**PHP Tutorial part**

# PHP Tutorial

Welcome to the modern PHP tutorial! This PHP tutorial helps you learn how to develop dynamic websites and web applications using PHP from scratch.

PHP is one of the most popular programming languages for web development.

PHP allows you to develop various web applications, including blogs, content management systems (CMS), and online stores.

## 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](https://en.wikipedia.org/wiki/Rasmus_Lerdorf" \t "https://www.phptutorial.net/php-tutorial/what-is-php/_blank) in 1994. It’s currently maintained by the [PHP Development Team](https://www.php.net/credits.php" \t "https://www.phptutorial.net/php-tutorial/what-is-php/_blank)

### PHP is a server-side language

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

The web browser sends an HTTP request to a web server where phptutorial.net 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.

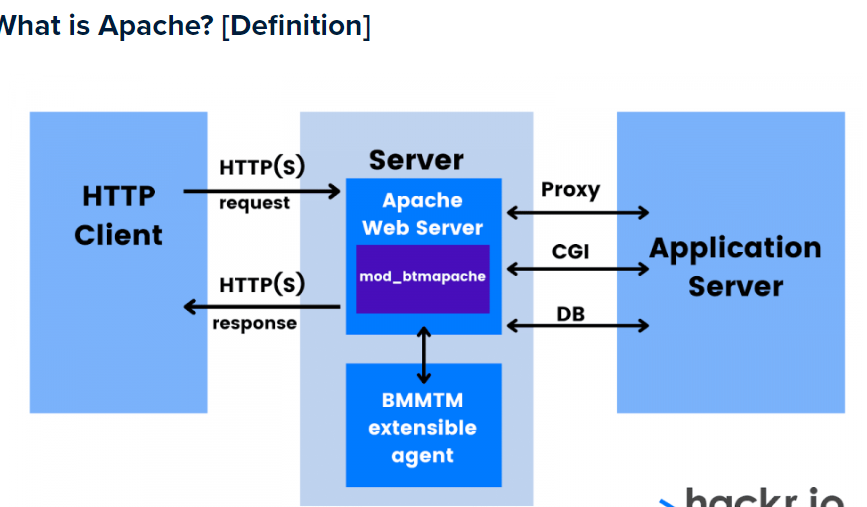
**NGINX VS Apache:**

<https://hackr.io/blog/nginx-vs-apache#:~:text=NGINX%20performs%202.5%20times%20faster,over%20Apache%20with%20static%20content.>

The two most popular open-source web servers powering the Internet today are the Apache HTTP server and NGINX. Over 50% of the websites in the world run on these two web servers. For almost two decades, the Apache Web server served around 60 percent of the world’s websites until NGINX (pronounced as “engine-x”) came into existence.

Due to the explosive growth in data traffic volumes and the number of users of the world-wide-web, NGINX was introduced to overcome the performance limitations of Apache web servers. NGINX, designed for higher concurrency, can be deployed as a standalone web server, and as a frontend proxy for Apache and other web servers.

But when it comes to the nitty-gritty, how do this two stack up against each other? That's what we look into here. This NGINX vs Apache head-to-head comparison covers everything you need to know about the two.

Apache is an open-source software developed and maintained by an open community of developers and it runs on a wide variety of operating systems. It was developed to deliver web content through the internet and is well known for playing a key role in the initial growth of the World Wide Web.

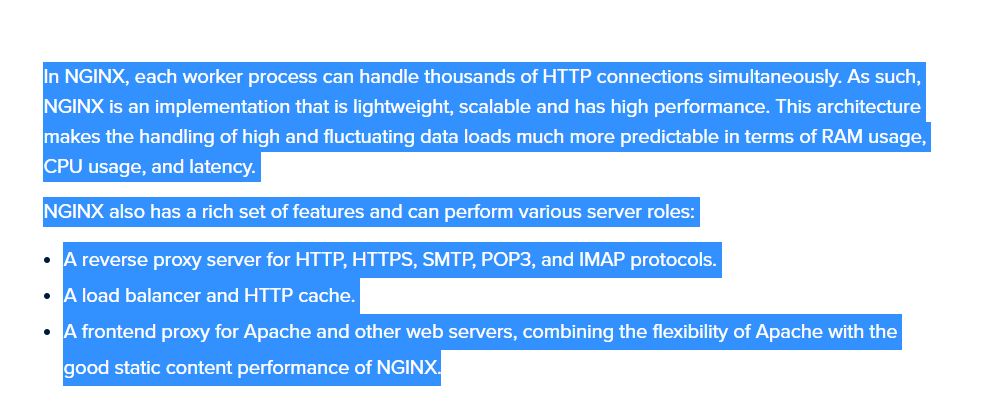
The Apache architecture includes the Apache Core and modules. The core component provides basic server-like capability, accepting connections and managing concurrency. Various modules correspond to different functionalities that are executed on each request. A specific Apache deployment can be configured to include different modules such as security features, management of dynamic content, or basic HTTP request processing

NGINX was written specifically to address the performance limitations of Apache web servers. The performance and scalability of NGINX arise from its event-driven architecture. It differs significantly from Apache’s process-or-thread-per-connection approach.

