

**Research Report**

**Server-side Javascript**

**The web development revolution**

Author: Bertocco Thomas-Killian - #13758322

Lecturer: Michael Hansen

CP5610 E-Business Technologies | SP52 | 2019

20 September 2019

# Abstract

Since the release of the JavaScript v8 engine, JavaScript has become a server-side technology with the emergence of NodeJS. The latter was created in 2008 and has been much talked about because of the many possibilities it offers in the development of web applications. Thus, it is now possible to make an application entirely in JavaScript, this is called Full-Stack JavaScript.

What is the potential of server-side JavaScript? Will it become an important technology?

Node.js has one of the most dynamic communities. "NPM", its official package manager, is one of the largest module libraries. Besides, the Node.js community creates the largest number of modules per day. Several frameworks have been set up, and this technology is experiencing a tremendous increase in power. Node.js owes this popularity to its lightness and performance.

Also, the Node.js community creates the largest number of modules per day. Several frameworks have been implemented, and this technology is experiencing a tremendous increase in power.Node.js owes its popularity to its lightness and performance.

Mature languages such as Java and PHP are still preferred to JavaScript in many companies because they have been used for many years, unlike other technologies. Is it a question of performance or do these companies take no risk in launching into a new technology?

Server-side JavaScript confirmed what was expected of him. There is a reason why it has been growing for several years.

Will JavaScript on the server-side be as durable as Java and PHP" This is no longer a question because this technology has been operational for many years and will probably become a standard on the web.

Despite this, many companies and developers do not want to take the risk of launching this technology and prefer to stay on their earnings as a precaution. It is therefore essential to show the potential of this technology because it is much more advantageous than others in some cases of use and that is the purpose of this research.

Table of Contents

[Abstract 2](#_Toc20019482)

[Introduction 4](#_Toc20019483)

[Java & PHP: Lack in the ecosystem 5](#_Toc20019484)

[Server-side JavaScript integration 6](#_Toc20019485)

[Concept 6](#_Toc20019486)

[Technical aspect 7](#_Toc20019487)

[Introduction 7](#_Toc20019488)

[Non-blocking 8](#_Toc20019489)

[Websocket 8](#_Toc20019490)

[JavaScript Fullstack 10](#_Toc20019491)

[Introduction 10](#_Toc20019492)

[Client-side JavaScript 10](#_Toc20019493)

[Server-side JavaScript 11](#_Toc20019494)

[Database 11](#_Toc20019495)

[Possible solution: MEAN 12](#_Toc20019496)

[Conclusion 13](#_Toc20019497)

[Server-side JavaScript frameworkization 14](#_Toc20019498)

[Introduction 14](#_Toc20019499)

[Concept 14](#_Toc20019500)

[Framework types 14](#_Toc20019501)

[Popular Node.js frameworks 15](#_Toc20019502)

[Conclusion 17](#_Toc20019503)

[JavaScript: Is it a Solution? 18](#_Toc20019504)

[For which case is JavaScript used on the server-side? 18](#_Toc20019505)

[In which case do not use Node.js? 20](#_Toc20019506)

[Will JavaScript match our project of tomorrow? 21](#_Toc20019507)

[Conclusion 28](#_Toc20019508)

[References 29](#_Toc20019509)

# Introduction

JavaScript technology on the server-side has been much talked about in recent years, especially with the Node.js web application development platform, which has opened the door to Full-Stack JavaScript. Node.js is currently experiencing a dazzling rise, but will it remain so for the next few years? Indeed, several technologies have emerged over time and were announced as future standards in their field. For many of them, their beginnings were shattering, but they did not last long.

Several questions arise.

Will it be a technology as durable as Java or PHP?

Will it become a standard on the web in the coming years, or will it disappear as fast as it has appeared?

Node.js is now ten years old and has been able to stay at the forefront. We can, therefore, say that it has almost ensured its sustainability. However, it is still a second choice compared to Java and PHP except for some specific cases. So what are the reasons for this second place? Does the IT world know JavaScript on the server-side or do companies not want to take the risk of migrating to this technology?

This document reviews the current status of server JavaScript to determine its and compare it with the competition. This will allow developers to have a more accurate look at and choose Node.js for the appropriate use cases. The purpose of this research is to final to offer server JavaScript technology as a solution.

# Java & PHP: Lack in the ecosystem

Despite the richness of languages like Java and PHP there is still a big gap in their ecosystem. Those who make them "incomplete" or non-full-stack languages.

Indeed, Java and PHP are server languages used only to manage the back-end of an application. These server-side languages are used to manage the information displayed on a screen.

They work by converting HTML files in the server into information that can be used by the browser.

Each time someone visit a website, his browser makes a request to the server that contains the content of the website. The request usually takes only a few milliseconds, but it ultimately depends on a multitude of factors (internet connection, server location, etc.).

The main gap in these languages is the client-side part. In order to be able to modify the design of a website once the web page has been loaded, programs must be able to access the DOM (Document Object Model) that we will see below. Roughly speaking, the DOM allows developers to modify the structure of a web page (change of colors, animations, etc.). Currently, Java and PHP are unable to access the client-side. This is why it is necessary to combine them with JavaScript.

JavaScript has been specially designed to master DOM, so it is the specialist in this field.

The disadvantage of this combination (Server language + JavaScript) is that you have to master several languages in order to create a complete web application.

JavaScript, therefore, intervenes to fill the gap in languages such as Java and PHP. However, in recent years JavaScript tends more and more to become a Fullstack language (client-side + server-side) especially with the Node.js platform which is a JavaScript execution server. It allows running a multitude of Node.js frameworks (Meteor.js, Express.js, Sails.js...).

The major advantage of FullStack JavaScript is that despite the specificities of each framework, the basic language remains the same. It is therefore no longer necessary to master several languages in order to have a complete website.

# Server-side JavaScript integration

Concept  
JavaScript on the server-side is dominated by Node.js, a platform based on the same engine V8 used by Google's Chrome browser. What is interesting about Node.js is not its ability to run JavaScript on the server-side, but rather its ability to provide asynchronous and non-blocking behavior, making it an ideal choice for real-time web applications.

We cannot define Node.js as a Framework because it has several modules. Node.js defines itself as a platform, which makes it many things. Node.js is an execution environment for JavaScript server applications, i.e., like a traditional web server, it simply uses an event loop design that makes it different and ideal for real-time applications. Node.js also has its own API for features such as HTTP requests, processing buffers and DNS requests, among others, this means that it is also a Framework. Besides, Node.js also comes with a package manager called npm, which provides access to a multitude of modules, libraries, and frameworks. Several thousand packages are written for Node.js.  
  
Number of modules and scalability  
The number and diversity of modules for a platform has a certain importance because it will determine the amount of functionality that will not have to be developed for a project. It is all the more important for a technology like Node.js which is quite limited without it.

However, in order to be able to judge if the number of modules available for Node.js is significant, it is necessary to be able to compare this with other libraries.

