CORE PHP Notes

Web Technologies

Web Technology refers to the various tools and techniques that are utilized in the process of communication between different types of devices over the internet. A web browser is used to access web pages. Web browsers can be defined as programs that display text, data, pictures, animation, and video on the Internet. Hyperlinked resources on the World Wide Web can be accessed using software interfaces provided by Web browsers.

Web Technology can be classified into the following sections:

World Wide Web (WWW): The World Wide Web is based on several different technologies: Web browsers, Hypertext Markup Language (HTML) and Hypertext Transfer Protocol (HTTP).

Web Browser: The web browser is an application software to explore www (World Wide Web). It provides an interface between the server and the client and requests to the server for web documents and services.

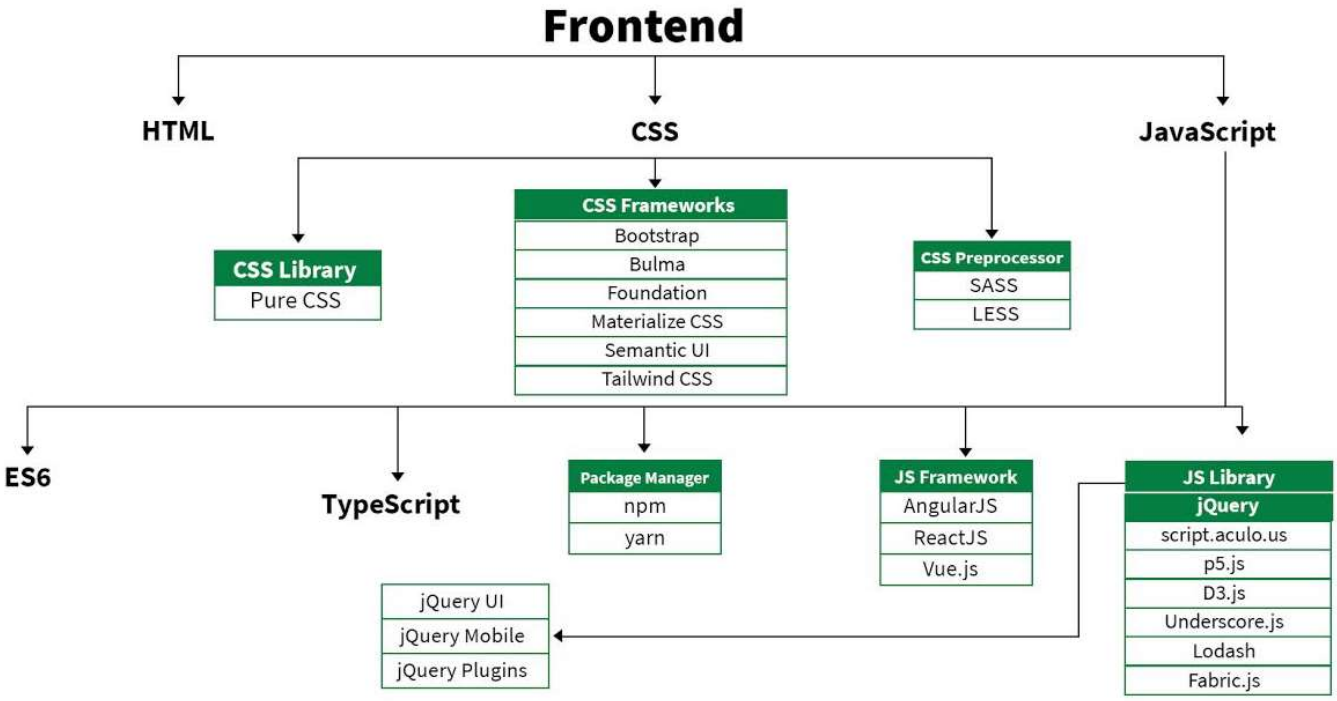
Web Server: Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP).

Web Pages: A webpage is a digital document that is linked to the World Wide Web and viewable by anyone connected to the internet has a web browser.

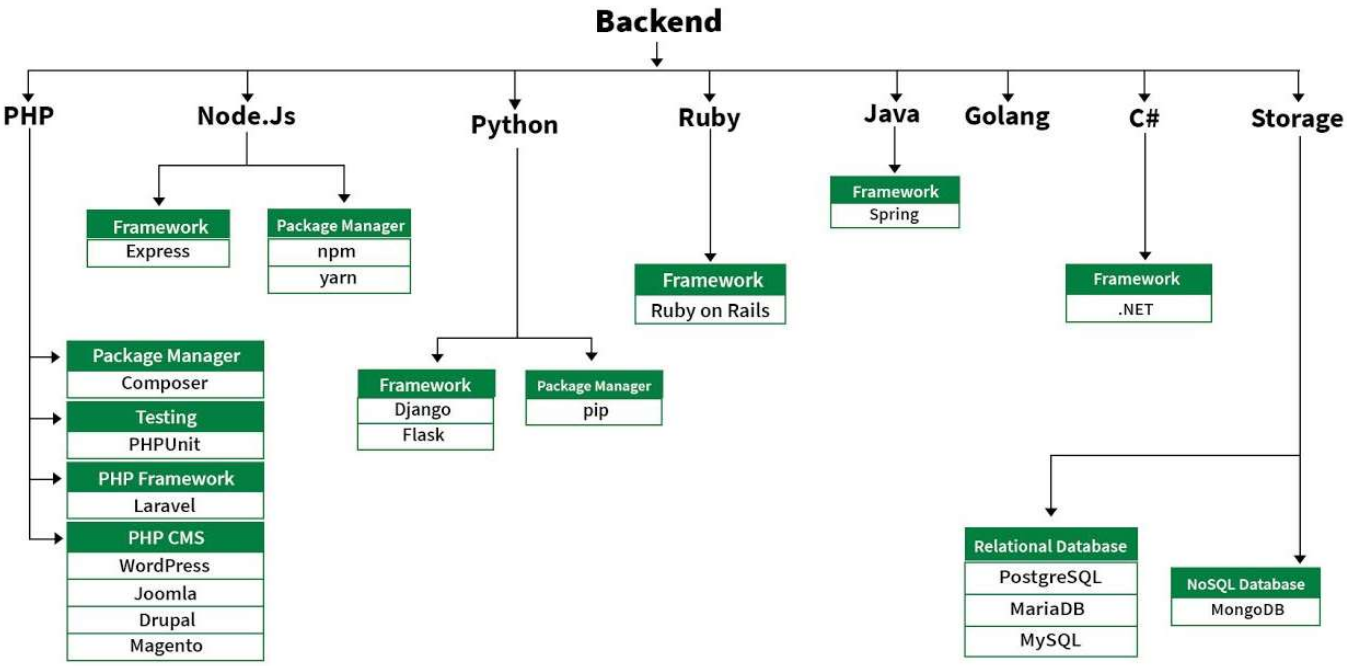
Web Development: Web development refers to the building, creating, and maintaining of websites. It includes aspects such as web design, web publishing, web programming, and database management. It is the creation of an application that works over the internet i.e. websites.

Web Development can be classified into two ways:

- **Frontend Development:** The part of a website that the user interacts directly is termed as front end. It is also referred to as the 'client side' of the application.



- **Backend Development:** Backend is the server side of a website. It is the part of the website that users cannot see and interact. It is the portion of software that does not come in direct contact with the users. It is used to store and arrange data.



Frontend Languages: The front-end portion is built by using some languages which are discussed below:

- **HTML:** HTML stands for Hypertext Markup Language. It is used to design the front-end portion of web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text documentation within the tag which defines the structure of web pages.
- **CSS:** Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.
- **JavaScript:** JavaScript is a famous scripting language used to create magic on the sites to make the site interactive for the user. It is used to enhancing the functionality of a website to running cool games and web-based software.
- **AJAX:** Ajax is an acronym for Asynchronous Javascript and XML. It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.

There are many other languages through which one can do front-end development depending upon the framework for example *Flutter* user *Dart*, *React* uses *JavaScript* and *Django* uses *Python*, and much more.

Front End Frameworks and Libraries:

- **CSS:**
  - **CSS Library:**
    - Pure CSS
  - **CSS Framework:**
    - Bootstrap
    - Bulma
    - Materialize
    - Semantic
    - Tailwind CSS

    - **CSS Preprocesser:**
        - SASS
        - LESS
- **JavaScript:**
    - **JavaScript Technology:**
        - ES6
        - TypeScript
    - **JavaScript Framework:**
        - AngularJS
        - ReactJS
        - Vue.JS
    - **JavaScript Library:**
        - jQuery
        - jQuery Mobile
        - script.aculo.us
        - P5.JS
        - D3.JS
        - Underscore.JS

Backend Languages: The back-end portion is built by using some languages which are discussed below:

- **PHP:** PHP is a server-side scripting language designed specifically for web development. Since PHP code executed on the server-side, so it is called a server-side scripting language.
- **Node.js:** Node.js is an open-source and cross-platform runtime environment for executing JavaScript code outside a browser. You need to remember that NodeJS is not a framework, and it's not a programming language. Most people are confused and understand it's a framework or a programming language. We often use Node.js for building back-end services like APIs like Web App or Mobile App. It's used in production by large companies such as Paypal, Uber, Netflix, Wallmart, and so on.
- **Python:** Python is a programming language that lets you work quickly and integrate systems more efficiently.

- **Ruby:** Ruby is a dynamic, reflective, object-oriented, general-purpose programming language. Ruby is a pure Object-Oriented language developed by Yukihiro Matsumoto. Everything in Ruby is an object except the blocks but there are replacements too for it i.e procs and lambda. The objective of Ruby's development was to make it act as a sensible buffer between human programmers and the underlying computing machinery.
- **Java:** Java is one of the most popular and widely used programming languages and platforms. It is highly scalable. Java components are easily available.
- **JavaScript:** JavaScript can be used as both (front end and back end) programming.
- **Golang:** Golang is a procedural and statically typed programming language having the syntax similar to C programming language. Sometimes it is termed as Go Programming Language.
- **C# :**C# is a general-purpose, modern and object-oriented programming language pronounced as "C sharp".
- **DBMS:** The software which is used to manage database is called Database Management System (DBMS).

Back End Frameworks and Technology:

PHP:

- Framework: Laravel, CodeIgniter
- CMS: WordPress

NodeJS:

- Framework: Express

Python:

- Framework: Django
- Package Manager: Python PIP

Ruby:

- Framework: Ruby on Rails

Java:

- Framework: Spring, Hibernate

C#:

- Framework: .NET

Database:

- **Relation Database:**
  - Postgre SQL
  - MariaDB
  - MySQL
- **NoSql Database:**
  - MongoDB

Data Format: Format of data is used by web applications to communicate with each other. It is light weight text-based data interchange format which means, it is simpler to read and write.

Below are two common data formats used in web development.

- **XML:** Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- **JSON:** JSON or JavaScript Object Notation is a format for structuring data.

API: API is an abbreviation for Application Programming Interface which is a collection of communication protocols and subroutines used by various programs to communicate between them.

Web Protocols: Web protocols are set of rules followed by everyone communicating over the web.

- **HTTP:** The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.
  HTTP works as a request-response protocol between a client and server.

A web browser may be the client, and an application on a computer that hosts a web site may be the server.

- **Other Protocols:**
  - TCP/IP Model
  - UDP
  - FTP
  - SMTP
  - SOAP

Graphics: Graphical elements are one of the key features of any webpage. They can be used to convey important points better than text does and beautify the webpage.

- **Canvas:** The HTML "canvas" element is used to draw graphics via JavaScript.
- **SVG:** SVG stands for Scalable Vector Graphics.
  It basically defines vector-based graphics in XML format.

Difference between Web Application and Website

Web Application: Web application is a piece of software that can be accessed by the browser. A Browser is an application that is used to browse the internet.

Web application needs authentication. The web application uses a combination of server-side scripts and client-side scripts to present information. It requires a server to manage requests from the users.

Example: Google Apps

Website: Website is a collection of related web pages that contains images, text, audio, video, etc. It can be consisting of one page, two pages, and n number of pages.

A website provides visual and text content that users can view and read.

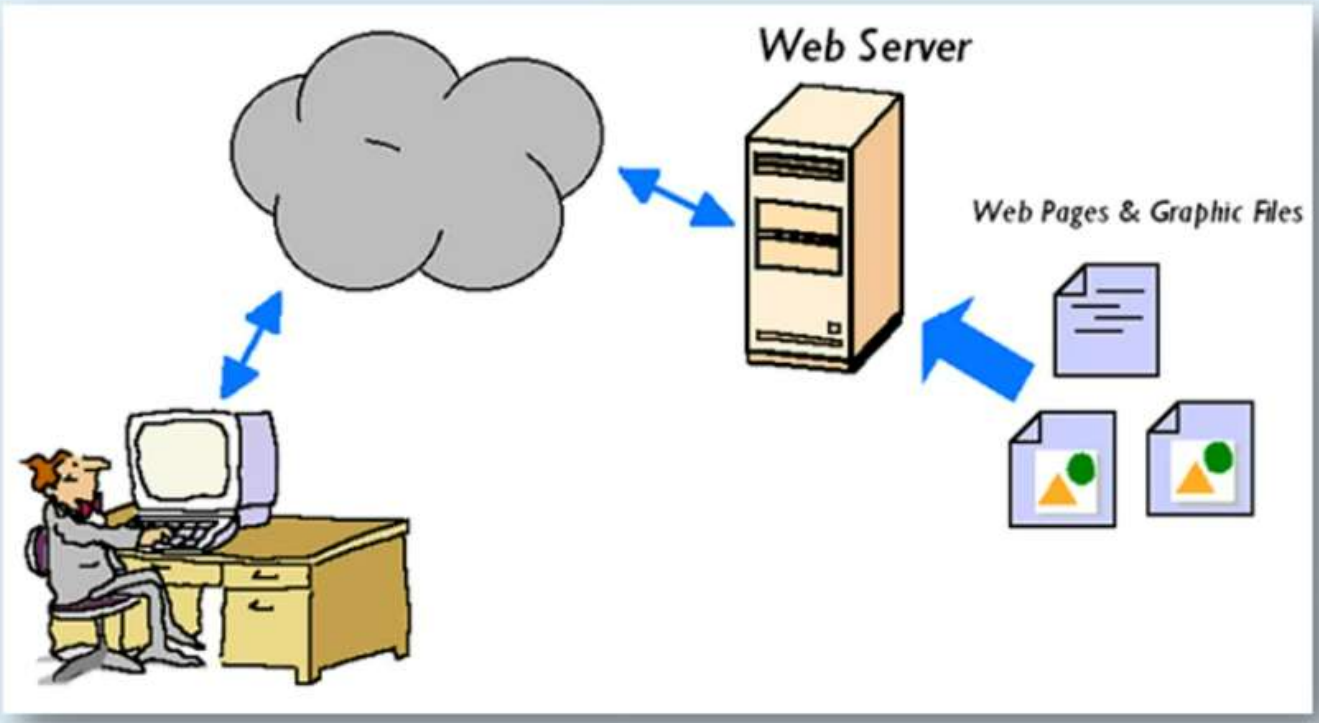
To view a website requires a browser (chrome, Firefox). There are many types of websites like Archive website, Blog, Community website, Dating website, etc.

Example: Amazon, YouTube, etc.

Types of Web Pages:

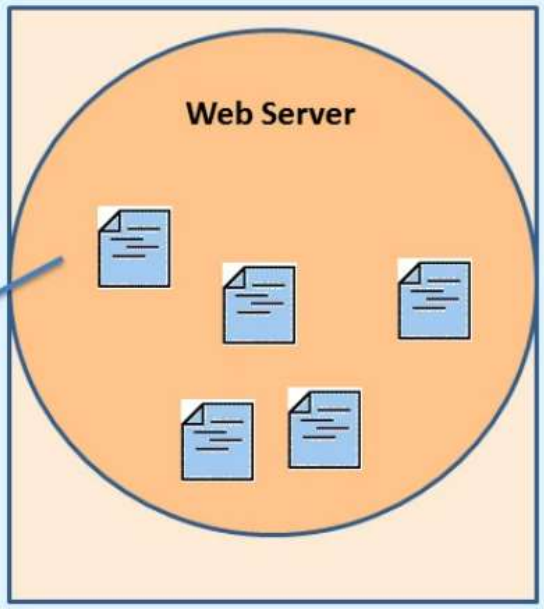
1. Static Web Pages
2. Dynamic Web Pages

Static Website

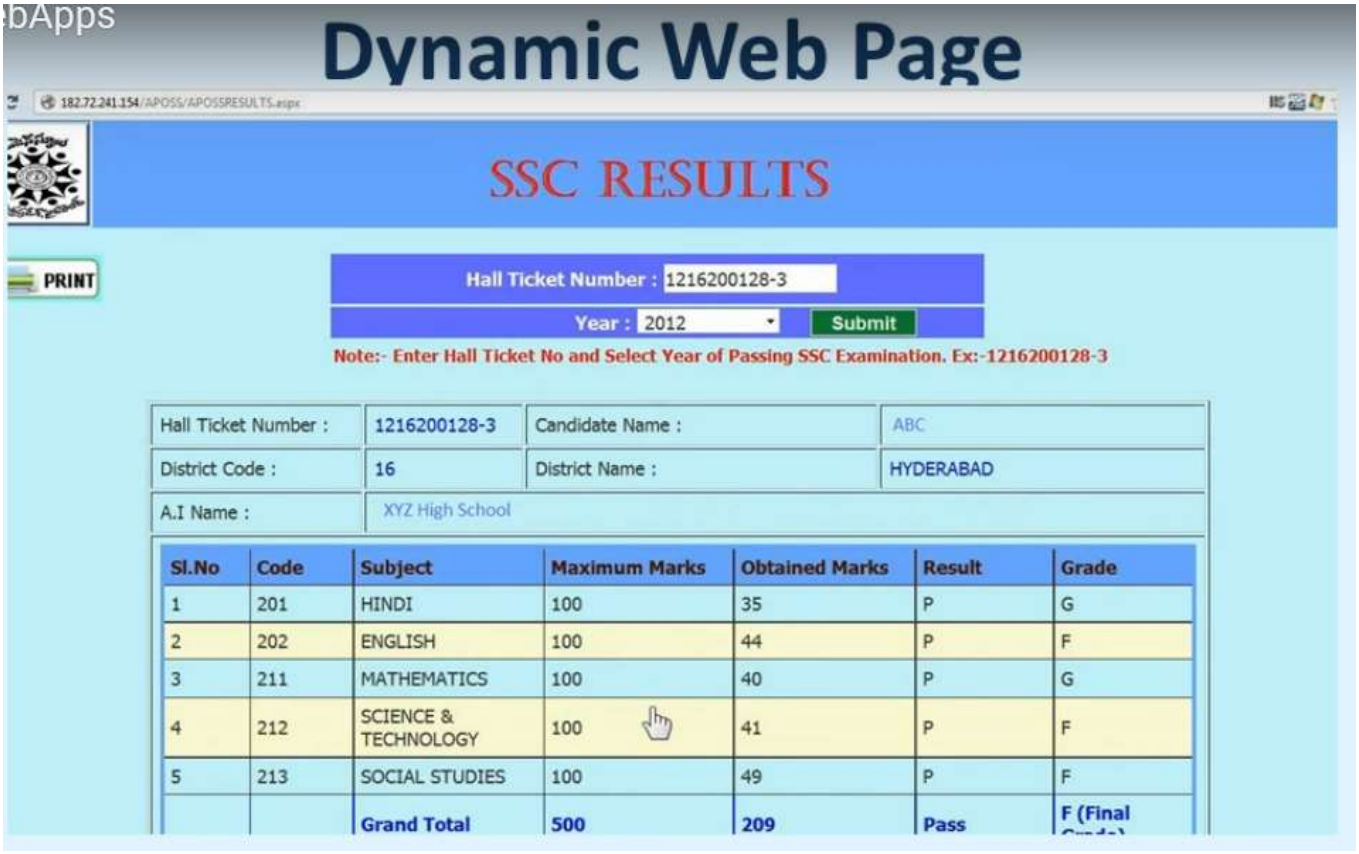


HTML
- Text
- Images
- Audio
- Video

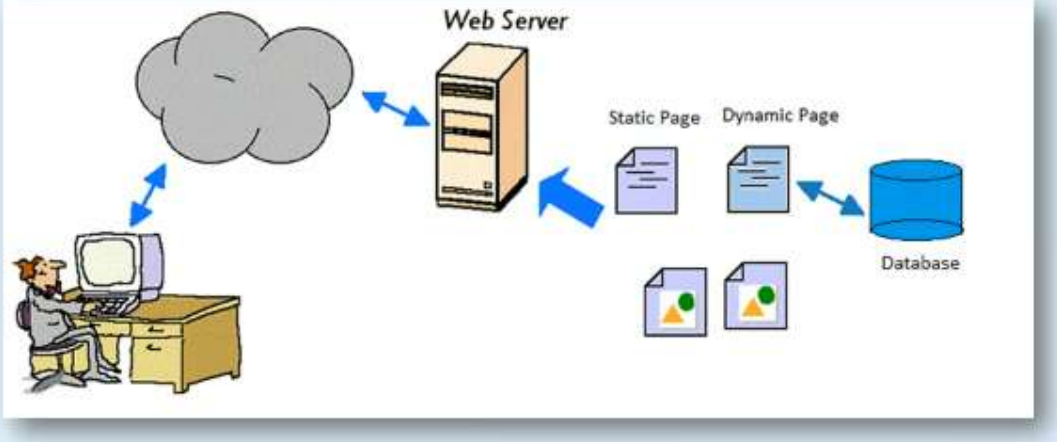
Static Website



Dynamic Web Page



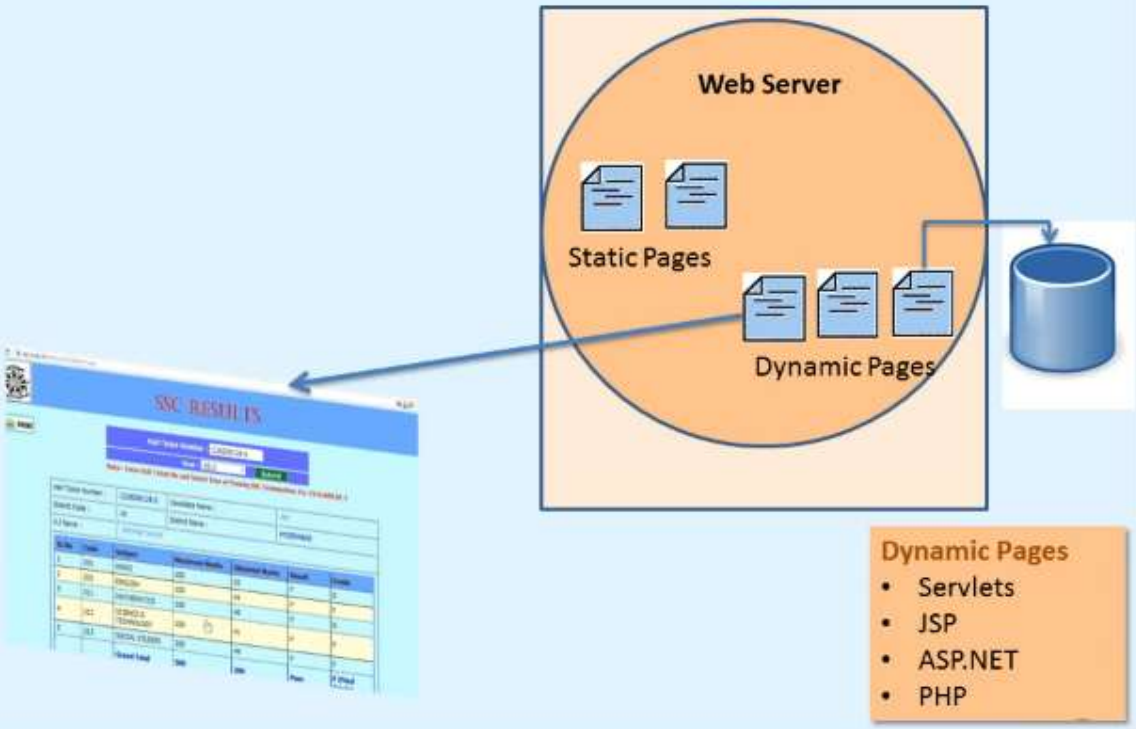
Dynamic Web Page



HTML
- Text
- Images
- Audio
- Video

Dynamic Pages
- Servlets
- JSP
- ASP.NET
- PHP

Dynamic Website



Technologies

Static Page

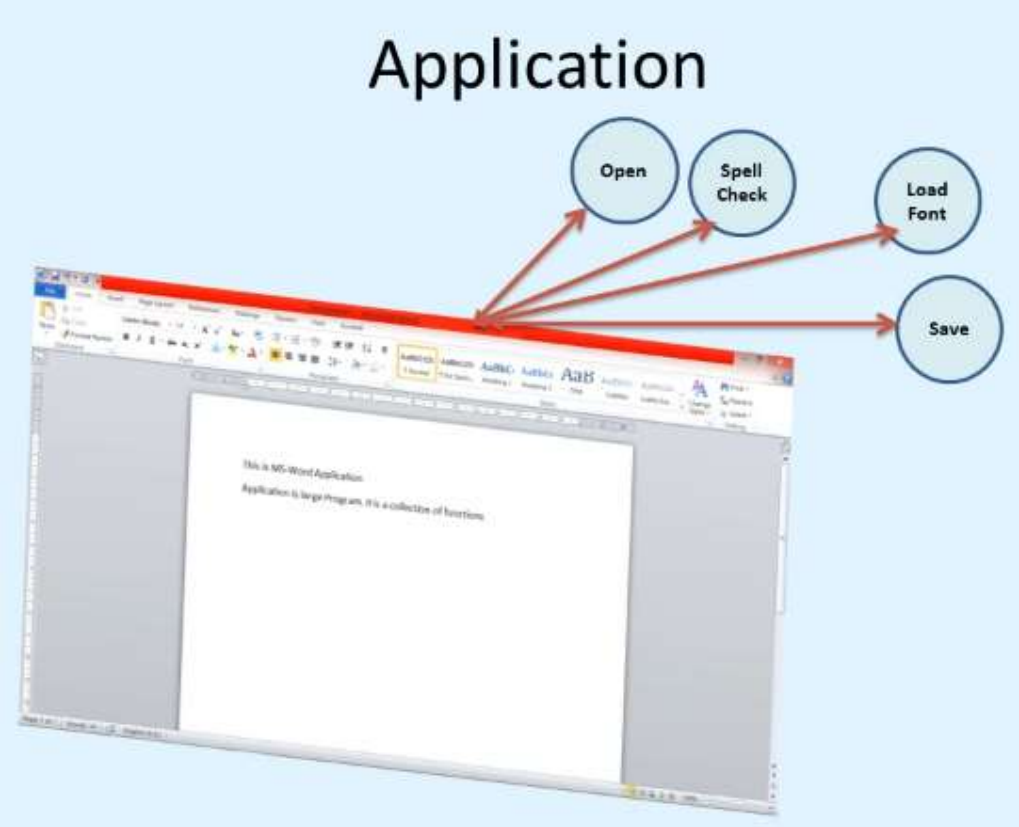
- HTML (Formatting Content/ User Interface)
- CSS (Formatting HTML)
- JavaScript (Interactivity)

Dynamic Page

- Servlet
- JSP

(Java + HTML +CSS + JavaScript)

- ASP.NET (VB.NET + HTML +CSS + JavaScript)
- PHP (PHP + HTML +CSS + JavaScript)



Web Application	Website
Web application is designed for interaction with end users.	Website basically contains static content.
The user of web application can read the content of web application and also manipulate the data.	The user of website only can read the content of website but not manipulate.
The web application site should be precompiled before deployment.	The website does not need to be precompiled.
The function of web application is quite complex.	The function of website is simple.
Web application is interactive for users.	Web site is not interactive for users.
The browser capabilities involved with web application is high.	The browser capabilities involved with web site is high.
Integration is complex for web application because of its complex functionality.	Integration is simpler for web site.

