All posts > Web Development

Node.js vs Python: which technology would be a better choice to develop your app and why?

03 July, 2019 | 21 min read

Claudia Slowik

The server side of the web application may not be visible to end users but it is the engine that makes your app run. The programming language that you choose to build it with determines many crucial flows and procedures in project development – that's why it is so important to make the right choice! However, with many technologies out there, the choice is not simple.

In this article, we will compare two popular solutions for web app development: Node.js (with Express as the main framework) vs Python (Django). Both loved for their simplicity, speed of development, and easy code maintenance, they are often compared and seem to be good alternatives to Java, PHP or Ruby on Rails. But which one is better? Let's see! In the following article, you will find information about:

1. **Comparison of Node.js and Python**

    1. **Performance/Speed**
    2. **Scalability**
    3. **Error handling**

2. **Advantages of Node.js and Python**

    1. **Node.js advantages**
    2. **Python advantages**

3. **Node vs Python: use cases**
4. **When to use Python and when Node.js?**

Web Development

What's the difference between quality assurance and testing in the software industry?

Dominik Wilk

Web Development

How to choose the best Node JS development company

Claudia Slowik

Comparison of Node.js and Python

First of all, there is one important thing: Python is a programming language and Node.js isn't (it's a runtime environment for JavaScript). So can we really compare Python and Node.js?

Well, both Python and Node.js are **solutions for server-side application development** and as such, they have a common thread and may be compared. Let's see how they perform in three aspects of software development: performance, scalability, and error handling.

Python vs Node.js – performance and speed

The speed of your application response (your app's performance) depends directly on how fast your code is executed. The faster it is executed, the better the app's performance gets. As Node.js is based on fast and powerful Chrome's V8 engine, **Node.js is faster than Python**, **and generally one of the fastest server-side solutions around**. Sure, there are technologies that are faster… in certain situations. If you want to check how Node.js, Python, Java or C++ may perform under different conditions, check **The Benchmarks Game (debian.net)**. And what's the benefit of Node's high performance? It makes Node.js a good choice for all real-time applications, such as **collaboration tools** or notification systems, e.g. stock notification system for a stock trader.

Also, Node.js operates on event-driven architecture which is based on asynchronous calls and stream modules that don't need to be available all at once. Simply put, it allows Node.js servers to process more concurrent requests than most conventional servers, as they don't have to "fit" in the memory limits.

So what about Python? Well... the performance is not really its strong point, especially when you use Django (Python's most popular framework), and it may require more hardware resources to work at the desired speed. To be fair: there are solutions that can improve Python app's performance, e.g. using backend serverless architecture without any framework but, in general, performance is not something that you would directly correlate with Python. However, according to Google's study

> the CPU time is rarely the limiting factor; the expressibility of the language means that most programs are small and spend most of their time in I/O and native run-time code.

Put simply: in many cases, you won't need that speed. If you're not building a real-time app, your customers may not experience any significant slowdown of the application. But if you are, Python – at least with Django – may not be the best choice.

Scalability

The scalability of your application is what we call **its ability to serve the increasing number of requests with no performance decrease**. It is important when you expect your product to grow and you want to be prepared for:

- a bigger number of users
- a bigger amount of data to be processed (content-heavy applications)
- a bigger number of features (meaning the bigger number of requests for the app to handle).

Node.js and Python deal with scalability in two different ways: Node with its architecture, and Python with its tools.

Node.js scalability

In the traditional, synchronous, approach to input/output operations, the app starts the access and then waits for it to complete, blocking the progress of a program while the communication is in progress. The bigger the application gets, the more data it has to process, making the app slower and slower. And how does it work with Node.js?

There are three main reasons that stand behind Node's scalability:

1. It can be easily broken down into microservices.
2. It has an event-based model.
3. It has a non-blocking I/O.

In an event-based architecture with asynchronous I/O, operations are completed outside the thread and, therefore, they are not blocking it. Although the **development of complex applications** with a lot of concurrent processes may require some expertise, in general, Node.js is considered to be a good choice when scalability is your focus.

Python scalability

Even though the scalability is achieved in a different way than in Node.js, the solution is rather satisfactory. And by “rather satisfactory” I mean good enough for such big services as Youtube, Pinterest, Reddit, Dropbox, or Quora. Not so bad, right?

When we choose Python (Django), we rely on tools such as Memcached, which uses caches to avoid recomputing data or accessing a slow database, or NGINX. When equipped with such tools, Python applications are able to handle data migrations even with the growing amount of data. Also, it’s worth mentioning that even though Python does not support asynchronous **programming** by default, it does support coroutines which can suspend their execution before reaching return, and indirectly pass control to another coroutine for some time.