Here is a summary of the evolution of the number of modules in the most important module libraries every day :

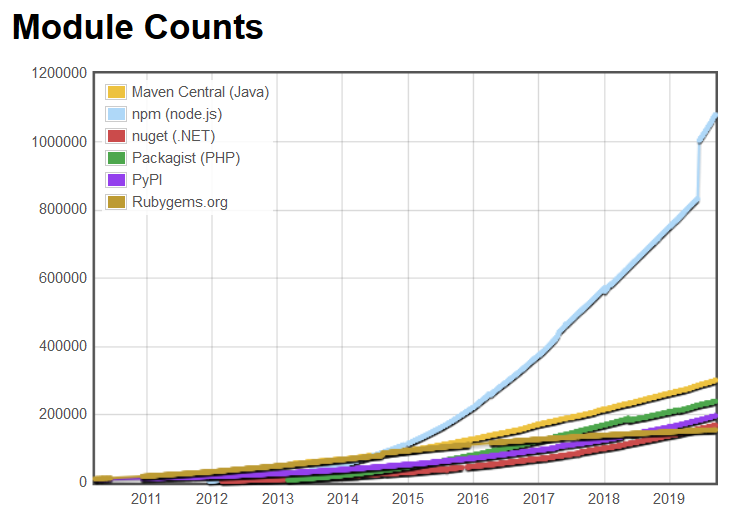


Figure 1. Evolution of the amount of modules. modulecounts.com (2019)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Package Manager | Maven Central (Java) | npm (node.js) | nuget (.NET) | Packagist (PHP) | PyPI (Python) | Rubygems (Ruby) |
| Total number of modules | 301000 | 1080000 | 169000 | 239000 | 196000 | 155000 |
| Average of  module per day | 324/day | 806/day | 135/day | 118/day | 136/day | 19/day |

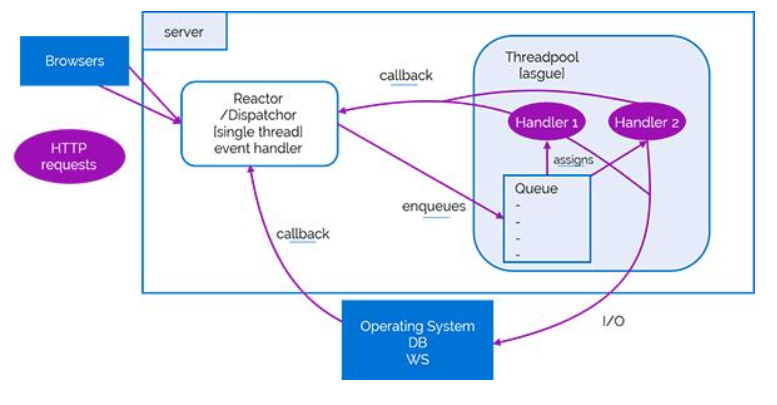
Table 1. Comparison of the amount of modules among the different packages manager.

After looking at this table, we can see that the number of modules developed for Node.js is far ahead of the other main module directories. We also note in a second time that the evolution of the number of modules is by far the most important on Node.js.

However, adding a new module in NPM is extremely simple and could explain the large number of modules offered by the Node.js community.

Technical aspect

IntroductionAs already mentioned, Node.js is a web development platform based on Google's V8 engine and thus exploiting JavaScript. However, what does this ultimately mean? What is the point? Here is an explanation of how this technology works and why it is of interest.

  
Figure 2. Google V8 engine.

Non-blocking  
Node.js works with a Non-blocking Input/Output system. This so-called asynchronous mode of operation is widely used in JavaScript. On the client-side, it is frequently used to communicate with the server, via AJAX (Asynchronous JavaScript and XML) requests(nodejs, 2019). It is one of the features of Node.js that makes it so powerful. To fully understand the interest of this operation, it is first necessary to understand the difference between the blocking (synchronous) mode and the non-blocking mode.

A server in blocking mode, when requested by a request, goes to assign a thread to the processing to be performed. This one will query a resource (a database, an API, the hard disk, etc...) and then wait for its answer. During the entire waiting time of the thread, it will do nothing and all the memory space assigned to it will be locked. The server will still have the ability to create new threads to handle new requests, but it will then have to pay the cost in memory space.

While on a server in non-blocking mode, when the server is requested, it will assign to its turn a thread to this request that will query the resource, but the thread will not wait for the answer(nodejs, 2019). We will let the system manage the wait for the return that will alert us when it arrives. Then we can take care of that answer. In the meantime, the thread will handle other requests. This system optimizes resources.

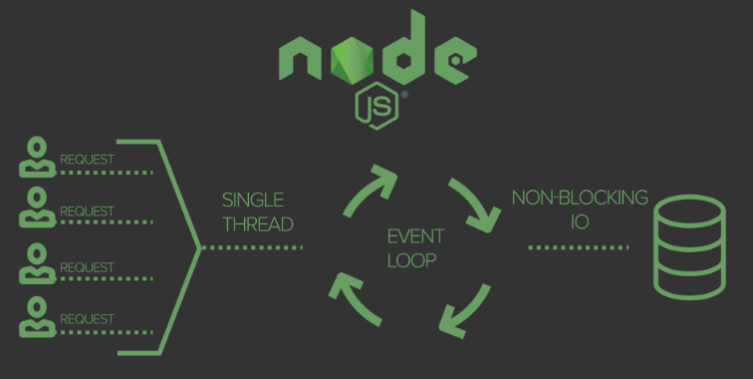


Figure 3. NodeJS

Websocket

#### Introduction

Websocket is a fairly recent protocol that has only recently been standardized by the IETF. Thus this protocol allows communication between browsers and web servers via a channel of bidirectional and full-duplex communication on a TCP socket.

#### Real-time

Node.js via the WebSocket protocol will be able to exploit a new concept, real-time. However, what does this concept consist of?

To put it simply, the server and client workstations are permanently connected. Thus the server is now able to send information to clients when it wants, it is no longer requires that this client send ita request so that the server can send it information. This is the "push" principle. Thus, as in the example below, an action is performed by a customer and then information can be forwarded to all connected clients.

#### Client/Server communication via Socket.io

A client will transmit information to the Node.js server via a WebSocket.

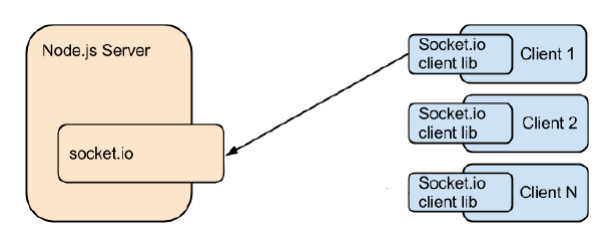


Figure 4. Client to Websocket

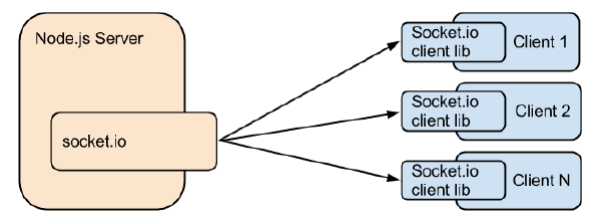
Node.js will interpret its data and transmit information to all currently connected users.

Figure 5. WebSocket to Clients

#### Use case

This concept of real-time will be very interesting for several types of use. First of all in everything that will require real-time communication between the different users, by, for example, instant messaging.

One of the other axes where real-time is used a lot is in multiplayer games, where each client goes have to transmit the actions in real-time to the other players.

In fact, real-time will be used when it comes to developing oriented features multi-user.

We also see the emergence of many collaborative tools that work with this feature.

## JavaScript Fullstack

IntroductionThe Node.js platform allowing to develop in JavaScript on the server-side allows a unique thing in web application development. Finally, it gives the possibility to develop with only one language (except HTML and CSS), both on the client-side and on the server-side. This possibility offered by Node.js allows developers who will work on this type of project to need only the expertise of a single language. Moreover, the communication between the client and the server will be easier because it uses the same communication syntax. Thus, the major defect of web projects which is intended to be the slowness of these projects compared to traditional applications is greatly reduced.