Web Application	Website
Web application mostly requires authentication	In web site authentication is not necessary.
EXAMPLE: - Amazon, Facebook, etc.	EXAMPLE: - Breaking News, Aktu website, etc.

Introduction to PHP

PHP is a server-side and general-purpose scripting language that is especially suited for web development.

PHP originally stood for **Personal Home Page**. However, now, it stands for **Hypertext Preprocessor**. It's a recursive acronym because the first word itself is also an acronym.

PHP was created by Rasmus Lerdorf in 1994. It's currently maintained by the PHP Development Team.

Version	Release Date
1.0	8 June 1995
2.0	1 November 1997
3.0	6 June 1998
4.0	22 May 2000
5.0	13 July 2004
6.x	cancel
7.0	3 December 2015
7.1	1 December 2016
7.2	30 November 2017
7.3	6 December 2018
7.4	28 November 2019
8.0	26 November 2020

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

PHP is a server-side language

When you open a website on your web browser, for example, https://www.saipali.education

The web browser sends an HTTP request to a web server where saipali.education locates. The web server receives the request and responds with an HTML document.

In this example, the web browser is a client while the web server is the server. The client requests for a page, and the server serves the request.

PHP runs on the web server, processes the request, and returns the HTML document.

PHP is a general-purpose language

When it comes to the purpose of the programming languages, there are two main types: domain-specific and general-purpose languages.

The domain-specific languages are used within specific application domains. For example, SQL is a domain-specific language. It's used mainly for querying data from relational databases. And SQL cannot be used for other purposes.

On the other hand, PHP is a general-purpose language because PHP can develop various applications.

PHP is a cross-platform language

PHP can run on all major operating systems, including Linux, Windows, and macOS.

You can use PHP with all leading web servers such as Nginx, OpenBSD, and Apache. Some cloud environments also support PHP like Microsoft Azure and Amazon AWS.

PHP is quite flexible. It’s not just limited to processing HTML. PHP has built-in support for generating PDF, GIF, JPEG, and PNG images.

One notable feature of PHP is that it supports many databases, including MySQL, PostgreSQL, MS SQL, db2, Oracle Database, and MongoDB.

What can PHP do

PHP has two main applications:

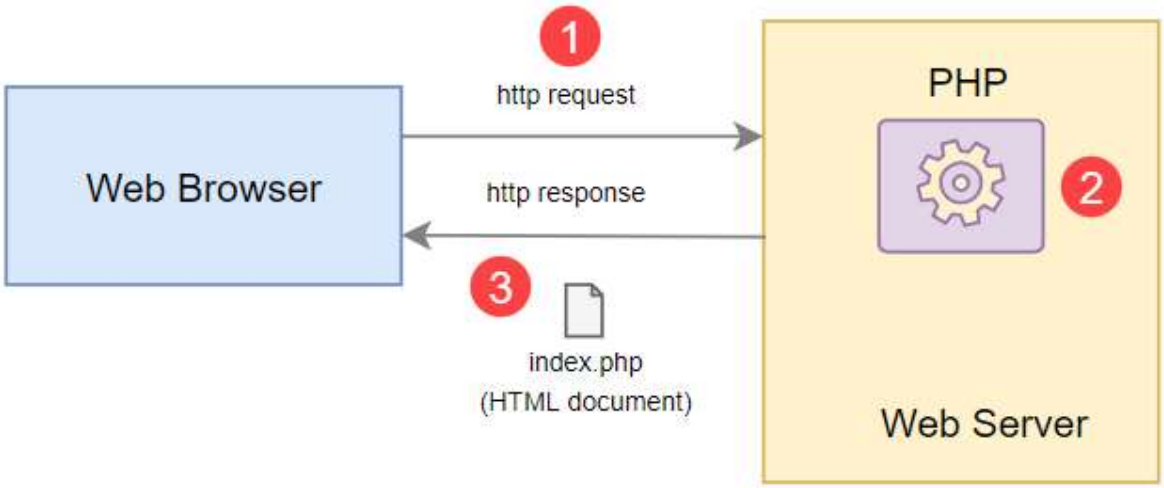
- Server-side scripting – PHP is well-suited for developing dynamic websites and web applications.
- Command-line scripting – like Python and Perl, you can run PHP script from the command line to perform administrative tasks like sending emails and generating PDF files.

Global Websites that uses PHP:

- Facebook
- Yahoo
- Wikipedia
- WordPress
- MailChimp
- Flickr
- Flipkart

How PHP Works

The following illustrates how PHP works:



How PHP works:

- First, the web browser sends an HTTP request to the web server, e.g., index.php.
- Second, the PHP preprocessor that locates on the web server processes PHP code to generate the HTML document.
- Third, the web server sends the HTML document back to the web browser.

Advantages of PHP

Since PHP is designed for the web in the first place, it brings many advantages to web development:

- Simple – PHP is quite easy to learn and get started.
- Fast – PHP websites typically run very fast.
- Stable – PHP is stable since it has been in existence for a long time.
- Open-source and free – PHP is open source and free. It means that you don't have to pay a license fee to use PHP to develop software products.
- Community support – PHP has an active online community that helps you whenever you face an issue.

Open Source

- PHP is an open-source project – the language is developed by a worldwide team of volunteers who make its source code freely available on the Web, and it may be used without payment of licensing fees or investments in expensive hardware or software.

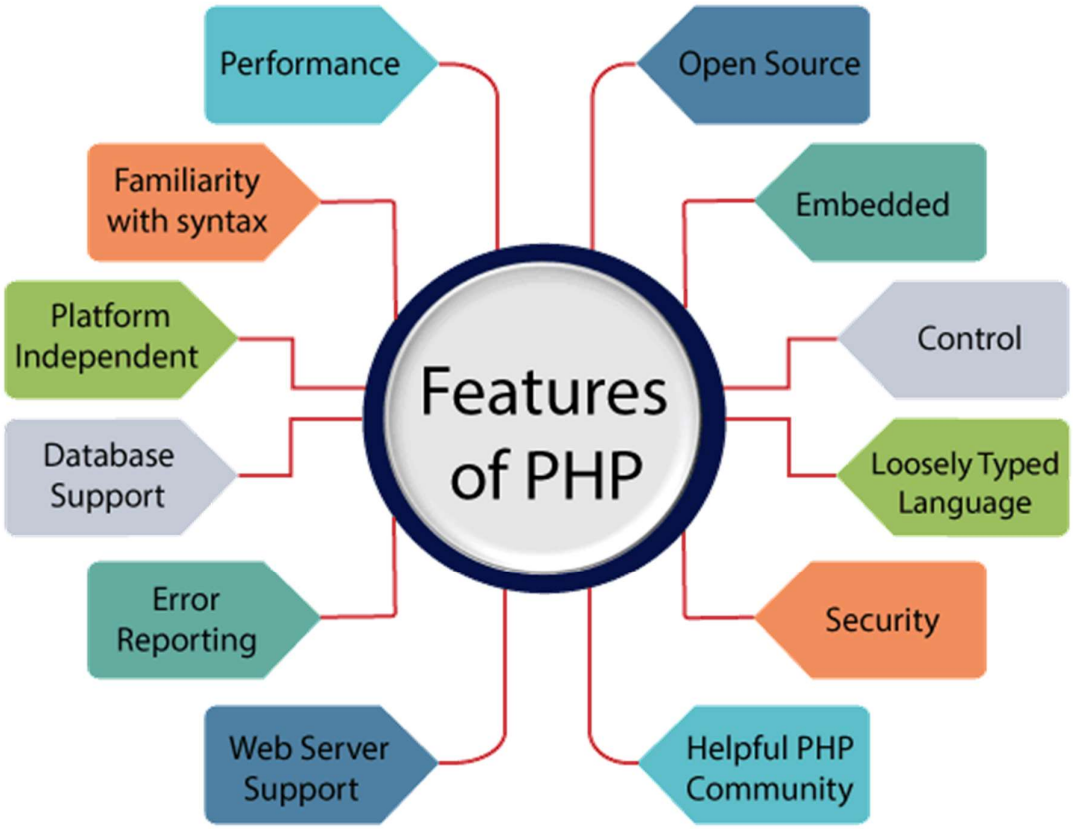
- This reduces software development costs without affecting either flexibility or reliability. The open-source nature of the code further means that any developer, anywhere, can inspect the code tree, spit errors, and suggest possible fixes, this produces a stable, robust product wherein bugs, once discovered, are rapidly resolved – sometimes within a few hours of discovery!

Third-Party Application Support

- One of PHP's Strengths has historically been its support for a wide range of different databases, including MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.
- PHP 5.3 Supports more than fifteen different database engines, and it includes a common API for database access.
- XML support makes it easy to read and write XML documents though they were native PHP data structures, access XML node collections using Xpath, and transform XML into other formats with XSLT style sheets.

PHP Features

PHP is very popular language because of its simplicity and open source. There are some important features of PHP given below:



Performance:

PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

Open Source:

PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

Familiarity with syntax:

PHP has easily understandable syntax. Programmers are comfortable coding with it.

Embedded:

PHP code can be easily embedded within HTML tags and script.

Platform Independent:

PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.

Database Support:

PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

Error Reporting -

PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.

Loosely Typed Language:

PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

Web servers Support:

PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

Security:

PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

Control:

Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

A Helpful PHP Community:

It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning PHP from the communities is one of the significant benefits

Applications of PHP

As mentioned before, PHP is one of the most widely used language over the web. I'm going to list few of them here:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

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

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

Web Development

PHP is widely used in web development nowadays. PHP can develop dynamic websites easily. But you must have the basic the knowledge of following technologies for web development as well.

- HTML
- CSS
- JavaScript
- Ajax
- XML and JSON
- jQuery

PHP Server

The PHP Community Provides Some types of Software Server solution under The GNU (General Public License).

These are the following:

1. WAMP Server
2. LAMP Server
3. MAMP Server
4. XAMPP Server

All these types of software automatic configure inside operating system after installation it having PHP, MySQL, Apache and operating system base configuration file, it doesn't need to configure manually.

Server	Stands for
WAMP	Microsoft window o/s, Apache MySQL PHP
LAMP	Linux Operating System Apache MySQL PHP

MAMP	Mac os Apache MySQL PHP
XAMPP	x-os(Cross Operating System) Apache MySQL PHP Perl. It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.
SAMP	Solaris

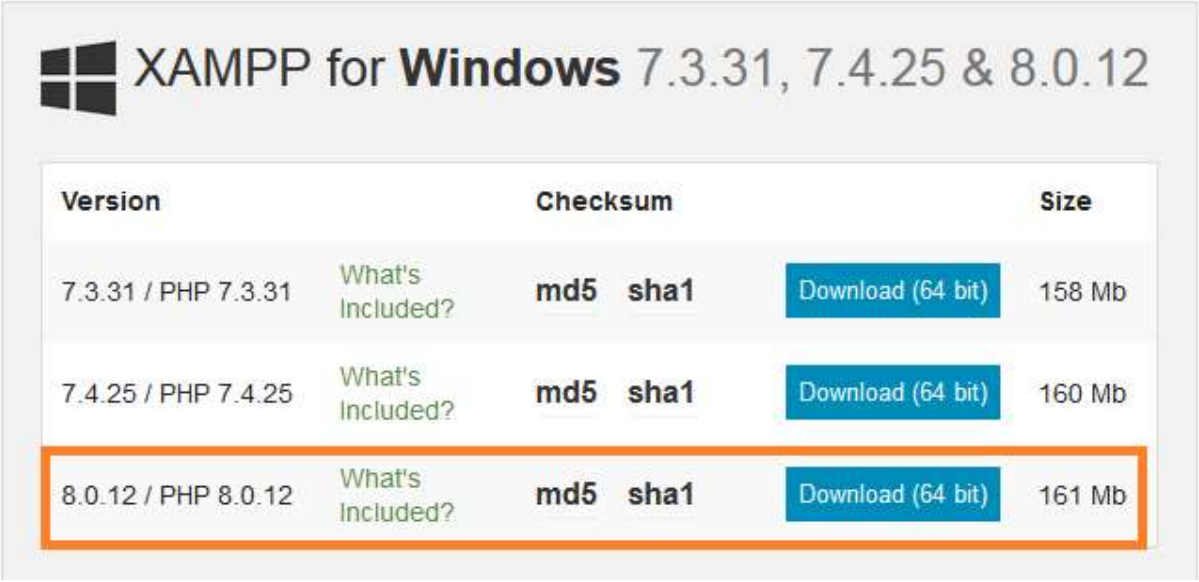
Note:

A cross-platform computer product or system is a product or system that can work across multiple types of platforms or operating environments

PHP XAMPP Installation

Download XAMPP from the following URL:

https://www.apachefriends.org/download.html



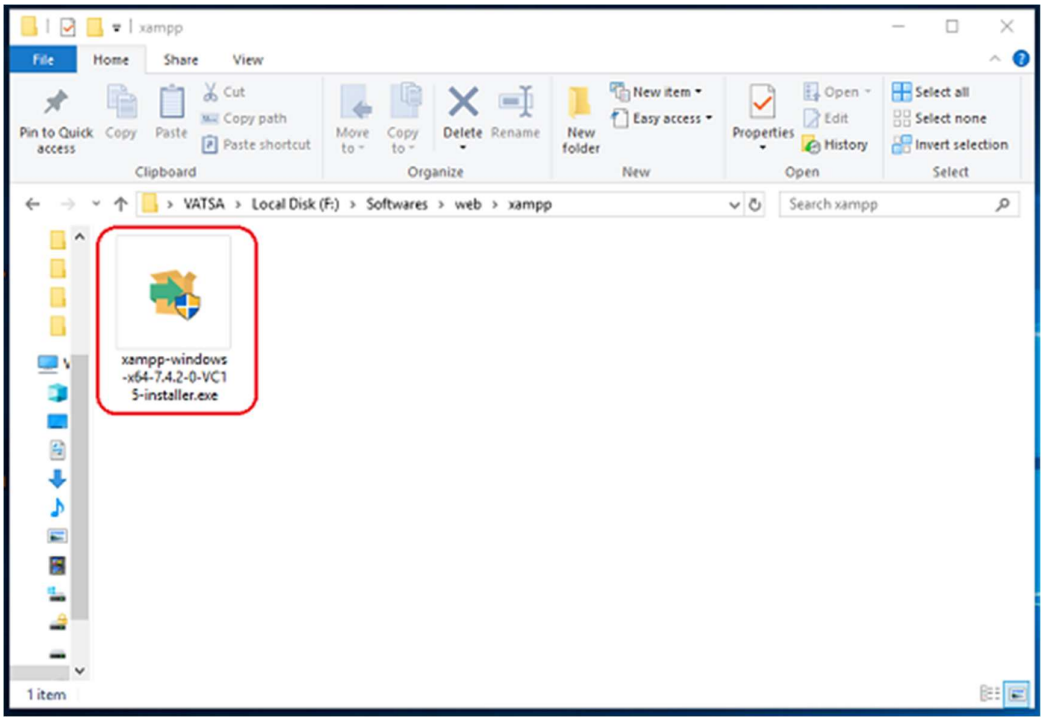
How to install XAMPP server on windows

We will learn how to install the XAMPP server on windows platform step by step. Follow the below steps and install the XAMPP server on your system.

Step 1: To download the XAMPP server, visit the "Apache Friends" website in your web browser.

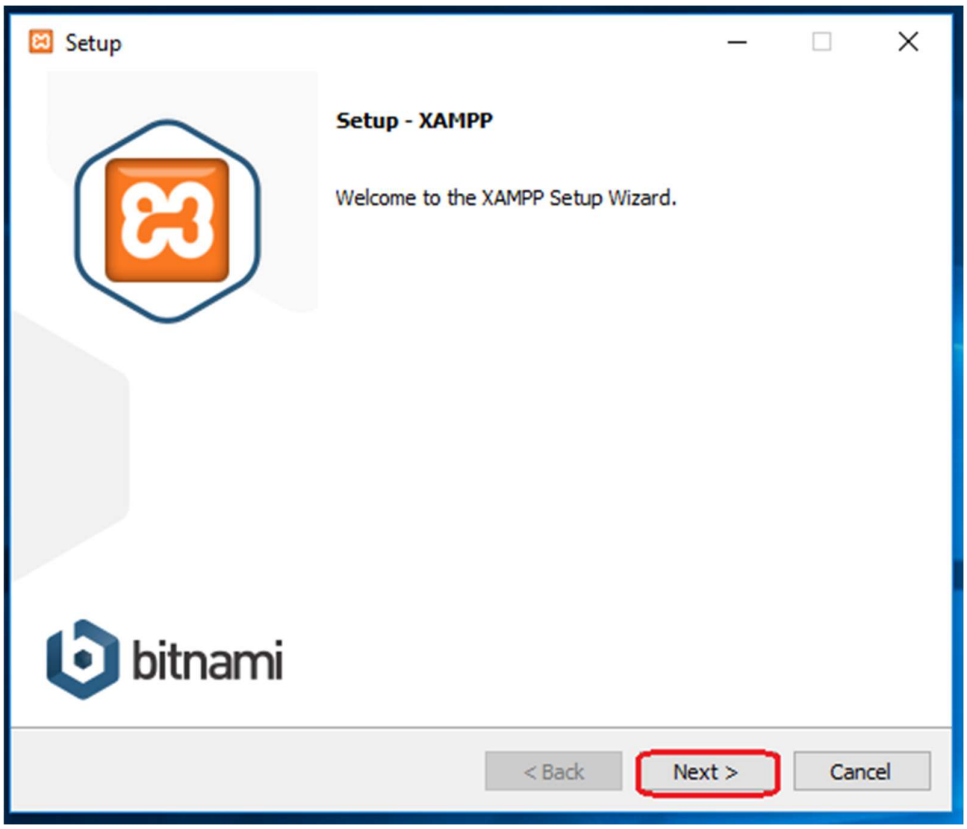
Step 2: Click on the above link provided to download the **XAMPP server** according to your window requirement.

Step 3: Double-click the downloaded file to launch the XAMPP installer.



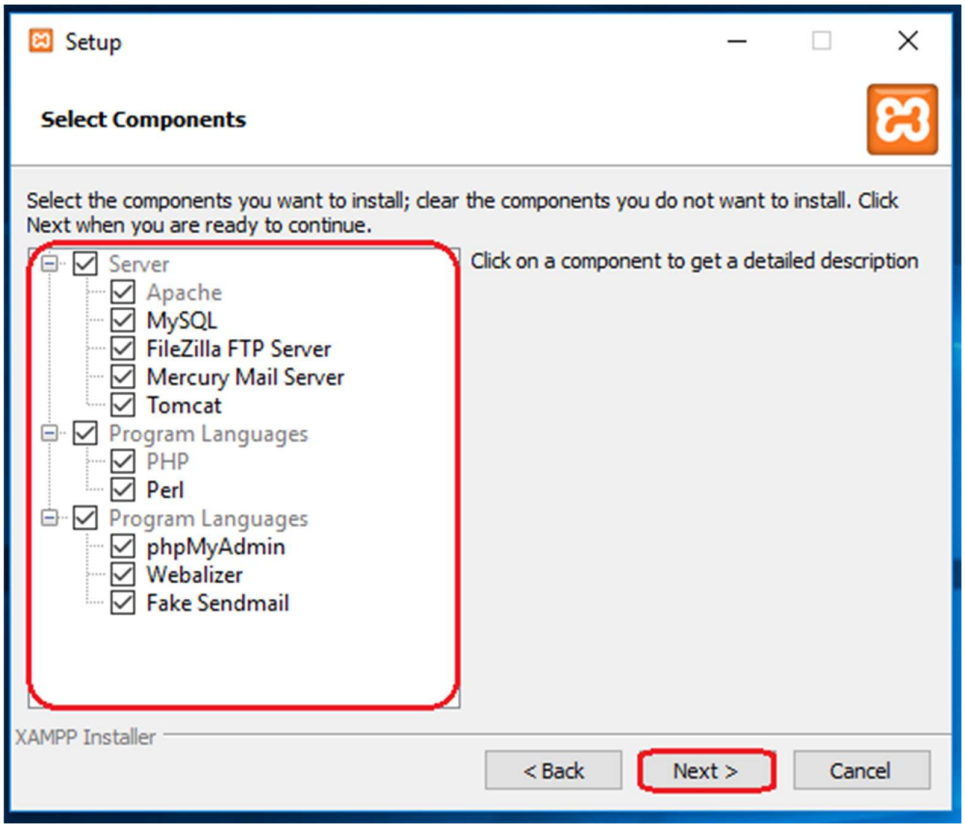
Step 4

"Setup" window will appear on the screen. Then, click on the "Next" button.



Step 5

Select the components that you want to install and click on the "Next" button.

