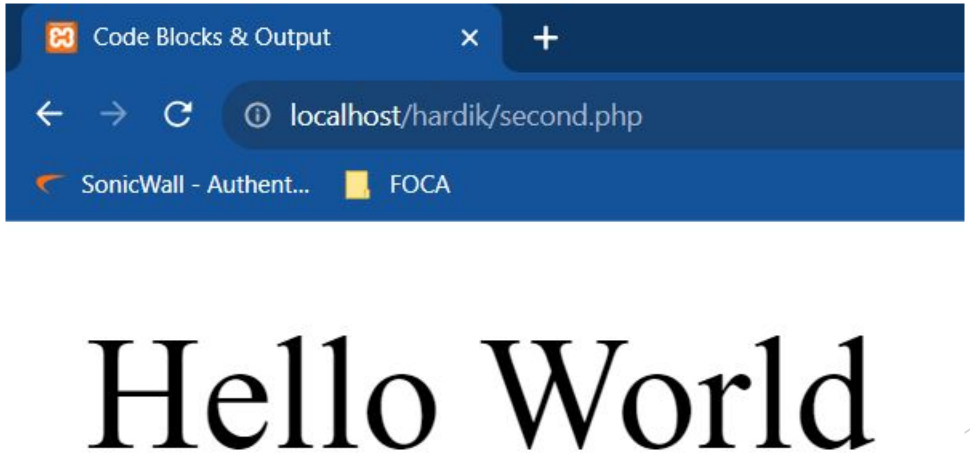
```
11
22
33
```

Code Blocks & Output

- You can incorporate PHP into an HTML document simply by adding HTML outside the PHP start and end tags.
- As you can see, incorporating HTML into a PHP document is simply a matter of typing in the code. The PHP engine ignores everything outside PHP open and close tags.
- As you can see, incorporating HTML into a PHP document is simply a matter of typing in the code. The PHP engine ignores everything outside PHP open and close tags.

```
<html lang="en">
<head>
<title>Code Blocks & Output</title>
</head>

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



```
Hello World
```

Comments

- Code that seems clear at the time of writing can seem like a hopeless tangle when you come to amend it six months later. Adding comments to your code as you write can save you time later on and make it easier for other programmers to work with your code.
- A comment is text in a script that is ignored by the PHP engine. Comments can be used to make code more readable, or to annotate a script.
- **Single line comments** begin with two forward slashes (//).
  - All text from either of these marks until either the end of the line or the PHP close tag is ignored.

```
// this is a comment
```

- **Multiline comments** begin with a forward slash followed by an asterisk (/*) and end with an asterisk followed by a forward slash (*/).

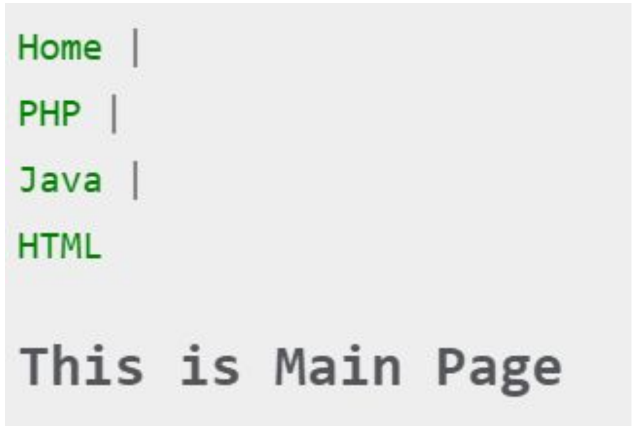
```
/*
this is a comment
none of this will
be parsed by the
PHP engine
*/
```

Include & Require

- PHP allows us to create various elements and functions, which are used several times in many pages. It takes much time to script these functions in multiple pages. Therefore, use the concept of **file inclusion** that helps to include files in various programs and saves the effort of writing code multiple times.
- "PHP allows you to include file so that a page content can be reused many times. It is very helpful to include files when you want to apply the same HTML or PHP code to multiple pages of a website." There are two ways to include file in PHP.
  - include
  - require
- **Both include and require are identical to each other, except failure.**
  - **include** only generates a warning, i.e., E_WARNING, and continue the execution of the script.
  - **require** generates a fatal error, i.e., E_COMPILE_ERROR, and stop the execution of the script.
- Advantage
  - **Code Reusability:** By the help of include and require construct, we can reuse HTML code or PHP script in many PHP scripts.
  - **Easy editable:** If we want to change anything in webpages, edit the source file included in all webpage rather than editing in all the files separately.

- **PHP include**
- PHP include is used to include a file on the basis of given path. You may use a relative or absolute path of the file.
- Syntax
  - **include** 'filename ';

    Or
  - **include** ('filename');
- **Example**
- *menu.html*
  - <a href="http://www.mu.com">Home</a> |
  - <a href="http://www.mu.com/php-tutorial">PHP</a> |
  - <a href="http://www.mu.com/java-tutorial">Java</a> |
  - <a href="http://www.mu.com/html-tutorial">HTML</a>
- *home.php*
  - <?php **include**("menu.html"); ?>
  - <h1>This is Main Page</h1>

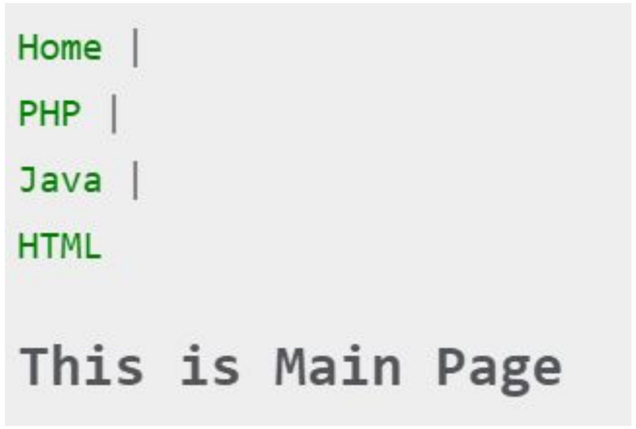


```
Home |
PHP |
Java |
HTML

This is Main Page
```

- **PHP require**
- PHP require is similar to include, which is also used to include files. The only difference is that it stops the execution of script if the file is not found whereas include doesn't.
- **Syntax**
  - **require** 'filename ';

    Or
  - **require** ('filename');
- **Example**
- *menu.html*
  - <a href="http://www.mu.com">Home</a> |
  - <a href="http://www.mu.com/php-tutorial">PHP</a> |
  - <a href="http://www.mu.com/java-tutorial">Java</a> |
  - <a href="http://www.mu.com/html-tutorial">HTML</a>
- *home.php*
  - <?php **require**("menu.html"); ?>
  - <h1>This is Main Page</h1>



```
Home |
PHP |
Java |
HTML

This is Main Page
```

- **PHP include vs PHP require**
- Both include and require are same.
- But if the file is missing or inclusion fails, **include** allows the script to continue but **require** halts the script producing a fatal E_COMPILE_ERROR level error.
- If included file is missing
  - Warning: include(menu.html): failed to open stream: No such file or directory in C:\xampp\htdocs\program\include.php on line
  - Warning: include(): Failed opening menu.html' for inclusion (include_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\program\include.php on line
  - This is Main Page **(this will be executed)**
- If require file is missing
  - Warning: require(menu.html): failed to open stream: No such file or directory in C:\xampp\htdocs\program\include.php on line
  - Fatal error: require(): Failed opening required menu.html ' (include_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\program\include.php on line