### Client-side JavaScript

There are now many frameworks that exploit the possibilities of the language and allow developers to create a more structured code. They are dissociated from Node.js but can be combined with. Here are some existing client-side JavaScript solutions for web application development.

#### Angular.js

Let us start with Angular.js that we briefly saw before. Angular.js is the Framework JavaScript from Google, the first version of this framework was released in 2009. As with all frameworks, it fills the gaps in JavaScript and implements new features at the language. However, more particularly, it provides a slightly different way of developing a web application.

Angular.js is designed to develop SPA (Single Page Application) applications, it is applications where there is no page refresh, everything is integrated into a single page web (Angular, 2019). Unlike other frameworks, the developer should avoid manipulating the DOM (Manipulation of the elements of a web page: image, block, text, etc.). Angular.js will allow adding attributes to the tags of our HTML page in order to integrate interactions between the elements without the need to develop them in JavaScript (This way you can directly on the HTML structure of your page, tell a button to display an element on the page when you click on it).

The JavaScript code will play the role of the controller and will link the HTML view to the queries sent to the server. This system greatly reduces the size of the code of a JavaScript project and simplifies code reading. This code will be structured in modules, to separate the different processing carried out on the page.

The other strength of Angular.js, which makes it the most widely used Framework currently in use, is its testability (Angular, 2019).

Angular.js, thanks to its modular operation, allows to set up very efficiently unit tests, which is often very complicated in JavaScript.

#### Backbone.js

The Backbone framework created in 2010 is one of the lightest frameworks in its category. It is its lightness and low dependence on other libraries that have made it popular. It is therefore only dependent on "Underscore.js" which is a library developed by the same team. Many popular applications have been developed on this Framework: Twitter, Foursquare, LinkedIn. This Framework is intended to specialize in single-page application development (Backbone.js, 2019).

One of the main shortcomings of Backbone is that it is often considered to be quite limited in these features. However, it is difficult to compensate for the lightness and richness of a framework.

ReactReact has been developed by Facebook since 2013. The library stands out from its competitors by its flexibility and performance, working with a virtual DOM and updating the rendering in the browser only when necessary. React does not use a template system and only works with JavaScript, allowing a complete encapsulation of the component within a single class (Reactjs, 2019). To facilitate the writing of the view, the initial team at Facebook developed a language, JSX, which allows generating Javascript objects with a notation similar to HTML (Reactjs, 2019).

The library is used by Netflix, Yahoo, Airbnb, Sony, Atlassian, and Facebook teams, applying dogfooding on the eponymous social network, Instagram and WhatsApp.

Vue.jsVue.js is an increasingly used framework, created by Evan You in 2014. It has the advantage of being easier to learn than other javascript frameworks such as React (VueJS, 2019). It also has the particularity of operating by components (code bricks that can be reused over and over again) (VueJS, 2019). This is particularly useful for large web and mobile applications: it saves time. Among its users are Alibaba, but also Nintendo, Euronews, and even Netflix. Companies with dense digital platforms that need reactive technology to facilitate UX.

### Server-side JavaScript

For the server part, on a full JavaScript application, it is possible to work with Node.js.

Being on server-side JavaScript code will simplify the reception of data. Because the exchanges will be done in "JSON" format, which is only a simple JavaScript object, so it avoids manipulating the data to adapt it to the server language.

### Database

Finally, the last important point in the development of a full JavaScript application is simply the communication with the database. Fortunately, with Node.js and NoSQL databases (Not only SQL), it is possible to make queries in JavaScript. However, it is still possible to work on SQL databases.

#### MongoDB

MongoDB is a NoSQL database, it was developed in 2007 by 10gen. It is the most widely used database with Node.js, it allows, thanks to many modules including "mongoose" to make queries to the database in JavaScript (MongoDB, 2019). It is a basis for data called document orientation because it allows storing objects such as "JSON" which is impossible in SQL. For a project in full JavaScript, this is an undeniable advantage, no need to heavy requests to store all the information of a JavaScript object so that it can store it immediately without prior processing.

Mongoose one of the most popular Node.js modules for MongoDB allows testing the modifications made on the database on the fly. This is to check if the information is transmitted in the correct format. Thus there is the possibility to test if the type of the value corresponds to the expected type, as well as its size or format if it is a string (in the case where a phone number (so 10 digits) is expected, or if a password with a minimum of complexity is expected for example).

#### Other NoSQL databases

Node.js works with many other NoSql databases, such as Redis, Apache CouchDB or DynamoDB, many modules exist on NPM to work on these different bases. Each base of data to its particularities, but they all have the advantage of being able to work directly with JSON" objects and therefore JavaScript, which is a real time saver.

### Possible solution: MEAN

To create a full-stack JavaScript project, several solutions already exist on the market. One of them of which is the MEAN stack solution.

The term MEAN stack refers to a collection of JavaScript-based technologies used to develop web applications. MEAN is an acronym for MongoDB, ExpressJS, AngularJS, and Node.js. From client to server to database, MEAN is a complete stack of JavaScript (TekSlate, 2018).

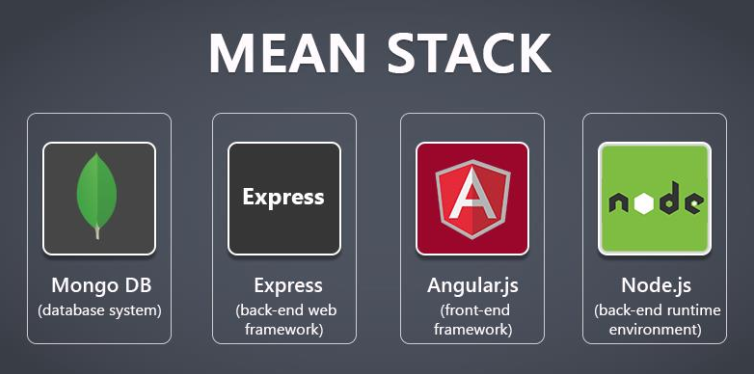


Figure 6. MEAN stack.

The MEAN solution offers a modern approach to web development. It also uses the power of modern SPAs (Single Page Applications), which does not require completely refreshing a web page for each server request as most traditional web applications do (TekSlate, 2018). Today, there is a tendency to develop several frameworks that correspond to a single programming language and form a full-stack solution. However, from now on, using MEAN is a very effective approach to web development.

MongoDB, Express.js, AngularJS, and Node.js appeared only a few years ago.

Now they have grown up and aligned together. There is a very good sustainability of these frameworks, many developers from LAMP have switched to the MEAN solution in order to be on mono language.

#### Who uses MEAN stack?

There are many well-known organizations that use Node.js in production, including PayPal, LinkedIn, Netflix and the New York Times. Here are some examples of application types that can benefit from the use of Node: REST APIs, chat applications and applications real-time monitoring (brokerage dashboards, real-time statistics on users, etc.)

This new approach requires the use of very fast, scalable, easy back-end applications to be deployed and maintained. All large companies have started to migrate their applications to Node.js and others are building applications from scratch using the technology MEAN stack. PayPal, Dow Jones, and Uber have already placed Node.js business solutions in production.

Yahoo, HP and many others have planned their next generation of products on this MEAN battery platform.

ConclusionIt is possible to work end-to-end on a web project with JavaScript. On the customer side, since 2010, many frameworks have been developed to develop structured web applications, to overcome the shortcomings of JavaScript.

On the server-side, Node.js and its various modules allow developers to work efficiently with the client-side by communicating in "JSON", the native JavaScript communication mode.

