International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-10, Issue-3, May 2022
https://doi.org/10.55524/ijircst.2022.10.3.13
Article ID Article ID IRP1267, Pages 67-70
www.ijircst.org

Benefits and Challenges of Using NodeJS

Bonjar Basumatary[1], and Nishant Agnihotri[2]

[1] Student, Department of Computer Science & Engineering, Lovely Professional University, Phagwara, Punjab, India
[2] Assistant Professor, Department of Computer Science & Engineering Lovely Professional University, Phagwara, Punjab, India

Correspondence should be addressed to Bonjar Basumatary; bonjar02@gmail.com

ABSTRACT- Node.js, the JavaScript runtime environment is very popular among Full Stack Developers for building Web applications. It gives the developer to write server-side code in JavaScript and able to manage both client and server side. Node.js ensures outstanding performance, it applies event-driven, non-blocking input output and asynchronous paradigm. The study is based on benefits and challenges of using Node.js. The results of study are from comparing different web development programming languages with JavaScript which is used in Node.js. And in the end some suggestions on how to improve those areas and make it more feasible.

KEYWORDS- Node.js, JavaScript, Asynchronous Function, Multi-Thread, Single-Thread, Front-end, Back-end, Full-Stack.

I. INTRODUCTION

Web application is application software that runs on a web server and are easily reachable to clients. The Web development industry is divided into two major kinds of developer that is Frontend developers and Backend Developers [2].

Frontend developers build front-end portion of websites and web applications, that is the part in that user see and interact with. To build a Frontend the developers require to have the knowledge of HTML (Hypertext Markup Language), CSS (Cascading Style Sheet) and JavaScript to add more dynamic effects to client side for interaction [2]. Backend developers are responsible for building the structure and logic behind functioning of the software application. Backend developers create, maintain, test and debug the entire backend, this includes the main application of the logic, databases, free flow of data. They play a crucial role in web development and make sure that there is free flow of data and manipulate the data based on user's request. The Backend developers must require the knowledge of languages like Java, .NET (Network Enabled Technologies) and Python [2]. They must also require some knowledge of databases like MySQL (My Structured Query Language) and Oracle, to maintain the free flowing of data [2].

Full Stack developers are those who works with both Backend or server side as well as the Frontend or client side of the application. Full Stack developers must require to have knowledge of both Frontend developers and Backend developers [2]. So, it is clear that to become a Full Stack developer one needs to have skills of different languages for client side like HTML, CSS, JavaScript, for server side like Python, PHP (Hypertext Pre-processor), Java and for databases like MySQL and Oracle [3].

Node.js have overcome the complication of learning multiple languages to become a Full Stack developer. Once you have mastered JavaScript which is used in Frontend, by just learning additional server-side functions is enough to become a Backend developer [2]. With consistently increasing number of Full-Stack Developer Node.js is regarded as viable solution for scalability. Many of the titans like Netflix, PayPal, Uber and Walmart have considered Node.js as optimal source for Web Development [1].

Node.js is a cross-platform runtime environment, built on V8 which is an open-source form google as high-performance JavaScript engine [1]. It converts JavaScript functions into machine codes with high performance and pace. Node.js provide outstanding performance in building high performance and scalable network applications, it uses event-driven, non-blocking input output and asynchronous paradigm [1]. In current market the requirements for real time processing applications are in high demand, and Node.js provides exceptional features for fast multi-user and real time performance. As Node.js is event-driven and not thread based, it is capable of fetching and sending millions of responses concurrently, while using event loop with single thread rather than splitting into multiple threads [1].

II. BENEFITS OF USING NODE.JS

A. *Fast Server Connections*

As we know now that Node.js applies non-blocking input output JavaScript applications by using event loop that can handle multiple requests at the same time. While other languages are not able to handle multiple tasks until the server fulfils the request sent form initial user form client-side as a result they will require multiple thread for execution. By using asynchronous in JavaScript usual multithreaded operations [1]. The reason behind scalability of Node.js is that it uses very few threads to handle multiple client's request and focuses more on system's computation capabilities working on clients rather than using space and time for handling threads. Since, other languages use multi-threading to handle multi-tasking, the cost of creating a thread will create more overhead than distributing the task [3]. A large block of memory has to be allocated and initialized for the thread stack.