In NGINX, each worker process can handle thousands of HTTP connections simultaneously. As such, NGINX is an implementation that is lightweight, scalable and has high performance. This architecture makes the handling of high and fluctuating data loads much more predictable in terms of RAM usage, CPU usage, and latency.

NGINX also has a rich set of features and can perform various server roles:

* A reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols.
* A load balancer and HTTP cache.
* A frontend proxy for Apache and other web servers, combining the flexibility of Apache with the good static content performance of NGINX.



## ****Differences Between Apache and NGINX****

We use 6 criteria to compare Apache versus NGINX. As you’ll see, this webserver comparison largely results in the conclusion that they both have their uses.

### ****1. Simplicity****

Developing and innovating applications on Apache is easy. The one-connection-per-process model makes it very easy to insert modules at any point in its web serving logic. Developers could add code in such a way that if there were failures only the worker process running the code would be affected. Processing of all other connections would continue undisturbed.

NGINX, on the other hand, has a sophisticated architecture making module development more challenging. NGINX module developers need to create efficient and accurate code, without any failures, that interacts appropriately with the complex event-driven kernel to avoid blocking operations.

### ****2. Performance****

Apache vs NGINX performance is measured by how the server delivers large volumes of content to the client browser and this is an important factor. Content can be static or dynamic.

#### ****2.1. Static Content****

NGINX performs 2.5 times faster than Apache, which tested running up to 1,000 simultaneous connections. Another benchmark running with 512 simultaneous connections showed that NGINX is about twice as fast and consumes less memory.

Undoubtedly, NGINX has an advantage over Apache with static content. So if you need to serve concurrent static content, NGINX is the right choice.

#### ****2.3. Dynamic Content****

For displaying dynamic content, Speedemy’s benchmark test results revealed that the Nginx vs Apache performance for both were exactly the same. One probable reason for this is that almost all of the request processing time is spent in the PHP runtime environment rather than the core part of the web server. The PHP runtime environment is pretty similar to both web servers.

Apache can also process dynamic content by embedding a processor of a language like PHP into each of its worker instances. This allows it to execute dynamic content within the web server itself without having to rely on external components. These dynamic processors can be enabled through the use of dynamically loadable modules.

NGINX does not have any ability to process dynamic content natively. To handle PHP and other requests for dynamic content, NGINX must pass to an external processor for execution and wait for the rendered content to be sent back. However, this method has some advantages as well. Since the dynamic interpreter is not embedded in the worker process, its overhead will only be present for dynamic content.

### ****3. OS support****

Apache runs on all operating systems such as UNIX, Linux, or BSD and has full support for Microsoft Windows. NGINX also runs on several modern Unix-like systems and has support for Windows, but its performance on Windows is not as stable as it is on UNIX platforms.

### ****4. Security****

Both Apache and NGINX are secure web servers. The Apache Security Team exists to provide help and advice to Apache projects on security issues and to coordinate the handling of security vulnerabilities. It is important to configure the servers properly and know what each option in the settings does. There are many guidelines to secure the servers to prevent security attacks.

### ****5. Flexibility****

Web servers can be customized by adding modules. Apache has had dynamic module loading for the longest time, so all Apache modules support this.

NGINX Plus, a software load balancer, web server, and content cache built on top of open source NGINX, uses a modular architecture too. New features and functionality can be added with software modules, which can be plugged into a running NGINX Plus instance on demand.

Dynamic modules add functionality to NGINX Plus such as geo-locating users by IP address, resizing images, and embedding Lua scripts into the NGINX Plus event-processing model. Modules are created both by NGINX and third-party developers.

The majority of the necessary core module functionality (e.g., proxy, caching, load-balancing) is supported by both web servers.

