RESEARCH ARTICLE OPEN ACCESS

Node.js Research Paper

ARYAN- 2302310140023
SHAILY- 2302310140085
SONIKA-230231014009
TANISHK-2302310140104
Department of Computer Applications
APJ Abdul Kalam University
Lucknow, Uttar Pradesh

ABSTRACT

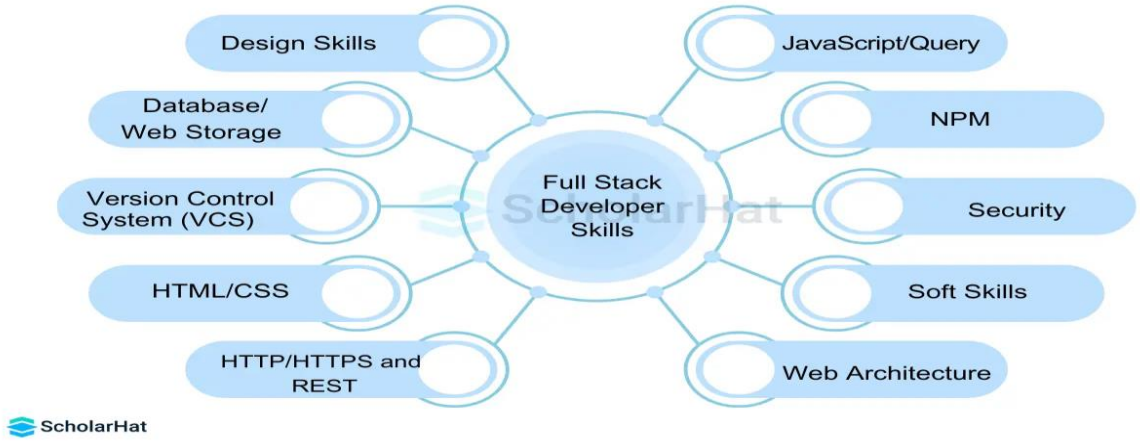
Node.js has led to a complete stack developer who can manage the server and client side themselves. With event-controlled and non-blocking and asynchronous approaches, the approach is quick and reliable for heavy files and heavy network applications that developers can maintain a complete project on individual pages (SPAs) and use for IoT. The results of this study conclude with an overview of the field of research and literature implementation and the challenges in node.js. Finally, suggestions are provided to improve the challenge.

Keywords: JavaScript, Node.js, event-driven, single-threaded, non-blocking, asynchronous.

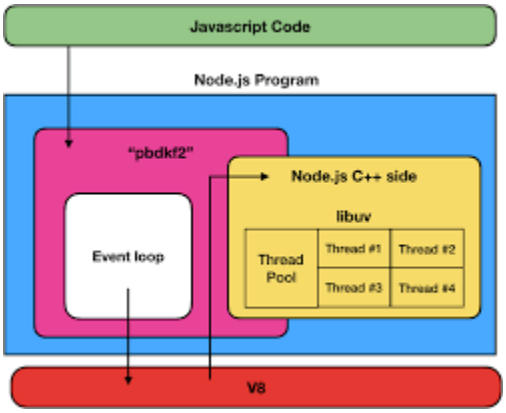
1. INTRODUCTION

Node.js is a JavaScript terminology environment. Based on the Chroms V8 JavaScript engine. This is a cross-platform running time environment originally developed by Ryan Dahl in 2009 for server-side application development. Knots allow you to easily create scalable, fast and easy applications. V8 and knots are primarily written in C and C++, focusing on low memory consumption and performance. The server side can be considered JavaScript. This was created to fix an issue where you can have a platform with performance over the network communication period, which means spending excessive time processing web requirements and answers. With node.js, JavaScript can be used on both the client and server edges. JavaScript is very well developed, highlighting the domination of server-side scripts [1] [2].

Full stack developers are jacks of all stores and they are the ones who do everything. In most cases, backend developers must have the skills of frontend developers and vice versa, putting an additional burden on learning additional skills [4]. Thus, developers of three types of languages, namely JavaScripts, JavaScripts, CSS, servers, client-side languages such as SERC, d. H. Client-side languages such as Java Script, Java Script, Java Script, Java, Java, Java, CSS, Servs, and more. Like SQL Server, MySQL Server, Oracle. As shown in Figure 1.



2. Node.js Libraries
The library **modules** in **node.js, such as FS, HTTP, paths, ciphers, etc** are very consistent and **in the end mostly reference features that live in Libuv projects, so you don't have direct access** to **your** C++ **code. Install node.js.** The idea **for the NPM module is pretty much similar to the idea** of **Ruby Gems. This** is a group of publicly reusable **components that can be used by** easy installation **through** a web **repository** with version and dependency management. Fig 2.



The library module in Node.js like fs, http, path, crypto etc. are very consistent API's and they all ultimately refer to a functionality that are mostly live inside the libuv project so that you do not have access directly to C++ code written in it.

3.1 NPM-
The NPM website often finds a complete list of **package** modules **or is accessed with** the **NPM** CLI **CLI tool.** The module ecosystem is **kind** and anyone can **expose** their own **modules** listed **in** the **NPM** repository. Some of the **most important** useful **NPM** modules **are today. Express-Express -Express.js** is a Sinatra-inspired **Node.js** framework for the **facto** standard **node.js** for **most** of Node.js applications. Connect **-Connect -Connect** is an extensible HTTP **-Server -Framework in node.js,** providing high-performance **plugins. Two** most common web **socket components: Socket.io and Sockjs-A server-side components.** mongodb and mongojs **- Provides** the API **of the mongodb** object **database** in **node.js Bluebird - a** fully **presented promise/A+** implementation with **particularly** good performance. **Wait** - A JavaScript **data** library for **data validation, analysis, manipulation,** and **formatting.** The list **continues.** There are **many** really useful packages available to **everyone (there are no shaming** to **those I have left** here) [6].

4. Key Features of Node.js
4.1 **blocking e/a:**
The **e/a method of** the **node.js** standard library **provides an unblocked** asynchronous **version** and **accepts backgruff** functions. Some methods **may** even **block name counterparts** that end **in sync.**
const content = fs.readfilesync('/file.txt);
// Here we block the file konsrosol.log(content) until it is read.
morework(); // console.log
asynchrones example:
const fs = **request('fs'); fs.readfile('/file.txt, (err, content)=> {if(err)throw err; console.log(content);});**
morework(); // console.log
Called before **console.log.** The ability to **perform more work (without** expecting **to be able to read** the file **can** be a key design choice that **allows for** higher **activation**

5. CONCLUSION

Node has transformed the usability of JavaScript, making Node a complete programming language. From browsers to server-side scripting outside of browsers, Node has made possible the availability of a runtime environment, a library full of free useful modules that can be imported by using an in-built tool named NPM. Node.js uses

eventdriven I/O, non-blocking asynchronous programming to be lightweight and be efficient. Essentially, any business utilizing Node can: utilize fewer servers, utilize less engineers and abatement page load times.

6. REFERENCES

[1] https://Node.js.org/en/docs
[2] Node.js in Action by Mike Cantelon, Marc Harter, T.J. Holowaychuk, Nathan Rajlich.
[3] https://brainhub.eu/blog/9-famous-apps-usingnode-js
[4] https://insights.stackoverflow.com/survey/2019
[5]https://www.journaldev.com/7462/node-jsarchitecture-single-threaded-event-loop
[6] https://www.toptal.com/nodejs/why-the-hell-wouldi-use-node-js
[7] https://nodejs.org/api
[8] A Comparative Analysis of Node.js (Server-Side JavaScript) Nimesh Chhetri