Comparing Node.js with PHP, Node.js is faster in execution and better option in multi-user applications because Node.js applies asynchronous, non-blocking, event-driven [4]. Whereas PHP does not use asynchronous method but uses synchronous programming model, which means that Node.js is better than PHP and other language in speeding up development.

B. Single Programming Language

Node.js overcome the problem of learning multiple languages to become a Full Stack Developer. Coding in Node.js is relatively easy to learn, since it uses JavaScript for Frontend and Backend side and it is enough to start. Advantage of using Node.js over others is that to become a Full Stack developer it requires the knowledge of only single language that is JavaScript for both client-side and server-side programming [1]. Node.js stores data like JavaScript Object into database called JSON (JavaScript Object Notation) [3]. While in Web development in PHP, you need to know Html, CCS, JavaScript for Front-end and PHP, JQUERY for Back-end programming [4], PHP is a programming language for Back-end development only. Similarly, Web development in .NET comprises both Frontend and Backend languages. In .NET C# is used for backend and Html, CSS are used as Frontend development, it is clear that in .NET Web development it requires more than one language for Frontend and Backend development [6]. Lastly, in Web development in Python, for the Frontend it uses Django and Flask [5]. Whereas Python language is only used in Backend development.

C. Most Used Language

JavaScript is known as to be the most used language amongst developers [1]. Since it is mostly preferred by developer implementation and learning of JavaScript for Front-end and Back-end programming will be of less burden for most of the new developers in Node.js [1].

III. CHALLENGES OF USING NODE.JS

A. Linear Input-Output Programming Operations

Since, Node.js follows the model of Asynchronous programming, it becomes a challenge for developers to carry out single thread input output programming operations because developer must structure the programming wisely. Too many asynchronous requests in single thread can make the program actually function slower [1]. When Node.js operates on a CPU (Central Processing Unit) driven task in its event loop, it utilizes all of its available CPU strength to accomplish the asynchronous operation. Which in result leads to slow performance of the overall event loop [1]. And the result is overall slow performance in handling all the requests. Furthermore, if there are nested asynchronous functions, the code can get overly complicated and such unorganized code can result in your program to not work properly. It also becomes much more difficult to debug and target errors when there are nested asynchronous functions [1].
Callback hell is seen in Node.js in asynchronous programming where you have too many nested callbacks [1]. In this case, each callback takes arguments that have been passed from the previous callback functions and ultimately it ends up in poor coding maintenance as well as complexity while writing codes. This kind of callback structure leads to lesser code readability and maintainability.

B. Hard To Learn For Beginners

The lousy and unstructured nature of JavaScript is one of its biggest drawbacks which makes your code difficult to understand and update. It is difficult for new learners in Node.js to learn complex JavaScript topics like callback and asynchronous function [1]. The syntax can be messy if you are trying to do things in asynchronous manner and also forced to implement programs asynchronously which is not easy to get with.
In Node.js there are several NPM (Node Package Manager) registries and libraries with lack of reusability and not properly documented [8]. Hence this lack of support is difficult for new developers in Node.js. Only the well experienced can drive projects to success [8]. For comparison in Node.js packages NPM need to be installed again and again for different projects. But in case of Python once you have downloaded a package “pip”, it does not need to download again and again for different projects, it can used in all projects. NPM is slower than pip.
The JavaScript language itself is adapting [9]. For example in 2015 JavaScript shown great progress by introducing “let” and “const” keyword [9]. Before that it had only global and function scope, but in ES6 JavaScript can now have block scope. In ES6 many features like Arrow functions, Promises and spread operator were also introduced. Similarly, in 2016 ES& was released with Exponential operator **(**)** was added [9]. In 2017 ES8 was released with Sync/Await, applied only inside async function waits to be rejected or resolve by a promise. Lastly, the latest version in 2018 ES9 was released with features of asynchronous iteration and few more [9]. This means even learning all of the previous methods there is still more developers need to learn.

C. Few Node.js Developer

The demands for Node.js Developers are growing exponentially in industries and there are not many Node.js Developers to quench the demands market necessity [8]. Even among the students there are not as many Node.js Developers as they are the demands in the markets [1]. There is also lack of knowledge among the developers about the concepts of asynchronous paradigm and most of the developers as well as the organizations are not ready to adopt to new technologies over existing technologies like .Net and PHP etc. Overall, there is lack of awareness in the markets.