Finally, many modules allow developers to work with any SQL or NoSQL database. NoSQL databases, mainly MongoDB, are gaining in popularity thanks to the possibilities it provides on Node.js.

## Server-side JavaScript frameworkization

### Introduction

Today, full-stack JavaScript is a common approach to building enterprise-wide systems with rich capabilities on both the client and server sides. This became possible after Node.js appeared in 2009. Unlike the Angular JS front-end framework, Node.js extends the capabilities of JavaScript to allow its use for work

on the server-side.

According to the 2019 StackOverflow survey, not only is JavaScript the most widely used language (67.8% out of 87,354 responses), but Node.js is the most popular framework (49.9% of 58,543 responses) ( Developer Survey Results, 2019). Such popularity lies in the proliferation of a versatile set of tools that can be used with Node.js.

### Concept

There is an ongoing debate around the term framework. Frameworks, by definition, are sets of conventions to interpret the code (Altexsoft, 2017). However, in simple terms, frameworks distort the original development environment (e. g. Node.js) to provide more than comfort to software engineers.

Convenience, in this case, has a very variable meaning depending on the length of a project, its complexity, its scalability and finally, engineering preferences. This is why all popular technologies, JavaScript, Node.js or others are generally surrounded by a lively community that continually suggests new frameworks for different use cases. Thus, Node.js frameworks have different tastes and generally from the point of view of technical, they are divided into four main groups.

### Framework types

#### Frameworks MVC

It is all in the name. These frameworks offer a great freedom of development, high flexibility without additions and are based on the ModelView-Controller design model.

#### Full-stack MVC frameworks

These instruments come with libraries, integrations, template engines, scaffolding and other add-ons that allow engineers to automate their work, but usually at the cost of flexibility (Altexsoft, 2017).

#### REST API frameworks

These tools streamline the development of the REST API server and can be used in combination with any other group of frameworks.

#### Other frameworks

There are many intermediate software, libraries and other specific solutions that can be considered as frameworks to some extent.

Popular Node.js frameworks

Here are the most popular Node.js server-side frameworks based on the number of Git stars they have. Frameworks develop the original Node.js or provide a light and flexible development with a limited number of features.

#### Express Js: the most popular and flexible middleware

IntroductionExpress is the most popular Node.js framework (the E stands for Express). MEAN is a complete set of technologies sufficient to cover the main software engineering tasks:

- MongoDB: database

- Express: back-end middleware

- Angular: front-end

- Node.js: runtime environment.

The term Middleware means that the framework provides tools to build the bridge between the server-side with a database and a client-side. Express is suggested as a lightweight, flexible and non-doped frame. It is not used because a software engineer is not limited to the use of cable development practices, so it offers much freedom. For example, Express does not dictate any of the design models (MVC, MVP, MVVM, etc.). This allows engineers to create applications in their own way. Yet, Express is commonly applied in the MVC model. Freedom also extends to database integration. While MEAN contains MongoDB, Express does not require any specific database integration (Express, 2019). Developers can choose the data storage technology they prefer and install the respective package as a database driver.

##### Express - The main features

* Freedom: Developers are only limited by their skills and preferences. This also guarantees a dual learning curve for newcomers (Express, 2019).
* High performance: Node.js is famous for its performance. Express only provides a thin layer without compromising performance (Express, 2019).
* Large community: Express is a technology that should be considered by default because it remains the common practice of most engineers (Express, 2019).

##### Express.js - Use case

If someone manage a small engineering team and his product is not likely to take quickly from the monstrosity to the company-wide scale, choose Express. It is the best choice for new projects that do not face the burden of inheritance. However, as the time comes, the project and team grow, the lack of standards can inadvertently lead to management of code.

#### Meteor: the all-in-one automated system

##### Introduction

Meteor is another popular JavaScript framework, which is located on the opposite pole of the globe Node.js. Unlike Express.js, Meteor takes the philosophy of standardization and out-off-the-box to the extreme. The package allows an engineer with sufficient integration tools to create server, mobile, web and software applications. Meteor integrates with Angular or React.JS to connect the Node.js back-end to front-end applications. It uses MongoDB as storage and can be integrated with Cordova to create hybrid applications using HTML, CSS, and JS via WebView. Although most parts of the battery can be exchanged, this packaging streamlines a large part of the groundwork.

In addition to that, Meteor is the stubborn MVC framework. Unlike non-pitted frameworks, it does not give engineers much freedom but ensures rapid development through automation.

##### Meteor: main characteristics

Magic of data synchronization: Meteor automatically activates synchronization bidirectional data transfer between a client application and a server (What is Meteor?, 2019). This is achieved by creating a copy of a mini-database on the client-side that contains a small piece of data to which a user has subscribed. Once a customer requests new data, the system connects them automatically to the server to retrieve the requested data from the database.

When changes are applied at one end, they are automatically and instantly reflected at the other end. Engineers do not need to configure the synchronization manually.

Ready-to-use integration: This may be the main quality of Meteor. It is an all-in-one solution and reduces much of the integration effort unless engineers choose technologies beyond the standard bundle (What is Meteor?, 2019).

Building automation: The engineers behind Meteor boast of having a system packaging that automates the packaging of the same code source across multiple platforms final, whether mobile, web or even software. For example, with the help of Cordova, a hybrid mobile development, developers can modify and package the HTML / JavaScript / CSS for iOS and Android without too much effort.

##### Meteor - Use cases

Being highly automated, Meteor encourages rapid prototyping and the creation of MVPs (minimum viable products). Developers quickly develop applications on several platforms, discover its appearance, update it iteratively, then access the initial comments. It is not that they cannot use it for large-scale applications of the company is that engineering teams will probably suffer from problems of long-term evolution. The creation of hybrid applications, for example, generally involves jeopardizes the user experience, as applications appear to be out of step and generally slow. Automatic data synchronization can also become a bottleneck once the user base reaches enterprise-wide loads.

#### Socket.io: The fastest and most reliable real-time engine

##### Introduction

Socket.io is a Node.js module that allows developers to create Web Sockets, i. e. bi-directional connections between clients and server that allow real-time communication on a protocol other than the HTTP protocol normally used in web pages (Nebra, 2018).

##### Socket.io: main characteristics

socket.io allows us to use WebSockets very easily. Since not all browsers support WebSocket, it can use other synchronous communication techniques if they are managed by the client's browser. Socket.io determines for each client what is the most suitable real-time communication method for the client (Nebra, 2018):

* WebSocket
* Adobe Flash Socket
* AJAX long polling
* AJAX multipart streaming
* Forever Iframe
* JSONP Polling

Thanks to all these different communication techniques, socket.io supports a very large number of browsers, even old ones

##### Socker.io - Use cases

Socket.io est utilisé pour construire rapidement des applications en temps réel. It can be used to create applications such as real-time communication systems, multiplayer games, collaboration applications, etc.

### Conclusion

The arrival of the Node.js execution platform has given life to a JavaScript frameworkization on the server-side. Indeed, many frameworks have been created and each one has its own particularity. They are born often as part of a specific project or use case. Indeed, the standard framework Node.js offers the possibility to extend its solution and to create several frameworks.

However, too many choices can turn our heads, so there are several steps to follow recommended in order to choose the framework adapted to specific needs.

First, it is necessary to evaluate the engineering team in order to identify technologies with which the team is in phase If someone want to start developing Node.js, it is better to keep to simple and commonly used frameworks that do not require long integration. Complete, standardized and complete solutions will certainly require more time.

