

Role of Node.js in Modern Web Application Development

Ghansham Jadhav¹, Flavia Gonsalves²

¹Student, Institute of Computer Science, Mumbai Educational Trust, Maharashtra, India

²Professor, Institute of Computer Science, Mumbai Educational Trust, Maharashtra, India

Abstract – A JavaScript Runtime Environment Node.js is quick and dependable for substantial records and overwhelming system load applications because of its event driven, non-blocking, and asynchronous approaches. Initially, we focus on Node’s modularity, its in-built package manager labelled Node Package Manager and Node’s working architecture. The main feature of Node.js is its use of non-blocking event-driven I/O with an asynchronous way to remain lightweight and efficient to handle concurrent request. Using Node.js, we can build complex real-time applications that can scale to millions of client connections. We also discuss the factors supporting choosing Node.js and why developers should use it. In this paper, we will describe the advantageous features of Node.js. The main features of Node.js- event-driven I/O, single-threaded and asynchronous programming are discussed with examples to give better insight into the working architecture of Node.js that led to Node’s success.

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

1. INTRODUCTION

Node.js is a JavaScript runtime environment. It is built on Chrome’s V8 JavaScript engine. It is a cross platform runtime environment originally developed in 2009 by Ryan Dahl for developing server-side applications. Using Node we can easily building scalable, fast, and lightweight applications. V8 and Node are mostly written in C and C++ focusing on low memory consumption and performance. It can be regarded as server-side JavaScript. It was created to address the issues platforms can have with the performance in network communication time dedicating excessive time processing web requests and responses. Node.js allows JavaScript to be used end to end, both on the client and on the server end. JavaScript has developed very well and has exceled to dominate server-side scripting [1] [2].

2. Node.js Internal Structure

V8: It is an open-source project created by Google. The purpose of this open-source project is to execute JavaScript code outside the browser. It gives access to node underlying networking and helps the Node to handle aspects of concurrency. (We know Node.js is all about concurrency). 70% of the code written in this project is in

C++ and the other 30% are written in JavaScript.

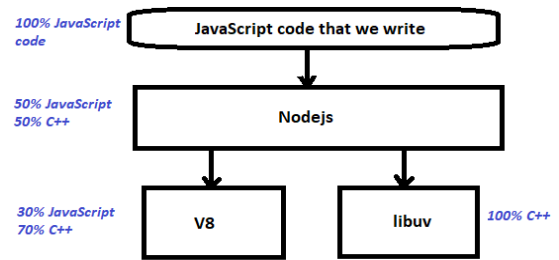


Fig -1: Node.js internal structure-I

libuv: It is an abstraction layer on the top of c-ares (for DNS) , iocp (for windows asynchronous-io) and libeio , libev. LibUv maintains and manages all the Input-Output and events in the event pool. If I have to put in simple words then I would say libuv allow your JavaScript code to perform I/O operation whether its networking or file operation etc. All the file/system operation and TCP level connectivity are performed by this library. This library is completely written in C++.

As JavaScript developer, you will be writing your project in JavaScript and want it to compile and perform its action. So you can see Node.js is just acting as an interface between the JavaScript code that we write and other open source code (V8 and libuv) that is written in either JavaScript or in C++. For JavaScript developer, they don’t interact with C/C++ code directly. So with this nice interface to use and to relate the JavaScript to actual C++ code running in our computer to compile and execute our JavaScript code. Node also provides a unified API’s to use in our JavaScript project. Give a look in Fig-2

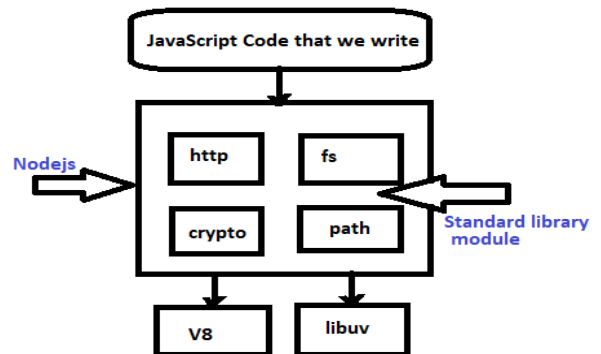


Fig -2: Node.js internal structure-II

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 C++ code written in it. You just use JavaScript functions and that functions calls the Node.js libuv project. You do not have to worry about its internal functionality written in C++ which libuv handles by itself.