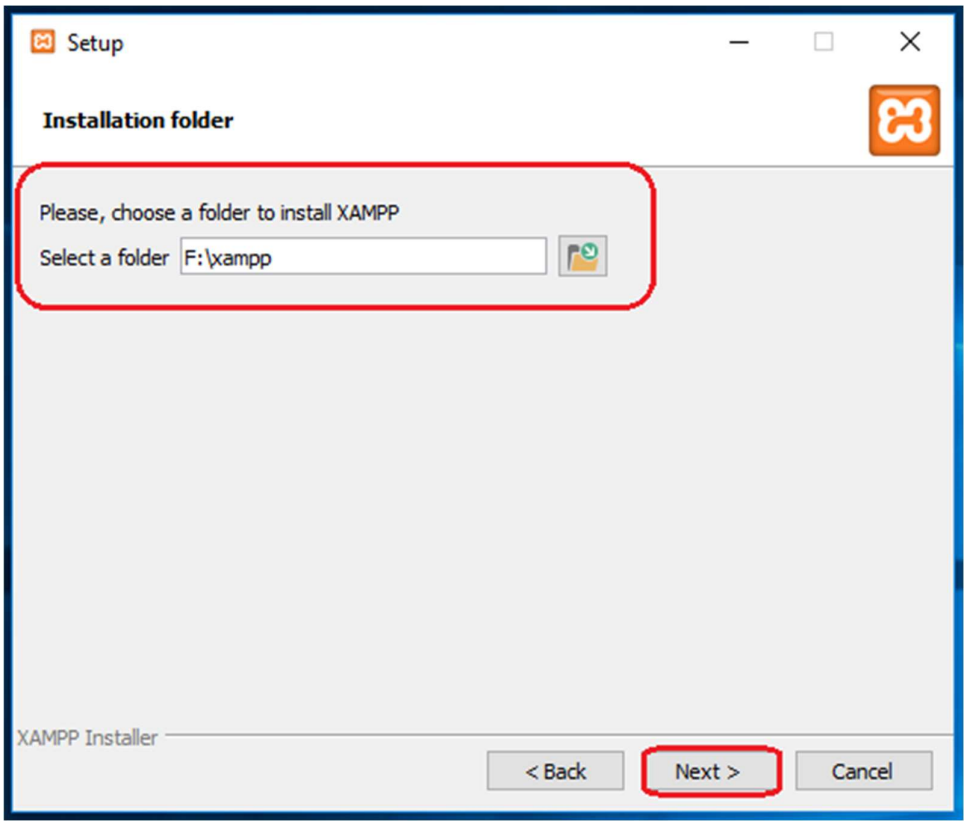


Note

By default, all components are selected in your XAMPP installation.

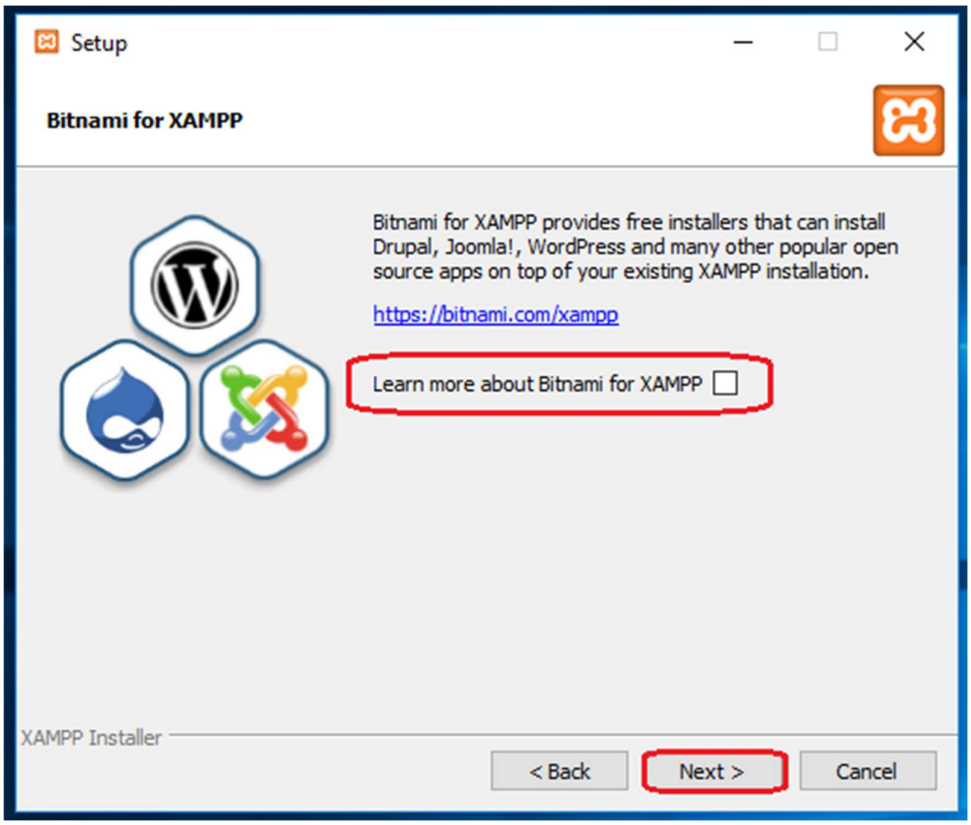
Step 6

Choose a folder to install the XAMPP and click on the "Next" button.



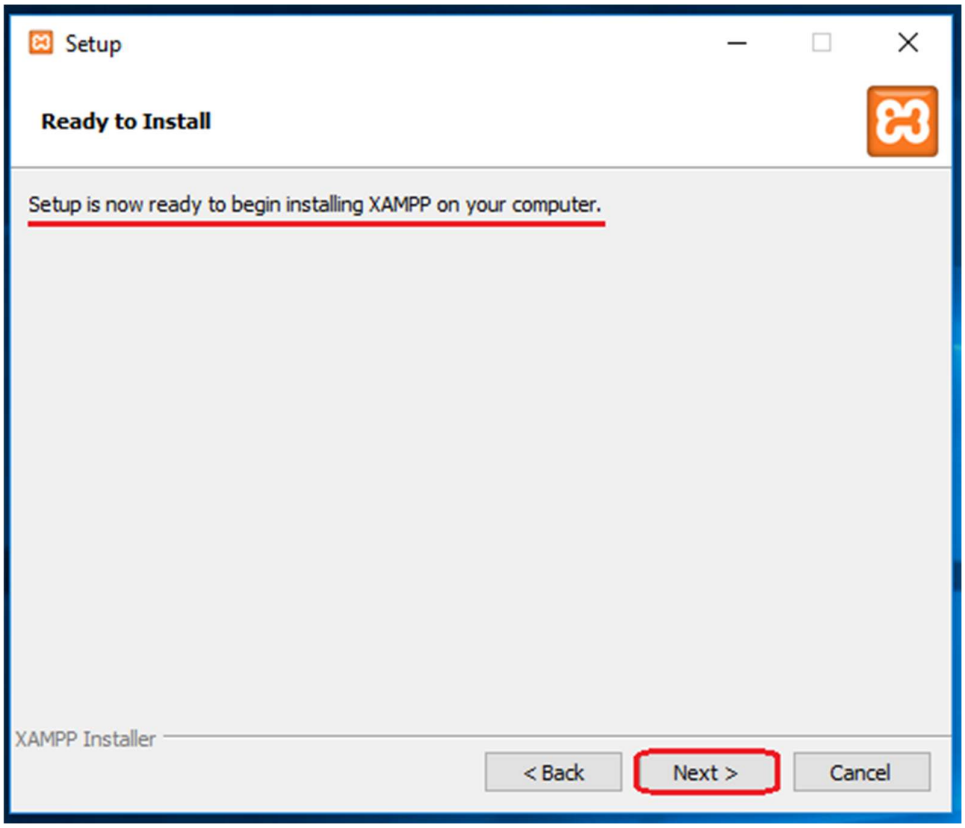
Step 7

Uncheck the "Learn more about Bitnami for XAMPP" option and click on the "Next" button.



Step 8

"Ready to Install" window will appear on the screen, then click on the "Next" button.



Step 9

Click on the "Finish" button.

Step 10

Select a language. (either English or German) and click on the "Save" button.

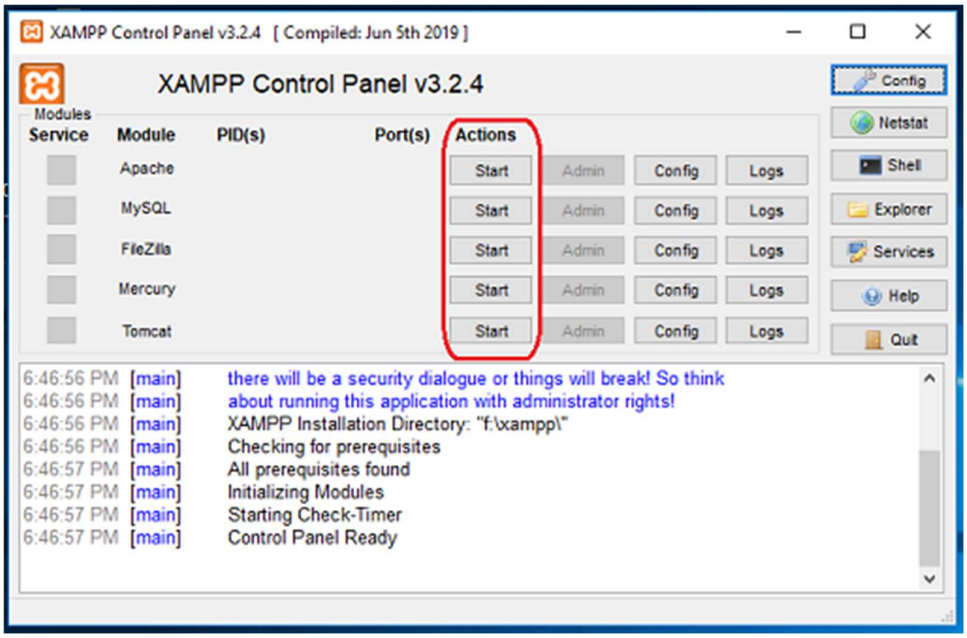
Configuration Process of XAMPP Server

Step 1

Start the XAMPP control panel through the "Run as administrator" option.

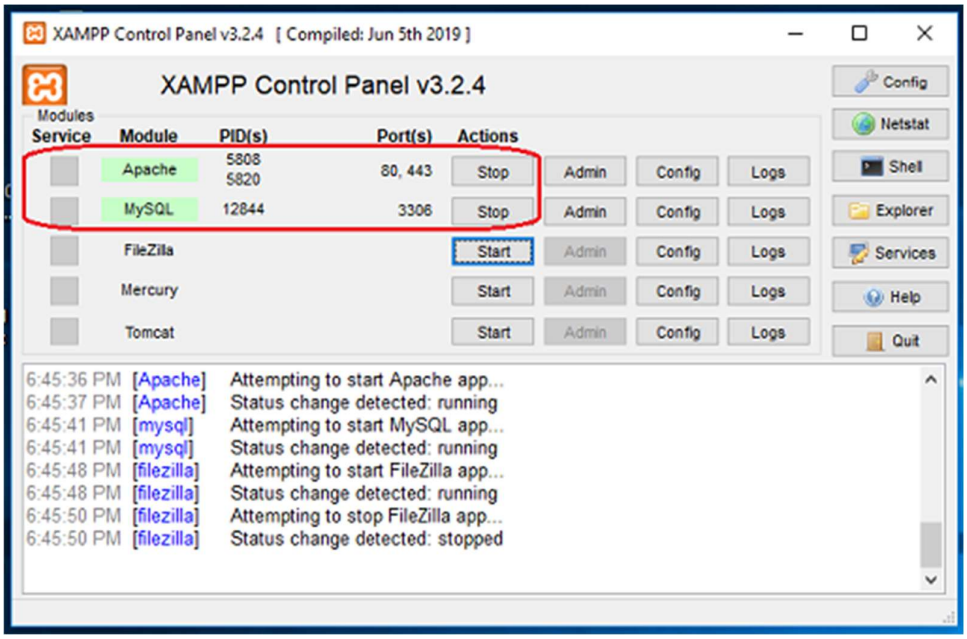
Step 2

"XAMPP Control Panel" will appear on the screen and click on "Start" action to start the "Apache" and "MySQL" modules.



Note: -

The default XAMPP server settings should work for most users. When you start the related modules (services) then, the color of the related modules (service) becomes change into the green color and the PID(s) and the Port(s) number will also be shown to the user.



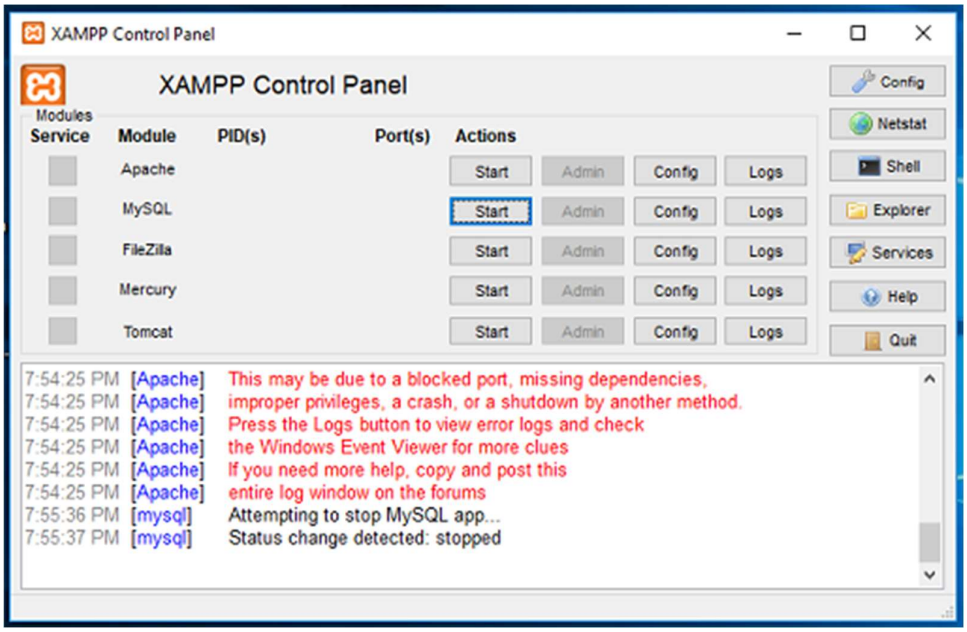
Fix an Issue of Apache not Starting in XAMPP

Some of the users may face an issue with Apache and MySQL module services not working in XAMPP. However, depending on the setup configuration or usage of your system, you may need to change the port number of the Apache and the MySQL. For example, because the "World Wide Web Services" under "Internet Information Services (IIS)" is also run on port 80 in your system, which is also the default port of apache in XAMPP. And, the two servers (applications) can not use (run) the same port simultaneously. Follow the instructions below to fix the problem with Apache and MySQL services.

Method 1. Change the default port of Apache

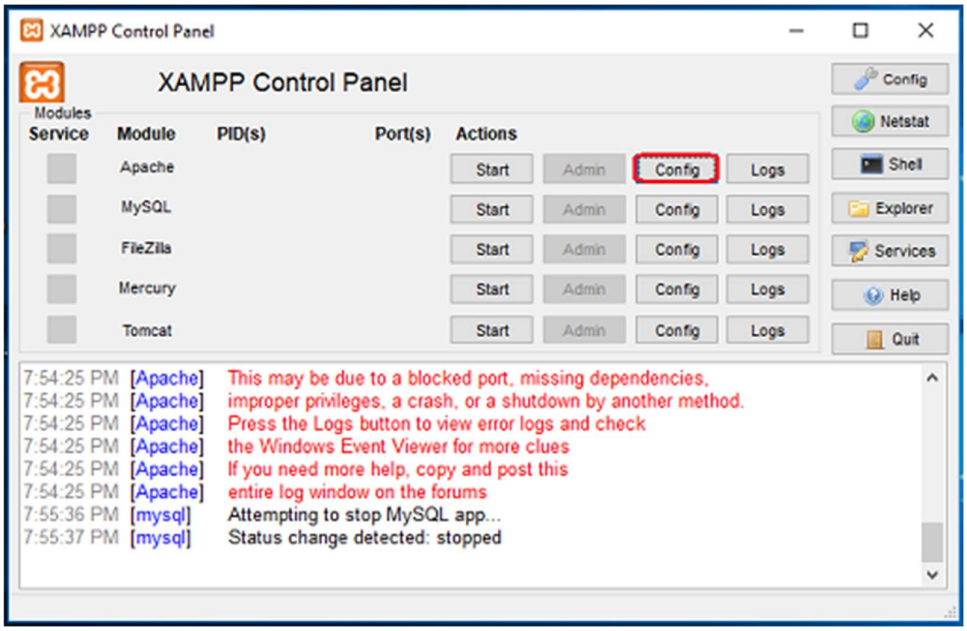
Step 1

Open the XAMPP Control Panel.



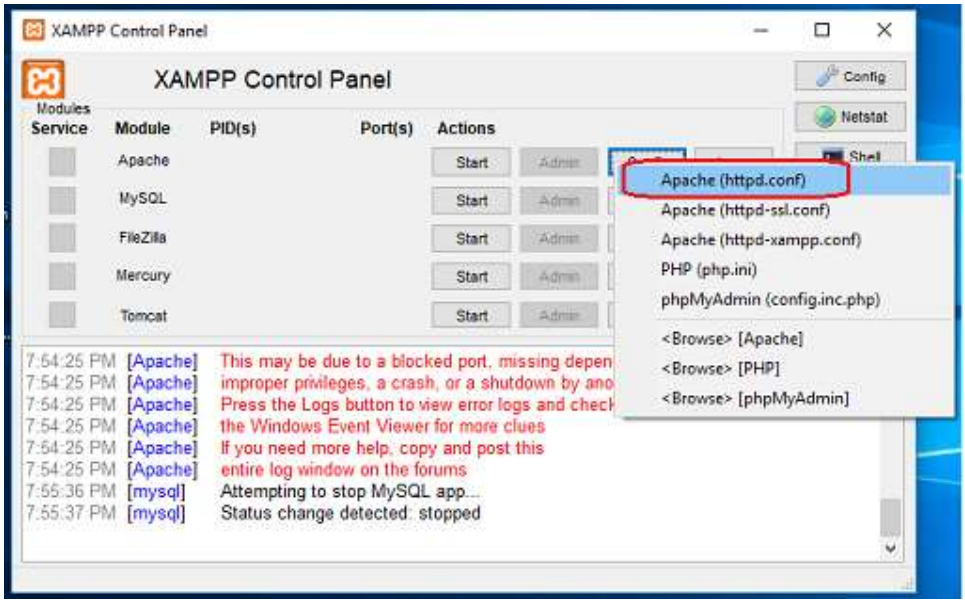
Step 2

In Apache Module Service, click on the "Config" button.



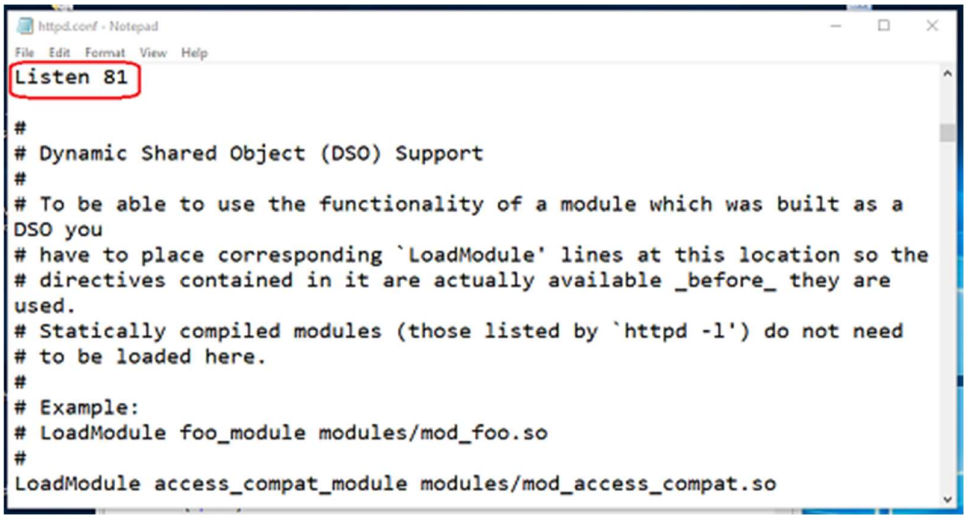
Step 3

Click on "Apache (httpd.conf)" option.



Step 4

By pressing the "Ctrl + F" key, find the "Listen 80" and replace it with another open port (like 81 or 9080) and save the file.

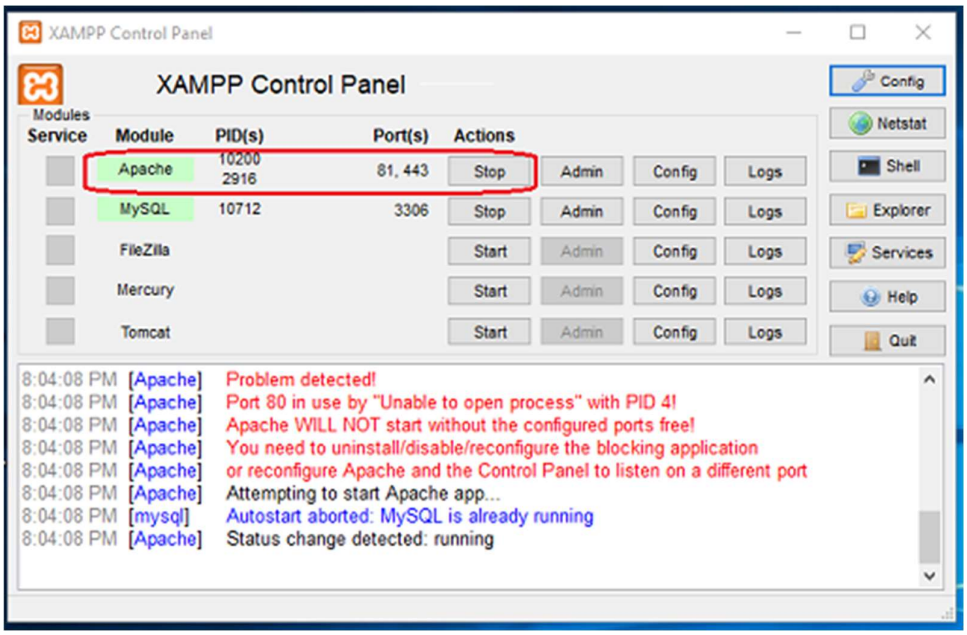


Step 5

Restart the XAMPP Server.

Step 6

Again, open the "XAMPP Control Panel" and click on the start option under the "Apache" module services.



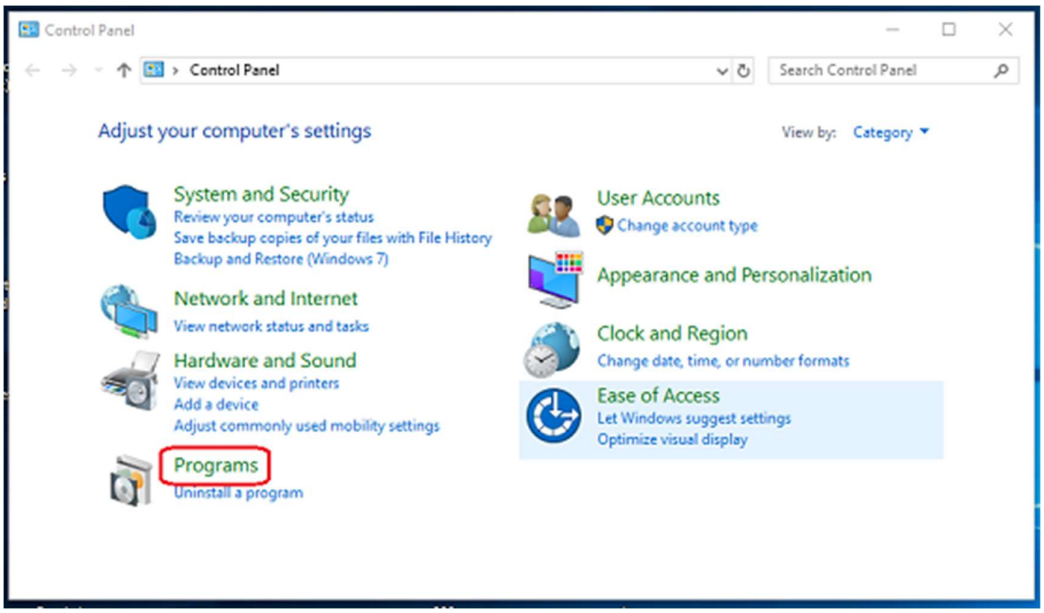
Method 2. By disabling the Internet Information Services (IIS)

Step 1

From the Start Menu, search for Control Panel.