Then, it is necessary to define the specificities of the project. Analyze how the functionalities of the framework correspond to the project. If we have a team of experienced engineers and we want to build a prototype or a fast MVC, it is recommended to opt for more standardized and full-stack solutions. For long-term development, it is better to stick to light and flexible instruments.

Before starting with a framework, it is necessary to check its research limits and the support of its community. Any software engineering tool only thrives if it is supported by a community powerful. Although there are good frameworks, the lack of peer support can become a problem.

Finally, it is also important to consider the maturity of a framework. Some tools are still in development. They may suffer from bugs or some features may be still in development. They must be evaluated to ensure that they have the full list of features that we need (Altexsoft, 2017).

## JavaScript: Is it a Solution?

### For which case is JavaScript used on the server-side?

#### Introduction

The choice of back-end technology is one of the most important decisions that every CEO makes and CTO are required to do. It determines how fast a product can be shipped on the What is the total cost and how important will pain maintenance be?

JavaScript has been one of the most popular client-side programming languages and a commonly used front-end web development tool. However, it has also gained ground in different application areas and on different platforms, such as React Native, Appcelerator Titanium, Apache Cordova / PhoneGap, NativeScript and Node.js, which is different from other commonly used JavaScript Frameworks (Netguru, 2017).

Node.js is an application runtime environment that allows developers to write server-side applications in Javascript. Thanks to its unique I/O model, it excels in the evolving and real-time situations (Netguru, 2017). It is also lightweight, efficient and allows us to use Javascript on both the frontend and backend, opening up new possibilities. It is not surprising that so many large companies have exploited it in production, namely Walmart, Netflix, Medium, LinkedIn or Groupon.

Internet of Things  
Since 2012, when the popularity of IoT (the Internet of Connected Things) increased dramatically, Node.js has become one of the preferred solutions for companies and organizations seeking to develop their private and public IoT systems (Badami, 2016). The most obvious advantage of Node.js as a back-end for such networks is its ability to handle several simultaneous requests and events from thousands or even millions of devices on the network. The avalanche of requests and data from IoT devices does not block Node.js servers thanks to their event-driven architecture and asynchronous processing adapted to heavy I/O operations on the IoT network (Badami, 2016). This makes Node.js fast as an application layer between these devices and the databases used to store the data from them. Besides, IoT developers working in data-intensive scenarios can take advantage of the low resource requirements of Node.js. Insufficient memory requirements allow easy integration of Node.js as software in single-card controllers such as Arduino, widely used for the construction of digital devices that constitute IoT systems. Finally, the Node community was one of the first to adopt IoT technology, create more than 80 packages for Arduino controllers and several packages for Pebble and Fitbit portable devices widely used in IoT systems.

Real-Time Chats  
Node.js provides all the basic features to create real-time chats of any complexity. In particular, Node has a powerful Event API that facilitates the creation of certain types of objects of ("transmitters") that periodically transmit events named "listened to". by event managers. With this feature, Node.js makes it easy to implement server-side events and push notifications widely used in messaging and other real-time applications (Infotech, 2019).

Node's event-based architecture also works well with the WebSockets protocol, which facilitates fast two-way message exchange between client and server via an open connection (Infotech, 2019). By installing WebSockets libraries on the server and client-side, we can implement real-time messaging that has overhead, lower latency and faster data transfer than most other, more conventional solutions (Infotech, 2019).

Node has excellent support for WebSockets via libraries such as socket.io, ws, or WebSocket-node, thanks to which we can easily deploy effective chats and applications in real-time (OPTASY, 2018). With socket.io, for example, all we have to do to create a basic live chat is to install the socket.io library on the server and client, create transmitters of events and broadcasters who will pass messages through the open WebSockets connection.

This basic functionality can be achieved with only a few lines of code.

Complex Single-Page Applications  
Node.js is an excellent choice for single-page applications (SPA) due to its efficient management of asynchronous calls and the heavy I/O workloads characteristic of these applications (OPTASY, 2018). The event loop of Node.js allows to "delay" several simultaneous customer requests, which ensures smooth transitions between views and continuous data updates. Besides, Node.js works well with data-driven SPAs, where the server acts as a back-end that provides data to the client while the client does all the HTML rendering.

Also, Node.js is good for SPAs because it uses the same language (JavaScript) as many popular JavaScript frameworks (Ember, Meteor, React, Angular) used in the construction of SPA. Node.js and browsers use JavaScript, there is less context change between them. Thus developers can use the same language data structures and modular approaches on both the server and the client-side. This results in a development faster and better maintenance of SPAs. The advantages of Node.js have been exploited by famous companies such as Netflix, Linkedin, and Medium, to name a few.

Real-Time Collaboration Tools  
As with real-time discussions, the asynchronous and event-driven architecture of Node is perfect for collaboration applications (OPTASY, 2018). On these applications, several events occur and input/output requests occur simultaneously. For example, several users can edit the same paragraph, comment, post messages and attach files. The changes to a content element can only be applied after a cascade of events where each step depends on the previous one.

The WebSockets and Event APIs of Node ensure that heavy input/output operations performed by many users do not block the server (OPTASY, 2018). Thus, all events and server-side data are returned to the client in time. By issuing push notifications on the client, Node.js will also instantly update the environment collaboration so that all users have a single and consistent representation of the application. It is precisely this the reason why the Trello project management application team (http://trello.com) uses the Node.js stack. Trello's engineering team thinks Node.js would be great for spreading instantly many updates and maintain many open connections, thanks to its event-driven and non-blocking architecture. Other collaborative applications include built on Node.js, we should also mention Yammer, a service of freemium social network facilitating private communication in companies.

Streaming apps  
Unlike remote server applications, in application streaming the program is run on the end user's local machine. Application streaming allows you to download parts of the application on demand without overloading the server and local computer. Initially, only some parts of the application needed for the bootstrap are downloaded, while the rest can be downloaded in the background if needed. When the application is completely downloaded, it can work without any internet connection. In the event that we want to save data in our account, the application can launch requests to the server. Similarly, server events can update the application without too much network traffic overhead.

Node.js is excellent for the development of such streaming applications thanks to its API a native of Stream. In particular, Node.js has a readable and writable flow interface that can be processed and monitored very effectively (Capan, 2019). The flow instances are essentially Unix pipes that allow transmitting parts of the executable code from the application to the local machine while maintaining an open connection for new components at download on demand. Flows allow users to channel each other and disseminate data directly to its final destination. As a bonus, the flows do not require caching and temporary data. All it takes is an open connection to broadcast application data from one place to another.

Microservices Architecture  
Node.js is an excellent solution for developing microservices and creating easy-to-use APIs to connect them. In particular, the Node.js repository offers the Express and Koa frameworks, which easily allows developers to set up several server instances for each micro-service and design routing addresses for them. Node.js with Express allows us to create very flexible modules responsible for certain parts of an application.

Besides, Node.js can be easily integrated with Docker and will allow us to encapsulate micro-services in sealed containers to avoid any conflict between the development of the application and the environments used in each of them. The use of Node.js for microservices also has some minor requirements. Node.js with the microservices architecture significantly reduces application deployment time and improves the efficiency, sustainability, and scalability of applications. The micro-service architecture also helps to effectively manage the division of labor in engineering teams, allowing them to work on specific tasks without affecting other parts of the application. These advantages have been successfully exploited by PayPal, the world's leading online payment system, which has been using Node.js to power its micro-service architecture since 2013. PayPal modularizes its application stack and divides the development process into several microservices. Thus, it organizes its teams to work more efficiently. PayPal was able to evolve Node.js so that several teams could work on the same project (Kadivar, 2018). The results of this were staggering.