IV. SUGGESTIONS TO OVERCOME CHALLENGES

As we know that Node.js applies input output model for processing multiple requests and single threaded in nature. In Node.js whenever there is an operation in which it requires to handle multiple requests at same time Node.js would set up blocks for other requests which results into a overall delay performance [1]. To overcome this, it is essential to introduce threads in Node.js to perform heavy JavaScript tasks. With this method heavy tasks can be performed without even distributing the main thread. Although, this concept is available but are still in the experimental phase called Worker Threads [1]. With this

Node.js can become suitable for processing CPU bound task and used for machine learning based calculations.

Although in Node.js NPM registries have vast numbers of packages but they are immature tools which provide poor quality and not quality driven. While the overall main products in Node.js proved to be feasible and well tested, but rest of the NPM registries is are not properly documented [8]. To overcome this, developers might need to restructure its registries in more towards quality driven. Try to assemble all different dependencies into a single large package and well document it, so that new developer does not need to install different dependencies in enumerate manner.

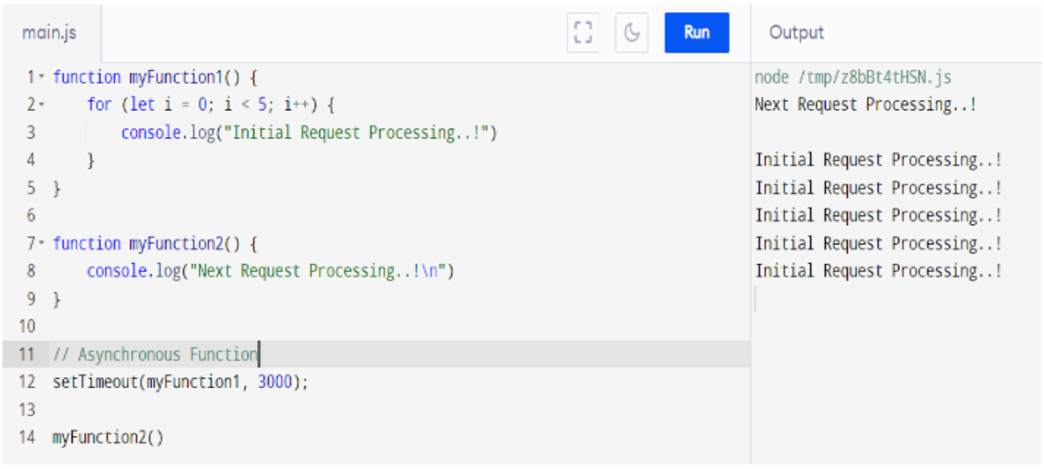
There is less in numbers of developers with experience in JavaScript that work with back-end. Demand is high in industry but typical Node.js developers are not matching up to the demands [8]. So, there is an urgent need of promotion of Node.js amongst the new developers as well as students in colleges and universities. Awareness should be spread about outstanding features of Node.js, make them learn about new paradigm of handling multiple requests with the features of asynchronous, event-driven and non-blocking input output.

V. RESULTS AND DISCUSSION

This study finds that Node.js plays a crucial role in the field of web development, if you want your application to be scalable and high performing you can approach with Node.js. With Node.js a developer can easily become a Full Stack Developer, as no extra burden to learn additional language for client-side and server-side language separately. Since, Node.js applies single-thread mechanism there is less burden on CPU memory for handling threads and more focus can be contributed to client-side multi-tasking.