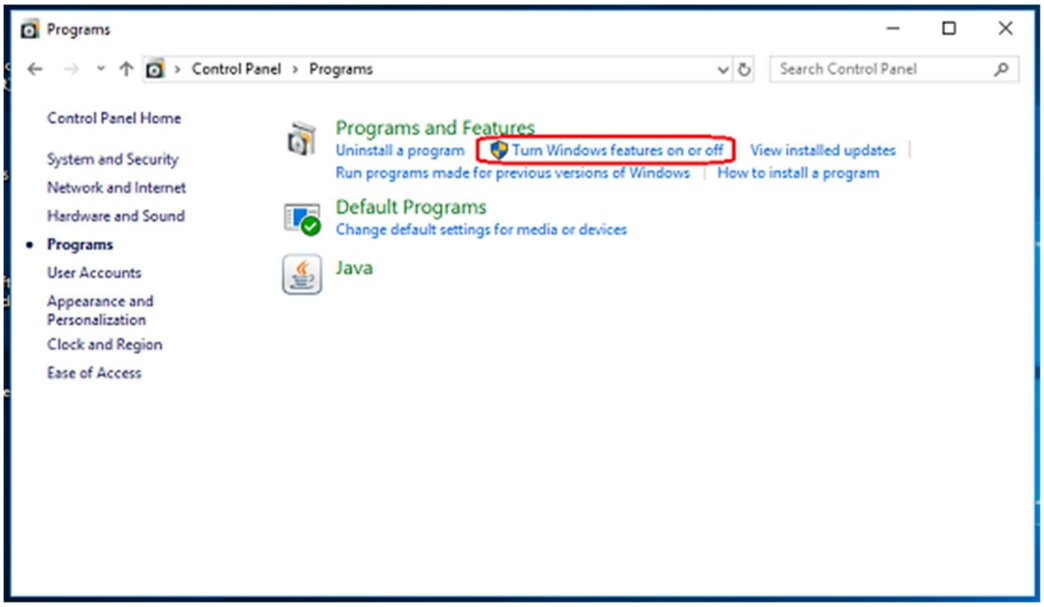
Step 2

Click on "Programs".



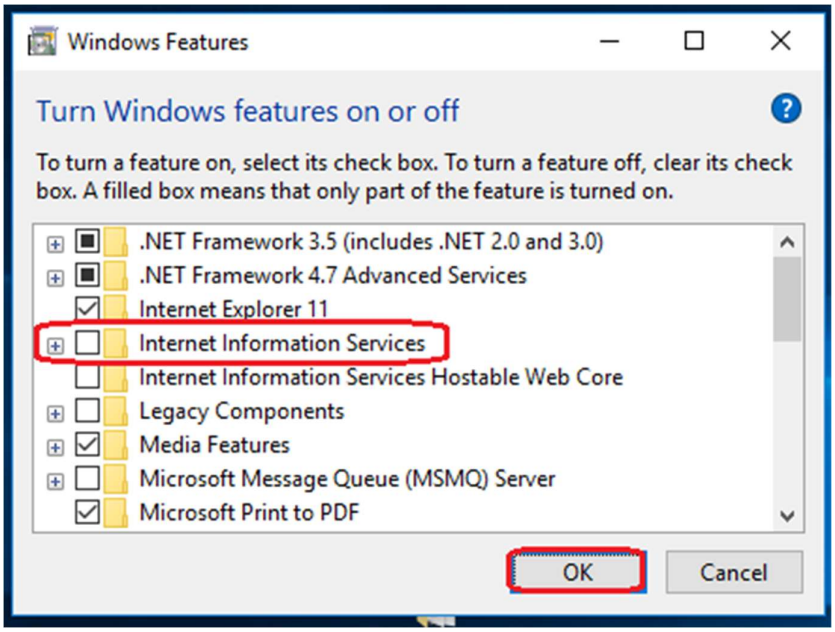
Step 3

Click on the "Turn Windows Feature on or off" option.



Step 4

Uncheck (clear) the "Internet Information Services" option and click on the "OK" button.

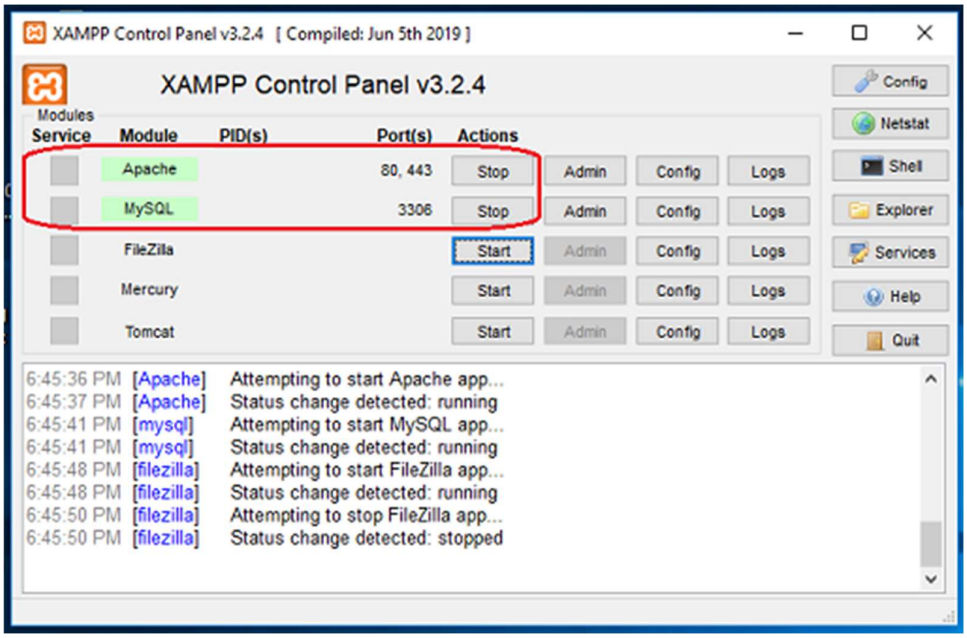


Step 5

Click on the "Restart Now" option (This will reboot/restart your computer/laptop).

Step 6

Again, open the "XAMPP Control Panel" and click on the start option in the "Apache" module services.



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:

First Program in PHP:

```
<!DOCTYPE html>
<html>

  <head>
    <title>Hello World</title>
  </head>

  <body>
    <?php
        echo "Hello, World!";
     ?>
  </body>

</html>
```

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>
```

Note: However; all variable names are case-sensitive!

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 Comments

Comments in PHP

A comment in PHP code is a line that is not executed as a 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 your code
- 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

Syntax for single-line comments:

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

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

# This is also a single-line comment
?>

</body>
</html>
```

Example

Syntax for multiple-line comments:

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

```
<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>

</body>
</html>
```

Example

Using comments to leave out parts of the code:

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

<?php
// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>

</body>
</html>
```

PHP Data Types

A Data type is the classification of data into a category according to its attributes;

- Alphanumeric characters are classified as strings
- Whole numbers are classified integers
- Numbers with decimal points are classified as floating points.
- True or false values are classified as Boolean.

PHP is a loosely typed language; it does not have explicit defined data types. PHP determines the data types by analyzing the attributes of data supplied. PHP implicitly supports the following data types

- Integer – whole numbers e.g. -3, 0, 69. The maximum value of an integer is platform-dependent. On a 32 bit machine, it's usually around 2 billion. 64 bit machines usually have larger values. The constant PHP_INT_MAX is used to determine the maximum value.

```
<?php
echo PHP_INT_MAX;
?>
```

Output:

```
9223372036854775807
```

- Floating point number – decimal numbers e.g. 3.14. they are also known as double or real numbers. The maximum value of a float is platform-dependent. Floating point numbers are larger than integers.
- Character string – e.g. Hello World
- Boolean – e.g. True or false.

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:

Example

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 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 = "Sai Pali Institute of Technology & Management";
echo "I Like $txt!";
?>
```

Example

```
<?php
$txt = " Sai Pali Institute of Technology & Management ";
```

```
echo "I Like " . $txt . "!";
?>
```

The following example will output the sum of two variables:

Example

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

Run PHP from command line:

Goto d:\xampp\htdocs\dse002

D:\xampp\htdocs\dse002> php p2.php

Note: If it doesn't work then change the path environment variable as d:\xampp\php

```
<?php
// Input section
$a = (int)readline('Enter an integer: ');
// read real number
$b = (float)readline('Enter a floating point number: ');
echo "Entered integer is ". $a.  " and entered float is " . $b;
?>
```

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

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

Variable with global scope:

```
<?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>";
?>
```

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

Example

Variable with local scope:

```
<?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>";
?>
```

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

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:

```
<?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();
myTest();
myTest();
?>
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

isset vs empty vs is_null in PHP

In PHP, isset vs empty vs is_null functions are most important part to start learning PHP programming.

isset(), empty() and is_null() functions are PHP in-built functions which are mainly used to check the value of the variable or initialization of the variable.

- **isset()** is used to check the variable has a value or not and also the variable is not NULL.
- **empty()** checks the given variable is empty. And it returns a Boolean value.
- **is_null()** is used to check the variable has a NULL value.

PHP isset() Function

isset() is a inbuilt PHP function which is used to check the value of the variable is set or not.

The function returns the result as Boolean value True or False.

Syntax:

```
isset(variable);
```

Example of PHP isset() Function

```
<?php
```

```
// PHP code to check without set variable value
$variable;

if( isset( $variable ) ) {
   echo "The variable value is set, checked by isset()";
}else{
   echo "The variable value is not set, checked by isset()";
}

echo "<br>";

// PHP code to check with set variable value
$variable2 = 'hello world';

if( isset( $variable2 ) ) {
   echo "The variable value is set, checked by isset()";
}else{
   echo "The variable value is not set, checked by isset()";
}
?>
```

Output:

```
The variable value is not set, checked by isset()
The variable value is set, checked by isset()
```

PHP empty () Function

empty () is also an PHP inbuilt function which is used to check the variable's value is empty or not.

If the value is empty then empty () function returns TRUE and if value is not empty then it returns FALSE.

Syntax:

```
empty( $variable )
```

Example:

```
<?php
// PHP Code to check first variable
$variable;
if( empty( $variable ) ) {
   echo "The variable value is empty, checked by empty()";
}else{
   echo "The variable value is not empty, checked by empty()";
}

echo "<br>";

// PHP Code to test variable with empty()

$variable2 = 'PHP world!';

if( empty( $variable2 ) ) {
   echo "The variable value is empty";

}else{
   echo "The variable value is not empty";
}
?>
```

Output:

```
The variable value is empty, checked by empty()
The variable value is not empty
```

PHP is_null() Function

PHP is_null() Function is used to find the variable value is NULL or not.

This function returns the result as a Boolean form (TRUE / FALSE).

Syntax:

is_null($variable)

Example:

```
<?php
// PHP Code or scritp to explain php is_null() function

$variable = null;

if( is_null( $variable ) ) {
   echo "The variable value is null";
}else{
   echo "The variable value is not null";
}

echo "<br>";

// PHP Code to test variable with is_null()
$variable2 = '20';

if( is_null( $variable2 ) ) {
   echo "The variable value is null";
}else{
   echo "The variable value is not null";
}
?>
```

Output:

```
The variable value is null
The variable value is not null
```

Output:

The variable value is null

The variable value is not null

Complete Difference Between isset, empty, and is_null in PHP by Using Table

Functions	“”	“phpcoder”	NULL	FALSE	0	undefined
empty()	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE
is_null()	FALSE	FALSE	TRUE	FALSE	FALSE	ERROR
isset()	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE

Difference Between isset, empty and is_null in PHP

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):

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.";
?>
```

Display Variables

The following example shows how to output text and variables with the echo statement:

Example

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "saipali.education";
$x = 5;
$y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

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):

Example

```php
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br />";
print "I'm about to learn PHP!";
?>
```

Display Variables

The following example shows how to output text and variables with the print statement:

Example

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "saipali.education";
```

```
$x = 5;
$y = 4;

print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
```

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:

Example

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

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

```
echo $y;
?>
```

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:

Example

```
<?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.

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

Example

```
<?php
$x = 10.365;
```

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

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:

Example

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

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

A class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

Example

```
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

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:

Example

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

PHP Strings

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

PHP String Functions

Here 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.

Example

Return the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

str_word_count() - Count Words in a String

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

Example

Count the number of word in the string "Hello world!":

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

strrev() - Reverse a String

The PHP strrev() function reverses a string.

Example

Reverse the string "Hello world!":

```
<?php
echo strrev("Hello world!"); // outputs !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.

Example

Search for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

Tip: The first character position in a string is 0 (not 1).

str_replace() - Replace Text Within a String

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

Example

Replace the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

PHP Math

PHP has a set of math functions that allows you to perform mathematical tasks on numbers.

PHP pi() Function

The `pi()` function returns the value of PI:

Example

```
<?php
echo(pi()); // returns 3.1415926535898
?>
```

PHP min() and max() Functions

The min() and max() functions can be used to find the lowest or highest value in a list of arguments:

Example

```
<?php
echo(min(0, 150, 30, 20, -8, -200));  // returns -200
echo(max(0, 150, 30, 20, -8, -200));  // returns 150
?>
```

PHP abs() Function

The abs() function returns the absolute (positive) value of a number:

Example

```
<?php
echo(abs(-6.7));  // returns 6.7
?>
```

PHP sqrt() Function

The sqrt() function returns the square root of a number:

Example

```
<?php
echo(sqrt(64));  // returns 8
?>
```

PHP round() Function

The round() function rounds a floating-point number to its nearest integer:

Example

```php
<?php
echo(round(0.60));  // returns 1
echo(round(0.49));  // returns 0
?>
```

Random Numbers

The rand() function generates a random number:

Example

```php
<?php
echo(rand());
?>
```

To get more control over the random number, you can add the optional *min* and *max* parameters to specify the lowest integer and the highest integer to be returned.

For example, if you want a random integer between 10 and 100 (inclusive), use rand(10, 100):

Example

```php
<?php
echo(rand(10, 100));
?>
```

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).

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

Example

Create a constant with a **case-sensitive** name:

```
<?php
define("GREETING", "Welcome to Sai Pali!");
echo GREETING;
?>
```

Example

Create a constant with a **case-insensitive** name:

```
<?php
define("GREETING", "Welcome to Sai Pali!", true);
echo greeting;
?>
```

PHP Constant Arrays

In PHP7, you can create an Array constant using the define() function.

Example

Create an Array constant:

```
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
?>
```

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:

```
<?php
define("GREETING", "Welcome to Sai Pali!");

function myTest() {
  echo GREETING;
}

myTest();
?>
```

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	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

Example:

```
<?php
$x = 10;
$y = 4;
echo($x + $y); // 0utputs: 14
echo($x - $y); // 0utputs: 6
echo($x * $y); // 0utputs: 40
echo($x / $y); // 0utputs: 2.5
echo($x % $y); // 0utputs: 2
?>
```

Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

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

Example:

```
<?php
$x = 10;
echo $x; // Outputs: 10
$x = 20;
$x += 30;
echo $x; // Outputs: 50
$x = 50;
$x -= 20;
echo $x; // Outputs: 30
$x = 5;
$x *= 25;
echo $x; // Outputs: 125
$x = 50;
$x /= 10;
echo $x; // Outputs: 5
$x = 100;
$x %= 15;
echo $x; // Outputs: 10
?>
```

Bitwise Operators

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise 0.
|	Or (Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor (Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$a << $b	Left shift the bits of operand $a $b steps
>>	Shift right	$a >> $b	Right shift the bits of $a operand by $b number of places

Comparison Operators

Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

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
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

Example:

```
<?php
    $x = 25;
    $y = 35;
    $z = "25";
    var_dump($x == $z); // Outputs: Boolean true
    var_dump($x === $z); // Outputs: Boolean false
    var_dump($x != $y); // Outputs: Boolean true
    var_dump($x !== $z); // Outputs: Boolean true
    var_dump($x < $y); // Outputs: Boolean true
    var_dump($x > $y); // Outputs: Boolean false
    var_dump($x <= $y); // Outputs: Boolean true
    var_dump($x >= $y); // Outputs: Boolean false
?>
```

Incrementing/Decrementing Operators

The increment and decrement operators are used to increase and decrease the value of a variable.

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

Example

```
<?php
$x = 10;
echo ++$x; // Outputs: 11
echo $x; // Outputs: 11
$x = 10;
echo $x++; // Outputs: 10
echo $x; // Outputs: 11
$x = 10;
echo --$x; // Outputs: 9
echo $x; // Outputs: 9
$x = 10;
echo $x--; // Outputs: 10
echo $x; // Outputs: 9
?>
```

Logical Operators

The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

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

```
<?php
$year = 2014; // Leap years are divisible by 400 or by 4 but not 100
if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 == 0)))
{
echo "$year is a leap year.";
}
Else
{
echo "$year is not a leap year.";
}
?>
```

String Operators

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

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

Example:

```
<?php
$x = "Hello";
$y = " World!";
echo $x . $y; // Outputs: Hello World!
$x .= $y;
echo $x; // Outputs: Hello World!
?>
```

Array Operators

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

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

Example:

```
<?php
$x = array("a" => "Red", "b" => "Green", "c" => "Blue");
$y = array("u" => "Yellow", "v" => "Orange", "w" => "Pink");
$z = $x + $y; // Union of $x and $y
```

```
var_dump($z);
var_dump($x == $y); // Outputs: Boolean false
var_dump($x === $y); // Outputs: Boolean false
var_dump($x != $y); // Outputs: Boolean true
var_dump($x <> $y); // Outputs: Boolean true
var_dump($x !== $y); // Outputs: Boolean true
?>
```

PHP Spaceship Operator PHP 7

PHP 7 introduces a new spaceship operator (<=>) which can be used for comparing two expressions. It is also known as combined comparison operator.

The spaceship operator returns 0 if both operands are equal, 1 if the left is greater, and -1 if the right is greater. It basically provides three-way comparison as shown in the following table:

Operator	<=> Equivalent
$x < $y	($x <=> $y) === -1
$x <= $y	($x <=> $y) === -1 \|\| ($x <=> $y) === 0
$x == $y	($x <=> $y) === 0
$x != $y	($x <=> $y) !== 0
$x >= $y	($x <=> $y) === 1 \|\| ($x <=> $y) === 0
$x > $y	($x <=> $y) === 1

The following example will show you how spaceship operator actually works:

```
<?php
// Comparing Integers
echo 1 <=> 1; // Outputs: 0
echo 1 <=> 2; // Outputs: -1
echo 2 <=> 1; // Outputs: 1
// Comparing Floats
echo 1.5 <=> 1.5; // Outputs: 0
```

```
echo 1.5 <=> 2.5; // Outputs: -1
echo 2.5 <=> 1.5; // Outputs: 1
// Comparing Strings
echo "x" <=> "x"; // Outputs: 0
echo "x" <=> "y"; // Outputs: -1
echo "y" <=> "x"; // Outputs: 1
?>
```

https://www.tutorialrepublic.com/php-tutorial/php-if-else-statements.php

The Ternary Operator

The ternary operator provides a shorthand way of writing the *if...else* statements. The ternary operator is represented by the question mark (`?`) symbol and it takes three operands: a condition to check, a result for `true`, and a result for `false`.

To understand how this operator works, consider the following examples:

Example:

```
<?php
if($age < 18)
{
    echo 'Child'; // Display Child if age is less than 18
} else
{
   echo 'Adult'; // Display Adult if age is greater than or equal to 18
} ?>
```

Using the ternary operator, the same code could be written in a more compact way:

```
<?php
echo ($age < 18) ? 'Child' : 'Adult';
```

```
?>
```

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.

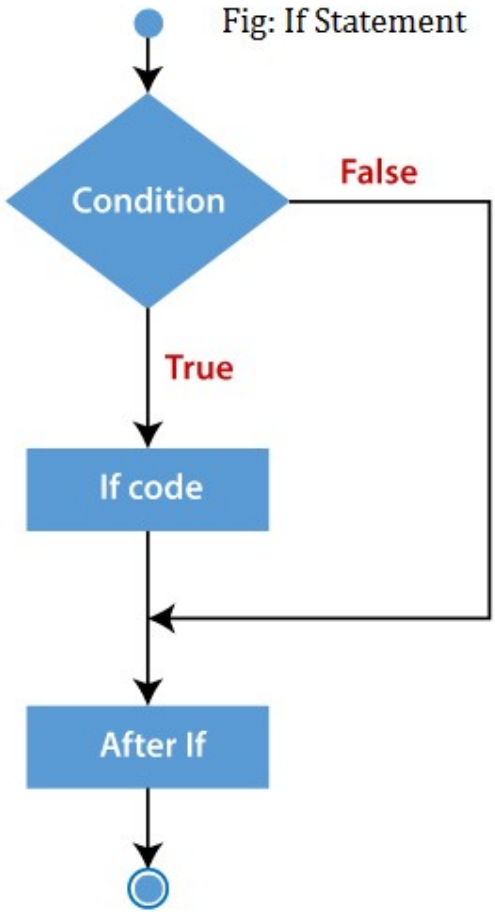
In PHP we have the following conditional statements:

- 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.

Flowchart



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:

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

```
<?php
$d = date("D");
if($d == "Fri")
{
    echo "Have a nice weekend!";
}
?>
```

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

Output:

12 is less than 100

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;
}
```

Flowchart

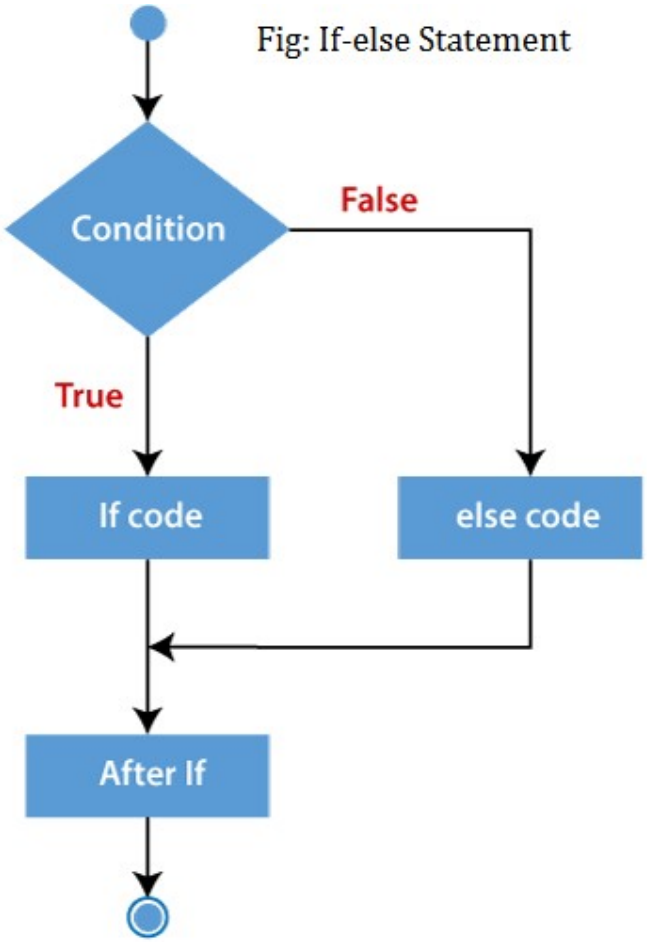


Fig: If-else Statement

Example

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

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

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

```php
<?php
        $d = date("D");
        if($d == "Fri")
        {
        echo "Have a nice weekend!";
        }
        Else
        {
        echo "Have a nice day!";
        }
?>
```

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

Output:

12 is even number

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
}
}
```

Flowchart

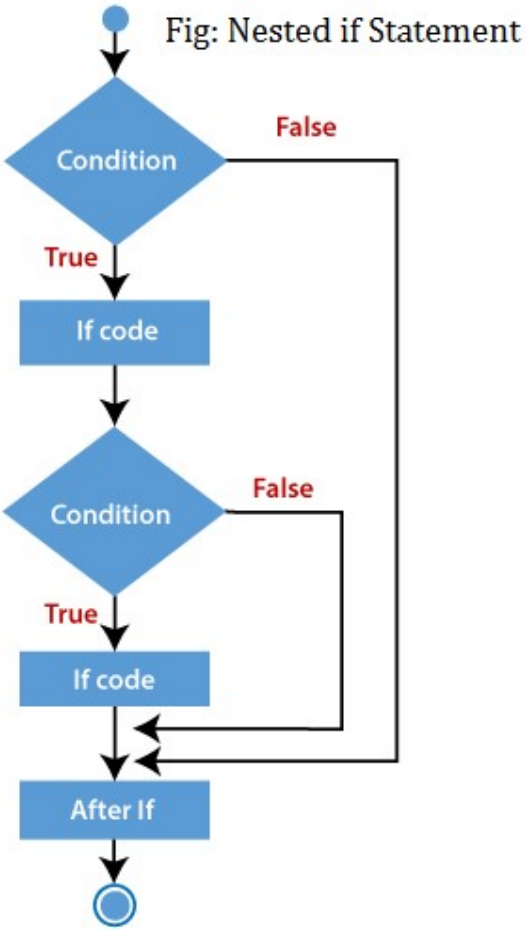


Fig: Nested if Statement

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

```
<?php
    $a = 34; $b = 56; $c = 45;
    if ($a < $b) {
      if ($a < $c) {
        echo "$a is smaller than $b and $c";
      }
    }
?>
```

Output:

34 is smaller than 56 and 45

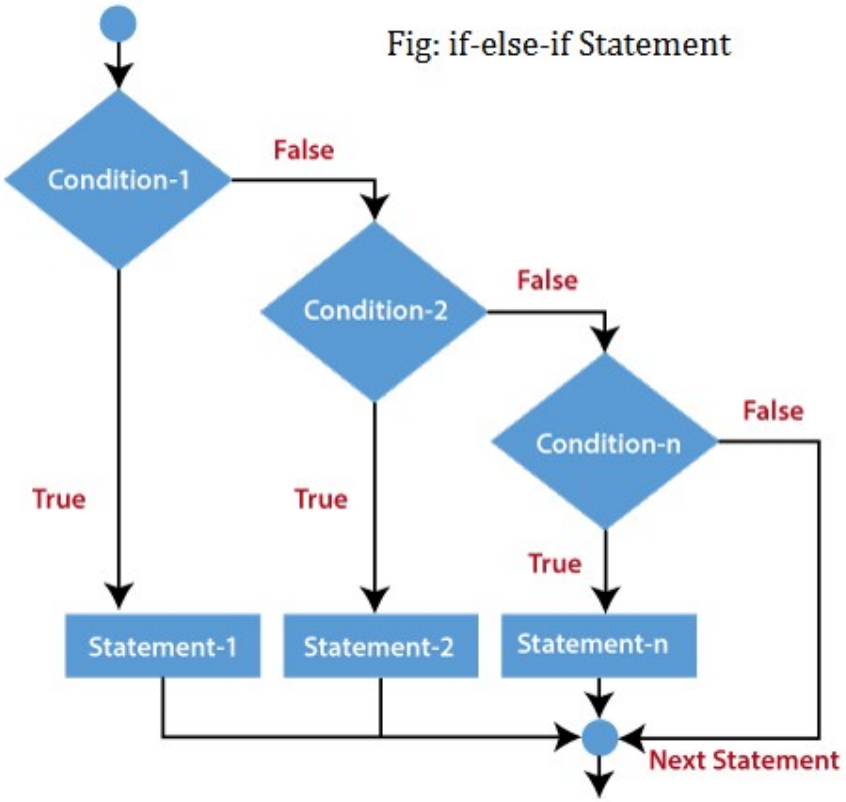
PHP - The if...elseif...else Statement

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;
}
```

Flowchart



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!":

```
<?php
$t = date("H");
if ($t < "10") {
```

```
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

Example:

```
<?php
   $marks=69;
   if ($marks<33){
      echo "fail";
   }
   else if ($marks>=34 && $marks<50) {
      echo "D grade";
   }
   else if ($marks>=50 && $marks<65) {
     echo "C grade";
   }
   else if ($marks>=65 && $marks<80) {
      echo "B grade";
   }
   else if ($marks>=80 && $marks<90) {
      echo "A grade";
   }
   else if ($marks>=90 && $marks<100) {
      echo "A+ grade";
   }
  else {
      echo "Invalid input";
   }
?>
```

Output:

B Grade

```
<?php
        $d = date("D");
        if($d == "Fri")
        {
          echo "Have a nice weekend!";
        } elseif ($d == "Sun")
        {
           echo "Have a nice Sunday!";
        }
        Else
        {
          echo "Have a nice day!";
        }
?>
```

PHP - The switch Statement

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

Syntax

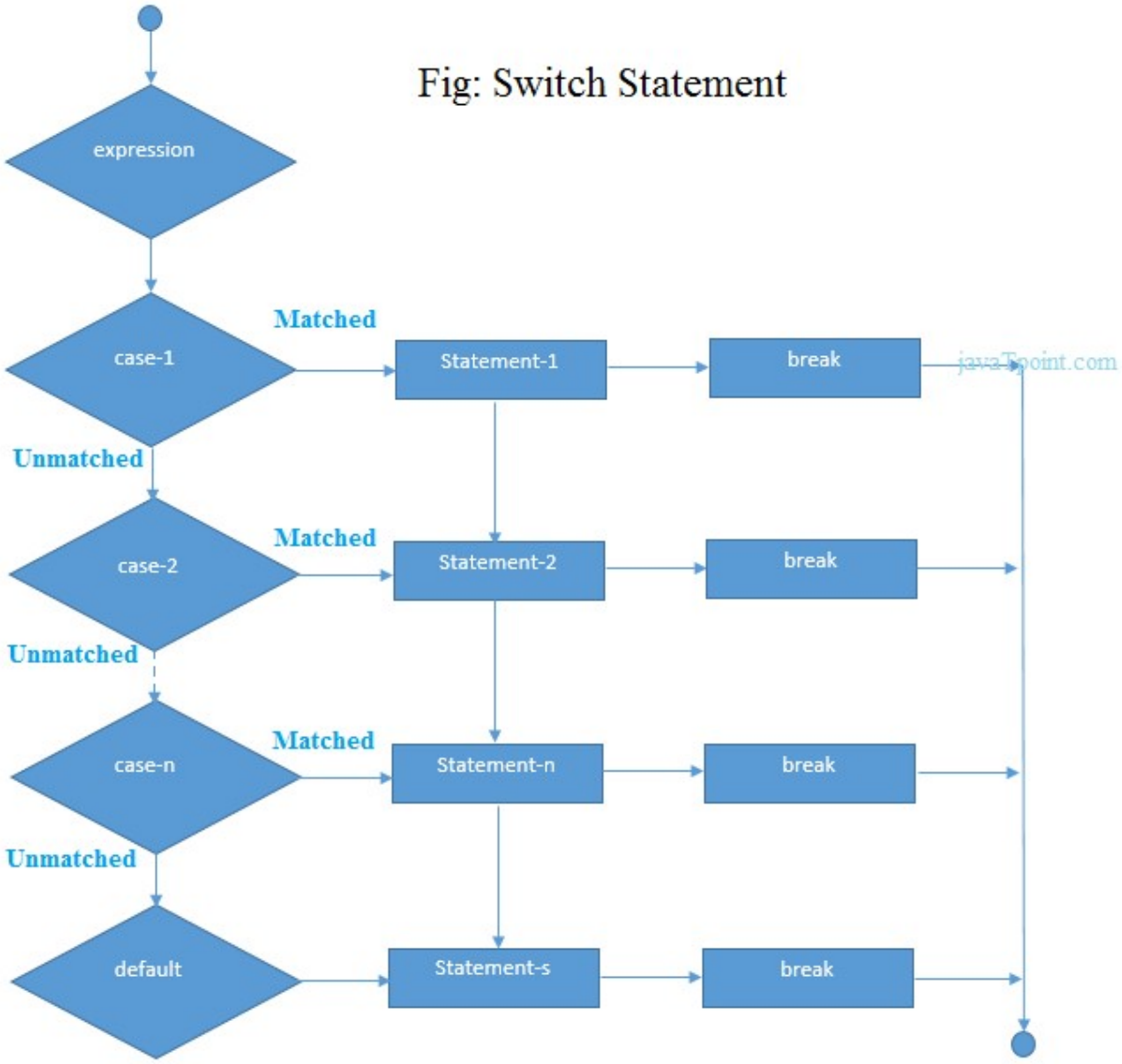
```
switch(expression){
case value1:
 //code to be executed
 break;
case value2:
 //code to be executed
 break;
.....
default:
```

```
    code to be executed if all cases are not matched;
}
```

Important points to be noticed about switch case:

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

PHP Switch Flowchart



PHP program to print daily task using switch statement

```
<?php
$today = date("D");
switch($today)
{
case "Mon":
   echo "Today is Monday. Clean your house.";
   break;
case "Tue":
  echo "Today is Tuesday. Buy some food.";
  break;
```

```
case "Wed":
        echo "Today is Wednesday. Visit a doctor.";
        break;
case "Thu":
        echo "Today is Thursday. Repair your car.";
        break;
case "Fri":
        echo "Today is Friday. Party tonight.";
        break;
case "Sat":
        echo "Today is Saturday. Its movie time.";
        break;
case "Sun":
        echo "Today is Sunday. Do some rest.";
        break;
default:
        echo "No information available for that day.";
break;
}
?>
```

PHP Switch Example

```
<?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
```

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

PHP switch statement with character

Program to check Vowel and consonant

We will pass a character in switch expression to check whether it is vowel or constant. If the passed character is A, E, I, O, or U, it will be vowel otherwise consonant.

```
<?php
```

```php
$ch = 'U';
switch ($ch)
{
   case 'a':
      echo "Given character is vowel";
      break;
   case 'e':
      echo "Given character is vowel";
      break;
   case 'i':
      echo "Given character is vowel";
      break;
   case 'o':
      echo "Given character is vowel";
      break;
   case 'u':
      echo "Given character is vowel";
      break;
   case 'A':
      echo "Given character is vowel";
      break;
   case 'E':
      echo "Given character is vowel";
      break;
   case 'I':
      echo "Given character is vowel";
      break;
   case 'O':
      echo "Given character is vowel";
      break;
   case 'U':
      echo "Given character is vowel";
      break;
   default:
```

```
        echo "Given character is consonant";
        break;
    }
?>
```

Output:

Given character is vowel

PHP switch statement with String

PHP allows to pass string in switch expression. Let's see the below example of course duration by passing string in switch case statement.

```
<?php
    $ch = "B.Tech";
    switch ($ch)
    {
        case "BCA":
            echo "BCA is 3 years course";
            break;
        case "Bsc":
            echo "Bsc is 3 years course";
            break;
        case "B.Tech":
            echo "B.Tech is 4 years course";
            break;
        case "B.Arch":
            echo "B.Arch is 5 years course";
            break;
        default:
            echo "Wrong Choice";
            break;
    }
?>
```

Output:

B.Tech is 4 years course

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.

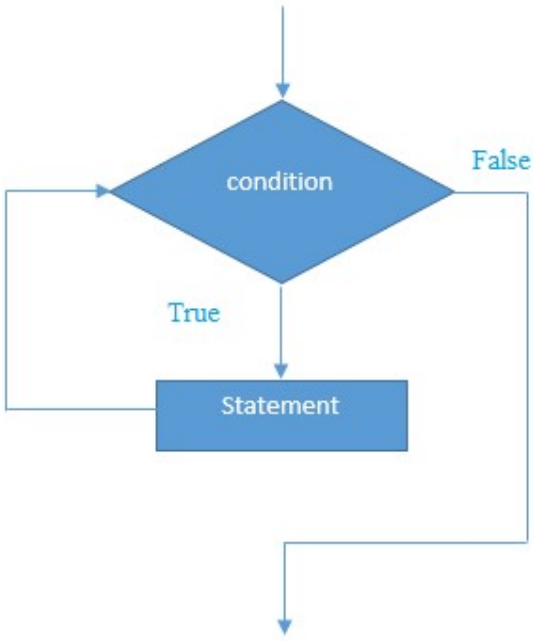
The PHP while Loop

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;
}
```

PHP While Loop Flowchart



PHP While Loop Example

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

Alternative Example

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

Example

Below is the example of printing alphabets using while loop.

```php
<?php
   $i = 'A';
   while ($i < 'H') {
      echo $i;
      $i++;
      echo "</br>";
   }
?>
```

Output:

A
B
C
D
E
F
G

Alphabets from A to Z using while loop:

```php
<?php
$i = '0';
while ($i < '26') {
      echo strtoupper(chr(97 + $i)) . '  ';
      $i++;
}
?>
```

Examples

The example below displays the numbers from 1 to 5:

Example

```php
<?php
$x = 1;
```

```
while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

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

This example counts to 100 by tens:

Example

```
<?php
$x = 0;

while($x <= 100) {
  echo "The number is: $x <br>";
  $x+=10;
}
?>
```

Example Explained

- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 100 - Continue the loop as long as $x is less than or equal to 100
- $x+=10; - Increase the loop counter value by 10 for each iteration

PHP Nested While Loop

We can use while loop inside another while loop in PHP, it is known as nested while loop.

In case of inner or nested while loop, nested while loop is executed fully for one outer while loop. If outer while loop is to be executed for 3 times and nested while loop for 3

times, nested while loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

Example

```
<?php
$i=1;
while($i<=3){
$j=1;
while($j<=3){
echo "$i  $j<br/>";
$j++;
}
$i++;
}
?>
```

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

PHP Infinite While Loop

If we pass **TRUE** in while loop, it will be an infinite loop.

Syntax

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

Example

```
<?php
   while (true) {
      echo "Hello Sai Pali!";
      echo "</br>";
   }
?>
```

The PHP do...while Loop

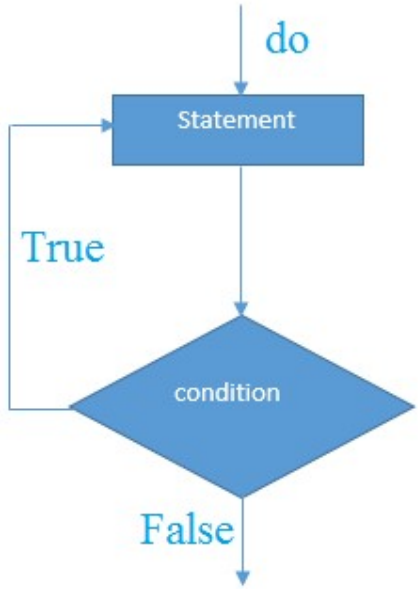
The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

The PHP do-while loop is guaranteed to run at least once.

Syntax

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

Flowchart



Example

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

Output:

1

2

3

4

5

6

7

8

9

10

Example

A semicolon is used to terminate the do-while loop. If you don't use a semicolon after the do-while loop, it is must that the program should not contain any other statements after the do-while loop. In this case, it will not generate any error.

```
<?php
   $x = 5;
   do {
      echo "Welcome to Sai Pali! </br>";
      $x++;
   } while ($x < 10);
?>
```

Example

The following example will increment the value of $x at least once. Because the given condition is false.

```
<?php
   $x = 1;
   do {
      echo "1 is not greater than 10.";
      echo "</br>";
      $x++;
   } while ($x > 10);
   echo $x;
?>
```

Output:

1 is not greater than 10.

2

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.

PHP 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

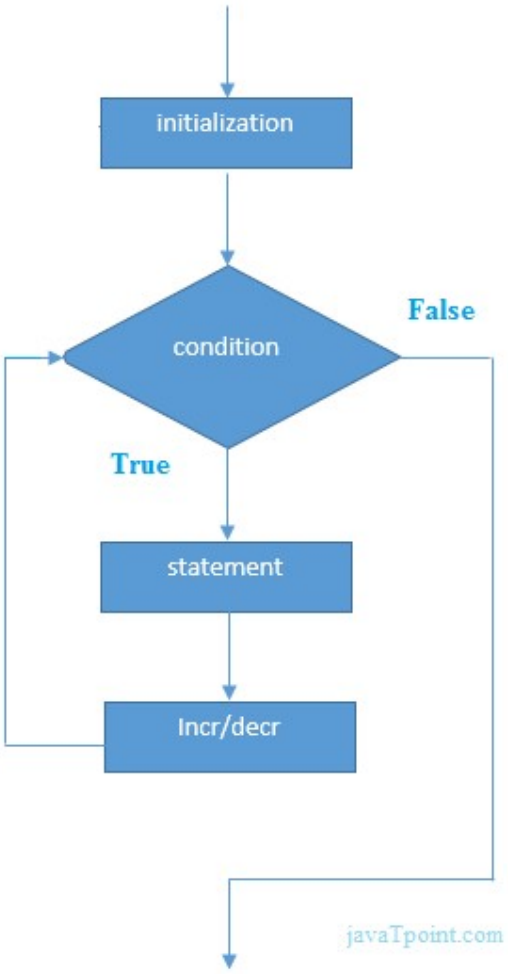
The php for loop is similar to the java/C/C++ for loop. The parameters of for loop have the following meanings:

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.

Flowchart



Example

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

Output:

```
1
```

2

3

4

5

6

7

8

9

10

Example

All three parameters are optional, but semicolon (;) is must to pass in for loop. If we don't pass parameters, it will execute infinite.

```
<?php
   $i = 1;
   //infinite loop
   for (;;) {
      echo $i++;
      echo "</br>";
   }
?>
```

Output:

1

2

3

4

.

.

.

Example

Below is the example of printing numbers from 1 to 9 in four different ways using for loop.

```
<?php
   /* example 1 */

   for ($i = 1; $i <= 9; $i++) {
   echo $i;
   }
   echo "</br>";

   /* example 2 */
   for ($i = 1; ; $i++) {
      if ($i > 9) {
         break;
      }
      echo $i;
   }
   echo "</br>";

   /* example 3 */

   $i = 1;
   for (; ; ) {
      if ($i > 9) {
         break;
      }
      echo $i;
      $i++;
   }
   echo "</br>";

   /* example 4 */
```

```
    for ($i = 1, $j = 0; $i <= 9; $j += $i, print $i, $i++);
?>
```

PHP Nested for Loop

We can use for loop inside for loop in PHP, it is known as nested for loop. The inner for loop executes only when the outer for loop condition is found **true**.

In case of inner or nested for loop, nested for loop is executed fully for one outer for loop. If outer for loop is to be executed for 3 times and inner for loop for 3 times, inner for loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

Example

```
<?php
for($i=1;$i<=3;$i++){
for($j=1;$j<=3;$j++){
echo "$i   $j<br/>";
}
}
?>
```

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

PHP For Each Loop

PHP for each loop is used to traverse array elements.

Syntax

```
foreach( $array as $var ){
 //code to be executed
}
?>
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
foreach( $season as $arr ){
  echo "Season is: $arr<br />";
}
?>
```

1. Season is: summer autumn

PHP foreach loop

The foreach loop is used to traverse the array elements. It works only on array and object. It will issue an error if you try to use it with the variables of different datatype.

The foreach loop works on elements basis rather than index. It provides an easiest way to iterate the elements of an array.

In foreach loop, we don't need to increment the value.

Syntax

```
foreach ($array as $value) {
    //code to be executed
}
```

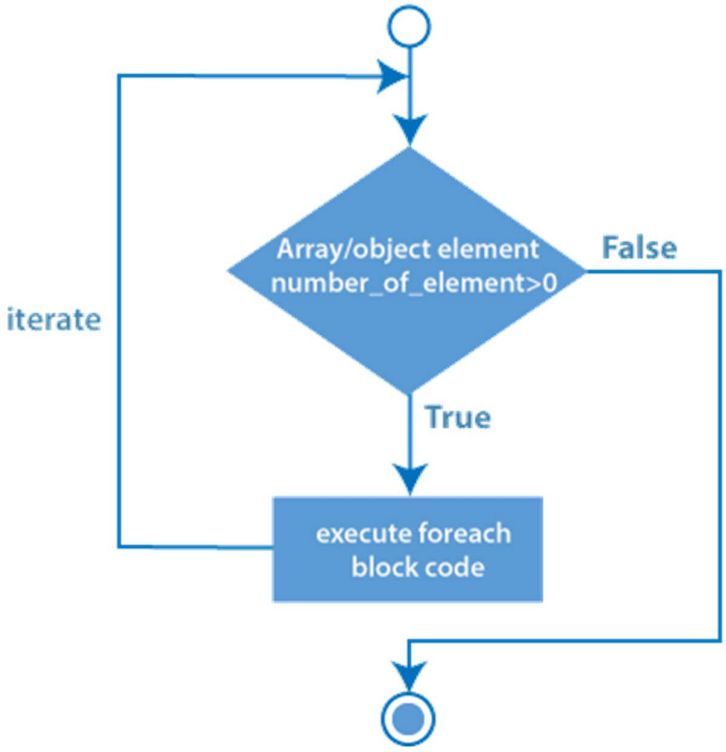
There is one more syntax of foreach loop.

Syntax

```
foreach ($array as $key => $element) {
```

```
    //code to be executed
}
```

Flowchart



Example 1:

PHP program to print array elements using foreach loop.

```
<?php
    //declare array
    $season = array ("Summer", "Winter", "Autumn", "Rainy");

    //access array elements using foreach loop
    foreach ($season as $element) {
        echo "$element";
        echo "</br>";
    }
?>
```

Example 2:

PHP program to print associative array elements using foreach loop.

```
<?php
   //declare array
   $employee = array (
      "Name" => "Alex",
      "Email" => "alex_jtp@gmail.com",
      "Age" => 21,
      "Gender" => "Male"
   );

   //display associative array element through foreach loop
   foreach ($employee as $key => $element) {
      echo $key . " : " . $element;
      echo "</br>";
   }
?>
```

Output:

Name : Alex

Email : alex_jtp@gmail.com

Age : 21

Gender : Male

Example 3:

Multi-dimensional array

```
<?php
   //declare multi-dimensional array
   $a = array();
   $a[0][0] = "Alex";
   $a[0][1] = "Bob";
   $a[1][0] = "Camila";
   $a[1][1] = "Denial";
```

```
   //display multi-dimensional array elements through foreach loop
   foreach ($a as $e1) {
      foreach ($e1 as $e2) {
         echo "$e2\n";
      }
   }
?>
```

Output:

Alex Bob Camila Denial

Example 4:

Dynamic array

```
<?php
   //dynamic array
   foreach (array ('s', 'a', 'i', 'p', 'a', 'l', 'i;) as $elements) {
      echo "$elements\n";
   }
?>
```

Output:

S a I p a l i

PHP 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;

Flowchart

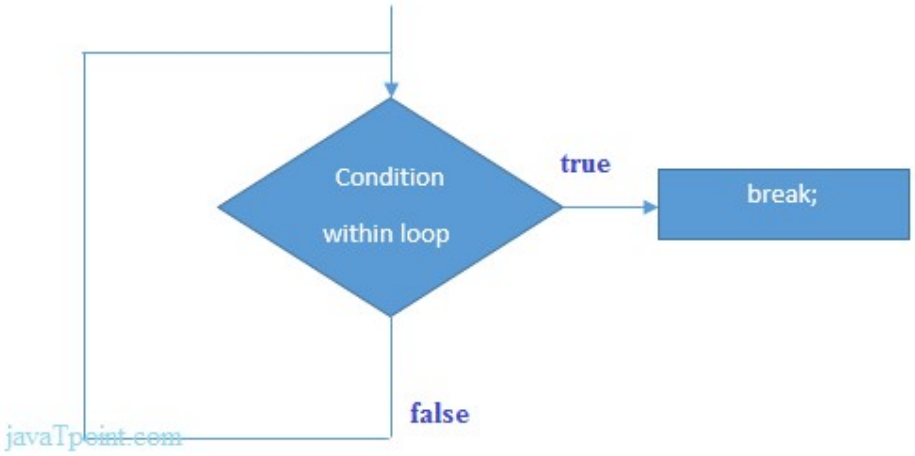


Figure: Flowchart of break statement

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

PHP Break: inside inner loop

The PHP break statement breaks the execution of inner loop only.

```
<?php
for($i=1;$i<=3;$i++){
 for($j=1;$j<=3;$j++){
  echo "$i   $j<br/>";
  if($i==2 && $j==2){
   break;
  }
 }
}
?>
```

Output:

1 1

1 2

1 3

2 1

2 2

3 1

3 2

3 3

PHP Break: inside switch statement

The PHP break statement breaks the flow of switch case also.

```
<?php
```

```
$num=200;
switch($num){
case 100:
echo("number is equals to 100");
break;
case 200:
echo("number is equal to 200");
break;
case 50:
echo("number is equal to 300");
break;
default:
echo("number is not equal to 100, 200 or 500");
}
?>
```

Output:

number is equal to 200

PHP Break: with array of string

```
<?php
//declare an array of string
$number = array ("One", "Two", "Three", "Stop", "Four");
foreach ($number as $element) {
if ($element == "Stop") {
break;
}
echo "$element </br>";
}
?>
```

Output:

One

Two

Three

You can see in the above output, after getting the specified condition true, break statement immediately ends the loop and control is came out from the loop.

PHP Break: switch statement without break

It is not essential to break out of all cases of a switch statement. But if you want that only one case to be executed, you have to use break statement.

```
<?php
$car = 'Mercedes Benz';
switch ($car) {
default:
echo '$car is not Mercedes Benz<br>';
case 'Orange':
echo '$car is Mercedes Benz';
}
?>
```

Output:

$car is not Mercedes Benz

$car is Mercedes Benz

PHP Break: using optional argument

The break accepts an optional numeric argument, which describes how many nested structures it will exit. The default value is 1, which immediately exits from the enclosing structure.

```
<?php
$i = 0;
while (++$i) {
    switch ($i) {
```

```
        case 5:
            echo "At matched condition i = 5<br />\n";
            break 1;  // Exit only from the switch.
        case 10:
            echo "At matched condition i = 10; quitting<br />\n";
            break 2;  // Exit from the switch and the while.
        default:
            break;
    }
}?>
```

Output:

```
At matched condition i = 5
At matched condition i = 10; quitting
```

Note: *The break keyword immediately ends the execution of the current structure.*

PHP continue statement

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.