3. Module System

JavaScript does not represent an API for module dependency and isolation in its specification. As a result, including multiple modules is only possible by disclosing a global variable. For example, the jQuery module can be included in a HTML document by including this line at the head tag (`script src=https://code.jquery.com/jquery-1.6.1.js`) Then, refer to this module through the global jQuery object. This process pollutes the global namespace and can result in potential naming collisions.

Instead of defining a number of globals, Node has introduced a modular system. One can define their own module or can use the core modules or third party modules. Node.js Modules are plugins, add-ons, and extensions for Node to help with the development process. The Node module discloses a public API (Application Programming Interface) that one can use after the module is imported into the current script. Node modules can be categorized as local modules, core modules and third party modules [7].

3.1 NPM- The Node Package Manage

Node.js has built-in support for package management using NPM, a tool that comes by default with every Node.js installation. The idea of NPM modules is sort of almost like that of Ruby Gems: a group of publicly available, reusable components, available through easy installation via a web repository, with version and dependency management. A full list of packaged modules are often found on the npm website, or accessed using the npm CLI tool that automatically gets installed with Node.js. The module ecosystem is hospitable all and anyone can publish their own module which will be listed within the npm repository.

Some of the foremost useful npm modules today are:

- **express** - Express.js is a Sinatra-inspired web development framework for Node.js, and the de-facto standard for the majority of Node.js applications.
- **connect** - Connect is an extensible HTTP server framework for Node.js, It is providing a collection of high performance "plugins" known as middleware;

- **socket.io** and **sockjs** – A server-side component of the two most common web sockets components out there today.
- **mongodb** and **mongojs** – To provide the API for MongoDB object databases in Node.js.
- **bluebird** - A fully featured Promises/A+ implementation with specially good performance.
- **moment** - A JavaScript date library for validating, parsing, manipulating, and formatting dates.

The list goes on. There are plenty of really useful packages out there, available to all or any (no offense to people who I've omitted here) [6].

4. Key Features of Node.js

4.1 Non-blocking I/O:

The I/O methods within the Node.js standard library provide asynchronous versions, which are non-blocking, and accept callback functions. Some methods even have blocking counterparts, which have names that end with Sync Example

```
const fs = require('fs');

const content = fs.readFileSync('/file.txt'); //
blocks here until file is read

console.log(content);

moreWork(); // will run after console.log
```

Asynchronous example:

```
const fs = require('fs');

fs.readFile('/file.txt', (err, content) => {

  if (err) throw err;

  console.log(content);

});

moreWork(); // will run before console.log
```

In the first example above, console.log are going to be called before moreWork(). In the second example fs.readFile() is non-blocking so JavaScript execution can continue and moreWork() are going to be called first. The ability to run moreWork() without expecting the file read to finish may be a key design choice that permits for higher throughput [1].

4.2 Single Threaded Event Loop

Node.js Platform doesn't follow Request/Response Multi-Threaded Stateless Model. It follows Single Threaded with Event Loop Model. Node.js Processing model mainly supported JavaScript Event based model with JavaScript call-back mechanism.

As Node.js follows this architecture, it can handle multiple concurrent client requests very easily.

The "Event Loop" is heart of Node.js Processing model.