The Node.js Paypal application was able to be built twice as fast and with less than human resources. The company has successfully reduced its code base and improved performance, where a single basic Node application could handle twice as much rps (requests per second) as 5 Java applications used by PayPal before (Kadivar, 2018).

In which case do not use Node.js?

Although Node.js is in many cases the ideal solution for building applications.

The first thing that usually comes to mind is heavy IT applications.

Node.js is based on a non-input/output blocking event model and uses only one CPU core. Heavy CPU operations will block incoming requests, making the biggest advantage of Node.js unnecessary. Therefore, if we are considering building some heavy CPU software, try a different, more appropriate technology that will give us the best result.

There are more things to consider before starting with Node.js. Many packages for the Node.js applications are available on the npm platform. The community is dynamic, the technology is maturing and the npm is the largest repository available at the moment. However, the quality of the packages varies. Sometimes we can have problems with packages supported only by individual users and not maintained correctly. For example, when connecting a node application to a base system of obscure or old data. Finally, the use of Node.js is unnecessary for HTML or CRUD (Create, Read, Update, Delete) simple in which you do not need a separate API and where all data comes directly from the server. The application may be slightly more scalable, but do not expect more traffic to the application just because it has used Node.js. In some cases, developers should stick to proven frameworks such as Ruby on Rails.

Ruby on Rails and Node can achieve the same results. However, there are different situations where each is more efficient than the other. With Rails, we can prototype and move quickly, break things and get a CRUD application in one hour.

Will JavaScript match our project of tomorrow?

Introduction  
Why have so many large groups chosen Node.js as their back-end technology? Here are the different comparisons between Node Js and other back-end technologies with the main advantages and disadvantages of these environments, but also some disadvantages that must be taken into account before making a choice. A bad decision can cost much money, so we have to choose wisely.

Node Js VS Ruby on Rails  
Node.js is more appropriate for dynamic applications with multiple server requests and frequent data mixing between client and server. What types of applications are they?

These are instant messaging applications such as chat and collaborative applications (drawing, video conferencing) collectively referred to as RTA (Real-Time Applications).

The event-based Node.js architecture is perfect for managing heavy I/O operations, server requests and data flows in these applications. For the same reason, Node.js is a preferred choice for single-page applications (SPAs) that involve heavy client-side processing, rendered and where the main function of the back end is to provide a REST API (Bohdan, 2019). Similarly, each time, the performance and evolution of web applications are a major concern when choosing a technology. Lightweight and fast Node.js surpasses the Rails framework which is an alternative to Node.js. This is why companies such as Linkedin have started using Node.js for performance and evolution reasons.

On the other hand, Ruby on Rails works better than Node.js in resource-intensive applications. Node.js is a mono thread environment that has not been designed for CPU intensive operations that handle graphics, images, and data. Doing calculations on large data tables can block all requests by input, making the main advantage of the Node.js useless. If we want to build a heavy application processor, Rails is certainly a better option than Node (Bohdan, 2019). Rails is also better when development time is critical. With all modules and generators available outside the box, Rails is very powerful in rapid application development (RAD) (Bohdan, 2019). Just in a few commands, we can have a fully functional prototype that can be modified with additional features later. Node.js can also provide generator scripts to accelerate development, but rapid prototyping is Rails' specialty.

So, when choosing between Node and Rails, we should take into account the time of development, performance, evolution of our application, as well as the type of case of use to which it applies. If there is a requirement for development time and if there are heavy CPU processing is necessary so it is better to focus on Rails. On the other hand, if we have requirements in terms of RTA, SPA and other heavy I/O solutions, go to Node.js.

Node Js VS PHP

IntroductionEven if PHP and Node.js can handle applications of any complexity, they are built around different concepts and architectures. If an application owner has to choose between these two environments, he should be aware of their main advantages and limitations.

Node.js and PHP are two very popular web development solutions. PHP, a scripting language created by Rasmus Lerdorf in 1994, was one of the best languages of the Web 1.0 era (Chrzanowska, 2017). The eloquent manifestations of PHP's success are CMS (Content Management Systems), such as WordPress, Joomla or Drupal, which feed millions of blogs and web portals. Node.js is a representative of a younger generation of web development. Unlike PHP, Node.js is not a language, but an execution environment that uses JavaScript for server-side application development. Launched in 2009, Node.js has demonstrated the power of JavaScript in the construction of event-based, data-driven applications, with a lot of I/O for the Web 2.0 era (Chrzanowska, 2017).

Comparison points**Mixing code and content -> PHP**

Open the magic PHP tags and start writing code in seconds. No need for models! No need for additional files or elaborate architectures, only a programmable logistic power at hand.

**Separation of concerns -> Node.js**

Programmers add structure and separate the cosmetic layer from the logical layer. The code becomes cleaner, easier to understand and easier to maintain. The frameworks running on Node.js are built by programmers who know that life is better when the model, view, and controller are separated (Wayner, 2019).

**Newness means more modern features -> Node.js**

Node.js plug-ins are not only newer than PHP plug-ins, but they have also been built in perfect knowledge of the latest architectural approaches.

**Simplicity (in a way) -> PHP**

There is not much with PHP: some basic variables and functions to juggle with the chains and numbers. It is a thin and simple layer.

**Dozens of language options -> Node.js**

Unlike PHP, Node.js can compile many major languages cross-referenced to work in JavaScript. There are popular languages such as Java, C# or Lisp and dozens of others like Scala, OCaml, and Haskell. There are even gifts for nostalgic lovers of the BASIC or Pascal.

**Service calls are thinner than PHP HTML calls -> Node.js**

Once the JavaScript code is in the browser's cache, the only thing that moves along of the wires is the new data. There are no repeated trips to download the entire page.

Only the data have changed. If we are willing to spend time creating an application web browser side, there is a significant advantage. Node.js is optimized to provide the data and only data via web services.

**The new code helps him to catch up -> PHP**

If a programmer wants to do more than interact with a database and format the results, he can now do more with PHP without shutting up. Facebook's HHVM adds support for Hack, a comprehensive language full of modern features such as type annotations, credits and lambda expressions. Using this limits the code to execute only on the HHVM, but it is not the worst thing in the world and it is pretty fast (Wayner, 2019).

**SQL -> PHP**

PHP was built to coexist with MySQL and its many variants. There are also other Oracle and Microsoft SQL databases. The code may change with some modifications to the requests. The vast SQL world does not stop at its borders. Some of the most stable and developed codes will interface with an SQL database, which means that all this power can also be easily integrated into a PHP project (Chrzanowska, 2017).

**JSON -> Node.js**

If we need to access SQL, Node.js has libraries for this. However, Node.js also speaks JSON to interact with many of the latest NoSQL databases. It does not mean that we cannot get JSON libraries for our PHP stack, but there is something fluid in the simplicity of working with JSON when using JavaScript because Let us remember that JSON means JavaScript Object Notation. It is a syntax of the browser to the webserver to the database.

**Development time -> PHP**

For most developers, writing PHP for web applications is faster: no compiler, no deployment, no JAR file or preprocessor, just an editor and some PHP files in a directory (Wayner, 2019). When it comes to creating a project together quickly, PHP is a good tool to use.

**Gross speed -> Node.js**

Writing JavaScript code is a little more difficult when we count the braces and parentheses, but when it is done, the Node.js code can fly. The return mechanism is great because it prevents us from juggling threads.