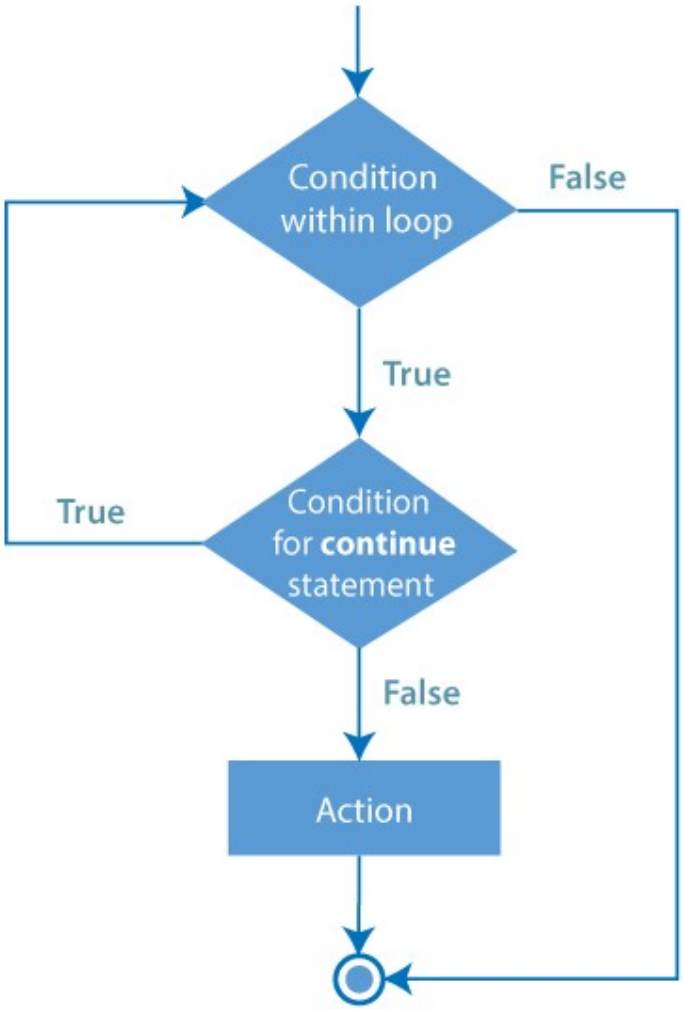
The continue statement can be used with all types of loops such as - for, while, do-while, and foreach loop. The continue statement allows the user to skip the execution of the code for the specified condition.

Syntax

The syntax for the continue statement is given below:

```
jump-statement;
continue;
```

Flowchart:



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

PHP continue Example in while loop

Example

In the following example, we will print the even numbers between 1 to 20.

```
<?php
   //php program to demonstrate the use of continue statement

   echo "Even numbers between 1 to 20: </br>";
   $i = 1;
   while ($i<=20) {
      if ($i %2 == 1) {
         $i++;
         continue;   //here it will skip rest of statements
      }
      echo $i;
      echo "</br>";
      $i++;
   }
?>
```

Output:

Even numbers between 1 to 20:

2

4

6

8

10

12

14

16

18

20

PHP continue Example with array of string

Example

The following example prints the value of array elements except those for which the specified condition is true and continue statement is used.

```
<?php
   $number = array ("One", "Two", "Three", "Stop", "Four");
   foreach ($number as $element) {
      if ($element == "Stop") {
         continue;
      }
      echo "$element </br>";
   }
?>
```

Output:

One

Two

Three

Four

PHP continue Example with optional argument

The continue statement accepts an optional numeric value, which is used accordingly. The numeric value describes how many nested structures it will exit.

Example

Look at the below example to understand it better:

```
<?php
   //outer loop
   for ($i =1; $i<=3; $i++) {
      //inner loop
      for ($j=1; $j<=3; $j++) {
         if (($i == $j) ) {      //skip when i and j have same values
            continue 1;     //exit only from inner for loop
         }
         echo $i.$j;
         echo "</br>";
      }
   }
?>
```

Output:

```
12
13
21
23
31
32
```

PHP Functions

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also.

Advantage of PHP Functions

1. **Code Reusability**: PHP functions are defined only once and can be invoked many times, like in other programming languages.
2. **Less Code**: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.
3. **Easy to understand**: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions

We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

Syntax

```
function functionname(){
//code to be executed
}
```

Note: Function name must be start with letter and underscore only like other labels in PHP. It can't be start with numbers or special symbols.

PHP Functions Example

File: function1.php

```
<?php
function sayHello(){
echo "Hello PHP Function";
}
```

```
sayHello();//calling function
?>
```

Output:

Hello PHP Function

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

Let's see the example to pass single argument in PHP function.

File: functionarg.php

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

Hello Sonoo

Hello Vimal

Hello John

File: functionarg2.php

```
<?php
function sayHello($name,$age){
```

```
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output:

Hello Sonoo, you are 27 years old

Hello Vimal, you are 29 years old

Hello John, you are 23 years old

PHP Call By Reference

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by-passing value as a reference.

By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

Let's see a simple example of call by reference in PHP.

File: functionref.php

```
<?php
function adder(&$str2)
{
   $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

Hello Call By Reference

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.

File: functiondefaultarg.php

```
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

Output:

```
Hello Rajesh
Hello Sonoo
Hello John
```

PHP Function: Returning Value

Let's see an example of PHP function that returns value.

File: functiondefaultarg.php

```
<?php
function cube($n){
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

Output:

Cube of 3 is: 27

PHP Parameterized Function

PHP Parameterized functions are the functions with parameters. You can pass any number of parameters inside a function. These passed parameters act as variables inside your function.

They are specified inside the parentheses, after the function name.

The output depends upon the dynamic values passed as the parameters into the function.

PHP Parameterized Example 1

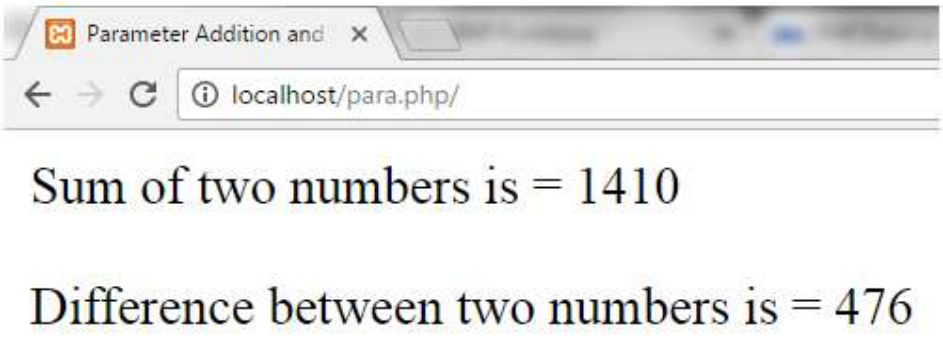
Addition and Subtraction

In this example, we have passed two parameters **$x** and **$y** inside two functions **add()** and **sub()**.

```
<!DOCTYPE html>
<html>
<head>
   <title>Parameter Addition and Subtraction Example</title>
</head>
<body>
<?php
      //Adding two numbers
       function add($x, $y) {
         $sum = $x + $y;
         echo "Sum of two numbers is = $sum <br><br>";
       }
       add(467, 943);

       //Subtracting two numbers
       function sub($x, $y) {
```

```
        $diff = $x - $y;
        echo "Difference between two numbers is = $diff";
      }
      sub(943, 467);
    ?>
</body>
</html>
```

Output:



PHP Parameterized Example 2

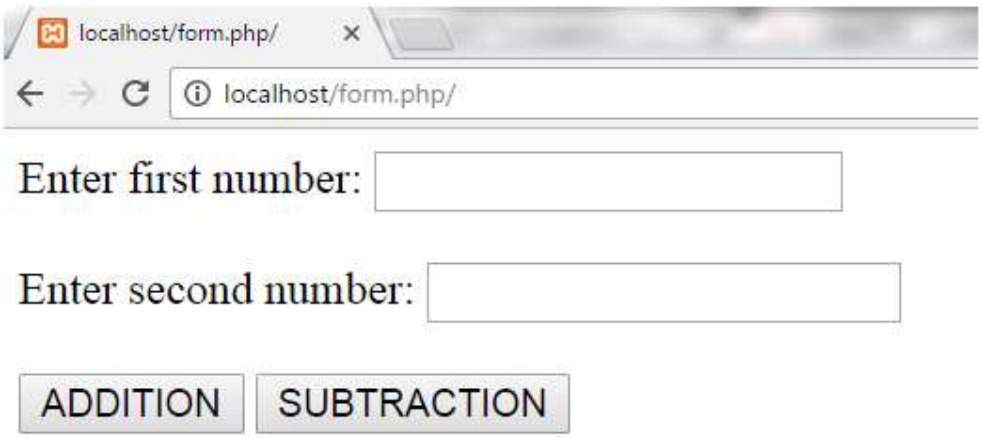
Addition and Subtraction with Dynamic number

In this example, we have passed two parameters **$x** and **$y** inside two functions **add()** and **sub()**.

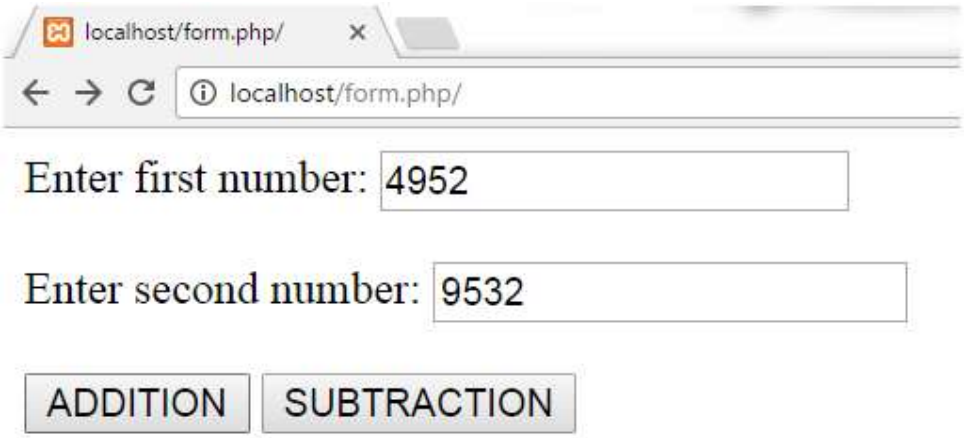
```
<?php
//add() function with two parameter
function add($x,$y)
{
$sum=$x+$y;
echo "Sum = $sum <br><br>";
}
//sub() function with two parameter
function sub($x,$y)
{
$sub=$x-$y;
```

```
echo "Diff = $sub <br><br>";
}
//call function, get  two argument through input box and click on add or sub button
if(isset($_POST['add']))
{
//call add() function
 add($_POST['first'],$_POST['second']);
}
if(isset($_POST['sub']))
{
//call add() function
sub($_POST['first'],$_POST['second']);
}
?>
<form method="post">
Enter first number: <input type="number" name="first"/><br><br>
Enter second number: <input type="number" name="second"/><br><br>
<input type="submit" name="add" value="ADDITION"/>
<input type="submit" name="sub" value="SUBTRACTION"/>
</form>
```

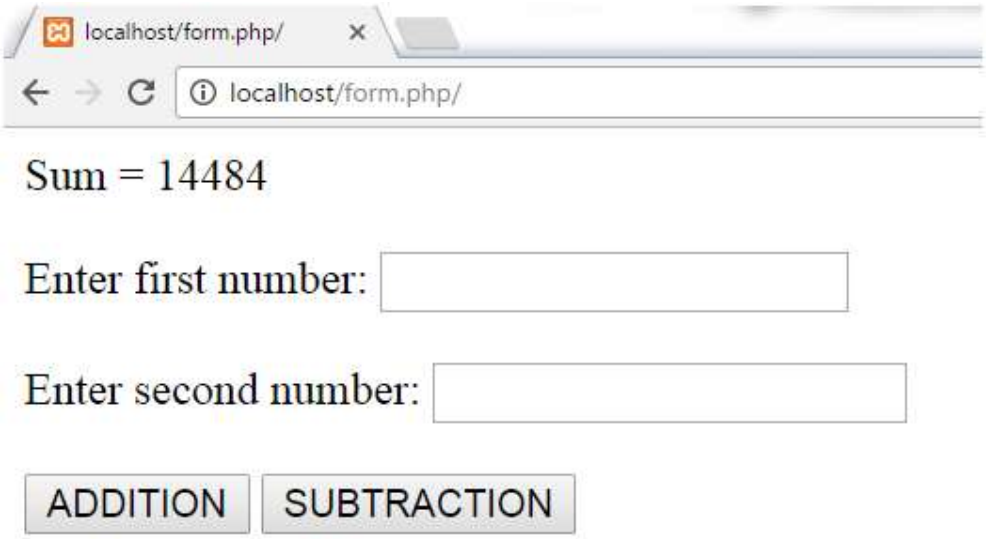
Output:



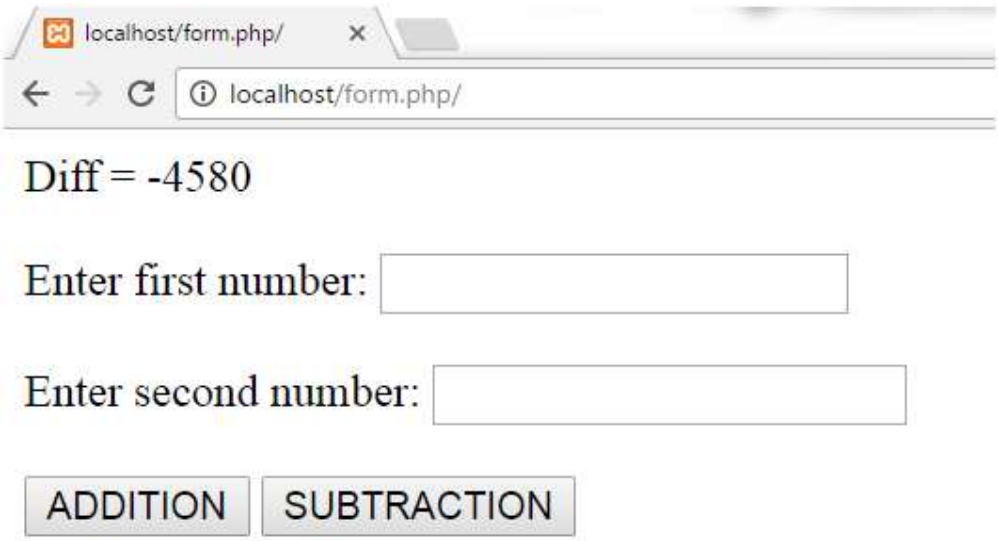
We passed the following number,



Now clicking on ADDITION button, we get the following output.



Now clicking on SUBTRACTION button, we get the following output.



PHP Call By Value

PHP allows you to call function by value and reference both. In case of PHP call by value, actual value is not modified if it is modified inside the function.

Let's understand the concept of call by value by the help of examples.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Value' string. But, printing $str variable results 'Hello' only. It is because changes are done in the local variable $str2 only. It doesn't reflect to $str variable.

```
<?php
function adder($str2)
{
   $str2 .= 'Call By Value';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

Hello

Example 2

Let's understand PHP call by value concept through another example.

```
<?php
function increment($i)
{
   $i++;
}
$i = 10;
increment($i);
```

```
echo $i;
?>
```

Output:

10

PHP Call By Reference

In case of PHP call by reference, actual value is modified if it is modified inside the function. In such case, you need to use & (ampersand) symbol with formal arguments. The & represents reference of the variable.

Let's understand the concept of call by reference by the help of examples.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Reference' string. Here, printing $str variable results 'This is Call By Reference'. It is because changes are done in the actual variable $str.

```
<?php
function adder(&$str2)
{
   $str2 .= 'Call By Reference';
}
$str = 'This is ';
adder($str);
echo $str;
?>
```

Output:

Example2

Let's understand PHP call by reference concept through another example.

```
<?php
```

```
function increment(&$i)
{
   $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```

Output:

11

PHP Default Argument Values Function

PHP allows you to define C++ style default argument values. In such case, if you don't pass any value to the function, it will use default argument value.

Let' see the simple example of using PHP default arguments in function.

Example 1

```
<?php
function sayHello($name="Ram"){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello();//passing no value
sayHello("Vimal");
?>
```

Output:

Hello Sonoo

Hello Ram

Hello Vimal

Since PHP 5, you can use the concept of default argument value with call by reference also.

Example 2

```
<?php
function greeting($first="Sonoo",$last="Jaiswal"){
echo "Greeting: $first $last<br/>";
}
greeting();
greeting("Rahul");
greeting("Michael","Clark");
?>
```

Output:

Greeting: Sonoo Jaiswal

Greeting: Rahul Jaiswal

Greeting: Michael Clark

Example 3

```
<?php
function add($n1=10,$n2=10){
$n3=$n1+$n2;
echo "Addition is: $n3<br/>";
}
add();
add(20);
add(40,40);
?>
```

Output:

Addition is: 20

Addition is: 30

Addition is: 80

PHP Variable Length Argument Function

PHP supports variable length argument function. It means you can pass 0, 1 or n number of arguments in function. To do so, you need to use 3 ellipses (dots) before the argument name.

The 3-dot concept is implemented for variable length argument since PHP 5.6.

Let's see a simple example of PHP variable length argument function.

```
<?php
function add(...$numbers) {
    $sum = 0;
    foreach ($numbers as $n) {
        $sum += $n;
    }
    return $sum;
}

echo add(1, 2, 3, 4);
?>
```

Output:

10

PHP Recursive Function

PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.

It is recommended to avoid recursive function call over 200 recursion level because it may smash the stack and may cause the termination of script.

Example 1: Printing number

```
<?php
function display($number) {
```

```
        if($number<=5){
         echo "$number <br/>";
         display($number+1);
        }
    }
    display(1);
    ?>
```

Output:

1

2

3

4

5

Example 2: Factorial Number

```
<?php
function factorial($n)
{
    if ($n < 0)
        return -1; /*Wrong value*/
    if ($n == 0)
        return 1; /*Terminating condition*/
    return ($n * factorial ($n -1));
}
echo factorial(5);
?>
```

Output:

Example:

```
<?php

function factorial( $n ) {
```

```
  // Base case
  if ( $n == 0 ) {
    echo "Base case: $n = 0. Returning 1...<br>";
    return 1;
  }

  // Recursion
  echo "$n = $n: Computing $n * factorial( " . ($n-1) . " )...<br>";
  $result = ( $n * factorial( $n-1 ) );
  echo "Result of $n * factorial( " . ($n-1) . " ) = $result. Returning $result...<br>";
  return $result;
}
echo "The factorial of 5 is: " . factorial( 5 );
?>
```

PHP is a Loosely Typed Language

In the examples above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives us an option to specify the expected data type when declaring a function, and by adding the strict declaration, it will throw a "Fatal Error" if the data type mismatches.

In the following example we try to send both a number and a string to the function without using strict:

```php
<?php
function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
?>
```

```php
<?php
declare(strict_types=1);
?>
<!DOCTYPE html>
<html>
<head>
	<title>User Defined functions Demo</title>
</head>
<body>
		<?php
function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5");
	?>
</body>
</html>
```

PHP Functions - Returning values

To let a function, return a value, use the return statement:

Example

```php
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
  $z = $x + $y;
```

```
  return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

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:

Example

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
  return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

You can specify a different return type, than the argument types, but make sure the return is the correct type:

Example

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : int {
  return (int)($a + $b);
}
```

```
echo addNumbers(1.2, 5.2);
?>
```

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:

Example

Use a pass-by-reference argument to update a variable:

```
<?php
function add_five(&$value) {
  $value += 5;
}

$num = 2;
add_five($num);
echo $num;
?>
```

PHP Arrays

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

Advantage of PHP Array

1. **Less Code**: We don't need to define multiple variables.
2. **Easy to traverse**: By the help of single loop, we can traverse all the elements of an array.
3. **Sorting**: We can sort the elements of array.

PHP Array Types

There are 3 types of arrays in PHP.

1. Indexed Array
2. Associative Array
3. Multidimensional Array

PHP Indexed Array

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.

There are two ways to define indexed array:

1st way:

```
$season=array("summer","winter","spring","autumn");
```

```
$colors = array(“Red”,”Blue”,”Green”);
```

Note: In an indexed or numeric array, the indexes are automatically assigned and start with 0, and the values can be any data type.

2nd way:

```
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
```

```
$colors[0] = "Red";
$colors[1] = "Green";
$colors[2] = "Blue";
```

Example

File: array1.php

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

Season are: summer, winter, spring and autumn

File: array2.php

```
<?php
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

Season are: summer, winter, spring and autumn

```
<?php
// Define array
$cities = array("London", "Paris", "New York");
// Display the cities array
print_r($cities);
?>
```

To get more information, use the following statement:

```
<?php
```

```
// Define array
$cities = array("London", "Paris", "New York");

// Display the cities array
var_dump($cities);
?>
```

This var_dump() statement gives the following output:

array(3) { [0]=> string(6) "London" [1]=> string(5) "Paris" [2]=> string(8) "New York" }

This output shows the data type of each element, such as a string of 6 characters, in addition to the key and value.

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

Example

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

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

Using foreach loop:

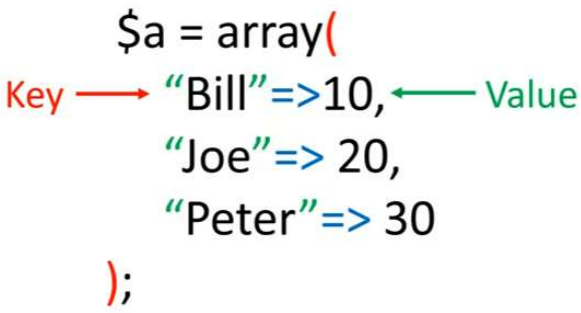
```
<?php

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

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

echo "Using foreach loop<br />";
foreach($cars as $c)
echo $c . '<br />';


?>
```

PHP Associative Array

Associative arrays are arrays that use named keys that you assign to them.

We can associate name with each array elements in PHP using => symbol.



There are two ways to define associative array:

1st way:

```
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
```

2nd way:

```
$salary["Sonoo"]="350000";
$salary["John"]="450000";
$salary["Kartik"]="200000";
```

Example

File: arrayassociative1.php

```
<?php

$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";

?>
```

Output:

```
Sonoo salary: 350000
John salary: 450000
Kartik salary: 200000
```

File: arrayassociative2.php

```
<?php
    $salary["Sonoo"]="350000";
    $salary["John"]="450000";
    $salary["Kartik"]="200000";
    echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
    echo "John salary: ".$salary["John"]."<br/>";
    echo "Kartik salary: ".$salary["Kartik"]."<br/>";

?>
```

Output:

```
Sonoo salary: 350000
John salary: 450000
Kartik salary: 200000
```

```
<?php
$age=array(
        "Bill"=>25,
        "Steve"=>28,
         "Allan"=>22
          );
echo $age["Steve"];

echo $age["Bill"]."<br />";
echo $age["Steve"]."<br />";
```

```
echo $age["Allan"]."<br />";
?>

OR
<?php
$age= [
          "Bill"=>25,
          "Steve"=>28,
           "Allan"=>22
            ];
echo $age["Steve"];

echo $age["Bill"]."<br />";
echo $age["Steve"]."<br />";
echo $age["Allan"]."<br />";
?>
```

Print_r function:

The print_r() function prints the information about a variable in a more human-readable way.

```
<?php
$age= [
          "Bill"=>25,
          "Steve"=>28,
           "Allan"=>22
            ];

print_r($age);
?>

<?php
$age= [
          "Bill"=>25,
```

```
            "Steve"=>28,
             "Allan"=>22
              ];

echo "<pre>";
print_r($age);
echo "</pre>";

?>
```

var_dump() function:

The var_dump() function dumps information about one or more variables. The information holds type and value of the variable(s).

Note: It shows the datatype of the variable and its value.

```
<?php
$age= [
            "Bill"=>25,
            "Steve"=>28,
             "Allan"=>22
              ];

echo "<pre>";
var_dump($age);
echo "</pre>";
?>
```

We can even change the value of the key

```
<?php
$age= [
            "Bill"=>25,
            "Steve"=>28,
             "Allan"=>22
              ];
```

```
$age[“Allan”] = 50;

echo “<pre>”;
var_dump($age);
echo “</pre>”;
?>
```

We also give different datatype for the values.