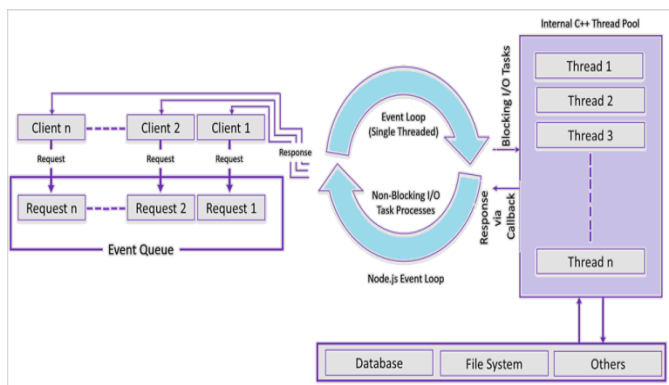


Fig -3: Node.js Application/Server

Single Threaded Event Loop Model Processing Steps:

- User Send request to the Server.
- Node.js Web Server inside keeps up a Limited Thread pool to offer types of assistance to the Client Requests.
- Node.js Web Server gets those requests and spots them into a Queue. It is known as "Event Queue".
- Node.js Web Server inside has a Component, known as "Event Loop". It got this name because it uses infinite loop to receive requests and process them. (Pseudo code to comprehend this underneath).

```
public class EventLoop {
    while(true){
        if(Event Queue receives a JavaScript Function Call){
            ClientRequest request = EventQueue.getClientRequest();
            If(request requires BlokingIO or takes more computation time)
                Assign request to Thread T1
            Else
                Process and Prepare response
        }
    }
}
```

- Event Loop utilizes Single Thread as it were. It is principle heart of Node.js Platform Processing Model.
- Even Loop checks any Client Request is put in Event Queue. Assuming no, athen wait for incoming requests for indefinitely.
- If yes, at that point get one Client Request from Event Queue
- Starts process that Client Request
- If that Client Request Does Not requires any Blocking IO Operations, at that point procedure everything, plan reaction and send it back to Client.
- If that Client Request requires some Blocking IO Operations like connecting with Database, File System, External Services then it will follow diverse methodology
- Checks Threads accessibility from Internal Thread Pool
- Picks up one Thread and allot this Client Request to that string.
- That Thread is answerable for taking that demand, process it, perform Blocking IO tasks, get ready reaction and send it back to the Event Loop
- Event Loop thusly, sends that Response to the respective Client [5].

5. Reasons for why Node.js used widely by Modern Web Developers.

5.1 Google V8 JavaScript Engine

Node.js executes JavaScript code using Google V8 engine. Unlike other JavaScript interpreters, V8 engine compiles the JavaScript code into native machine language. Thus, it enables the runtime environment to boost the performance of web server applications by executing JavaScript code during a faster and more efficient way [1].

5.2 Asynchronous I/O Operations

Node.js further performs all I/O operations asynchronously through a single-threaded event loop. The advanced methodology makes the Node.js application to perform I/O operations by sending asynchronous task to an event loop alongside a call-back function. After sending the async task to the event loop, the application continues executing the remaining code. After completing the asynchronous operation, the event loop returns to the task, and executes the call-back function. In addition to consuming less memory, the feature enables Node.js to

handle a large number of concurrent connections efficiently. The programmers can use the runtime environment to perform common tasks like file system or network connections, reading or writing to the database.

5.3 Robust Tooling

While using Node.js, the developers can take advantage of a dependable package manager like npm. In addition to being fast, consistent and robust, npm also simplifies the method of specifying and installing project dependencies. At an equivalent time, it keeps the project dependencies separate to eliminate chances of version conflict. The users can further take advantage of the strong file streams capabilities of JavaScript using tools like broccoli, gulp and brunch, while using a popular task runner like grunt.

5.4 Complements Real-Time and Multi-Use Requirements

In addition to using responsive web design, the developers nowadays need to build real-time and multi-user web applications. Node.js enables programmers to make complex gaming, chatting and communication applications without putting any overtime and energy. The developers can use websocket protocols to create real-time web applications. As a two-way channel between the client and web server, websockets can make the online server push data to the client in a very faster and more efficient way without increasing the overhead of HTTP. At an equivalent time, the developers can create multi-user applications by taking advantage of the event loop feature of Node.js [7].