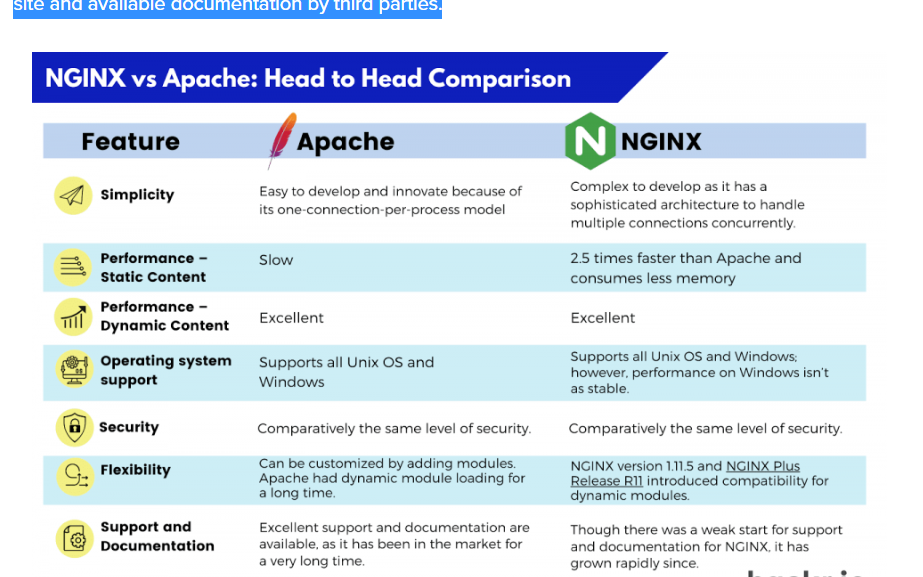
### ****6. Support and Documentation****

A major point to consider is the available help and support for the web servers among other software. Because Apache has been popular for so long, support for the server is fairly ubiquitous. There is a large library of first and third-party documentation available for the core server and for task-based scenarios involving hooking Apache up with other software.

Along with documentation, many tools and [web projects](https://hackr.io/blog/best-web-development-projects" \t "https://hackr.io/blog/_blank) include tools to bootstrap themselves within an Apache environment. This may be included in the projects themselves, or in the packages maintained by your distribution's packaging team.

Apache, in general, has more support from third-party projects simply because of its market share and the length of time it has been available.

For NGINX, in the past, it was difficult to find comprehensive English-language documentation due to the fact that most of the early development and documentation were in Russian. However, today the documentation has been filled out and there are now plenty of administration resources on the NGINX site and available documentation by third parties.



## ****Conclusion****

As seen, both Apache and NGINX are powerful, flexible, and capable. The latest versions of both servers are competitive in all areas. Deciding which server is best for you is largely a function of evaluating your specific requirements and deciding the best option. For example, for static content, NGINX would be much better.

There’s no cut-and-dry answer as to which is better. Like many decisions when it comes to development, the choice will come down to what your requirements are.

## ****Frequently Asked Questions****

#### ****1. Is Apache Better than NGINX?****

In terms of performance, NGINX is much better than Apache. NGINX performs 2.5 times faster than Apache — and consumes less memory as well. However, Apache has more functionality and features. It is worth noting that it is possible to use both together.

#### ****2. Are Apache and NGINX the Same?****

No, Apache and NGINX are not the same. They are both open-source web servers, but Apache is an HTTP server and NGINX can also be used as a reverse proxy server.

#### ****3. Can NGINX Replace Apache?****

It is unlikely. While NGINX is more efficient, it does not have Apache’s configuration system, and they both have their advantages and disadvantages.

#### ****4. Why is NGINX Faster than Apache?****

The architecture of NGINX makes it more capable at handling heavy static content. It does not require context switching so a single thread can serve all requests.

#### ****5. Can PHP Run on NGINX?****

Yes, you can run PHP on NGINX. You can easily find the installation instructions online.

#### ****6. Is Apache Still Used?****

Yes, Apace is still widely used. It remains one of the most popular web servers.

#### ****7. Does Facebook Use Apache or NGINX?****

Neither. Facebook uses a custom server that they designed themselves and which runs on Linux.