```
<?php
$age= [
          "Bill"=>”25”,
          "Steve"=>28.55,
           "Allan"=>22
            ];

$age[“Allan”] = 50;

echo “<pre>”;
var_dump($age);
echo “</pre>”;
?>



<!DOCTYPE html>
<html>
<body>

<?php
$a = array("red", "green", "blue");
print_r($a);

echo "<br>";
```

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

</body>
</html>
```

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

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

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

Output:

Key=Peter, Value=35

Key=Ben, Value=37

Key=Joe, Value=43

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

foreach($age as $key => $value) {
  echo $key . " = “  . $value;
  echo "<br>";
```

```
}
?>
```

Output:

Peter = 35

Ben = 37

Joe = 43

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

foreach($age as $key => $value) {
  echo "$key  =  $value <br />";
}
?>
```

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

echo "<ul>";
foreach($age as $key => $value) {
  echo "<li>$key  =  $value </li>";
}
echo "</ul>";
?>
```

PHP Multidimensional Array

Multidimensional Array or Nested Array :

Emp No.	Name	Designation	Salary
1	Krishana	Manager	50000
2	Salman	Salesman	20000
3	Mohan	Computer Operator	12000
4	Amir	Driver	5000

```
$emp = array (
    array(1,"Krishana","Manager",50000),
    array(2,"Salman","Salesman",20000),
    array(3,"Mohan","Computer Operator",12000),
    array(4,"Amir","Driver",5000)
);
```

From PHP Version 5.4 the same can be written as:

```
$emp = [
    [ 1 ,"Krishana“ , "Manager“ , 50000 ],
    [ 2 , "Salman“ , "Salesman“ , 20000 ],
    [ 3 , "Mohan“ , "Computer Operator“ , 12000 ],
    [ 4 , "Amir“ , "Driver“ , 5000 ]
];
```

Example:

```
<?php
$emp = [
```

```
            [1,"Mark","Manager",5000],
            [2,"Smith","Salesman",2000],
            [3,"Peter","Admin",2500],
            [4,"Wilson","Developer",5000]
            ];

echo $emp[0][0]."  ";
echo $emp[0][1]."  ";
echo $emp[0][2]."  ";
echo $emp[0][3]."  ";
echo "<br />";
echo $emp[1][0]."  ";
echo $emp[1][1]."  ";
echo $emp[1][2]."  ";
echo $emp[1][3]."  ";

echo "<pre>";
print_r($emp);
echo "</pre>";

for($row=0;$row<4;$row++)
{
  for($col=0;$col<4;$col++)
    {
        echo $emp[$row][$col]."  ";
     }
echo "<br />";
}
?>
```

Using foreach loop:

```
foreach($emp as $v1)
    {
      foreach($v1 as $v2)
```

```
        {
            echo $v2."    ";
        }
    echo "<br />";
}
```

Example:

```
echo "<table border='2px' cellpadding='5px' cellspacing='0px'>";
echo "<tr><th>Emp Id</th><th>Emp
Name</th><th>Designation</th><th>Salary</th></tr>";
   foreach($emp as $v1)
    {
        echo "<tr>";
       foreach($v1 as $v2)
        {
            echo "<td>$v2</td>";
        }
    echo "<tr />";
}
echo "</table>";
```

Multidimensional Associative Array :

Name	Physics	Chemistry	Math
Krishana	85	89	78
Salman	68	79	73
Mohan	62	92	67

```
$marks = array (
    “Krishana” => array(“physics“ => 85, “chemistry“ => 89, “math“ => 78),
    “Salman” => array(“physics“ => 68, “chemistry“ => 79, “math“ => 73),
    “Mohan” => array(“physics“ => 62, “chemistry“ => 92, “math“ => 67)
);
```

```
$marks = [
        " Krishana" => [
                "physics" => 85,
                "maths" => 78,
                "chemistry" => 89
        ],
        “Salman" => [
                "physics" => 68,
                "maths" => 73,
                "chemistry" => 79
        ],
        “Mohan" => [
                "physics" => 62,
                "maths" => 67,
                "chemistry" => 92
        ]
];
```

Examples:

```
<?php
$marks = [
                "Krishna" => [
```

```
                "Physics" => 85,
                "Maths" => 78,
                "Chemistry" => 90
               ],
        "Salman" => [
                "Physics" => 64,
                "Maths" => 80,
                "Chemistry" => 89
               ],
        "Mohan" => [
                "Physics" => 75,
                "Maths" => 68,
                "Chemistry" => 80
               ]
        ];
//used for testing
echo "<pre>";
  print_r($marks);
echo "</pre>";

foreach($marks as $key=>$v1)
  {
     echo $key. " <br />";
 }
================================

foreach($marks as $key=>$v1)
  {
     echo $key;
     foreach($v1 as $v2)
        {
            echo $v2."  ";
        }
     echo "<br />";
```

```
 }
==================================
echo "<table border='2px' cellpadding='5px' cellspacing='0px'>
<tr>
<th>Student Name</th>
<th>Physics</th>
<th>Maths</th>
<th>Chemistry</th>
</tr>";

foreach($marks as $key=>$v1)
  {
     echo "<tr><td>$key</td>";
     foreach($v1 as $v2)
        {
            echo "<td>$v2</td>";
        }
     echo "<tr />";
 }
echo "</table>";
```

Multidimensional Array or Nested Array :

Emp No.	Name	Designation	Salary
1	Krishana	Manager	50000
2	Salman	Salesman	20000
3	Mohan	Computer Operator	12000
4	Amir	Driver	5000

```
$emp = array (
    array(1,"Krishana","Manager",50000),
    array(2,"Salman","Salesman",20000),
    array(3,"Mohan","Computer Operator",12000),
    array(4,"Amir","Driver",5000)
);
```

```
$array = [
        [1,2],
        [30,40]
    ];

foreach($array as list($a, $b)){
    echo $a . $b;
}
```

```
<?php

$emp = [
        [1,"Mark","Manager",5000],
```

```
            [2,"Smith","Salesman",2000],
            [3,"Peter","Admin",2500],
            [4,"Wilson","Developer",5000]
            ];


foreach($emp as list($id,$name,$desig,$salary)) {
   echo "$id  $name  $desig  $salary<br />";
}

?>


====================
echo "<table border='1px' cellpadding='5px' cellspacing='0px'>";
foreach($emp as list($id,$name,$desig,$salary)) {
   echo "<tr><td>$id</td><td>  $name</td><td>  $desig</td><td>
$salary</td></tr>";
}
echo "</table>";
```

```
echo "<table border='1px' cellpadding='5px' cellspacing='0px'>
<tr>
    <th>Emp Id</th>
    <th>Name</th>
    <th>Designation</th>
    <th>Salary</th>
</tr>";

foreach($emp as list($id,$name,$desig,$salary)) {
   echo "<tr><td>$id</td><td>  $name</td><td>  $desig</td><td>
$salary</td></tr>";
}
echo "</table>";
```

MULTIDIMENSIONAL ASSOCIATIVE ARRAY

```
<?php

$emp =  [
             [
                "id" =>1,
                "name" => "Smith",
                "designation" => "Manager",
                "Salary" => 50000
             ],
      [
                "id" =>2,
                "name" => "Salman",
                "designation" => "Salesman",
                "Salary" => 20000
             ],
      [
                "id" =>3,
                "name" => "Mohan",
                "designation" => "Admin",
                "Salary" => 15000
             ],
            [
                "id" =>4,
                "name" => "Allan",
                "designation" => "Developer",
                "Salary" => 25000
             ],
];

echo "<table border='1px' cellpadding='5px' cellspacing='0px'>
<tr>
    <th>Emp Id</th>
```

```
    <th>Name</th>
    <th>Designation</th>
    <th>Salary</th>
</tr>";

foreach($emp as
list("id"=>$id,"name"=>$name,"design"=>$desig,"salary"=>$salary)) {
   echo "<tr><td>$id</td><td>  $name</td><td>  $desig</td><td>
$salary</td></tr>";
}
echo "</table>";


?>
```

PHP Array Functions

PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

1) PHP array() function

PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

Syntax

1. **array array** ([mixed $...])

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

Season are: summer, winter, spring and autumn

2) PHP array_change_key_case() function

PHP array_change_key_case() function changes the case of all key of an array.

Note: It changes case of key only.

Syntax

1. **array** array_change_key_case (**array** $array [, int $case = CASE_LOWER])

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_LOWER));
?>
```

Output:

```
Array ( [SONOO] => 550000 [VIMAL] => 250000 [RATAN] => 200000 )
```

Example:

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_LOWER));
?>
```

Output:

```
Array ( [sonoo] => 550000 [vimal] => 250000 [ratan] => 200000 )
```

3) PHP array_chunk() function

PHP array_chunk() function splits array into chunks. By using array_chunk () method, you can divide array into many parts.

Syntax

```
array array_chunk ( array $array , int $size [, bool $preserve_keys = false ] )
```

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_chunk($salary,2));
?>
```

Output:

```
Array (
[0] => Array ( [0] => 550000 [1] => 250000 )
[1] => Array ( [0] => 200000 )
)
```

4) PHP count() function

PHP count() function counts all elements in an array.

Syntax

1. int count (mixed $array_or_countable [, int $mode = COUNT_NORMAL])

Example

```
<?php
    $season=array("summer","winter","spring","autumn");
    echo count($season);
?>
```

Output:

4

5) PHP sort() function

PHP sort() function sorts all the elements in an array.

Syntax

```
bool sort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

Example

```
<?php
   $season=array("summer","winter","spring","autumn");
   sort($season);
   foreach( $season as $s )
   {
     echo "$s<br />";
   }
?>
```

Output:

```
autumn
spring
summer
winter
```

Example:

```
<?php
// Define array
$colors = array("Red", "Green", "Blue", "Yellow");

// Sorting and printing array
sort($colors);
print_r($colors);
?>
```

Similarly, you can sort the numeric elements of the array in ascending order.

```php
<?php
// Define array
$numbers = array(1, 2, 2.5, 4, 7, 10);
// Sorting and printing array
sort($numbers);
print_r($numbers);
?>
```

Sorting Indexed Arrays in Descending Order

The rsort() function is used for sorting the elements of the indexed array in descending order (alphabetically for letters and numerically for numbers).

```php
<?php
// Define array
$colors = array("Red", "Green", "Blue", "Yellow");

// Sorting and printing array
rsort($colors);
print_r($colors);
?>
```

This `print_r()` statement gives the following output:

Array ([0] => Yellow [1] => Red [2] => Green [3] => Blue)

Similarly, you can sort the numeric elements of the array in descending order.

```php
<?php
// Define array
$numbers = array(1, 2, 2.5, 4, 7, 10);

// Sorting and printing array
rsort($numbers);
print_r($numbers);
?>
```

This `print_r()` statement gives the following output:

Array ([0] => 10 [1] => 7 [2] => 4 [3] => 2.5 [4] => 2 [5] => 1)

Sorting Associative Arrays in Ascending Order By Value

The asort() function sorts the elements of an associative array in ascending order according to the value. It works just like sort(), but it preserves the association between keys and its values while sorting.

```
<?php
// Define array
$age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);

// Sorting array by value and print
asort($age);
print_r($age);
?>
```

This `print_r()` statement gives the following output:

Array ([Harry] => 14 [Peter] => 20 [Clark] => 35 [John] => 45)

Sorting Associative Arrays in Descending Order By Value

The arsort() function sorts the elements of an associative array in descending order according to the value. It works just like rsort(), but it preserves the association between keys and its values while sorting.

```
<?php
// Define array
$age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);

// Sorting array by value and print
arsort($age);
print_r($age);
?>
```

This `print_r()` statement gives the following output:

Array ([John] => 45 [Clark] => 35 [Peter] => 20 [Harry] => 14)

Sorting Associative Arrays in Ascending Order By Key

The ksort() function sorts the elements of an associative array in ascending order by their keys. It preserves the association between keys and its values while sorting, same as asort() function.

```
<?php
// Define array
$age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);

// Sorting array by key and print
ksort($age);
print_r($age);
?>
```

This print_r() statement gives the following output:

Array ([Clark] => 35 [Harry] => 14 [John] => 45 [Peter] => 20)

Sorting Associative Arrays in Descending Order By Key

The krsort() function sorts the elements of an associative array in descending order by their keys. It preserves the association between keys and its values while sorting, same as arsort() function.

```
<?php
// Define array
$age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);

// Sorting array by key and print
krsort($age);
print_r($age);
?>
```

This print_r() statement gives the following output:

Array ([Peter] => 20 [John] => 45 [Harry] => 14 [Clark] => 35)

```
array array_reverse ( array $array [, bool $preserve_keys = false ] )
```

Example

```
<?php
    $season=array("summer","winter","spring","autumn");
    $reverseseason=array_reverse($season);
    foreach( $reverseseason as $s )
    {
      echo "$s<br />";
    }
?>
```

Output:

```
autumn
spring
winter
summer
```

7) PHP array_search() function

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

Syntax

```
mixed array_search ( mixed $needle , array $haystack [, bool $strict = false ] )
```

Example

```
<?php
    $season=array("summer","winter","spring","autumn");
    $key=array_search("spring",$season);
    echo $key;
?>
```

Output:

```
2
```

8) PHP array_intersect() function

PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.

Syntax

```
array array_intersect ( array $array1 , array $array2 [, array $... ] )
```

Example

```
<?php
$name1=array("sonoo","john","vivek","smith");
$name2=array("umesh","sonoo","kartik","smith");
$name3=array_intersect($name1,$name2);
foreach( $name3 as $n )
{
  echo "$n<br />";
}
?>
```

Output:

```
sonoo
smith
```

PHP Sessions

Although you can store data using cookies but it has some security issues. Since cookies are stored on user's computer it is possible for an attacker to easily modify a cookie content to insert potentially harmful data in your application that might break your application.

Also, every time the browser requests a URL to the server, all the cookie data for a website is automatically sent to the server within the request. It means if you have stored 5 cookies on user's system, each having 4KB in size, the browser needs to upload 20KB of data each time the user views a page, which can affect your site's performance.

You can solve both of these issues by using the PHP session. A PHP session stores data on the server rather than user's computer. In a session-based environment, every user is identified through a unique number called session identifier or SID. This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

Tip: The session IDs are randomly generated by the PHP engine which is almost impossible to guess. Furthermore, because the session data is stored on the server, it doesn't have to be sent with every browser request.

What is a PHP Session?

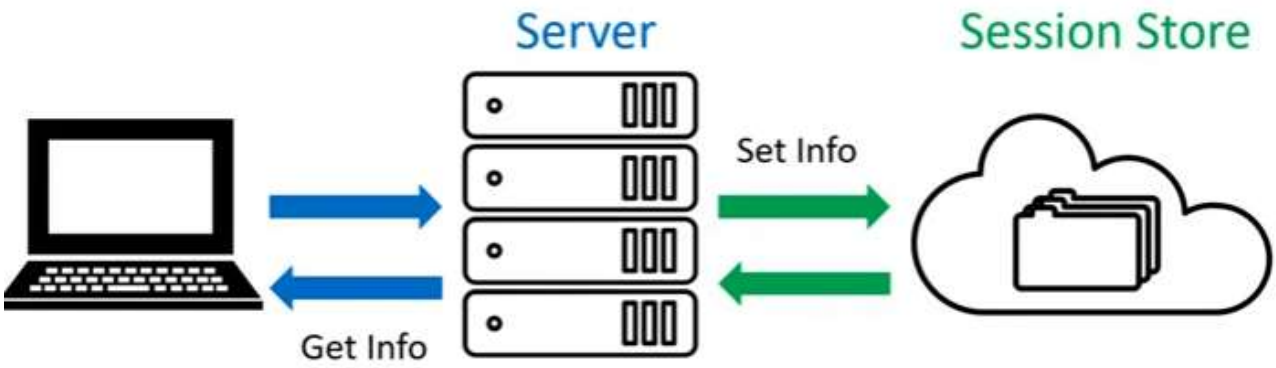
When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

Tip: If you need a permanent storage, you may want to store the data in a database.

PHP : $_SESSION



User Login Form

PHP : Three Steps to Set & Get SESSION Value

Step 1 : session_start();

Step 2 : $_SESSION[name] = value; Set Session name & value

Step 3 : echo $_SESSION[name]; Get Session value

Delete Session

Step 1 : session_unset(); Remove all session variables

Step 2 : session_destroy(); Destroy the session

Start a PHP Session

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Session Set:

```
<?php
session_start();
$_SESSION["favcolor"]="orange";
echo "Session Variable is set.";
?>
```

Session View:

```
<?php
session_start();
print_r($_SESSION);
?>

<html>
<body>
<?php
if(isset($_SESSION["favcolor"]))
{
    echo "Favourite Color: " .$_SESSION["favcolor"];
}
?>
</body>
</html>
```

Session Delete:

```
<?php
  session_start();
  session_unset();
  session_destroy();
  echo "Session is destroyed.";
?>
```

Example 2:

```php
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```php
<?php
session_start();
```

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

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();
```

```
// destroy the session
session_destroy();
?>
</body>
</html>
```

Example 3:

Storing and Accessing Session Data

You can store all your session data as key-value pairs in the `$_SESSION[]` superglobal array. The stored data can be accessed during lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

```
<?php
// Starting session
session_start();
// Storing session data
$_SESSION["firstname"] = "Peter";
$_SESSION["lastname"] = "Parker";
?>
```

To access the session data we set on our previous example from any other page on the same web domain — simply recreate the session by calling `session_start()` and then pass the corresponding key to the `$_SESSION` associative array.

```
<?php
// Starting session
session_start();

// Accessing session data
echo 'Hi, ' . $_SESSION["firstname"] . ' ' . $_SESSION["lastname"];
?>
```

The PHP code in the example above produce the following output.

Hi, Peter Parker

Note: To access the session data in the same page there is no need to recreate the session since it has been already started on the top of the page.

Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the `$_SESSION` associative array, as shown in the following example:

```
<?php
// Starting session
session_start();
// Removing session data
if(isset($_SESSION["lastname"]))
{
unset($_SESSION["lastname"]);
} ?>
```

However, to destroy a session completely, simply call the `session_destroy()` function. This function does not need any argument and a single call destroys all the session data.

```
<?php
// Starting session
session_start();
// Destroying session
session_destroy();
?>
```

Note: Before destroying a session with the session_destroy() function, you need to first recreate the session environment if it is not already there using the session_start() function, so that there is something to destroy.

Every PHP session has a timeout value — a duration, measured in seconds — which determines how long a session should remain alive in the absence of any user activity. You can adjust this timeout duration by changing the value of `session.gc_maxlifetime` variable in the PHP configuration file (`php.ini`).

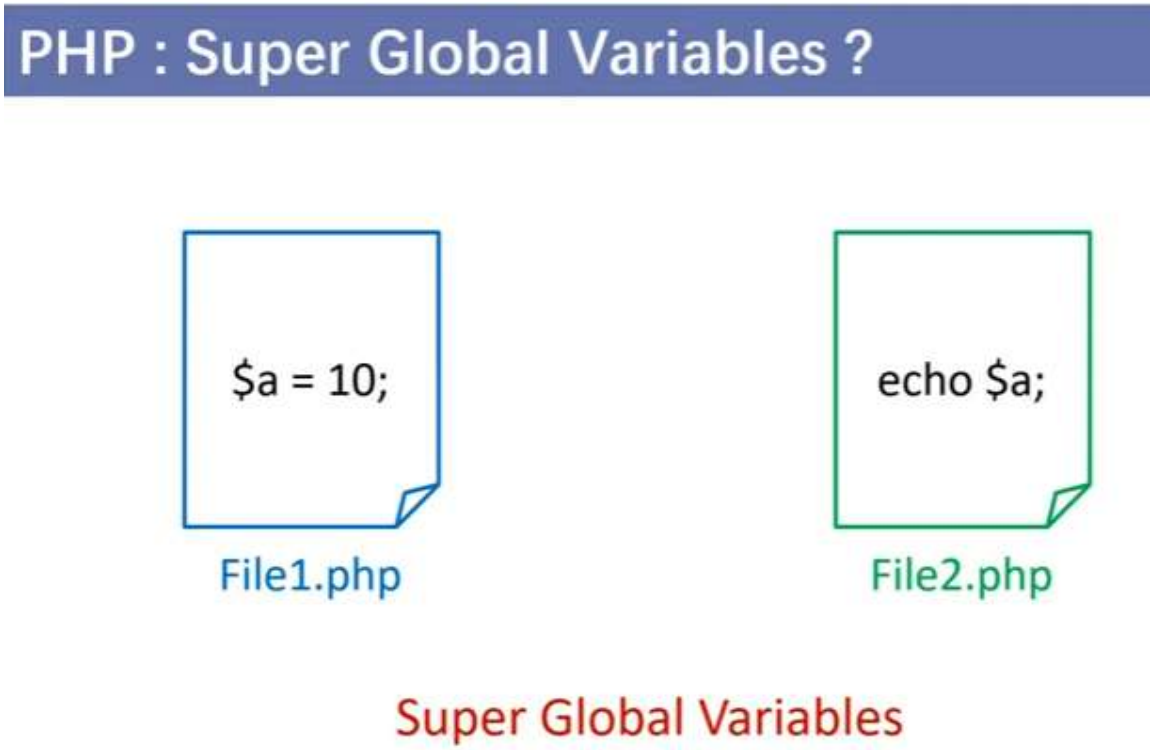
https://www.youtube.com/watch?v=NL20n0x0raI

GET & POST REQUEST

https://www.youtube.com/watch?v=fTs3deU7n10

https://www.tutorialrepublic.com/php-tutorial/php-mysql-insert-query.php

$_GET, $_POST and $_REQUEST Variables:



PHP Global Variables - Superglobals

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.

Super global variables are built-in variables that are always available in all scopes.

PHP Super Global Variables:

1. $_GLOBAL
2. $_GET
3. $_POST
4. $_REQUEST

5. $_SERVER
6. $_SESSION
7. $_COOKIE
8. $_FILES

PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS:

Example

```
<?php
$x = 75;
$y = 25;

function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

In the example above, since z is a variable present within the $GLOBALS array, it is also accessible from outside the function!

PHP $_SERVER

$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $_SERVER:

Example

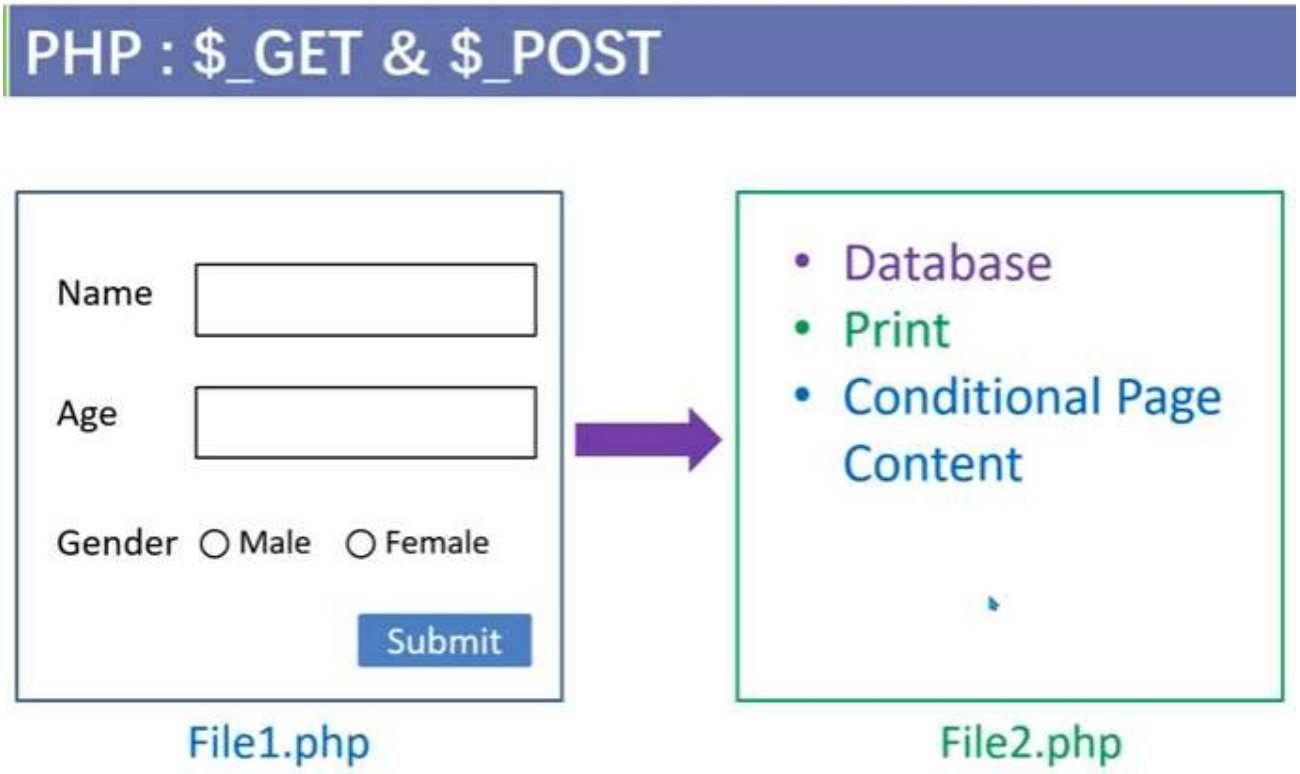
```php
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

The following table lists the most important elements that can go inside $_SERVER:

Element/Code	Description
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.saipali.education)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)

$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@saipali.education)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)

$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page



PHP $_GET

PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

Example:

form.php

```
<html>
<head>
<title>Demo</title>
</head>
<body>
<form method="get" action="test.php">
Name:<input type=text name="fname"><br>
Age:<input type=text name="age"><br>
<input type="submit" name="submit">
</form>
</body>
</html>
```

test.php

```
<?php
echo "<pre>";
print_r($_GET);
echo "</pre>";

echo $_GET['fname'];
echo $_GET['age'];

?>
```

Note: *Ctrl+F5 to remove cache from the browser.*

PHP $_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

form.php

```
<html>
```

```
<head>
<title>Demo</title>
</head>
<body>
<form method="post" action="test.php">
Name:<input type=text name="fname"><br>
Age:<input type=text name="age"><br>
<input type="submit" name="submit">
</form>
</body>
</html>
```

test.php

```
<?php
echo "<pre>";
print_r($_POST);
echo "</pre>";

echo $_POST['fname'];
echo $_POST['age'];

?>
```

PHP $_REQUEST

PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag.

form.php

```
<html>
<head>
```

```
<title>Demo</title>
</head>
<body>
<form method="post" action="test.php">
Name:<input type=text name="fname"><br>
Age:<input type=text name="age"><br>
<input type="submit" name="submit">
</form>
</body>
</html>
```

test.php

```
<?php
echo "<pre>";
print_r($_REQUEST);
echo "</pre>";

echo $_REQUEST['fname'];
echo $_REQUEST['age'];

?>
```

In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

```
 <html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

OR

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

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 < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Note: $_REQUEST will work for both GET & POST request.

PHP - Web Concepts

This session demonstrates how PHP can provide dynamic content according to browser type, randomly generated numbers or User Input. It also demonstrated how the client browser can be redirected.

Identifying Browser & Platform

PHP creates some useful **environment variables** that can be seen in the phpinfo.php page that was used to setup the PHP environment.

One of the environment variables set by PHP is *HTTP_USER_AGENT* which identifies the user's browser and operating system.

PHP provides a function getenv() to access the value of all the environment variables. The information contained in the HTTP_USER_AGENT environment variable can be used to create dynamic content appropriate to the browser.

Following example demonstrates how you can identify a client browser and operating system.

NOTE – The function preg_match() is discussed in PHP Regular expression session.

