**AL FINAL SE RESUME, EN QUE CONDICIONES SE RECOMIENDA CADA UNO Y SUS LIMITACIONES, EN RESUMEN, PARA SISTEMAS PEQUENOS PHP.**

**#1 Coding**

If the syntax is easy and well engineered, you’ll be able to do more with less code.

**Node.js**

On the surface, it may seem that Node.js requires more lines of code to execute the same functions as PHP. Thankfully, you need not remember the syntax of different languages while coding because the entire development environment is programmed with Javascript. So Node.js may sometimes require relatively longer lines of code, but it operates with only one language.

**PHP**

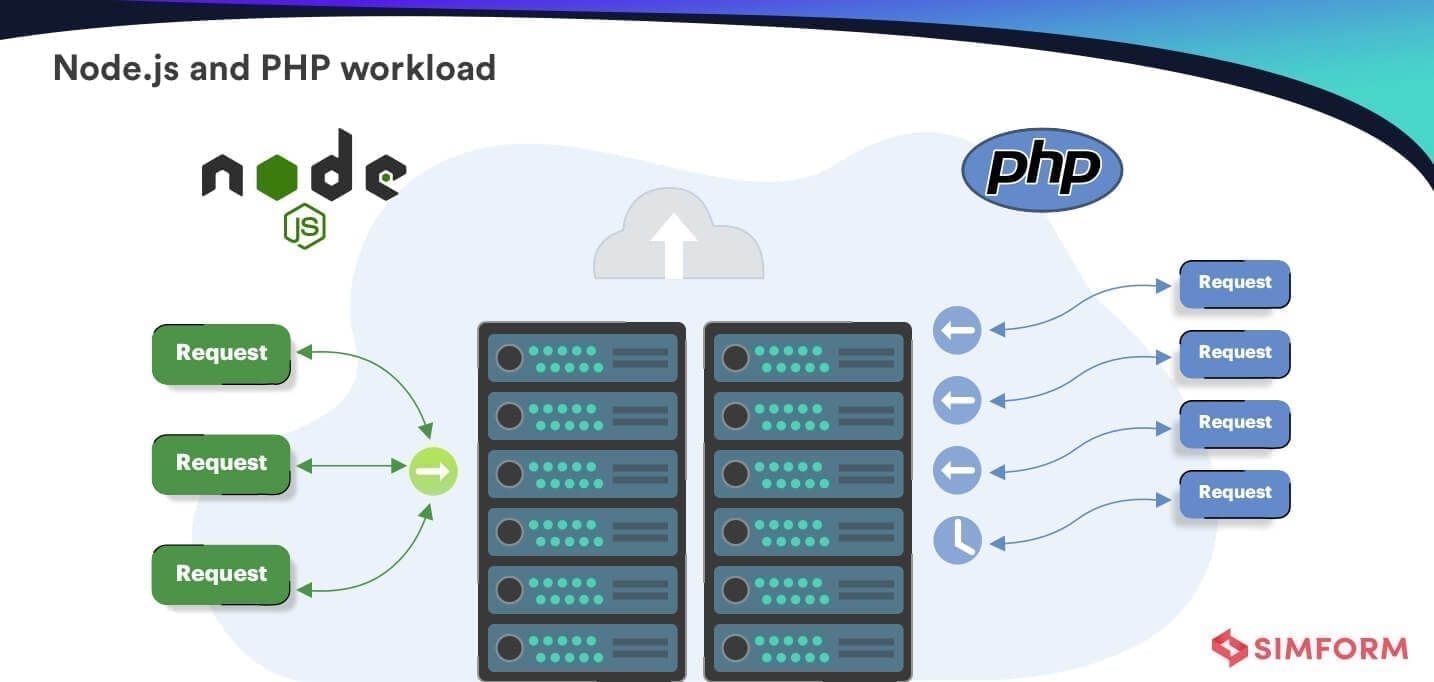
PHP has been around for a while and has flourished significantly over the years. It may seem from afar that PHP requires lesser lines of code to perform a function and is thus easier. But then, you need to be fluent in how operators and generators work in Linux, Apache HTTP Server, MySQL, and PHP itself (commonly referred to as LAMP).