5.5 Facilitates File Streaming

The web programmers can take advantage of the efficient I/O handling capacity to Node.js to reduce the amount of time required for streaming a file from the file system. They can use the runtime to read/write streams to both HTTP and websockets. Thus, they have can reduce the overall transcode audio or video processing time in a number of ways. For instance, a programmer can simply shift the stdout from the web server to a web browser through websockets, and allow the web page to display output to the user in real time [7].

5.6 Popularity of JavaScript

Since early in the evolution of WWW, JavaScript has been there in the browser. Even available when AJAX emerged, JavaScript was vital. This has led to the popularity of JavaScript among developers, despite some criticism. No matter which server-side scripting language is used, JavaScript has been the choice for client-side scripting. Familiarity with JavaScript and adherence of Node to JavaScript, with capabilities to code in the server-side and numerous other features has developers to adopt Node. By leveraging the best features of JavaScript as a language and

nurturing a vibrant community, Node has become a popular platform and framework, with continued adoption growth

5.7 Build and Maintain a Single Code Base

Node.js enables developers to write down both client-side and server-side code in JavaScript. So a programmer can use JavaScript for building both frontend and backend of an internet application. Thus Node.js eliminates the gap between frontend and backend development [8].

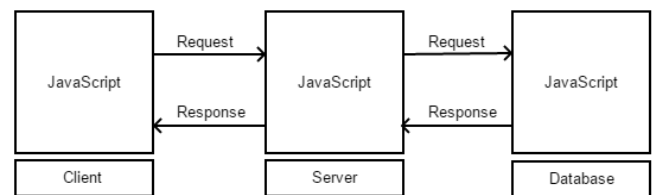


Fig -4: JavaScript end-to-end

6. What reason are the biggest companies using Node.js for?

6.1 PayPal

PayPal is one among the most important and most well-known worldwide systems for Internet payments. The platform enables its users to transact with one another online quickly and simply in additional than 100 currencies.

According to Payal, the service has over 184 million active customer accounts (as of 2015). The Company has been using Node.js to create the consumer-facing side of its web applications.

Why did they choose Node.js? Senior Director 'Jeff Harrel' of Payments Products and Engineering at PayPal says: "Node.js helps us solve boundary between the browser and server by enabling both the browser and server applications to be coded in JavaScript. It brings together our building claims to fame into one group which permits us to know and respond to our clients' needs at any level inside the technology stack".

Results:

The Node.js application was built twice as snappy with less people, in 33% less lines of code and 40% fewer files (in connection with past Java-based application) [3].

6.2 LinkedIn

LinkedIn is a business-situated person to person communication administration found in 2002 in Mountain View, California. LinkedIn permits clients to welcome

anybody (regardless of whether a site client or not) to turn into an association.

It is accessible in 24 dialects and as of now has in excess of 400 million individuals in more than 200 nations and regions. LinkedIn utilizes Node.js to engage the server side of its mobile application.

For what reason did they pick Node.js?

As indicated by LinkedIn's Mobile Development Lead, Kiran Prasad "One explanation was scale. The second is, on the off chance that you take a gander at Node, the thing its best at doing is conversing with different services."

Results:

When contrasted and the past Ruby on Rails-based adaptation, the new portable application is up to multiple times quicker and utilizes just a small amount of assets – servers were sliced from 30 to 3. The advancement itself was uncommonly quick [3].

6.3 Yahoo

Yahoo is an American worldwide technology organization, centering and all around perceived for its web portal, search engine, and related services. As per Yahoo, service attracts more than 500 million consumers in more than 30 languages every month. For what reason did they pick Node.js? Eric Ferraiuolo, Principal Software Engineer at Yahoo clarifies: "It can be scalable, and every property that we've moved toward the Node.js stack has seen an expansion in execution" Results: Node.js has changed and bound together the frontend designer culture at Yahoo, it as of now controls numerous locales [3].

6.4 Netflix

Netflix is the world's greatest worldwide supplier of video streaming including movies and TV series, accessible in more than 190 nations. As of April 2016, Netflix detailed more than 81 million endorsers around the world, with in excess of 46 million in the U.S itself. At Netflix, the entire UI is worked with Node. The innovation demonstrated so viable, that the organization needs to utilize it likewise in different layers of the stack. For what reason did they pick Node.js? The group chose to utilize Node.js to accomplish lightweight, modular and fast application. Therefore, the start-up time of their new application has been diminished by 70% [3].