A. Methods and Materials

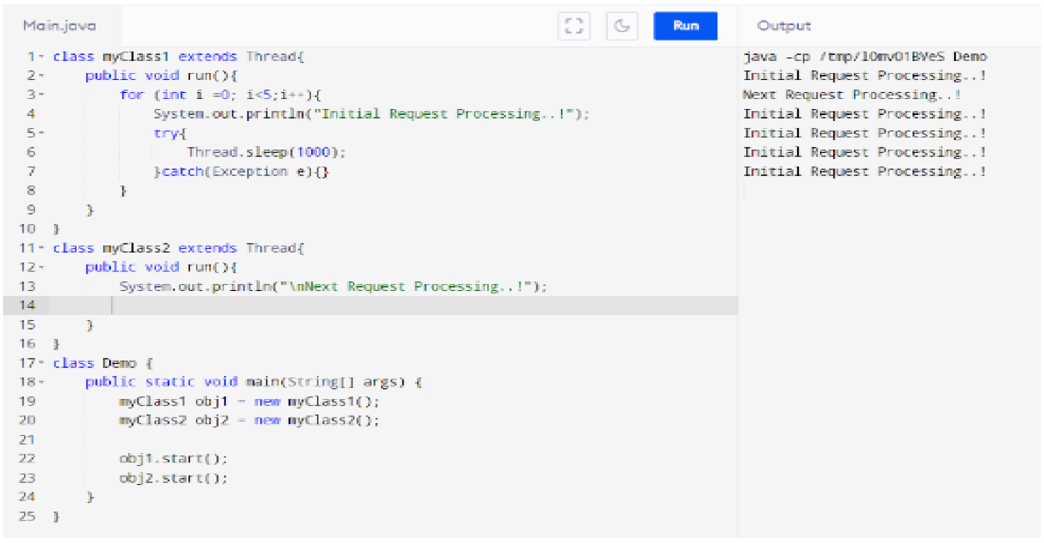
This study finds that Node.js plays a crucial role in the field of web development, if you want your application to be scalable and high performing you can approach with Node.js. With Node.js a developer can easily become a Full Stack Developer, as no extra burden to learn additional language for client-side and server-side language separately. Since, Node.js applies single-thread mechanism there is less burden on CPU memory for handling threads and more focus can be contributed to client-side multi-tasking.



```javascript
function myFunction1() {
    for (let i = 0; i < 5; i++) {
        console.log("Initial Request Processing..!")
    }
}

function myFunction2() {
    console.log("Next Request Processing..!\n")
}

// Asynchronous Function
setTimeout(myFunction1, 3000);

myFunction2()
```

```
node /tmp/z8bBt4tHSN.js
Next Request Processing..!

Initial Request Processing..!
Initial Request Processing..!
Initial Request Processing..!
Initial Request Processing..!
Initial Request Processing..!
```

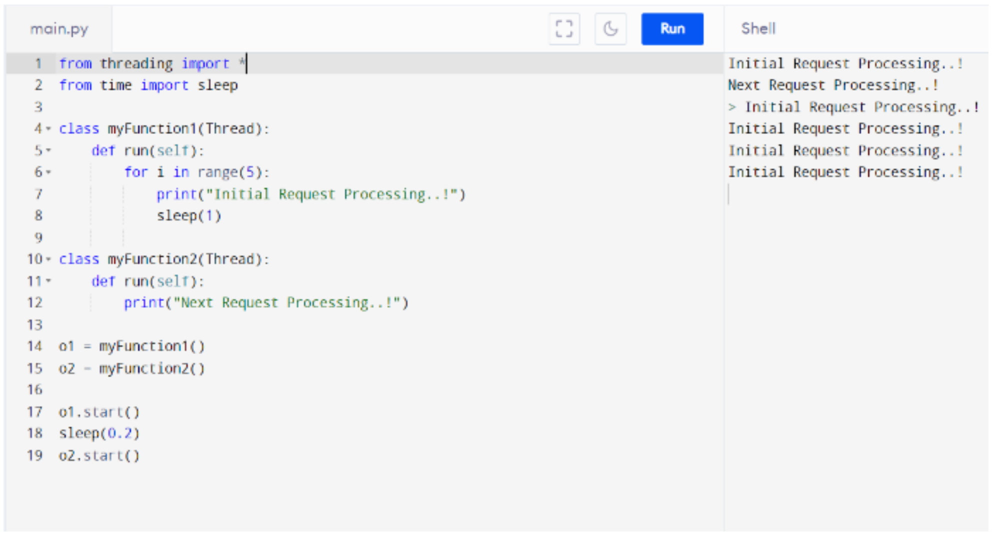
Figure. 1: Asynchronous Program in JavaScript



```java
class myClass1 extends Thread{
    public void run(){
        for (int i =0; i<5;i++){
            System.out.println("Initial Request Processing..!");
            try{
                Thread.sleep(1000);
            }catch(Exception e){}
        }
    }
}
class myClass2 extends Thread{
    public void run(){
        System.out.println("\nNext Request Processing..!");

    }
}
class Demo {
    public static void main(String[] args) {
        myClass1 obj1 = new myClass1();
        myClass2 obj2 = new myClass2();

        obj1.start();
        obj2.start();
    }
}
```

```
java -cp /tmp/10mv01BVeS Demo
Initial Request Processing..!
Next Request Processing..!
Initial Request Processing..!
Initial Request Processing..!
Initial Request Processing..!
Initial Request Processing..!
```