ConclusionBoth are back-end technologies, but Node.js can offer an advantage if we are looking to have a fully JavaScript technology stack that is both client and server. If we are trying to choose between backbone technologies or if we are building a complete stack of solutions, it is useful to go into a little more detail.

Developers should consider PHP if a project involves:

* Software stacks such as the LAMP stack (Linux, Apache, MySQL, PHP)
* CMS like WordPress, Drupal, or Joomla, etc.
* Servers such as MySQL, SQL, MariaDB, Oracle, Sybase, and Postgresql, etc.

Developers should consider Node.js if a project involves:

* Software batteries such as MEAN stack (MongoDB, Express.js, AngularJS, Node.js)
* Single Page Dynamic Applications (SPA)
* Frontend technologies such as jQuery, AngularJS, Backbone.js, Ember.js, ReactJS, etc.
* Server-side technologies such as Node.js, MongoDB, Express.js, etc.

Node Js VS Java EE

*Introduction*Java EE is the leader in enterprise applications. Since 2009, NodeJS has invaded the small world of web development and gradually makes its way into the reference platforms to build an application, site or service API. The old Java EE resists, however, and anyone who now wants to start a new project can be confronted with this problem: NodeJS or Java EE?

*Comparison points*

**A rock-solid base -> Java**Java has problems and bugs, but compared to other technologies, it is a very reliable language. It can take decades for the JavaScript team to write as many regression tests as Sun / Oracle developed to test the Java virtual machine (Sharma, 2017). When we start a JVM, we benefit from 20 years of experience with a reliable registrar committed to dominating the enterprise server.

**Ubiquity -> Node.js**Thanks to Node.js, JavaScript finds a home on both the server and client sides. The code we write for one side will most likely be executed in the same way on both sides (Sharma, 2017). It is much easier to stick with JavaScript for both client/server sides than to write something once in Java and again in JavaScript, which we would likely need to do if we decided to move business logic we wrote in Java for the server to the browser.

**Simple construction process -> Node.js**Sophisticated construction tools like Ant and Maven have revolutionized Java programming.

However, there is one problem. Specifications are written in XML, a data format that was not designed to support programming logic. Of course, it is relatively easy to express a branch with embedded tags, but it is annoying to switch from Java to XML to build something. With JavaScript, there is no need to switch.

**Remote debugging -> Java**Java has incredible tools to monitor machine clusters. The JVM has deep hooks and profiling tools to help identify errors. The Java enterprise stack runs some of the most sophisticated servers on the planet and the companies that use these servers have demanded the best in telemetry (Madasamy, 2018).

**Database queries -> Node.js**Queries for some of the most recent databases, such as CouchDB, are written in JavaScript. Mixing Node.js and CouchDB does not require any gear change and even less the need to remember syntax differences.

Meanwhile, many Java developers are using SQL. Even when they use Java DB (formerly Derby), a database written in Java for Java developers, they write their queries in SQL. They must write their database code in SQL, then let Derby analyze the SQL. It is an excellent language, but it is completely different and many development teams need different people to write SQL and Java (Pentechs, 2019).

**Libraries -> Java**There is a vast collection of libraries available in Java and they offer some of the most serious work. Text indexing tools like Lucene and computer vision tools like OpenCV are two examples of Open Source projects that are ready to be the basis of a serious project. There are many libraries written in JavaScript and some are incredible, but the depth and quality of the Java codebase are superior.

**JSON -> Node.js**When databases send responses, Java has difficulties to turn the results into Java objects.

Finally, the Java code retrieves the Java objects after the entire conversion.

Many web services and databases return data in JSON, a natural part of JavaScript. JSON is now so common and useful that many Java developers use the format, so several good JSON parsers are also available as Java libraries (Pentechs, 2019). However, JSON is part of the JavaScript foundation. We do not need libraries. Everything is there and ready to use.

**Solid engineering -> Java**This is a little difficult to quantify, but many complex packages for serious scientific work are written in Java because Java has a solid mathematical foundation (Madasamy, 2018). Sun spent a long time sweating the details of the utility classes and it shows. There are BigIntegers, elaborate I/O routines and complex date code with Gregorian and Julian calendar implementations (Sharma, 2017). JavaScript is suitable for simple tasks, but there is much confusion in the gut. An easy way to see this is in the three different results of JavaScript for functions that have no answers: undefined, NaN and null. Which one is right? Well, everyone has their role, one of them being to push programmers to try to keep them straight.

**Speed -> Node.js**People like to praise the speed of Node.js. The data comes in and the answers come out like lightning. Node.js does not disturb with the configuration of separate threads with all locking headache (Madasamy, 2018).There are no overhead costs to slow down anything. We write simple code and Node.js makes the right choice as quickly as possible.

This praise comes with a warning. The Node.js code must be simple and work better correctly. If interlocked, the entire server could crash. The developers of the operating system have removed their hair by creating safety nets that can withstand programming errors, but Node.js throws these nets away.

**Threads -> Java**The fast code is great, but it is usually more important that it is correct. Here is where the additional features of Java make sense.

Java's web servers are multithreaded. Creating multiple threads can take time and memory, but it pays off. If one thread gets blocked, the others continue. If one thread requires a more extended calculation, the other threads are not hungry (usually).

If a Node.js query runs too slowly, everything slows down. There is only one thread in Node.js and it will arrive at an event when it is good and ready. It may seem superfluous, but underneath it, it uses the same architecture as a one-stop post office the week before Christmas (Sharma, 2017).

There have been decades of work devoted to building intelligent operating systems that can juggle many different processes at the same time. Why go back in time to the sixties when computers could only manage one thread?

**Momentum -> Node.js**Yes, Java can follow, but there is old code everywhere. Of course, Java has new IO routines, but it also has old IO routines.

Many applet and tool classes can be annoying.

**The cross-compile between Java and Node.js -> Node.js & Java**The debate about using Java or Node.js on servers can and will continue for years to come. Unlike most debates, however, we can have it both ways. Java can be compiled in JavaScript. Google does this frequently with Google Web Toolkit and some of its most popular websites use Java code, Java that has been translated into JavaScript (Madasamy, 2018).

There is also a path in the other direction. JavaScript engines like Rhino run JavaScript in a Java application. If we are ambitious, we can directly create a link in Google's V8 engine.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technologies /**  **Features and functions** | **Node.js** | | **Java EE** | |
| **+** | **-** | **+** | **-** |
| **Environment and framework** | Easy library integration and operation | Reduced operating features | Many libraries and operating features | Dependency management and configuration of application servers |
| **Getting Started** | Ultra-modular software architecture | Need to be documented on each imported module | Frameworks with high abstraction layers | Difficulties related to the use of these frameworks |
| **Execution engine and performance** | Good performance, resistant to scalability | Some shadows on the management of partial compilation by V8 | Good performance | A decrease in speed or even crash of the server during scale-up |
| **Maintenance and evolution** | A very dynamic platform, many new features to come | The very average readability of the syntax is not helped by the absence of a proper EDI | IDE | Few new features to expect |
| **Security** | The possibility to fully customize the security strategy through lower-level management | The need to know the subject, or, if we choose to integrate pre-existing modules, the difficulty of setting them up, as in Java EE | Everything is already implemented and ready to be integrated | It is sometimes difficult to find ourself in the jungle of settings |

Table 2. Comparison of NodeJS and Java EE.

*Conclusion*The two platforms are quite different, and each has its own identity marked. However, it is not easy to decide between the two solutions.