6.5 GoDaddy

GoDaddy is a traded on an open market Internet space enlistment center and web hosting organization. As of January 2016, with in excess of 13 million clients and 61 million domains under management, GoDaddy makes the world's biggest recorder. As of late organization has

patched up its whole backend to a completely open-source Node.js-based framework. For what reason did they pick Node.js? As indicated by Stephen Comisso, Senior Software Developer at GoDaddy, Node enables to build quality applications, deploy new features immediately, write unit and integration tests easily, REST easily. Another key feature is NPM access. Results: GoDaddy's Vice President of Engineering, Antonio Silveira states: "we are currently utilizing about 10x less servers to have our client sites and we decreased the Time To First Byte (TTFB) significantly from ~60ms to something around ~12ms. Execution is a key differentiator when we talk about web applications, remembering a superior situation for Google's query items." GoDaddy is currently ready to deal with a similar burden with just 10% of the hardware [3].

According to Stack Overflow survey 2019 the Node.js is the most commonly used and most wanted technology [4] Give a look in Fig-5 and Fig-6.

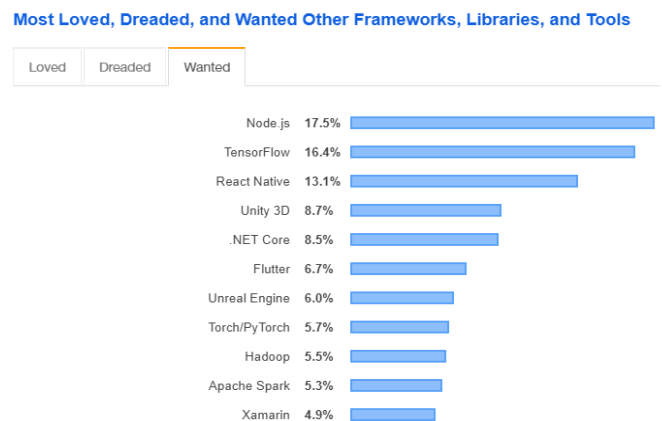


Fig -5: Stack Overflow Survey-2019

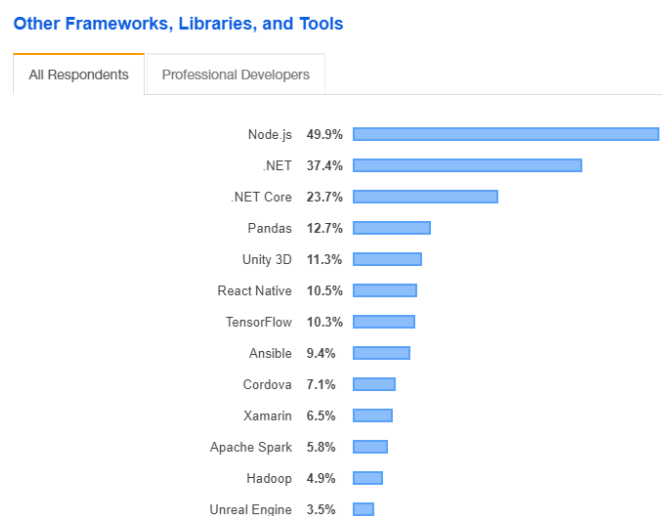


Fig -6: Stack Overflow Survey-2019

Potential application areas of Node.js

- Media
- Payment gateways
- Ecommerce
- Social media
- Enterprise web apps
- Backend/API for mobile apps

7. 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 event-driven 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.

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-using-node-js>
- [4] <https://insights.stackoverflow.com/survey/2019>
- [5] <https://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop>
- [6] <https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js>
- [7] <https://nodejs.org/api>
- [8] A Comparative Analysis of Node.js (Server-Side JavaScript) Nimesh Chhetri.