Figure. 2: Multi-Thread in Java



```python
from threading import *
from time import sleep

class myFunction1(Thread):
    def run(self):
        for i in range(5):
            print("Initial Request Processing..!")
            sleep(1)

class myFunction2(Thread):
    def run(self):
        print("Next Request Processing..!")

o1 = myFunction1()
o2 = myFunction2()

o1.start()
sleep(0.2)
o2.start()
```

```
Initial Request Processing..!
Next Request Processing..!
> Initial Request Processing..!
Initial Request Processing..!
Initial Request Processing..!
Initial Request Processing..!
```

Figure. 3: Multi-Thread in Python

Table 1: Comparison Table of different methods for handling multiple requests.

S. No.	Programming Language	Method of handling multiple requests	Expected Result
1	JavaScript (Figure 1)	Single Thread Asynchronous	Send multiple responds concurrently
2	Java (Figure 2)	Multi-Thread	Send multiple responds concurrently
3	Python (Figure 3)	Multi-Thread	Send multiple responds concurrently

Thread - Thread is a single sequential execution of your code. Multiple threads can run multiple different tasks in a single program.
Asynchronous – Asynchronous is the processing of running multiple different tasks independently of one another in a single program.

VI. CONCLUSION

Node.js in the market have made the jobs of Full Stack Developer easier. Developers can now learn single language for maintaining both client-side and server-side. Developers can build high-performance real time applications using single threaded, event-driven architecture. Its event loop and non-blocking I/O model allow tremendous code execution which seems to look that it works in parallel manner. Node.js offers outstanding scalability and offers community support for learning, the community is backed by Google, Facebook, Netflix and Amazon contributing to Node.js advancement. Node.js is easy to learn since it uses JavaScript, and JavaScript is considered as one of the most popular programming languages that is why it is easy to learn and adapt to Node.js.

But there are also some challenges because most of the organizations are not ready to adapt new technologies like Node.js over existing ones like .NET, PHP etc. Hard to learn new concepts like asynchronous model and difficult to maintain large size of multiple callbacks in a single page as it applies single thread paradigm. These are some of the challenges which can be also overcome by introducing new features in Node.js and keep optimizing based on market demands.
Overall Node.js has tremendous scope in the future to meet the market's need as each and every day millions of users are increasing. Developers must consider learning Node.js technology for your smooth scalability of applications, performance and speed.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] Anzhelika Danielkievich. (2021, September 10). Exploring Node.js Pros and Cons. Article title. Available: www.forbytes.com/blog/nodejs-pros-ands-cons.

[2] By Indeed Editorial Team. (2021, July 28). Front-End vs Back-End vs Full Stack: What's the Difference. Article title. Available: Front-End vs. Back-End vs Full-Stack What's the Difference?

[3] Shah, Hezbullah. Nodej Challenges in Implementation. Global Journal of Computer Science and Technology. Available: https://computerresearch.org/index.php/computer/article/view/1735.

[4] Priya Padamkar. Node.js vs PHP Performance | 7 Successful Comparisons To Learn. Available at: Node.js vs PHP Performance | 7 Successful Comparison To Learn (educba.com)

[5] Jhon Tera. (2022, May 13). How to Become a Php Developer Article. Article title. Available at: How to Become a PHP Developer in 2022(simplilearn.com)

[6] By STEPS. How to Become a Python Full-Stack Developer. Article title. Available at: How to Become a Python Full Stack Developer-(stepskochi.com).

[7] Sundaram Subramanian. Skills required to become a Full-Stack Developer. Article title. Available at: skills-requires to become full stack developer net (csharpcorner.com).

[8] Tejas Kaneriya. (2020, June 2). Advantages and Disadvantages of Node.js: Why to Use Node.js? Available at: www.simform.com/blog/nodejs-advantages-disadvantages.

[9] Ramanadham, Kiran. JavaScript versions. (12 Oct. 2021). Article. Available at: JavaScript Version (greycampus.com).

ABOUT THE AUTHORS

Bonjar Basumatary is a student of B.Tech final year, Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India

Nishant Agnihotri, Assistant Professor, Department of Computer Science & Engineering Lovely Professional University, Phagwara, Punjab, India. His Specialization in Network and Security-I,