In summary, Java EE is a very complete, reliable and proven platform, which offers facilities for development, but little flexibility. Its performance is good, to be qualified in case of an increase of charge. We will also note the complexity of the settings and the age of the technology, which leaves expect a few new features to come.

For its part, NodeJS is an innovative, dynamic, flexible and high-performance platform that allows a modular software architecture. Nevertheless, event development can be a sophisticated approach and some technical issues such as security may sometimes have to be addressed, managed at a low level. It is also sometimes difficult to choose a module for a particular feature, as well as to access documentation worthy of the name.

To conclude, we would say that Java EE is more suitable for large, highly focused projects with prospects for medium to high evolutions and a long maintenance cycle.

NodeJS will be perfect for smaller projects with more flexible project management, low or average development prospects and a medium to long maintenance cycle. He requires developers (or at least one architect) with a slightly higher technical level high.

Conclusion  
Will JavaScript on the server-side be as durable as Java and PHP? To answer this question, let us take a look at what we have just seen in the different parts. JavaScript on the server-side (Node.js) currently has the most important community of all languages. It has been growing since its creation in 2009, almost ten years ago. We can say that this technology is no longer in the category where developers are interested in it out of curiosity or to discover something new. Many articles, books, and videos allow everyone to try Node.js. The various actors of the web have already adopted this technology, there is no JavaScript IU that does not provide examples of scripting for Node.js. Many IDEs allow developers to develop on this platform and many hosting providers offer solutions to host applications. We were able to see the technical possibilities of Node.js, its asynchronous operation which allows it to be very efficient and fast. However, also the real-time that allows developers to design new concepts. Node.js is not just an idea, many companies have already taken the plunge and large groups like Groupon or PayPal have applied this technology to their products, with extremely positive feedback. This technology has not stopped at large companies, as many startups are now launching on Node.js. It is these startups who have already developed and popularized applications such as Trello or Voxer, innovative products in their field.

JavaScript technologies have undergone significant evolution in recent years. The convergence of new web technologies, NoSQL databases and the Node.js runtime environment have brought a significant breakthrough and have enabled JavaScript to extend its hold on the server-side.

However, if this convergence is beneficial to JavaScript and allows programming with only one language (Full-stack JavaScript), it would seem that this would impact other server languages such as the PHP, which is, therefore, trying to push its limits with Facebook's implementation of the whole thing of HHVM software. In terms of server JavaScript, this has made it possible to create new functionalities such as real-time with its socket.io module. Moreover, with the computerization of the world and the rapid growth of machines, especially the power of current processors, Node.js which is mono-threaded can exploit to the maximum a processor core to be more powerful.

All these crucial points show the power of JavaScript on the server-side. Its technology Node.js is implemented on many web projects, but technologies like Java and PHP remain more advantageous on specific use cases.

To conclude, I would say that Node.js is the most appropriate technology for some use cases and Java or PHP are appropriate for other use cases. This allows everyone to have their share of the cake, but the future is full of surprises. Who would have ever thought that JavaScript would be a direct competitor of Java and PHP? Therefore, it is essential not to draw too hasty conclusions because it would be very likely that JavaScript could compete with these languages on project frameworks that involve large-scale mathematical calculations where Java is dominant at the moment.

# References

*Developer Survey Results*. (2019). Retrieved from StackOverflow: https://insights.stackoverflow.com/survey/2019

Altexsoft. (2017, October 23). *Node.js Frameworks Comparison for Your Back-end Solution: Express.js, Meteor.js, Sails.js, and more*. Retrieved from Altexsoft: https://www.altexsoft.com/blog/engineering/node-js-frameworks-comparison-for-your-back-end-solution-express-js-meteor-js-sails-js-and-more/

Angular. (2019). *Features & Benefits*. Retrieved from angular.io: https://angular.io/features

Backbone.js. (2019). *Backbone.js*. Retrieved from backbonejs.org: https://backbonejs.org/

Badami, V. (2016, December 30). *Why JavaScript is used in IoT*. Retrieved from hackerearth.com: https://www.hackerearth.com/blog/developers/javascript-in-iot/

Bohdan. (2019, February 26). *Ruby On Rails Vs Node JS*. Retrieved from jelvix.com: https://jelvix.com/blog/ruby-on-rails-vs-node-js

Capan, T. (2019). *Why The Hell Would I Use Node.js? A Case-by-Case Tutorial*. Retrieved from toptal.com: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

Chrzanowska, N. (2017, September 7). *Node.js vs. PHP: Which Environment To Choose For Your Next Project?* Retrieved from Netguru: https://www.netguru.com/blog/nodejs-vs-php

*Express*. (2019). Retrieved from expressjs.com: https://expressjs.com/

Infotech, S. (2019). *6 Cool Things You Can Do With Node.JS*. Retrieved from Solace Infotech: https://solaceinfotech.com/blog/6-cool-things-you-can-do-with-node-js/

Kadivar, N. (2018, May 4). *How Netflix and Paypal did product transformation using Node.js?* Retrieved from hackernoon.com: https://hackernoon.com/how-netflix-and-paypal-did-product-transformation-using-node-js-22074e13caad

Madasamy. (2018, March 20). *Explanation and Detail about NodeJs & epic battle of Java and NodeJs*. Retrieved from Medium: https://medium.com/@madasamy/explain-and-detail-about-nodejs-epic-battle-of-an-java-vs-nodejs-c0f90f1f0d5f

MongoDB. (2019). *The MongoDB 4.2 Manual*. Retrieved from mongodb.com: https://docs.mongodb.com/manual/

Nebra, M. (2018, January 23). *Ultra fast applications using Node.js*. Retrieved from Openclassrooms: https://openclassrooms.com/en/courses/2504541-ultra-fast-applications-using-node-js/2505653-socket-io-let-s-go-to-real-time

Netguru. (2017, September 13). *Should you use Node.js in your application?* Retrieved from Netguru: https://www.netguru.co/hubfs/Downloadables/Should%20you%20use%20Node.js%20in%20your%20application.pdf

nodejs. (2019). *Overview of Blocking vs Non-Blocking*. Retrieved from nodejs.org: https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/

OPTASY. (2018, May 16). *What Is Node.js Used for? What Projects Can You Build Using It? 7 Best Use Cases*. Retrieved from medium.com: https://medium.com/@OPTASY.com/what-is-node-js-used-for-what-projects-can-you-build-using-it-7-best-use-cases-1d62cac5fb70

Pentechs. (2019, April 10). *Node.js vs. Java: An epic battle for developer mind share*. Retrieved from Pentechs: http://pentechs.com/java-vs-nodejs-an-epic-battle-for-developer-mindshare

Reactjs. (2019). *reactjs*. Retrieved from reactjs.org: https://reactjs.org/

Sharma, T. (2017, October 1). *Node js & Java Epic Battle*. Retrieved from Medium: https://medium.com/tkssharma/node-js-java-epic-battle-b1362ebb458b

*socket.io*. (2019). Retrieved from What Socket.IO is: https://socket.io/docs/

TekSlate. (2018). *MEAN Stack Interview Questions*. Retrieved from TekSlate: https://tekslate.com/interview-questions-on-mean-stack

VueJS. (2019). *What is Vue.js?* Retrieved from vuejs.org: https://vuejs.org/v2/guide/

Wayner, P. (2019, April 22). *Node.js vs. PHP: An epic battle for developer mindshare*. Retrieved from InfoWorld: https://www.infoworld.com/article/3166109/nodejs-vs-php-an-epic-battle-for-developer-mindshare.html?page=2

*What is Meteor?* (2019). Retrieved from meteor.com: https://docs.meteor.com/#/full/