**Simform Recommends**

On a standalone basis, both languages are easily understood. It’d take no time for a rookie developer to attain proficiency in either language. However, Node.js is easier to execute despite its longer code thanks to its setup of just one programming language. So when coding is concerned, Node.js is a winner in our books.

**#2 Client-Side Coding**

Server-side coding focuses on the effective execution of functions that would eventually constitute the UI. This is a critical parameter since it largely determines the user-experience, data used on the client’s device, and page loading speeds of an app.

**Node.js**

The key benefit of this development environment is that you can import all the logic of the backend code for the server-side code. It then becomes easy to access the features you have allocated on both fronts. Also, Node.js moves a lot of workload to the client side. So if your target audience has smartphones with decent processing power, Node.js is ideal for your use-case. However, it uses AJAX engine to create new web pages upon client requests, so it must be connected to the server consistently.

**PHP**

PHP was designed to maintain dynamic web-pages. Hence, several functionalities that you might need for UX elements of the web-page are already inbuilt.

Beyond that, PHP depends on the HTML code created for each static web page. Since there is no change necessary for each web-page, it easily loads the page without exerting stress on the processing power of client devices.

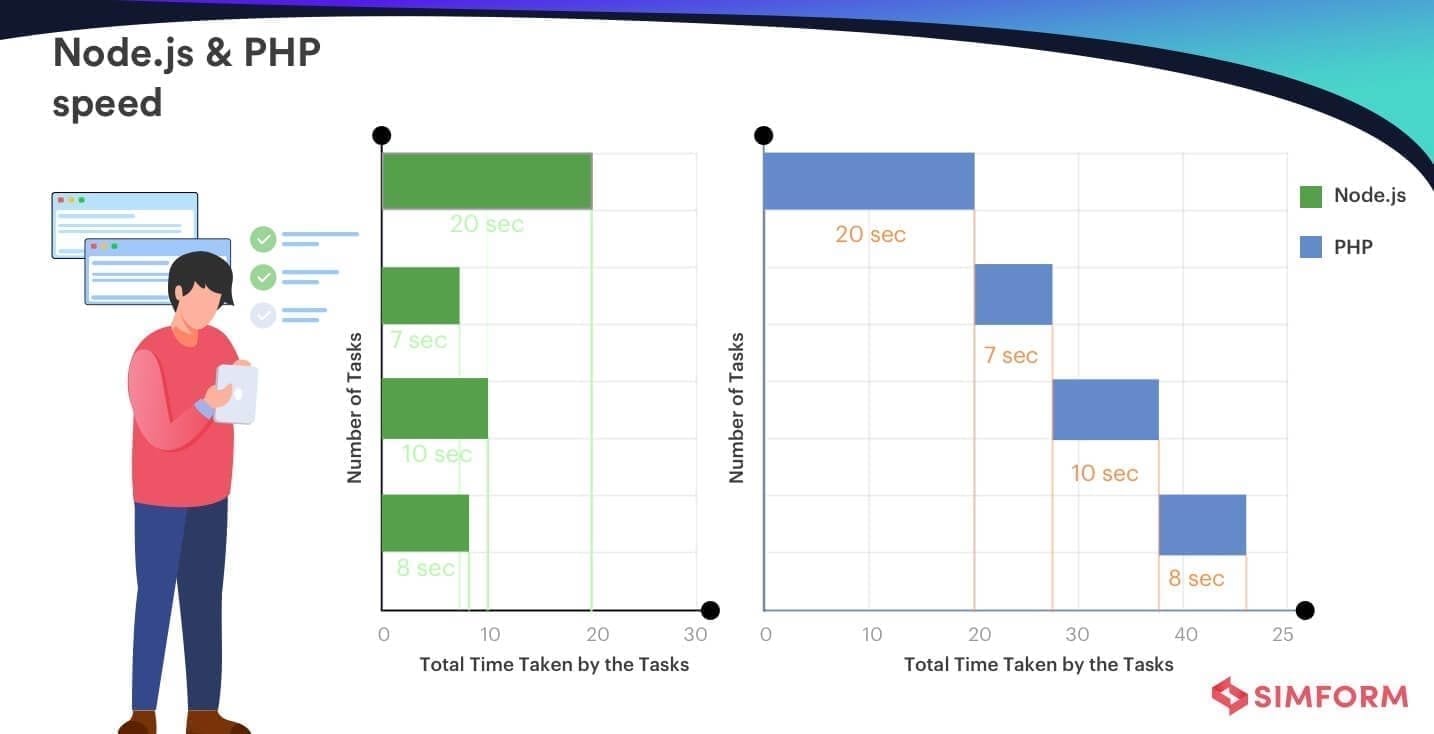
**Simform Recommends**

Both PHP and Node.js derive their value from the use-case you have in place. If your project has a static website requiring few client requests, and your target audience uses low-powered devices, PHP would be just what the doctor ordered.

But if your use-case includes dynamic websites frequently sending server-requests and has a dynamic UI, you want to go with Node.js.

**#3 Speed**

This metric refers to the execution speed inherent to both technologies. High speed of execution means a quicker development process and more cost-effective projects.

**Node.js**

Node.js is among very few asynchronous development environments. This, being a major differentiator, gives Node.js an edge over other languages. Being asynchronous means that it need not wait for a module to be successfully executed before loading the next one. Such execution drastically reduces the downtime for the web app and makes for a seamless user experience.

**PHP**

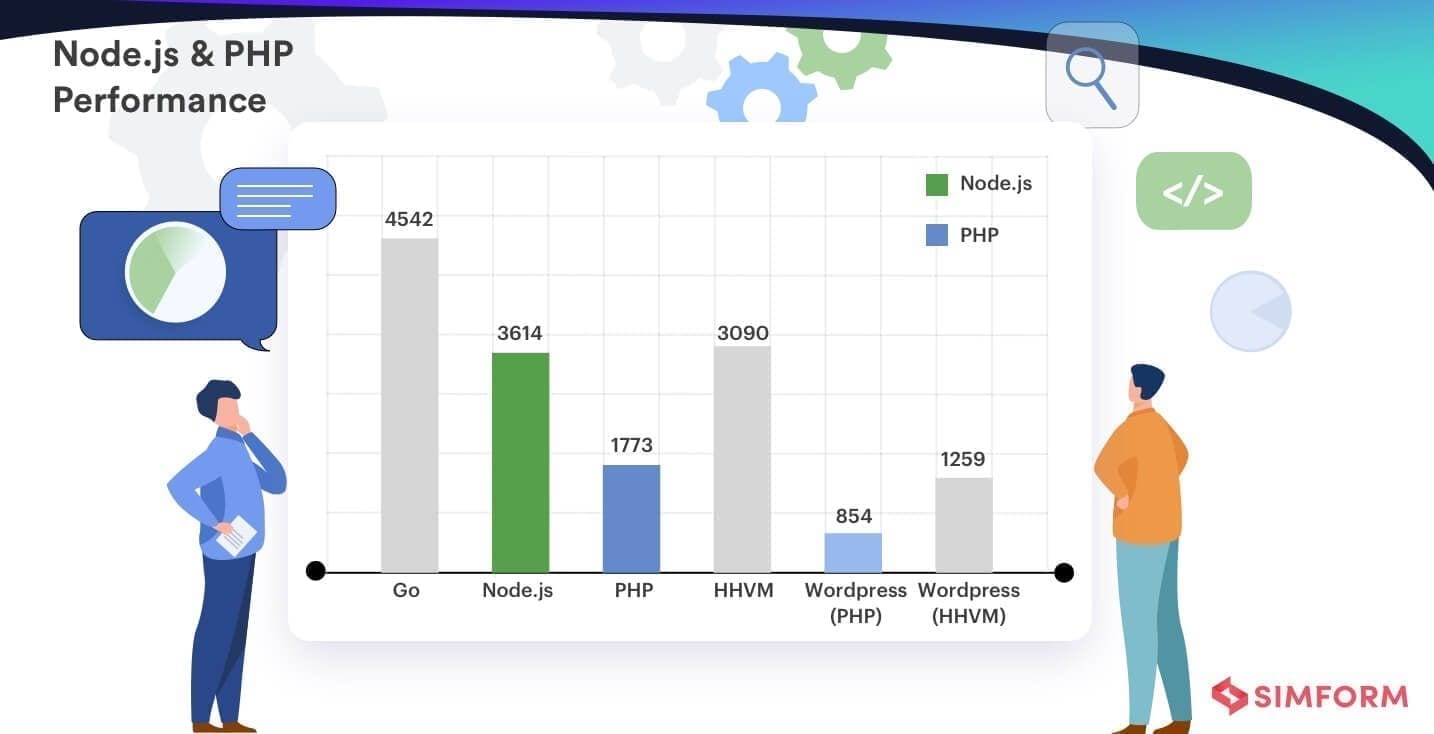
Like most languages and technologies from the pre-2K era, PHP operates on synchronicity. That means every module and function is executed in the code-specified order. If one function or module is not executed, the consequential ones will not start until it’s completed.