Assuring scalability of any application is quite a challenge and depends more on good practices implemented from the beginning of the development process than

on the technology. In this category, there is no true winner – at least when we think about Node.js and Python.

Error Handling

Both Node.js and Python are said to deal well with catching errors that occur during code execution. From the developer’s perspective, it may be worth mentioning that Node.js prints top-bottom and Python prints bottom-top. It doesn’t really matter, however, in terms of business as it affects neither the speed of development nor the app’s quality.

Again, we’ve got a draw.

Advantages of Node.js and Python

Advantages of Node.js

JS everywhere

As the so-called Atwood's Law states: "**Any application that can be written in JavaScript will eventually be written in JavaScript"**. Using JS from back- to front-end is an optimization factor that may shorten time-to-market and make future

maintenance easier. Also, the same language on the client side and the server side makes Node.js applications faster than the apps that utilize different languages.

High scalability

It's not a coincidence that companies such as LinkedIn, Netflix, or Twitter are switching to Node.js. As it was already described **above**, there are three main factors that contribute to the scalability of Node.js: it can be easily broken down into microservices, it has an event-based model, and it has a non-blocking I/O that helps it to make the most of the CPU and computer memory. Scalability makes Node a good choice for applications that are expected to quickly grow their number of users.

High performance

Node.js is based on fast and powerful Chrome's V8 engine and it is one of the fastest server-side solutions around. Also, thanks to its event-driven architecture, Node.js servers are able to process more concurrent calls than other servers. This makes Node.js a perfect choice for all real-time applications, such as chats or collaboration tools, and all tools that require high performance.

Fast development process

The stories of companies switching to Node.js from Java and other technologies are well-known, have been already described, and they are actually the best proof of the value that Node.js brings to the development process. When PayPal decided to switch to Node.js, they built the first Node app in parallel with an equivalent Java application. The Node.js application was:

- built almost twice as fast with fewer people,
- written in 33% fewer lines of code,
- constructed with 40% fewer files.

Shorter development time translates directly into shorter time-to-market, which often determines the success of your product. After all, it doesn't matter how many good ideas you come up with if your competitors deliver similar solutions before you do.

Awesome package manager

Originally, Node.js was intended as a server environment for applications, but developers started using it to create modules to aid them in local task automation. Since then, a whole new ecosystem of Node-based tools has evolved. Node package manager (NPM) installs the packages you want to use and provides a useful interface to work with them.

Huge user community

The community behind Node.js is large and getting bigger each day. The bigger the community gets, the easier it is to get support. Also, the technology itself is growing really fast: a new version is released every 6 months, there is a great choice of open-source scripts, libraries, and applications supporting Node.js.

Handling concurrent requests and operations

Thanks to Node's asynchronous calls and non-blocking I/O, multiple users can be editing the same file, moving tasks between the boards, commenting, adding media files – all at the same time. The ability to handle the requests simultaneously makes

Node a perfect environment for real-time web apps such as chats, games or collaboration tools.

Is Node.js safe?

Read about 13 best Node.js safety practices

Read now

Advantages of Python

Speedy development process

Even though it may be difficult to compare the speed of the development process between Python and Node.js – both are recognized for their high speed of development – it is estimated that developing a Python application is about 5 to 10 times faster than developing the same application with Java (or even more if we compare it to C++).

Thanks to Python's rich standard library, there is no need of searching for additional modules and packages for your application. Instead, it is possible to quickly start building a prototype that can be developed with other features later.

Rich standard library

Many useful features come natively in Python. This means you don't need many additional libraries to build your application. And if its standard library lacks some functionality that you need, you have PIP which is an ecosystem of modules to choose from (just like NPM in Node.js!).

Very scalable

As mentioned before, when we choose Python (Django), we rely on its coroutines which can suspend execution of the operation before reaching return, and on the tools such as Memcached or NGINX. Equipped with them, Python applications are able to handle data migrations even with the growing amount of data.

If scalability is one of your concerns, Python will not disappoint you. If it’s enough for Youtube, Pinterest, Reddit, and Dropbox, there is a chance that it will serve you well – at least in the beginning 😉

Huge user community

As stated by Python Software Foundation: “Great software is supported by great people, and Python is no exception”. A vivid community behind it indicates good support but also fast growth of this technology.

