**Javascript is the most common programming language used in many startups and large enterprises for software development. It is used as a client-side development tool in 95% of the websites. Before 2009 Javascript was only employed in front-end development and it was used along with various server-side languages. Around one decade back (in 2009) when Ryan Dahl (the original developer of NodeJS) released the initial version of NodeJS it was surprising for developers because they couldn’t have imagined before that this language is so powerful and it can be used to write the code for backend application as well. Today NodeJs is getting a lot of well-deserved hype and many companies switched their tech stack to NodeJS including Paypal, LinkedIn, Uber, Yahoo, Medium, GoDaddy, Groupon, Walmart.**

**NodeJS is the hottest technology across the world, especially in Silicon Valley. It is the perfect skill to open up amazing career opportunities for any software developer. Well, every programming language has some benefit of using it but what makes Node so special? why it is getting so much hype among developers? what are some of its cool features and why so many companies are using it for different kinds of applications? Let’s discuss some reasons to learn NodeJS in 2020…**

**1. Real-Time Applications**

**Today the web has become much more about interaction. Users want to interact with each other in real-time. Chat, gaming, constant social media updates, collaboration tools, eCommerce websites, real-time tracking apps, marketplace- each of these features requires real-time communication between users, clients, and servers across the web. Building a real-time application is challenging because it happens at a massive scale, supporting hundreds, thousands, even millions of users. The real-time communication between client and server requires fast and persistent I/O.**

**NodeJS is best suited for these kinds of applications. The synchronization process with NodeJS is fast and in an organized manner as events drive the architecture serves both the client-side and server-side. The event loop through the web socket protocol handles the multiuser function. It works in TCP and avoids HTTP overload. NodeJS also make RTA lightweight, scalable, maintainable, and usable from a software development standpoint.**

**2. Low Learning Curve**

**No matter what language you are using for backend application you’re gonna need JavaScript for front-end anyway so instead of spending your time learning a server-side language such as Php, Java or Ruby on Rails, you can spend all your efforts in learning JS and mastering in it. Javascript the same language can be shared on both client-side and server-side. So a developer who knows JS can act as full-stack devs without having to learn additional languages. The frontend and backend are also easier to keep in sync because of the single language use on both sides.**

**For startups, this is one of the big advantages to get their job done quickly with fewer developers. There is no need to divide the team for both sides. It provides higher productivity and the ability to share or reprocess the code and within the team offers a smooth exchange of knowledge.**

**3. Performance and Scalability**

**NodeJS is built upon Chrome V8’s engine powered by Google. It allows Node to provide a server-side runtime environment that compiles and executes JavaScript at lightning speeds. The V8 engine compiles JavaScript into native machine code, instead of interpreting it or executing it as bytecode and that makes Node really fast. Lightweight Javascript achieves high performance with fewer lines of code when compared to Java or C. The Chrome V8 engine also being updated constantly as Google continues to invest heavily in it.**

**The reason behind the fast execution of Javascript is it’s Event Loop. In a typical application server model that uses blocking I/O – in this instance, the application has to handle each request sequentially, suspending threads until they can be processed. This can add complexity to an application and, of course, slows an application down.**

**4. NPM Support With Rich Modules**

**Only a few programming languages offer a rich package ecosystem as NodeJS. When you install Node.js, it automatically installs the programs of NPM (Node Package Manager). Any Node.js developer can package their libraries and solutions into a module that anyone can install using Node’s official package manager, NPM. Thousands of libraries and tools for Javascript development are gathered on NPM. With the constant support of the NodeJS community, NPM focus on encouraging users to add new packages, so you have countless readymade solutions for the specific issue.**

**Till now it has more than 60K modules and it’s still growing every day. Isn’t it a good idea to use these modules for some common features instead of writing the code from scratch? This great feature of Node reduces the complexity, makes the development simpler, faster and allows you to share, refresh and even reuse the code effortlessly.**

**5. Useful Single Codebase**

**Using Node It’s easy to send and synchronize the data between server-side and client-side coding. Because of using the same language Javascript on both sides your source code will be cleaner and more consistent. You will be using the same naming convention, the same tools, and the same best practices. The time of the developers is saved to a greater extent because of this feature.**

**6. Data Streaming**

**Just like Array in data structures, streams are the collection of data and to handle this data it requires top-notch I/O data processing methods. Node.Js comes to rescue since it’s good at handling such an I/O process which allows users to transcode media files simultaneously while they are being uploaded. It takes less time compared to other data processing methods for processing data. Node.js streams help simplify the I/O tasks tremendously.**

**There are four types of streams used by node.js- Writable, Readable, Duplex, and Transform, along with the Pipe method to handle data. The developers can take amazing advantage while forming features like processing files when they are uploaded. NodeJS streams enable apps to consume less memory while dealing with massive amounts of data to function faster. This feature gives more benefit to the developer working on real-time audio or video encoding.**

**7. Well Suited For Building Microservices**

**As we have discussed that NodeJS is highly scalable and lightweight that’s why its heavy favorite for microservice architectures. In a nutshell, microservice architectures mean breaking down the application into isolated and independent services. It makes it easier to update and maintain the architecture as your services are decoupled and you can add new or fix the existing architecture without dealing with the other parts of the applications. NodeJS fix well for designing such architectures with the help of Node modules which represent building blocks of NodeJS functions. With this architecture, applications can be independently developed, handled, operated, and tested, which saves you of the infrastructure risks. This feature allows code-reusability between client-side and server-side also it reduces the time and cost of development because you need only to consider new services introduced or updated.**

**8. Strong Corporate Support**

**In 2015 a number of companies including IBM, Microsoft, PayPal, Fidelity, SAP organized a NodeJS Foundation. It’s an independent community and aimed at facilitating the development of NodeJS core tools. The foundation NodeJS was formed to speed up the development of NodeJS, and it was intended to allow broad adoption of it. There is the continuous growth of the organizations that make use of Node.js in the production. Almost three hundred prominent companies like Medium, Uber, are included in it.**

**Follow these steps:**

**Import required module: The first step is to use ?require? directive to load http module and store returned HTTP instance into http variable. For example:**

**var http = require("http");**

**Create server: In the second step, you have to use created http instance and call http.createServer() method to create server instance and then bind it at port 8081 using listen method associated with server instance. Pass it a function with request and response parameters and write the sample implementation to return "Hello World". For example:**

**http.createServer(function (request, response) {**

**// Send the HTTP header**

**// HTTP Status: 200 : OK**

**// Content Type: text/plain**

**response.writeHead(200, {'Content-Type': 'text/plain'});**

**// Send the response body as "Hello World"**

**response.end('Hello World\n');**

**}).listen(8081);**

**// Console will print the message**

**console.log('Server running at http://127.0.0.1:8081/');**

**Combine step1 and step2 together in a file named "main.js".**

**File: main.js**

**var http = require("http");**

**http.createServer(function (request, response) {**

**// Send the HTTP header**

**// HTTP Status: 200 : OK**

**// Content Type: text/plain**

**response.writeHead(200, {'Content-Type': 'text/plain'});**

**// Send the response body as "Hello World"**

**response.end('Hello World\n');**

**}).listen(8081);**

**// Console will print the message**

**console.log('Server running at http://127.0.0.1:8081/');**