**Simform Recommends**

Thanks to its asynchronous execution, Node.js is a clear winner owing to its speedy performance.

**#4 Performance**

Performance is the metric of how the code is written in PHP or Node.js and its performance on KPIs like page-loads and seamlessness. A high-performing technology means better results for your product’s performance using KPIs that have a deep impact on the UX.

**Node.js**

The asynchronous nature of Node.js with JavaScript V8 Engine gives Node.js a remarkable execution speed along with quick startup time. Since Node.js is event-driven, it doesn’t block requests to finish existing ones.

This order-independent execution is generally referred to as ‘concurrency’. Concurrent module execution means that multiple modules are executed at a given time but not simultaneously.

Simply put, their beginning and completion times may differ even if they’re using the same resource and executed in the same environment.

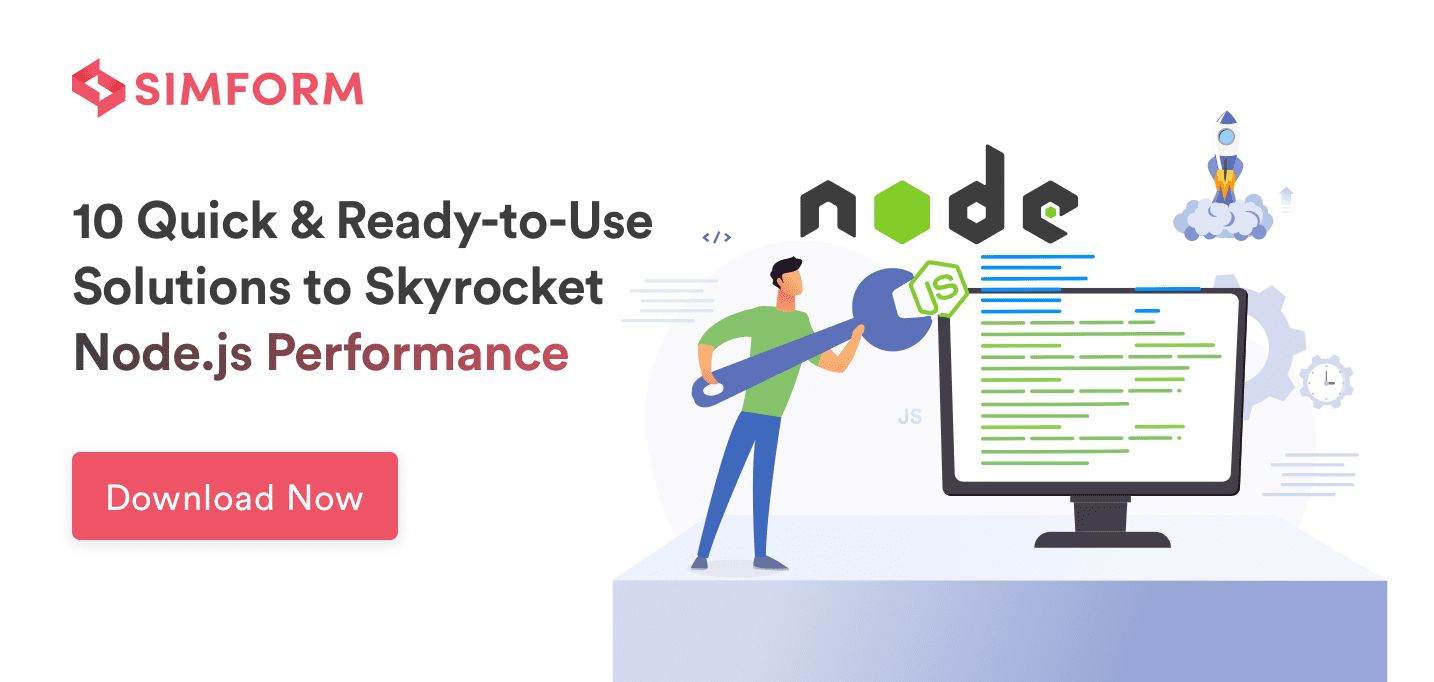
**PHP**

PHP was developed in a much earlier timeframe, and it shows in the way it performs on page-load time efficiency. It blocks a process till it’s been entirely calculated – which translates into slow loading with no scope for concurrency.

But If you still want to continue with PHP, you can pair it with HHVM Virtual Machine used to execute PHP web apps. It can boost the performance of your PHP web app by almost 75%. Even so, Node.js is still a faster alternative.

**Simform Recommends**

Both PHP and Node.js use Just-In-Time Compilation. However, if performance is high on your list of priority features, Node.js is hands down the tech that delivers.

**[](https://www.simform.com/node-js-performance-checklist/?itm_source=NodeJs_vs_PHP_Blog&itm_medium=Inline_Banner&itm_campaign=NodeJs_vs_PHP_Blog-NodeJs_Performance_Checklist_eBook)**

**#5 Frameworks**

Frameworks help you focus on the fine details of your project as the most commonly used code is already provided with functions, libraries, and APIs. The more extensive the framework, the lesser the amount of redundant code you need to write.

**Node.js**

With frameworks like Meteor, Derby, Express, and Sails, Node.js has seen a massive expansion of its libraries in a relatively shorter period. While these frameworks can boost productivity reducing development time and resource consumption, PHP’s frameworks outnumber those of Node.js by a huge margin.

**PHP**

Around since 1995, PHP has a very rich library of frameworks with niche markets of their own. In fact, some development agencies focus specifically on forms of PHP frameworks like Laravel, CodeIgniter, CakePHP, and Phalcon.

**Simform Recommends**

While the list of PHP frameworks is long, there goes more nuance in deciding which technology performs better. Many PHP frameworks were engineered to import features like non-blocking which are inherent to Node.js. Besides, several PHP frameworks have one common issue – mixed content and code.

Since Node.js was developed in a time when demarcating code and content between modules became a norm, consistency is maintained across most of its frameworks keeping the code organized and interpretable.

Also, its most popular frameworks are developed on Javascript, which makes it easier for developers to maintain their backend and frontend logic of developing web applications.

**Conclusively, Node.js will serve as a better choice if your project requires extensive use of frameworks.**

**#6 Database**

Here, we’re examining how conveniently the technology is able to connect with the database along with the type of the database – relational or traditional. Conventional databases traditionally store data in a navigational or hierarchical format.

Relational databases store data in the form of tables; that way, it’s super convenient to understand the relationships and dependencies between different data points.

There’s also the remarkable development of NoSQL (Not Only SQL) databases that allow data storage in formats like documents, key-pairs, and even graphs.

NoSQL databases can store almost all forms of data structures including unstructured, semi-structured, and structured databases. With this, NoSQL databases are very developer friendly.

**Node.js**

Since Node.js has grown with the advent of NoSQL databases, it’s well-synchronized with all forms of databases. With Javascript Object Notation- a unique format of data used while interacting with databases- Node.js can work swimmingly with NoSQL databases like MongoDB and CouchDB.

It also supports graph databases and has substantial libraries to access SQL databases.

**PHP**

Being the older technoloy, PHP is designed to work with relational and conventional databases like MySQL and MariaDB. While it’s possible to import libraries to work with NoSQL databases, the process is tedious and eats up a large chunk of processing time.

**Simform Recommends**

**If you plan to have a scalable web application that will frequently pull data from conventional, relational, or NoSQL databases, Node.js is what you’re looking for.**

**#7 Node.js vs PHP – Request handling**

This is the measure of how swiftly a technology processes client-side requests. Executing the request accurately is no doubt crucial, but it also needs to be done using minimum resources and time. Error handling would further help us understand which technology best optimizes the UX.