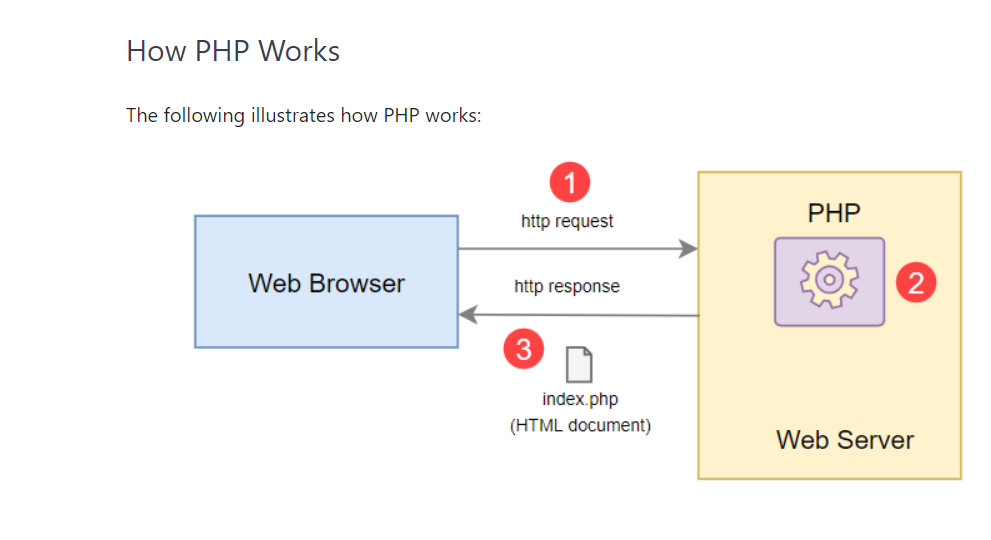
****People are also reading:****

* [Best Nginx Tutorials](https://hackr.io/tutorials/learn-nginx?ref=blog-post)
* [Types of Software Testing](https://hackr.io/blog/types-of-software-testing)
* [AWS Certifications](https://hackr.io/blog/aws-certifications)
* [Top AWS Interview Questions](https://hackr.io/blog/aws-interview-questions)
* [Difference between Google Cloud vs AWS vs Azure](https://hackr.io/blog/google-cloud-vs-aws-vs-azure)
* [Top Docker Interview Questions](https://hackr.io/blog/docker-interview-questions)
* [Docker vs Kubernetes](https://hackr.io/blog/kubernetes-vs-docker)
* [Best Linux Certifications](https://hackr.io/blog/best-linux-certifications)
* [Basic Linux Commands with Examples](https://hackr.io/blog/basic-linux-commands)
* [Linux Interview Questions and Answers](https://hackr.io/blog/linux-interview-questions)

## 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.



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.

<https://www.phptutorial.net/php-tutorial/what-is-php/>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <div>

        <?php

        // $character=array(array("name"=>"ajman","occupation"=>"Engineer","age"=>24),

        // array("name"=>"kalim","occupation"=>"ASE Engineer","age"=>25),

        // array("name"=>"Noman","occupation"=>"AE Engineer","age"=>26),

        // array("name"=>"Rehamn","occupation"=>"ME Engineer","age"=>28));

        // print $character[1]['occupation'];

    //    print $character[2]['age'];

    // $character[]=array("name"=>"janala","occupation"=>"ASE EE","age"=>28);

    // print $character[3]['name'];//Rehman

    // print $character[4]['name'];//janala

    // $character[0]="specialist";

    // $character[0]["specialist"]="IT";

    // print $character[0]['specialist'];//

    // print $character[2]['age'];

    // print $character[3]['name'];

    // print $character[1]['age'];

    // print end($character);

    // $user=array("Bertt","Janana","Kalim","Hananana");

    // print $user[count($user)-1];// hannannannanna

    // $user[66]="gananannana";

    // $user[67]="gananannana";

    // $user[68]="gananannana";

    // $user[69]="gananannana";

    // $user[80]="gananannana";

    // $user[47]="gananannana";

    // $user[68]="gananannana";

    // $user[89]="gananannana";

    // $user[41]="gananannana";

    // print end($user);//ganannanannanannanana

    // $character=array("name"=>"ajman","occupation"=>"Engineer","age"=>24);

    // foreach($character as $key=>$temp){

    //     print "$key=$temp<br/>";

    ///////////////////////////////////////////////

   // }

    //  foreach($user as $key=>$temp){

    //      print "$key=$temp<br/>";

    //    }

    $character=array(array("name"=>"Hassan khan","age"=>25,"City"=>"islamabad","Country"=>"Pakistan"),

    array("name"=>"Hira sohail","age"=>25,"City"=>"Lahore","Country"=>"Pakistan"),

    array("name"=>"Noman khan","age"=>27,"City"=>"Lahore","Country"=>"Pakistan"),

    array("name"=>"Zubair khan","age"=>26,"City"=>"Multan","Country"=>"Pakistan"),

    array("name"=>"Arsalan shouakat","age"=>25,"City"=>"islamabad","Country"=>"Pakistan"),

    array("name"=>"ajman ullah khan","age"=>24,"City"=>"islamabad","Country"=>"Pakistan"),

    array("name"=>"ahmad Munir ","age"=>27,"City"=>"Lahore","Country"=>"Pakistan"),

    array("name"=>"Shahid akhtar","age"=>24,"City"=>"Lahore","Country"=>"Pakistan"),

    array("name"=>"Zeeshan ahmad","age"=>25,"City"=>"Minawali","Country"=>"Pakistan"),

    array("name"=>"Shuja butt","age"=>24,"City"=>"Multan","Country"=>"Pakistan"));

    // print\_r($character);

    // looping through the associatve array

    // foreach($character as $val)

    // {

    //    print "<p>";

    //    foreach($val as $key=>$final\_val)

    //    {

    //     print "$key:$final\_val <br/>";

    //     print"</p>";

    //    }

    // }

    print\_r($character);

          ?>

    </div>

</body>

</html>