It’s the leading programming language of data science

Python comes with a huge amount of inbuilt libraries dedicated to Artificial Intelligence and Machine Learning, such as Tensorflow or scikit-learn. This advantage may not apply to most web application scenarios but in terms of AI development or any projects related to Machine Learning, Python is undoubtedly the best choice.

Node vs Python: use cases

Node.js use cases

It’s not a coincidence that companies such as LinkedIn, Netflix, or Twitter are switching to Node.js. It makes the process of app development faster (PayPal built their Node.js application almost twice as fast as an equivalent Java application with fewer people), it's lightweight, efficient, and performant – with the use of Node, the startup time of Netflix was reduced by 70%!

So what types of apps will benefit most from using Node.js? On the top of the list, we should probably put real-time applications (RTAs) such as chats, games or collaboration tools, and streaming applications. Thanks to Node's architecture that works well with the WebSocket protocol, you can create real-time messaging with faster data transfer and lower latency. Then, we've got Single Page Applications (SPAs) and other web applications that need to be fast and scalable. Node.js is also said to be a good choice for the Internet of Things (IoT) solutions as it allows the processing of multiple concurrent requests and a vast number of devices on the network.

Python use cases

As a high-level general-purpose programming language, Python can be applied to many different use cases.

One use case where Python excels at is the whole area of Artificial Intelligence development services. There are a few reasons for that. Firstly, it has great libraries for that: scikit-learn for handling basic ML algorithms, Tensorflow for working with deep learning by setting up, training, and utilizing artificial neural networks with massive datasets, or PyBrain for neural networks, unsupervised and reinforcement learning – just to mention a few of them. Secondly, it's simple (unlike the whole Artificial Intelligence, which rather is a complicated field!). Machine Learning processes rely on extremely complex algorithms and multi-stage workflows.

Python's simplicity allows developers to release their mental resources so that they can concentrate on solving the problems and achieving project goals. Last but not least, it has a relatively low entry barrier which allows more data scientists to quickly pick up Python and start using it for AI development.

Apart from AI development, Python comes useful when you are building audio/video applications (again, thanks to dedicated libraries such as Librosa or PyAudioAnalysis), Progressive Web Applications, or system administration applications.

When to use Python and when Node.js?

Just like when we compared Node.js with Java, PHP or with Ruby on Rails, it's almost impossible to say that one technology is "better" than the other. As Python has a lot of similarities with Node.js – they are both recognized as fast, scalable technologies with good communities behind them, and they are both said to shorten the development time of software projects – the comparison is even more difficult here.

There are some projects, however, that would benefit more from using Python (Django) or from using Node.js. For Python, it's the whole field of AI development, where Python is the most popular programming language, but also some web development projects – especially audio/video applications, Progressive Web Applications, or system administration applications. For Node.js, these are mainly real-time applications, streaming applications, Single Page Applications, and other web applications. Note, however, scalability and other individual strengths of particular solutions matter later in the product development lifecycle. Being able to start the project with the right team has a much bigger impact on your success.

One thing to remember is that the choice of the perfect technology for your project should always start with the assessment of your challenges and needs – may it be the performance of the app or just… short time-to-market. Once you realize what they are, it is much easier to tell which technology will be more beneficial for your product.

Netflix, Uber, PayPal are built with Node.JS

Learn more about the types of applications you can build with Node.JS

Learn more

Do you like our work?

Do you like our work?
Let’s talk about your project!

Related posts

The Node.js and JS foundations want to merge

The two major open-source JavaScript foundations, Node.js and JavaScript, have recently issued an announcement on their merging plans....

read more

Docker containers with root privileges

A significant part of the IT world relies on Docker containers. They are easy to use & portable. But are they always good? Let's see how to use them s...

read more

How To Increase Gym Revenue In COVID-19 era?

Are gyms going to extinct? Learn 5 proven ways to increase gym revenue with a custom app in the Covid-19 era....

read more

Services

Web app development services

AI development services

Product design services

Node.js development services

Fitness app development

Remote software development

Blog

Web development

Artificial Intelligence

Product design

Delivery process

Knowladge base

A start-smart guide to successful AI adoption

Dos and don’ts of building online fitness applications

Tech Talks: Digital Fitness

AI Talks

Articles

Data science, machine learning, and AI in fitness – now and next

AI in business: What are the benefits of artificial intelligence?

12 challenges of AI adoption

What’s the difference between quality assurance and testing in the software industry?

Estimate your project ⟶

Neoteric sp. z o.o. | VAT-ID: PL 957-107-23-74 | Marynarki Polskiej 163, 80-868 Gdańsk, Poland

Find us on

Awards:

Best AI and ML Company in 2021 according to

Partnerships: