A Comparison of Server Side Scripting Technologies

Tyler Crawford, Tauqeer Hussain*
tcraw850@live.kutztown.edu, hussain@kutztown.edu
Computer Science & Information Technology Department
Kutztown University of Pennsylvania
Kutztown, PA, U.S.A.

Abstract — In the presence of so many scripting languages or technologies available for server-side processing, it is usually a difficult decision for a beginner which technology to learn or use for development. In this paper, we have presented four leading server-side scripting technologies –PHP, Django, Ruby on Rails, and Node.js. We have identified five comparison attributes namely ease of getting started, availability of help and support, popularity, availability of development tools and packages, and perofrmance for the purpose of comparison. We have rated each technology based on these comparison attributes and provided our recommendations for the learners and developers. It is expected that the developers, software engineers, as well as the instructors teaching web development courses would benefit from this research.

Keywords — web technologies; Node.js; PHP; Django; Ruby on Rails

Type of submission: Regular Research Paper

I. Introduction

Today a number of scripting languages or technologies are available for server side processing and integration with the databases. These technologies have their advantages and disadvantages. This makes it difficult for developers to choose an appropriate server side environment for their projects development. The instructors teaching computer science or information technology courses in web engineering or web development areas are also confronted by similar problem, that is, which language they should adopt in their program curriculum to teach their students.

In this paper, we are providing a compariosn of four major server-side scripting technologies – Node.js, PHP, Django, and Ruby on Rails. For the purpose of this comparison, we have chosen five comparison attributes: 1) the ease of getting started with a particular technology, 2) the help and support available to the developers for that particular technology, 3) its popularity, 4) the development and package management tools available for that specific technology, and 5) its performance.

It is worth mentioning here that many other server side technologies are not included in this comparison because they are either not scripting languages or are frameworks. For example, two of the largest server side technologies, ASP.NET MVC and Java servlets including frameworks, were not included as they are not classified as a scripting language like JavaScript, PHP, or Ruby. Other languages excluded are Go which is a system language and not a scripting language, and Coldfusion which is a commercial platform by Adobe. Hack, created by Facebook, is sort of a dialect of PHP and was not included due to not being used as widely as PHP on the web. Finally, Vapor, an upcoming framework for the Swift programming language, was not included in this comparison as well due to the infancy of the technology which results in lack of information, lack of help and support. Swift itself is not a scripting lanuage either.

In the rest of this section, we have provided a brief history of each scripting language that we are comparing. In the following sections, we compare these languages with respect to each comarison attribute mentioned above. In the last section, we conclude this research by rating each teachnology, giving our recommnedation, and providing pointers to further work.

A. PHP

PHP's history dates back to when the internet was still young. PHP is actually a successor to a product name PHP/FI which was created in 1994 by Rasmus Lerdorf. Lerdor originally created PHP to track visits to his online resume and named the suite of scripts "Personal Home Page Tools". Development continued on to PHP 3, which most closely resembles PHP as it is today [53]. PHP is written primarily in C with some code in C++.

B. Django

Django was created in the fall of 2003 when programmers for the Lawrence Journal-World newspaper began using Python to build applications. The World Online team, which was responsible for production and maintenance of multiple new sitesm, thrived in a development environment that was dictated by journalism deadlines. Simon Willson and Adrian Holovaty developed a time-saving web development framework out of their necessity to maintain web applications under extreme deadlines. In the summer of 2005, the developers decided to release the framework as open source software and named it after the jazz guitarist Django Reinhardt. Since 2008, the newly formed Django Software Foundation continued maintaining Django [54].

C. Ruby on Rails

Unlike PHP, Django, (and Node.js), Ruby on Rails was not quite created out of necessity. Rather, it was pulled from David Heinmeier Hansson's work while working for the company Basecamp on their product Basecamp. The first open source release of Rails was in July 2004 with commit rights to the project following soon after. Rails uses the Model-View-Controller (MVC) programming pattern to organize application programming so that it is easier to work with.

D. Node.js

Node.js, was first introduced in May 2009 by Ryan Dahl. When looking at the prevailing server side programming

languages around 2009, Dahl felt that there was a problem with input and output (I/O), and the database processing at the back-end. Dahl observed that instead of waiting for the database to respond with a result set, that wastes millions of clock cycles, an application can proceed in non-blocking mode [52]. The goal of Node.js was therefore set to provide an event-driven, non-blocking I/O model that was lightweight and efficient [4]. It allows the application to proceed its execution without wasting clock cycles thus making the application run faster. Node.js is primarily written in C, C++, and JavaScript and touts that it is a JavaScript runtime built on Google Chrome's V8 JavaScript engine.

II. Getting Started

In many programming languages, the starter program is to say "Hello, World!". For comparison sake, we are doing the same thing in each of the four server-side technology sections given below. Though syntax for many other things can be compared, considering loops as an essential part of every scripting and programming language, a simple for loop is also coded to show that for primitive statements the syntax in the four technologies is quite comparable.

A. Node.js

To get started with Node.js, a simple installer may be used to obtain the technology locally in order to start developing [4]. The installation typically includes the node package manager as well any command line tools to get started with local server development. However, the installation package does not include any database management tools, but rather is left to the developer to choose freely which database system would be necessary to create their desired product.

The code for creating a "Hello, World!" project in Node.js can be seen in Figure 1.

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World!\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at
http://${hostname}:${port}/`);
});
```

Fig. 1. Node.js "Hello, World!" server code [48].

Node.js relies heavily on the use of packages from the Node Package Manager (npm) to complete tasks. The first line of code requires that an http package be used to create a server. The next lines specify the hostname and port number. In this case, it is the localhost address and port 3000. To create a server, the createServer function is utilized where the response status code, the header, and the data are set. Finally, the server listens on the port and specified host.

The for loop in Figure 2 should be quite familiar in that it is very similar to a for loop in many other languages. It is important to note however, that JavaScript is a dynamically typed language. This means that *var* can be used instead of explicitly defining a variables type but rather its type is inferred.

```
for (var i = 0; i < n; i++) {
    // code
}
```

Fig. 2. For loop in Node.js/JavaScript

B. PHP

To get started with developing and testing PHP code locally, a local server environment is desired. Such tools to assist in creating this environment include XAMPP, MAMP, or WampServer [1-3]. Each of these are quite simple to install and get started on Linux, MacOS, or Windows, but not all technologies support each platform. Typically, the aforementioned technologies include a database management system that interacts well with the PHP server tools.

The code for creating a PHP "Hello, World!" project is quite simple and involves only a few lines of code. The code, seen in Figure 3, simply echos "Hello, World!" onto the page and is used inside PHP tags. This is similar to bash scripting when printing data to a document or to the console.

```
<?php
    echo 'Hello, World!';
?>
```

Fig. 3. PHP "Hello, World!" code

The code for a PHP for loop shown in Figure 4 is similar to Node.js. The variables however are prefixed with a *$* symbol which is typical when programming in PHP.

```
<?php
    for ($i = 0; $i < $n; $i++) {
        # code
    }
?>
```

Fig. 4. For loop in PHP

C. Django

Getting started with Django is quite simple. The first step is to install python if it is not already installed, and then use the python package manager, *pip*, to install Django [5]. The development version of Django can also be installed by cloning the repository from Github. Django does not provide any sort of database management system and is left to the discrection of the developer creating a project.

The code for creating a Django "Hello, World!" project is simple as well, but it is important to modify the file controlling the server settings to be able to properly develop locally. The project code shown in Figure 5 defines a function called index

that takes a request parameter and returns an HttpResponse with the "Hello, World!" text as the parameter.

```
def index(request):
    return HttpResponse("Hello, World!")
```

Fig. 5. Primary code responsible for creating "Hello, World!" in Django [49]

Django fortunately uses Python to make creating a for loop quite simple. The code, found in Figure 6, creates a for loop by using a range of values between *0* and *n*.

```
for i in range(0, n):
    # code
```

Fig. 6. For loop in Django/Python

D. Ruby on Rails

Finally, getting started with Ruby on Rails has proven to be more difficult compared to that of the other technologies. The first step in getting started with Ruby on Rails, or Rails as it is widely known, is to ensure that the latest compatible version of Ruby is installed for Rails. For Windows this involves an installer to install Ruby. For Linux and macOS, a Ruby version manager is necessary to manage the versions of Ruby installed. This task can become more complicated if the developer is unfamiliar with version management tools or lacks some knowledge of how command line tools typically behave. Once ensuring that the latest compatible verison of Ruby is installed, the *gem* package manager that comes with Ruby may be used to install the Rails package. The Install Rails website provides clear instructions of how to install Rails on each platform and ensures that Ruby is installed as well [6].

The "Hello, World!" project code gets accepted a little differently with Rails due to a number of files required to start developing and running a local server. No code will be shown for Rails, but rather the process to get a project started to see the "Yay! You're on Rails" webpage will be given instead. The commands for creating the project will be shown however for the sake of being able to replicate this application. The first necessary step is to create a new rails application via the command line tools, then change directories into the new folder created for the application, and finally starting the server. The commands for doing so may be found in Figure 7.

```
$ rails new <app name>
$ cd <app name>
$ rails server
```

Fig. 7. Commands used to create a new Rails application

Finally, Rails utilizes the Ruby language to create a for loop in a range of values similar to that of Django. The differences is that in Django, a range function is called, versus the two periods used between the lower and upper bounds of the range in Rails. This difference can be quickly identified in Figure 8.

```
for i in 0..n
    # code
end
```

Fig. 8. For loop in Rails/Ruby

E. Comparison on "Getting Started"

It can be easily observed that getting started with PHP is the simplest. Upon setting up a local server environment, it takes minimal effort to create a new PHP file and view it on the local server. Node.js on the other hand proves to contain more difficult syntax that may not be quite clear to a beginner. Django continues the trend of difficulty proving to be a bit more complicated to get started with. While writing "Hello, World!" in a function may not be difficult, setting up the initial URL patterns, hostname, and port number creates extra steps that can be confusing and and hard to understand. Finally, Rails may be difficut to set up but makes up for it in the ease of creating a new project and running this on a local server. Installing Ruby and Rails can be a daunting task if one is unfamiliar with how the technologies work, but creating a new project and running it is quite straight forward.

III. Help And Support

For each of the following sections, the age, guides available to get started, help, and documentation provided are discussed regarding each technology.

A. PHP

PHP first appeared in June of 1995 which makes it 21 years old and is the oldest of all technologies discussed in the document. When navigating to the PHP website, a developer should notice that getting started guides are avaible from those that develop the language making it quite clear and simple how to get started writing a basic PHP web application. Due to its age, there are a number of posts on the internet related to the language which makes it easy to get help.

B. Django

Django, the second oldest technology, was founded on July 21, 2005, currently making it 11 years old. Django provides well documented starter guides on their site for developers begin learning the framework. While Django is used by less than 0.1% of the internet, getting help with it should not be too difficult due to the documentation and quick start guides on their site. Aside from this, it is important to note that Django is written in Python, which is widely used, making it easier to obtain specific help related to the Python lanugage itself.

C. Ruby on Rails

Ruby on Rails, or Rails, was founded on December 18, 2005, which currently makes it 10 years old. RailsGuides provides numerous guides to working with Rails and getting started as a new developer to the framework [11].

D. Node.js

Finally, the newest of technologies, Node.js, was founded on May 27, 2009 which currently makes it 7 years old. Node.js provides a minimal usage and example guide on the site which simply shows how to get started but provides no further examples or guides to continue after replicating the "Hello, World!" project. There is however API documenation for the long-term status (LTS) and newest version of Node.js.

E. Comparison on "Help and Support"

It can be concluded that PHP is probably the easiest technology to get started with given the guides, documentation, and other help available. This is partially due to the age of the technology compared to the other three. PHP has ten years more of data compared to the next oldest technology, Django. While age and popularity alone does not necessarily mean that a technology is easier to get started with, it helps due to it being more prevalent and therefore more people have knowledge of the technology and are able to assist when help is needed. Django on the other hand is used in only a small portion of websites on the internet, but it provides many guides to get started as well as API documentation. Following closely behind Django is Rails which has many guides available on web via RailsGuides making it a suitable choice if desired [11]. Finally, the latest technology, Node.js – though it is gaining popularity gradually, finding well documented starter guides has been more difficult than desired. The Node.js website provides a usage and example guide in the API but provides no further guides to getting started with Node.js. It seems that outside sources will be absolutely necessary to continue further if a developer is just getting started with the technology.

IV. Popularity

The following section discusses the popularity of each technology in in terms of its use on the web today. However, this may not correctly reflect popularity amongst developers. Trending popularity is derived from the number of packages that are currently available in each package management system as it shows a general interest level of developers to a specific technology.

A. Node.js

Based on statistics from W3Techs, Node.js is currently used by approximately 0.24% of all websites whose server they know [13]. Figure 9 below shows the trend of websites using Node.js from November 2015 to November 2016.

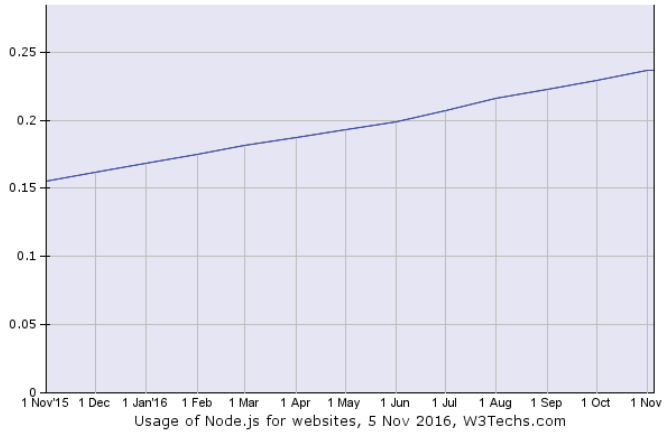

Fig. 9. Usage statistics from W3Techs of Node.js for Nov '15 – Nov '16 [13]

B. PHP

Based on the statistics from the site BuiltWith, and as of November 2016, 11.7% of the entire internet uses PHP [8]. Figure 10 shows the trend of websites using PHP from November 2015 to November 2016. It is unclear why there is a sudden drop in data. One conclusion could be that the application used to find the number of websites using PHP had a bug during this time. As of November 2016, approximately 42 millions websites use PHP.

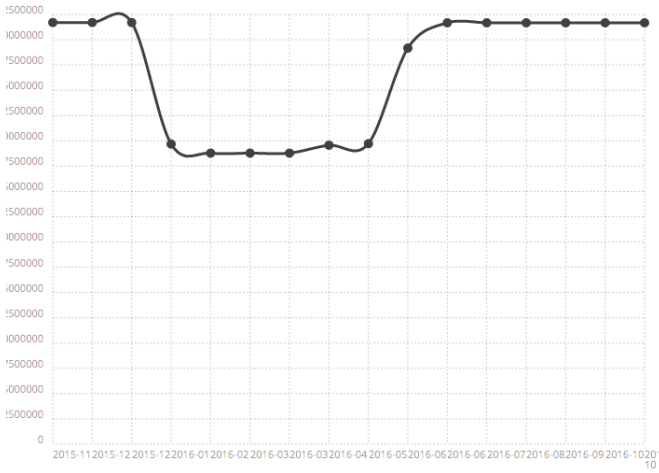

Fig. 10. Line graph of websites using PHP from Nov '15 to Nov '16 [8]

C. Django

Based on statistics from W3Techs and BuiltWith, Django is used by less than 0.1% of the internet [9, 10]. As of November 2016, Django is used on approximately 5,200 websites which is very small compared to that of PHP. Figure 11 below shows the trend of websites using Django for November 2015 – November 2016.

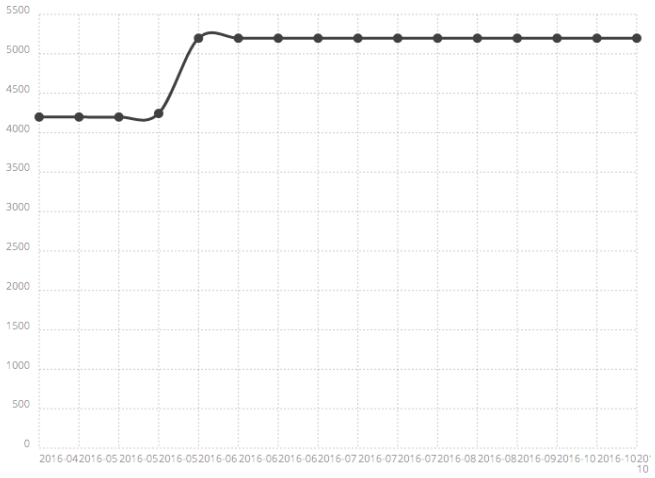

Fig. 11. Line graph of websites using Django from Nov '15 to Nov '16 [10]

D. Ruby on Rails

Currently, based on statistics from BuiltWith, approximately 0.3% of the entire internet uses Rails [12] as shown in Figure 12. Once again, it is unclear what caused the dip in data for a few months over the year long period. Nonetheless, there are approximately 1.09 million website using Rails as of November 2016 based on statistics from BuiltWith [12].

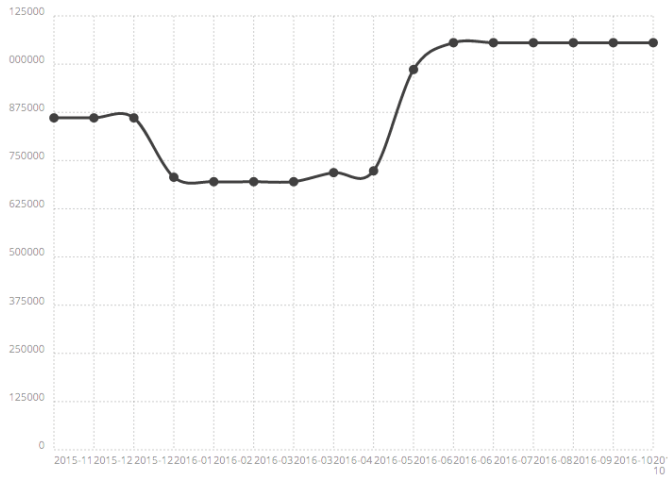

Fig. 12. Line graph of websites using Rails from Nov '15 to Nov '16 [12]

V. Development Tools and Package Management Systems

Each of the technologies discussed in this paper has development tools and package management systems that support the corresponding languages. The following subsections present the number of packages each package management system currently has. The numbers obtained for each of these package managers are noted at a website called module counts [14].

A. Node.js

As stated prior in this document, npm is the package manager used for Node.js projects and is bundled with Node.js on installation. Node.js relies on npm applications to get tasks done when creating projects. When installing Node.js a developer will receive a Node.js JavaScript runtime which allows the developer to run Node.js applications. In order to start programming specific applications, like a web server, a package will be necessary. By not bundling everything desired into the technology, it helps keep each application lightweight. Section 6 discusses a number of the environments that Node.js can run on. If Node.js were to include everything necessary to run the application in all environments, this would make Node.js very slow and undesirable to use. It is quite necessary to say, that in order to run any application using Node.js, npm is mandatory to use. If it is not used, the application likely becomes a simple JavaScript application instead. Npm packages can be browsed via the npmjs website where approximately 345,000 packages currently live [14, 15]. Packages can be installed via the command line tools on a per project basis and documentation for each package is typically displayed on the package's page on the npmjs site.

B. Ruby on Rails

RubyGems is the package manager used for Ruby and Ruby on Rails and is provided with the installation of Ruby. RubyGem packages can be browsed via the RubyGems website where approximately 125,000 packages are currently listed [14, 16]. Packages for Ruby and Rails can be installed via command line. Typically documentation for the packages is provided under the "Links" section and is shown on an external site.

C. PHP

PHP utilizes a technology called Composer as its package management system which must be installed separately from PHP [17]. Packagist is the site used to browse or search packages to use with Composer which currently contains approximately 115,000 packages [14, 18]. Documentation is typicallyed linked to on each of the packages page to assist in getting started. Composer follows similarly to the rest of the package managers in that in order to install a package, the command line tool that comes with Composer is utilized.

D. Django

Django uses the package manager that comes with Python called PyPI, or Python Package Index. Packages, installed via command line, can be searched using the PyPI site, which is an extension of the main Python website where approximately 90,000 packages are currently listed [14, 19]. Documentation for each of the packages is either given on the package page, link to, or not shown on the package page. This can cuase problems when desiring to use a tool but no information is clearly given on the package page. This requires a developer to seek other resources all together or to search the internet for the documentation to the package.

E. Comparison on "Development Tools and PM Systems"

Based upon these findings, Node.js provides almost three times the amount of packages compared to that of the next competitor, RubyGems, which provides some indication that there are more developers working with, and interested in Node.js. This number could be skewed due to Node.js lacking features that other technologies have and therefore need packages to make up for the missing abilities. This should not warrant over 200,000 packages worth of features missing from Node.js however. Node.js also provides a section on each package page for a developer to include documentation for the package if it is able to be shown on a single page. RubyGems and Composer both prove that they are a good package manager overall and provide a large number of packages to developers with well rounded package pages. The packages listed on Python's package manager website, PyPI, on the other hand, seems to lack documentation occasionally, as well as a good interface for browsing and discovering new packages. Overall, PyPI is usable but lacks in information and friendliness.

VI. Integration with Databases and Drivers

In the following sections, integration with the top five databases is studied for each of the technologies compared in this paper. The top five databases are being pulled from DB-Engines and are as follows: Oracle, MySQL, Microsoft SQL Server, MongoDB, and PostgreSQL [22-27].

A. Node.js

Node.js does not have support upon installation for any database. It does, however, inherit support from packages. All of the top five databases are supported by at least one or more package. The following packages are a few examples that provide support: oracledb for Oracle, mysql for MySQL, mssql

for Microsoft SQL Server, mongodb for MongoDB, and pg for PostgreSQL [28-32].

B. PHP

PHP appears to integrate with MySQL, Oracle, PostgreSQL, and MongoDB upon installation. As of PHP 7.x.x, it appears that support for Microsoft SQL Server has been deprecated, but support can be added via the SQL Server Driver on Microsoft's website [33].

C. Django

Django relies on support added by drivers to the Python language. Support is provided for each of the top five database systems but may not be clear how to add or utilize and may be hard to find. Examples of packages or drivers that support each of the top five databases include the follwing: psycopg2 for PostgreSQL, cx_Oracle for Oracle, Django MongoDB Engine for MongoDB, pymssql for Microsoft SQL Server, and MySQL Python Connector for MySQL [34-38].

D. Ruby on Rails

Rails inherits database support primarily from RubyGem packages as support for databases is not added to the Ruby language. A few examples of RubyGem packages that provide support for the top five databases are MySQL2 for MySQL, pg for PostgreSQL, mongoid for MongoDB, activerecord-sqlserver-adapter for Microsoft SQL Server, and activerecord-oracle_enhanced-adapter for Oracle [39-43].

E. Comparison on "Integration with Databases"

After studying how each database can be connected to with each technology, it is evident that PHP has the most support upon initial installation and may take some work to utilize Microsoft SQL Server with the newest version of PHP. While Node.js does not support database connections by default, it is made up for in the packages that the community provides with well rounded documentation for each. Ruby and Rails inherit support via packages as well but falls just slightly behind Node.js in terms of community support and documentation. Finally, support for connecting to a database using Python for Django seems to be less intuitive and requires research to understand how to install and utilize the drivers added.

VII. Performance

The following section is included to discuss performance differences among the four technologies. However, there is not any significant data or information available in literature or on web to compare the performance of these technologies. The only source that we could find is [50] that compares requests per second for plaintext, JSON, and SQLite on Vapor (Swift), Ruby on Rails (Ruby), Laravel (PHP), Lumen (PHP), Express (JavaScript), Django (Python), Flask (Python), Spring (Java), Nancy (C#), and Go. Of these technologies compared, the important ones for this document are Express (which is a Node.js web application framework), Laravel and Lumen, Ruby on Rails, and Django. Figures 13–15 show the results found in the comparison. Of the technologies being compared, it is evident that Node.js is the clear winner in each test run with Django coming in second and finally Rails and PHP sitting in the last three positions of the bar graphs. One can infer that Node.js has the ability to process more requests per second due to the event-driven, non-blocking I/O strategy. Due to this strategy, fewer clock cycles are wasted and also the program does not wait on another process resulting in significantly improved performance.

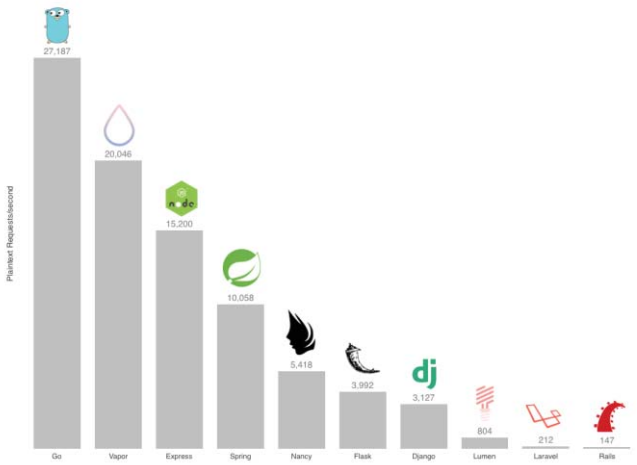

Fig. 13. Plaintext Requests/second [50]

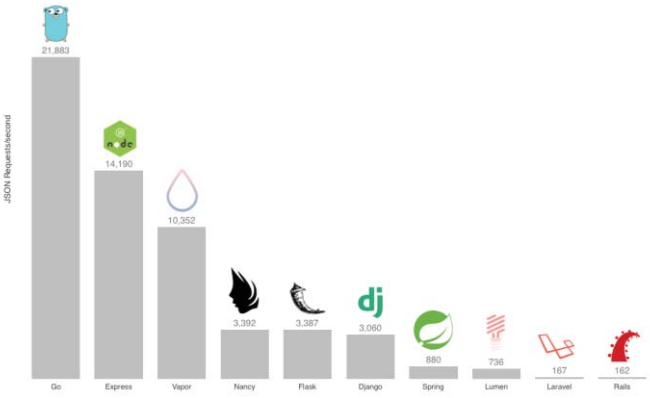

Fig. 14. JSON Requests/second [50]

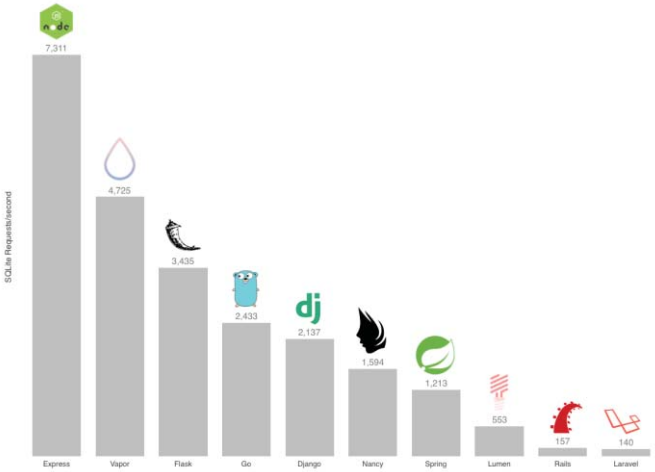

Fig. 15. SQLite Requests/second [50]

VIII. Conclusion And Future Work

In this paper, we have compared four web-based server-side scripting technologies: PHP, Django, Ruby on Rails and Node.js based on five comparison attributes – ease of getting started, availability of help and support, popularity, availability of development tools and packages, and perofrmance. To summarize and make it convenient for the reader, we have rated in the following table (Table 1) each technology for each comparison attribute on a qualitative scale of excellent, very good, good, fair, and poor. Though these ratings are subjective, they serve the purpose of providing a quick comparison.

Table 1: Qualitative comparison of Node.js, PHP, Django, and Rails

	Node.js	**PHP**	**Django**	**Rails**
Getting Started	Very good	Excellent	Very good	Good
Help And Support	Good	Excellent	Excellent	Excellent
Popularity	Very Good	Excellent	Fair	Very Good
Development Tools and Package Management Systems	Excellent	Good	Fair	Very Good
Environments	Excellent	Poor	Good	Good
Integrations with Databases	Excellent	Very Good	Fair	Very Good
Performance	Excellent	Fair	Very Good	Fair

Based on the above ratings, our overall recommendation is Node.js. Node.js is a very fast technology that allows for event-driven, non-blocking I/O giving it an extra boost compared to the other technologies. It is also a very popular platform amongst developers when comparing the amount of packages available in npm to other package management systems. It can also run virtually everywhere you want to run it. Aside from being able to run on a server, Node.js can be used to create console applications, run on the desktop with the help of Electron, in the browser, and embedded systems. The integration with five of the most popular databases through packages makes it easy to continue using whichever database system you desire. The documentation provided by these popular packages is usually very good which makes getting started with them even easier. The limitation of Node.js however, is its learning curve. Developers are required to find other resources to get started and they may not be readily available. PHP continues to be the dominator of the internet primarily due to its old age and ease of getting started as well as the help and support available on the internet. It is limited in the number of packages available, the environments it can run on, and its overall performance.

For future work, we intend to run experiments that can help evaluate performance of each technology. Also, more comparison attributes can be studied, for instance, the native environment supported by each technology, their hosting and deployment, and support for complex programming structures.

References

[1] "MAMP & MAMP PRO." *MAMP & MAMP PRO*. Appsolute GmbH, 2016. Web. 25 Oct. 2016.

[2] Bourdon, Romain. "WampServer." *WampServer*. N.p., n.d. Web. 25 Oct. 2016.

[3] "XAMPP Installers and Downloads for Apache Friends." *Apache Friends*. Apache Friends, 2016. Web. 25 Oct. 2016.

[4] "Node.js." *Node.js*. Node.js Foundation, 2016. Web. 25 Oct. 2016.

[5] "Download Django." *Django*. Django Software Foundation, 2016. Web. 25 Oct. 2016.

[6] "Install Rails: A Step-by-step Guide." *Install Rails*. One Month, Inc., 2016. Web. 25 Oct. 2016.

[7] N/A

[8] "PHP Usage Statistics." *BuiltWith*. BuiltWith Pty Ltd, 2016. Web. 25 Oct. 2016.

[9] "Usage Statistics and Market Share of Django CMS for Websites." *Web Technology Surveys*. Q-Success, n.d. Web. 25 Oct. 2016.

[10] "Django Language Usage Statistics." *BuiltWith*. BuiltWith Pty Ltd, 2016. Web. 25 Oct. 2016.

[11] "Ruby on Rails Guides (v5.0.0.1)." *Ruby on Rails Guides*. N.p., n.d. Web. 25 Oct. 2016.

[12] "Ruby on Rails Usage Statistics." *BuiltWith*. BuiltWith Pty Ltd, 2016. Web. 25 Oct. 2016.

[13] "Usage Statistics and Market Share of Node.js for Websites." *Web Technology Surveys*. Q-Success, n.d. Web. 25 Oct. 2016.

[14] DeBill, Erik. "Module Counts." *Modulecounts*. N.p., n.d. Web. 26 Oct. 2016.

[15] "Build Amazing Things." *Npm*. Npm, Inc, n.d. Web. 26 Oct. 2016.

[16] "Find, Install, and Publish RubyGems." *RubyGems*. RubyGems, n.d. Web. 26 Oct. 2016.

[17] "Dependency Manager for PHP." *Composer*. N.p., n.d. Web. 31 Oct. 2016.

[18] Boggiano, Jordi. "Packagist The PHP Package Repository." *Packagist*. N.p., n.d. Web. 31 Oct. 2016.

[19] "PyPI - the Python Package Index." *PyPI - the Python Package Index*. Python Software Foundation, 2016. Web. 31 Oct. 2016.

[20] "Electron." *Electron*. Github, n.d. Web. 31 Oct. 2016.

[21] Prévost, Rémi, Mike McQuaid, and Danielle Lalonde. "Homebrew." *Homebrew*. N.p., n.d. Web. 31 Oct. 2016.

[22] "DB-Engines Ranking." *DB-Engines*. Solid IT Gmbh, 2016. Web. 31 Oct. 2016.

[23] "Oracle Database." *Oracle*. Oracle, n.d. Web. 31 Oct. 2016.

[24] "MySQL." *MySQL*. Oracle Corporation, 2016. Web. 31 Oct. 2016.

[25] "SQL Server 2016." *Microsoft*. Microsoft, 2016. Web. 31 Oct. 2016.

[26] "MongoDB." *MongoDB*. MongoDB, Inc., 2016. Web. 31 Oct. 2016.

[27] "PostgreSQL." *PostgreSQL: The World's Most Advanced Open Source Database*. The PostgreSQL Global Development Group, 2016. Web. 31 Oct. 2016.

[28] "Oracledb." *Npm*. Npm, Inc, 2016. Web. 01 Nov. 2016.

[29] "Mysql." *Npm*. Npm, Inc, 2016. Web. 01 Nov. 2016.

[30] "Mssql." *Npm*. Npm, Inc, 2016. Web. 01 Nov. 2016.

[31] "Mongodb." *Npm*. Npm, Inc, 2016. Web. 01 Nov. 2016.

[32] "Pg." *Npm*. Npm, Inc, 2016. Web. 01 Nov. 2016.

[33] "Microsoft | Developer Network." *System Requirements for the PHP SQL Driver*. Microsoft, 2016. Web. 01 Nov. 2016.

[34] Varrazzo, Daniele. "PostgreSQL + Python | Psycopg." *Psycopg*. N.p., 2014. Web. 01 Nov. 2016.

[35] "Cx_Oracle." *Cx_Oracle*. N.p., n.d. Web. 01 Nov. 2016.

[36] "Django MongoDB Engine." *Django MongoDB Engine*. N.p., 21 May 2016. Web. 01 Nov. 2016.

[37] "Pymssql." *Pymssql*. Pymssql Developers, 2016. Web. 01 Nov. 2016.

[38] "Download Connector/Python." *MySQL*. Oracle Corporation, 2016. Web. 01 Nov. 2016.

[39] "Mysql2." *RubyGems*. RubyGems, n.d. Web. 01 Nov. 2016.

[40] "Pg." *RubyGems*. RubyGems, n.d. Web. 01 Nov. 2016.

[41] "Mongoid." *Mongoid*. RubyGems, n.d. Web. 01 Nov. 2016.

[42] "Activerecord-sqlserver-adapter." *RubyGems*. RubyGems, n.d. Web. 01 Nov. 2016.

[43] "Activerecord-oracle_enhanced-adapter." *RubyGems*. RubyGems, n.d. Web. 01 Nov. 2016.

[44] "DigitalOcean: Cloud Computing Designed for Developers." *DigitalOcean*. DigitalOcean Inc., 2016. Web. 01 Nov. 2016.

[45] "Deploy Pre-built Applications." *DigitalOcean*. DigitalOcean Inc., 2016. Web. 01 Nov. 2016.

[46] "Cloud Application Platform | Heroku." *Heroku*. Salesforce, 2016. Web. 01 Nov. 2016.

[47] "Amazon Web Services (AWS) - Cloud Computing Services." *Amazon Web Services*. Amazon Web Services, Inc., 2016. Web. 01 Nov. 2016.

[48] "Usage | Node.js V7.2.0 Documentation." *Node.js V7.2.0 Documentation*. Node.js Foundation, 2016. Web. 28 Nov. 2016.

[49] "Writing Your First Django App, Part 1." *Django*. Django Software Foundation, 2016. Web. 28 Nov. 2016.

[50] Harrison Shaun, Paul Wilson, Willie Abrams, Jonathan Channon, Chris Johnson, and Sven Schmidt. "Server Side Swift vs. The Other Guys - 2: Speed." *Medium*. N.p., 13 June 2016. Web. 28 Nov. 2016.

[51] Anonymous. "Perl Python Ruby PHP C C++ Lua Tcl Javascript and Java Benchmark/comparison." *Onlyjob*. N.p., 08 Mar. 2011. Web. 28 Nov. 2016.

[52] Stri8ted. "Ryan Dahl: Original Node.js Presentation." *YouTube*. YouTube, 08 June 2012. Web. 28 Nov. 2016.

[53] "PHP: History of PHP - Manual." *PHP*. The PHP Group, n.d. Web. 28 Nov. 2016.

[54] Holovaty, Adrian, and Jacob Kaplan-Moss. "Introducing Django." *The Django Book*. N.p., Nov. 2008. Web. 28 Nov. 2016.