**Node.js**

Thanks to the asynchronous processing of multiple requests, the system isn’t waiting on one process to end before the next one begins. CPU and RAM-times are also not wasted. The only concern here is that if an unresolved error of one request isn’t addressed in due time, it could interfere with other requests creating a pan-system error.

**PHP**

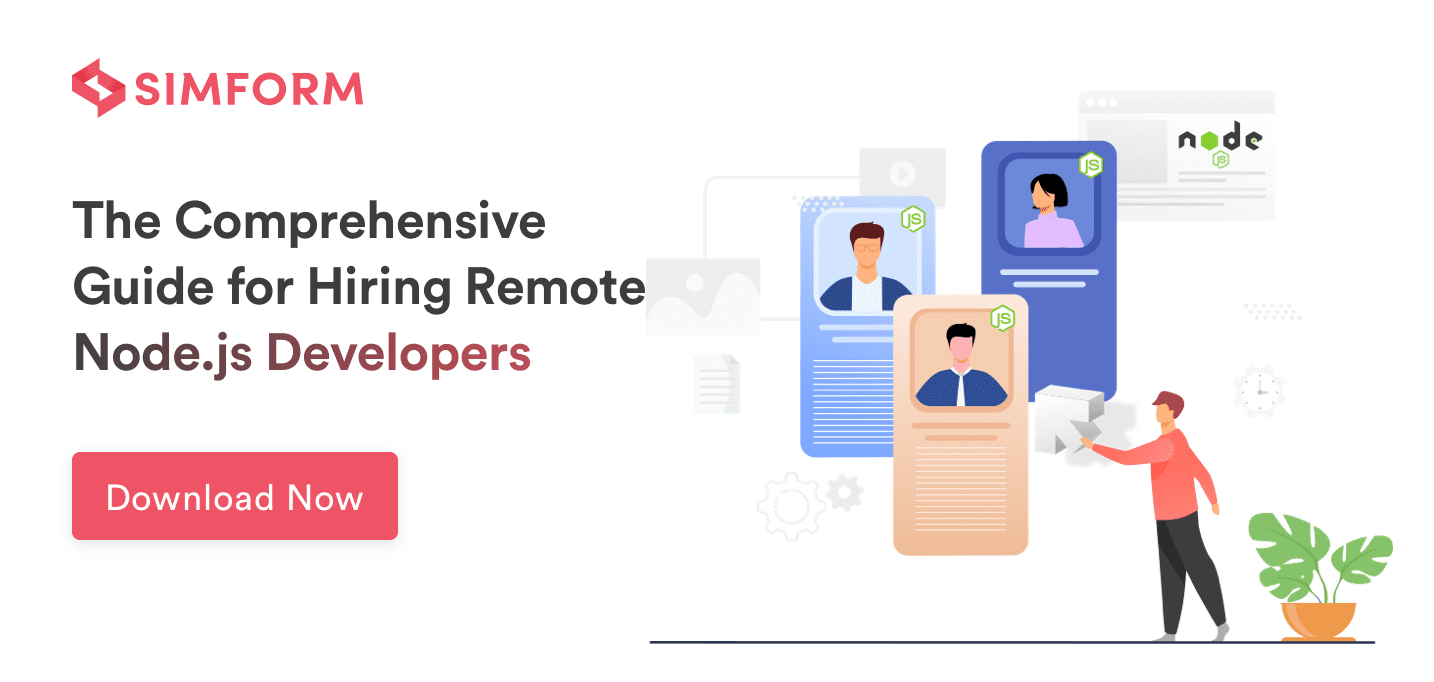
PHP handles one request at a time. Naturally, it consumes more resources on the CPU and the RAM. Since it’s on the opposite end of the spectrum for this parameter, every request is executed in a silo.

This prevents cross-contamination of request errors, but it also creates a lag in the system with its one-request-at-a-time approach.

**Simform Recommends**

Node.js has an edge in this comparison. It executes more processes using fewer resources. Plus, with effective error-handling mechanisms, you can safeguard the system against slogging that results from a faulty execution.