```
<html>
  <body>

    <?php
      function getBrowser() {
        $u_agent = $_SERVER['HTTP_USER_AGENT'];
        $bname = 'Unknown';
        $platform = 'Unknown';
        $version = "";

        //First get the platform?
        if (preg_match('/linux/i', $u_agent)) {
          $platform = 'linux';
        }elseif (preg_match('/macintosh|mac os x/i', $u_agent)) {
          $platform = 'mac';
        }elseif (preg_match('/windows|win32/i', $u_agent)) {
```

```
            $platform = 'windows';
        }

        // Next get the name of the useragent yes seperately and for good reason
        if(preg_match('/MSIE/i',$u_agent) && !preg_match('/Opera/i',$u_agent)) {
            $bname = 'Internet Explorer';
            $ub = "MSIE";
        } elseif(preg_match('/Firefox/i',$u_agent)) {
            $bname = 'Mozilla Firefox';
            $ub = "Firefox";
        } elseif(preg_match('/Chrome/i',$u_agent)) {
            $bname = 'Google Chrome';
            $ub = "Chrome";
        }elseif(preg_match('/Safari/i',$u_agent)) {
            $bname = 'Apple Safari';
            $ub = "Safari";
        }elseif(preg_match('/Opera/i',$u_agent)) {
            $bname = 'Opera';
            $ub = "Opera";
        }elseif(preg_match('/Netscape/i',$u_agent)) {
            $bname = 'Netscape';
            $ub = "Netscape";
        }

        // finally get the correct version number
        $known = array('Version', $ub, 'other');
        $pattern = '#(?<browser>' . join('|', $known) . ')[/ ]+(?<version>[0-9.|a-zA-
Z.]*)#';

        if (!preg_match_all($pattern, $u_agent, $matches)) {
            // we have no matching number just continue
        }

        // see how many we have
```

```
    $i = count($matches['browser']);

    if ($i != 1) {
      //we will have two since we are not using 'other' argument yet

      //see if version is before or after the name
      if (strripos($u_agent,"Version") < strripos($u_agent,$ub)){
        $version= $matches['version'][0];
      }else {
        $version= $matches['version'][1];
      }
    }else {
      $version= $matches['version'][0];
    }

    // check if we have a number
    if ($version == null || $version == "") {$version = "?";}
    return array(
      'userAgent' => $u_agent,
      'name'      => $bname,
      'version'   => $version,
      'platform'  => $platform,
      'pattern'   => $pattern
    );
  }

  // now try it
  $ua = getBrowser();
  $yourbrowser = "Your browser: " . $ua['name'] . " " . $ua['version'] .
    " on " .$ua['platform'] . " reports: <br >" . $ua['userAgent'];

  print_r($yourbrowser);
?>
```

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

This is producing following result on my machine. This result may be different for your computer depending on what you are using.

It will produce the following result –

```
Your browser: Google Chrome 54.0.2840.99 on windows reports:
Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/54.0.2840.99 Safari/537.36
```

Display Images Randomly

The PHP **rand()** function is used to generate a random number.i This function can generate numbers with-in a given range. The random number generator should be seeded to prevent a regular pattern of numbers being generated. This is achieved using the **srand()** function that specifies the seed number as its argument.

Following example demonstrates how you can display different image each time out of four images –

```
<html>
  <body>

    <?php
      srand( microtime() * 1000000 );
      $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>
```

Using HTML Forms

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

Try out following example by putting the source code in test.php script.

```
<?php
  if( $_POST["name"] || $_POST["age"] ) {
    if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
      die ("invalid name and name should be alpha");
    }

    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";

    exit();
  }
?>
<html>
  <body>

    <form action = "<?php $_PHP_SELF ?>" method = "POST">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
```

```
    </form>

  </body>
</html>
```

It will produce the following result −

- The PHP default variable **$_PHP_SELF** is used for the PHP script name and when you click "submit" button then same PHP script will be called and will produce following result −
- The method = "POST" is used to post user data to the server script. There are two methods of posting data to the server script which are discussed in PHP GET & POST chapter.

Browser Redirection

The PHP **header()** function supplies raw HTTP headers to the browser and can be used to redirect it to another location. The redirection script should be at the very top of the page to prevent any other part of the page from loading.

The target is specified by the **Location:** header as the argument to the **header()** function. After calling this function the **exit()** function can be used to halt parsing of rest of the code.

Following example demonstrates how you can redirect a browser request to another web page. Try out this example by putting the source code in test.php script.

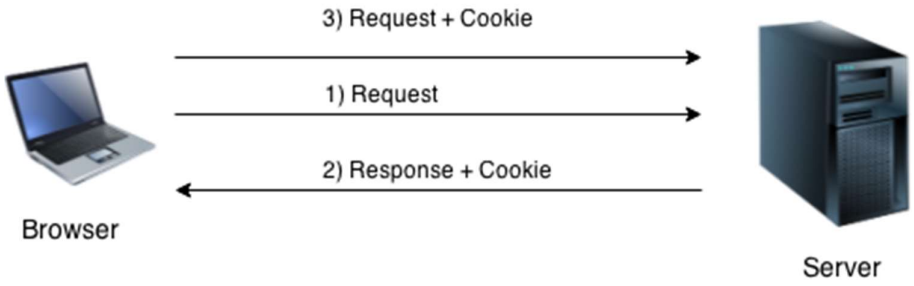
```
<?php
  if( $_POST["location"] ) {
     $location = $_POST["location"];
     header( "Location:$location" );

     exit();
  }
?>
<html>
```

```
  <body>

    <p>Choose a site to visit :</p>

    <form action = "<?php $_SERVER['PHP_SELF'] ?>" method ="POST">
      <select name = "location">.

        <option value = "https://www.saipali.education">
          Sai Pali Institute of Technology & Management
        </option>

        <option value = "http://www.google.com">
          Google Search Page
        </option>

      </select>
      <input type = "submit" />
    </form>

  </body>
</html>
```

PHP – Cookies

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

Note: PHP Cookie must be used before <html> tag.

PHP setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Syntax

1. bool setcookie (string $name [, string $value [, int $expire = 0 [, string $path

2. [, string $domain [, bool $secure = false [, bool $httponly = false]]]]])

Example

```
1. setcookie("CookieName", "CookieValue");/* defining name and value only*/
2. setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 h
   our(1*60*60 seconds or 3600 seconds)
3. setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "mydom
   ain.com", 1);
```

PHP $_COOKIE

PHP $_COOKIE superglobal variable is used to get cookie.

Example

```
$value=$_COOKIE["CookieName"];//returns cookie value
```

PHP Cookie Example

File: cookie1.php

```
<?php
```

```
setcookie("user", "Sonoo");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
   echo "Sorry, cookie is not found!";
} else {
   echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

Output:

```
Sorry, cookie is not found!
```

Firstly, cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Output:

```
Cookie Value: Sonoo
```

PHP Delete Cookie

If you set the expiration date in past, cookie will be deleted.

File: cookie1.php

```
1. <?php
2. setcookie ("CookieName", "", time() - 3600);// set the expiration date to one h
   our ago
3. ?>
```

Following example will create two cookies **name** and **age** these cookies will be expired after one hour.

```
<?php
  setcookie("name", "John Watkin", time()+3600, "/","", 0);
  setcookie("age", "36", time()+3600, "/", "",  0);
```

```
?>
<html>

  <head>
    <title>Setting Cookies with PHP</title>
  </head>

  <body>
    <?php echo "Set Cookies"?>
  </body>

</html>
```

Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables. Following example will access all the cookies set in above example.

```
<html>
  <head>
    <title>Accessing Cookies with PHP</title>
  </head>

  <body>

    <?php
      echo $_COOKIE["name"]. "<br />";

      /* is equivalent to */
      echo $HTTP_COOKIE_VARS["name"]. "<br />";

      echo $_COOKIE["age"] . "<br />";

      /* is equivalent to */
```

```
      echo $HTTP_COOKIE_VARS["age"] . "<br />";
    ?>

  </body>
</html>
```

You can use **isset()** function to check if a cookie is set or not.

```
<html>

  <head>
    <title>Accessing Cookies with PHP</title>
  </head>

  <body>

    <?php
      if( isset($_COOKIE["name"]))
        echo "Welcome " . $_COOKIE["name"] . "<br />";

      else
        echo "Sorry... Not recognized" . "<br />";
    ?>

  </body>
</html>
```

Deleting Cookie with PHP

Officially, to delete a cookie you should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired –

```
<?php
  setcookie( "name", "", time()- 60, "/","", 0);
  setcookie( "age", "", time()- 60, "/","", 0);
```

```
?>
<html>

  <head>
    <title>Deleting Cookies with PHP</title>
  </head>

  <body>
    <?php echo "Deleted Cookies" ?>
  </body>

</html>
```

1 Minutes = 60 secs

1 Hour = 60 mins

60 * 60 = 3600 secs

3600 * 24 = 86400

1 hrs

1 Day

86400 * 30

days

1 month

```
<?php
$cookie_name="user";
$cookie_value="Technox IT";
setcookie($cookie_name,$cookie_value,time() +(86400 *30),"/");

?>
<html>
<head>
<title>Demo</title>
```

```
</head>
<body>
  <?php
 if(!isset($_COOKIE[$cookie_name]))
   {
      echo "Cookie is not set.";
    }
else
{
       echo $_COOKIE[$cookie_name];   //retrieve the cookie
    }
   ?>
```

test.php

```
<?php
       echo "Cookie Value: " .$COOKIE["user"];
       setcookie("user","",time() - (86400 * 30), "/");
?>
```

PHP - File Inclusion

You can include the content of a PHP file into another PHP file before the server executes it. There are two PHP functions which can be used to included one PHP file into another PHP file.

- The include() Function
- The require() Function

The include and require statements are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

Syntax

include '*filename*';

or

require '*filename*';

NOTE:

Use require when the file is required by the application.

Use include when the file is not required and application should continue when file is not found.

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.

The include () Function

The include () function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the **include()** function generates a warning but the script will continue execution.

Assume you want to create a common menu for your website. Then create a file menu.php with the following content.

```
<a href="http://www.tutorialspoint.com/index.htm">Home</a> -
<a href="http://www.tutorialspoint.com/ebxml">ebXML</a> -
<a href="http://www.tutorialspoint.com/ajax">AJAX</a> -
<a href="http://www.tutorialspoint.com/perl">PERL</a> <br />
```

Now create as many pages as you like and include this file to create header. For example now your test.php file can have following content.

```
<html>
  <body>

    <?php include("menu.php"); ?>
    <p>This is an example to show how to include PHP file!</p>
```

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

It will produce the following result –

The require () Function

The require () function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the **require()** function generates a fatal error and halt the execution of the script.

So there is no difference in require() and include() except they handle error conditions. It is recommended to use the require() function instead of include(), because scripts should not continue executing if files are missing or misnamed.

You can try using above example with require() function and it will generate same result. But if you will try following two examples where file does not exist then you will get different results.

```
<html>
  <body>

    <?php include("xxmenu.php"); ?>
    <p>This is an example to show how to include wrong PHP file!</p>

  </body>
</html>
```

This will produce the following result –

```
This is an example to show how to include wrong PHP file!
```

Now lets try same example with require() function.

```
<html>
  <body>

    <?php require("xxmenu.php"); ?>
    <p>This is an example to show how to include wrong PHP file!</p>

  </body>
</html>
```

This time file execution halts and nothing is displayed.

NOTE – You may get plain warning messages or fatal error messages or nothing at all. This depends on your PHP Server configuration.

The include and require statements **are identical**, except upon failure: require will produce a fatal error (E_COMPILE_ERROR) and stop the script. include will only produce a warning (E_WARNING) and the script will continue.

Example 1

Assume we have a standard footer file called "footer.php", that looks like this:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " Sai Pali Institute of Technology &
Management</p>";
?>
```

To include the footer file in a page, use the include statement:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>
```

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

Example 2

Assume we have a standard menu file called "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
<a href="/css/default.asp">CSS Tutorial</a> -
<a href="/js/default.asp">JavaScript Tutorial</a> -
<a href="default.asp">PHP Tutorial</a>';
?>
```

All pages in the Web site should use this menu file. Here is how it can be done (we are using a element so that the menu easily can be styled with CSS later):

Example

```
<html>
<body>

<div class="menu">
<?php include 'menu.php';?>
</div>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>

</body>
</html>
```

Example 3

Assume we have a file called "vars.php", with some variables defined:

```
<?php
$color='red';
$car='BMW';
?>
```

Then, if we include the "vars.php" file, the variables can be used in the calling file:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'vars.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

PHP include vs. require

The require statement is also used to include a file into the PHP code.

However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php require 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

PHP - Files & I/O

This chapter will explain following functions related to files −

- Opening a file
- Reading a file
- Writing a file
- Closing a file

Opening and Closing Files

The PHP **fopen()** function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.

Files modes can be specified as one of the six options in this table.

Sr.No	Mode & Purpose
1	**r** Opens the file for reading only. Places the file pointer at the beginning of the file.
2	**r+** Opens the file for reading and writing. Places the file pointer at the beginning of the file.
3	**w** Opens the file for writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.

4	**w+** Opens the file for reading and writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
5	**a** Opens the file for writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.
6	**a+** Opens the file for reading and writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.

If an attempt to open a file fails then **fopen** returns a value of **false** otherwise it returns a **file pointer** which is used for further reading or writing to that file.

After making a changes to the opened file it is important to close it with the **fclose()** function. The **fclose()** function requires a file pointer as its argument and then returns **true** when the closure succeeds or **false** if it fails.

Reading a file

Once a file is opened using **fopen()** function it can be read with a function called **fread()**. This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.

The files length can be found using the **filesize()** function which takes the file name as its argument and returns the size of the file expressed in bytes.

So here are the steps required to read a file with PHP.

- Open a file using **fopen()** function.
- Get the file's length using **filesize()** function.
- Read the file's content using **fread()** function.
- Close the file with **fclose()** function.

The following example assigns the content of a text file to a variable then displays those contents on the web page.

```
<html>

  <head>
    <title>Reading a file using PHP</title>
  </head>

  <body>

    <?php
      $filename = "tmp.txt";
      $file = fopen( $filename, "r" );

      if( $file == false ) {
        echo ( "Error in opening file" );
        exit();
      }

      $filesize = filesize( $filename );
      $filetext = fread( $file, $filesize );
      fclose( $file );

      echo ( "File size : $filesize bytes" );
      echo ( "<pre>$filetext</pre>" );
    ?>
```

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

It will produce the following result –

Writing a file

A new file can be written or text can be appended to an existing file using the PHP **fwrite()** function. This function requires two arguments specifying a **file pointer** and the string of data that is to be written. Optionally a third integer argument can be included to specify the length of the data to write. If the third argument is included, writing would will stop after the specified length has been reached.

The following example creates a new text file then writes a short text heading inside it. After closing this file its existence is confirmed using **file_exist()** function which takes file name as an argument

```
<?php
  $filename = "/home/user/guest/newfile.txt";
  $file = fopen( $filename, "w" );

  if( $file == false ) {
     echo ( "Error in opening new file" );
     exit();
  }
  fwrite( $file, "This is  a simple test\n" );
  fclose( $file );
?>
<html>

  <head>
```

```
    <title>Writing a file using PHP</title>
  </head>

  <body>

    <?php
      $filename = "newfile.txt";
      $file = fopen( $filename, "r" );

      if( $file == false ) {
        echo ( "Error in opening file" );
        exit();
      }

      $filesize = filesize( $filename );
      $filetext = fread( $file, $filesize );

      fclose( $file );

      echo ( "File size : $filesize bytes" );
      echo ( "$filetext" );
      echo("file name: $filename");
    ?>

  </body>
</html>
```

It will produce the following result −

```
File size : 23 bytes
This is  a simple test
file name: newfile.txt
```

https://www.youtube.com/watch?v=FrJUJwf6YDM

FILES & DIRECTORIES:

1) List all the files in a directory
2) Check for specific files in a directory
3) Check if the name is a directory or file
4) Create directory
5) Copy files between directories

```
<?php
//$current_dir = getCwd() . '/images';
$current_dir = getCwd();
echo "Current directory is $current_dir";
?>

$path="saipalidemo";
$result = scandir($path);
var_dump($result);
```

```
<?php
// $result = mkdir("/testingTechnox", "0777");

$current_dir = getCwd() . '/images';

echo "Current directory is $current_dir";

$path = "saipalidemo";
$result = scandir($current_dir);
var_dump($result);
?>
```

```
//removing . and .. from the directory
 foreach($result as $dir)
{
   if($dir != "." && $dir != "..")
     {
        echo $dir . PHP_EOL;
     }
}

One more method
 $directory = array_diff($result,['.','..']);
var_dump($directory);

foreach($directory as $dir)
{
   echo $dir . PHP_EOL;
}

============================
2) Check for specific files in a directory
3) Check if the name is a directory or file.

//is_file or is_dir

$result = scandir($path);
$result = array_diff($result,['.','..']);

  foreach($result as $dir)
{
   if(is_file($path ."/" .$dir))
     {
        echo $dir . PHP_EOL;
     }
}
```

Show only directoires:

```
$result = scandir($path);
$result = array_diff($result,['.','..']);

  foreach($result as $dir)
{
   if(is_dir($path ."/" .$dir))
     {
        echo $dir . PHP_EOL;
     }
}
```

==========

Let us create a directory

```
$result = glob("*.php");
var_dump($result);

if(!file_exists("saipalidemo1")) {
mkdir("saipalidemo1");
}
```

Copy files from some other folder:

```
<?php
// Copy the file from /user/desktop/geek.txt
// to user/Downloads/geeksforgeeks.txt'
// directory

// Store the path of source file
$source = 'f1.php';

// Store the path of destination file
$destination = 'saipalidemo1/f1.php';

// Copy the file from /user/desktop/geek.txt
```

```
// to user/Downloads/geeksforgeeks.txt'
// directory
if (!copy($source, $destination)) {
	echo "File can't be copied! \n";
} else {
	echo "File has been copied! \n";
}
?>
```

Delete a file from a folder:

```
<?php
// PHP program to delete a file named gfg.txt
// using unlike() function

$file_pointer = "uganda.txt";

// Use unlink() function to delete a file
if (!unlink($file_pointer)) {
	echo ("$file_pointer cannot be deleted due to an error");
} else {
	echo ("$file_pointer has been deleted");
}

?>
```

Create a file and delete

```
<?php
// PHP program to delete a file named gfg.txt
// using unlike() function

$file_pointer = fopen('gfg.txt', 'w+');
```

```
// writing on a file named gfg.txt
fwrite($file_pointer, 'A computer science portal for developers!');
fclose($file_pointer);

// Use unlink() function to delete a file
unlink('gfg.txt');
?>
```

PHP - File Uploading

A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script.

Information in the **phpinfo.php** page describes the temporary directory that is used for file uploads as **upload_tmp_dir** and the maximum permitted size of files that can be uploaded is stated as **upload_max_filesize**. These parameters are set into PHP configuration file **php.ini**

The process of uploading a file follows these steps –

- The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
- The user clicks the browse button and selects a file to upload from the local PC.
- The full path to the selected file appears in the text filed then the user clicks the submit button.
- The selected file is sent to the temporary directory on the server.
- The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
- The PHP script confirms the success to the user.

As usual when writing files it is necessary for both temporary and final locations to have permissions set that enable file writing. If either is set to be read-only then process will fail.

An uploaded file could be a text file or image file or any document.

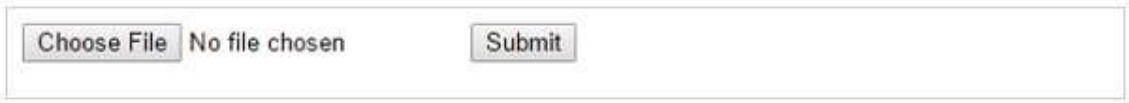
Creating an upload form

The following HTM code below creates an uploader form. This form is having method attribute set to **post** and enctype attribute is set to **multipart/form-data**

```
<?php
  if(isset($_FILES['image'])){
     $errors= array();
     $file_name = $_FILES['image']['name'];
     $file_size =$_FILES['image']['size'];
     $file_tmp =$_FILES['image']['tmp_name'];
     $file_type=$_FILES['image']['type'];
     $file_ext=strtolower(end(explode('.',$_FILES['image']['name'])));

     $extensions= array("jpeg","jpg","png");

     if(in_array($file_ext,$extensions)=== false){
        $errors[]="extension not allowed, please choose a JPEG or PNG file.";
     }

     if($file_size > 2097152){
        $errors[]='File size must be excately 2 MB';
     }

     if(empty($errors)==true){
        move_uploaded_file($file_tmp,"images/".$file_name);
        echo "Success";
     }else{
        print_r($errors);
     }
```

```
  }
?>
<html>
  <body>

    <form action="" method="POST" enctype="multipart/form-data">
      <input type="file" name="image" />
      <input type="submit"/>
    </form>

  </body>
</html>
```

It will produce the following result –



Creating an upload script

There is one global PHP variable called **$_FILES**. This variable is an associate double dimension array and keeps all the information related to uploaded file. So, if the value assigned to the input's name attribute in uploading form was **file**, then PHP would create following five variables –

- **$_FILES['file']['tmp_name']** – the uploaded file in the temporary directory on the web server.
- **$_FILES['file']['name']** – the actual name of the uploaded file.
- **$_FILES['file']['size']** – the size in bytes of the uploaded file.
- **$_FILES['file']['type']** – the MIME type of the uploaded file.
- **$_FILES['file']['error']** – the error code associated with this file upload.

Example

Below example should allow upload images and gives back result as uploaded file information.

```php
<?php
  if(isset($_FILES['image'])){
    $errors= array();
    $file_name = $_FILES['image']['name'];
    $file_size = $_FILES['image']['size'];
    $file_tmp = $_FILES['image']['tmp_name'];
    $file_type = $_FILES['image']['type'];
    $file_ext=strtolower(end(explode('.',$_FILES['image']['name'])));

    $extensions= array("jpeg","jpg","png");

    if(in_array($file_ext,$extensions)=== false){
      $errors[]="extension not allowed, please choose a JPEG or PNG file.";
    }

    if($file_size > 2097152) {
      $errors[]='File size must be excately 2 MB';
    }

    if(empty($errors)==true) {
      move_uploaded_file($file_tmp,"images/".$file_name);
      echo "Success";
    }else{
      print_r($errors);
    }
  }
?>
<html>
  <body>

    <form action = "" method = "POST" enctype = "multipart/form-data">
      <input type = "file" name = "image" />
      <input type = "submit"/>
```

```
      <ul>
        <li>Sent file: <?php echo $_FILES['image']['name'];  ?>
        <li>File size: <?php echo $_FILES['image']['size'];  ?>
        <li>File type: <?php echo $_FILES['image']['type'] ?>
      </ul>

    </form>

  </body>
</html>
```

It will produce the following result –