It’s possible to use libraries in PHP to get the asynchronous functionalities. However, since this is not a native feature, its efficacy just doesn’t compare to that of Node.js.

[](https://www.simform.com/the-comprehensive-guide-for-hiring-remote-node-js-developers/?itm_source=NodeJs_vs_PHP_Blog&itm_medium=Inline_Banner&itm_campaign=NodeJs_vs_PHP_Blog-Comprehensive_Guide_Hiring_Remote_NodeJs_Developers_eBook)

**#8 Community**

The strength and expertise of a community decide what kind of updates come to various the frameworks, libraries, and projects being launched under each technology. It’s a given that your team won’t be able to write code for each feature and module from scratch.

Thus, using popular and tested libraries brings down development time and increases productivity. While a larger community might have more projects, the quality of an individual project/library/framework is what helps create value for your team.

**Node.js**

Most Node.js projects are indexed on npmjs.com registry. Since Node.js is relatively new, its community is smaller compared to PHP. Yet the projects developed by the community lean more toward the development needs of today. In fact, many projects focus on adding unique functionalities to Node.js instead of serving as a library to import features from other languages.

**PHP**

PHP has been around longer than most technologies, so it has a bigger community and a larger number of projects under its belt. But one key issue here is that the newer projects developed by the community just appear uninteresting when compared to Node.js.

Initially, the projects they developed were received well for adding to the platform features that already existed in other languages. But now that many similar projects do exist, some PHP loyalists feel the need for more interesting projects to come their way.

**Simform Recommends**

Node.js projects are uploaded on the npmjs.com package registry. No doubt the number of projects is smaller compared to PHP, but its newer projects are way more relevant. Given the existing projects in PHP and the ones up and coming for Node.js, it’s a tie.

**#9 Modules**

Modules generally act as sub-programs within a program. They carry a certain set of functionalities and are often interchangeable – giving the program its ‘modularity’.

**Node.js**

Some PHP loyalists often complain that the Node.js projects are not as stable as their PHP counterparts. As the community is still growing, quality-control systems are barely in place.

As a measure of control, the Node.js package registry has introduced the [npm-audit](https://www.npmjs.com/package/npm-audit" \t "_blank) initiative. With the help of this, you can now check each package for malicious code.

**PHP**

PHP has had an edge in this area for years now. Recently, however, some PHP loyalists have expressed displeasure toward receiving lackluster projects. This has prompted them to migrate to other platforms especially because the inherent features of PHP can be imported there as well.

**Simform Recommends**

For now, PHP has a richer library of modules. However, Node.js is making great strides in this space. So PHP would still be recommended with Node.js falling in a close second.

**#10 Ecosystem**

A platform’s community is responsible for its ecosystem. It shows how many open-source libraries, APIs, modules, frameworks, and projects are being pushed out by the community at any given point in time.

**Node.js**

The Node.js ecosystem has a wide range of libraries and frameworks, but they lag way behind PHP’s numbers. Although what Node.js lacks in quantity, it makes up for with its variety of projects. Since it’s used for both server-side and backend programming, the types of projects available range freely fitting into a larger number of use cases.

**PHP**

WordPress has been a major contribution to the PHP ecosystem. The company is directly responsible for running a large percentage of total websites on the internet, and nothing else shows PHP’s godlike reach quite like this.

Moreover, the PHP community has developed a ton of training material and supporting technology over the years to bring new developers onboard.

**Simform Recommends**

PHP has a larger ecosystem while Node.js has a richer one with a greater variety for projects, frameworks, and modules. For that reason, Node.js wins this round as well.

**#11 Node.js vs PHP – Hosting**

It measures the range of compatibility that a technology platform shares with different hosting service providers. The number of alternatives available directly affects hosting expenses. The speed and security of the web app must also be considered while selecting a host.

**Node.js**

Node.js has a range of alternatives that help it perform competitively. Joyent, the company that maintains Node.js, provides a SmartOS system. This is great for debugging, performance enhancement, and ease of deployment. With Heroku and Nodejitsu, it is easy to use Node.js in a Platform-as-a-Service setup.

**PHP**

PHP is, directly and indirectly, powering almost 79% of the internet. Due to its majestic reach, it’s designed for compatibility with all major hosting service providers. With its LAMP stack, it meets the needs of many servers. However, LAMP isn’t considered the most secure technology stack for hosting.

**Simform Recommends**

PHP clearly has wider compatibility with hosting service providers, but so does Node.js. If security is not a major concern, you could freely choose either platform for your web-app development project. There’s no difference that’d prove a deal-breaker.

**When to Choose Node.js Over PHP?**

If you’re looking for the following features, Node.js can provide optimal value to your web-app development:

1. **Development Efficiency-**If you are creating a dynamic single-page application, you might be using MongoDB, ExpressJS, or AngularJS. Node.js is generally used in conjunction with this stack for ease of development and optimal performance.
2. **High Speed and Consistent Callback from Servers-**Web applications developed using Node.js tend to perform better while consistently sending requests to the server. Its asynchronous architecture allows non-blocking execution which is great for any project that needs speed.
3. **Real-Time Data-**Since the data-transmission speed from the server to the client-side application is optimal, Node.js is perfect for web apps that need real-time data functionality.

[**Companies**](https://www.simform.com/companies-using-nodejs/) like Netflix, LinkedIn, and Medium have been using Node.js for a long time. You can easily see the common thread of real-time data, consistent and fast connection with the server, and ease of development among the three.

**Check how we build SentMap - a real-time sentiment-based market analytics tool using Node.js**

[**READ DETAILED CASE-STUDY**](https://www.simform.com/case-studies/sentmap/?itm_source=NodeJs_vs_PHP_Blog&itm_medium=Inline_Cta&itm_campaign=NodeJs_vs_PHP_Blog-Sentmap_Case_Study)

**When to Choose PHP Over Node.js?**

PHP would be the go-to technology for your stack if you covet the following properties for your web app:

1. **Centralized Server and No Scaling Needs-**If your plan is to allocate a specific centralized server to your web app that’s not scaled across a wider range of servers, PHP is the perfect addition to your tech stack. Also, you can use it alongside Linux, Apache, and MySQL.
2. **Portability-**While PHP definitely puts a tab on the number of servers you can effectively connect with, it also provides you great portability among servers. That’s why, you can practically port your web app to any server that has Apache, IIS, and other database-support functionalities. And if you use it with content management systems like WordPress, Jumla, or Drupal, you can practically have your website up and running in no time.

Wikipedia, MailChimp, and Tumblr are some of the most popular users of PHP. They show what all you can achieve with a centralized server and effective use of PHP.

**Limitations of Node.js and PHP**

It’s true that both technologies can do wonders for your web app, but they aren’t free of flaws either. Before you make a choice, you should be aware of their limitations:

**Node.js**

1. **Ineffective With CPU-Heavy Apps-**Despite the speed it provides, Node.js often struggles with web apps with heavy graphics and high CPU usage. This is because it uses a single thread to handle concurrent requests. In case the execution of one request is halted, it can potentially affect the functioning of other requests as well.
2. **Lack of Quality Control in Frameworks & Libraries-**Many libraries and frameworks that give Node.js its extensibility have been developed by its community and third-party developers. While the documentation and support for many of them are great, the same isn’t true for every library and framework.

**PHP**

1. **Poor Error Handling**– Even with the inbuilt error-handling tools, most professional developers maintain that PHP doesn’t have a great ability to highlight, filter, and manage errors. It’s not a deal-breaker for smaller projects where codebase can be scanned for errors. However, handling errors can never be a standalone manual task when a web application is scaled. PHP simply performs poorly in this regard.
2. **Synchronous Execution**: In stark contrast to Node.js, PHP executes each module sequentially. While this ensures that the errors in one code do not cross over to other codes, the process is sluggish and often causes a crash. This further piles on to the scalability issues that plague PHP.

**Conclusion**

There is no standardized process to identify if Node.js is better than PHP or vice versa. However, instead of trying to evaluate which technology outshines the other, it’d serve you well to first map out the requirements of your project and then select the technology that best fits it.

In addition to technology relevance, also pay special attention to the human expertise you possess. If your team is more proficient at one technology over the other, it’s advisable to inquire if the expertise can compensate for the potential drawbacks of the technology.