University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

Evaluation of common front-end JavaScript frameworks

Autodesk
210 King Street East
Toronto, Ontario, Canada

Prepared by
David James Lee
ID 20516557
userid dj6lee
3A Computer Engineering

8 January 2017

110 Cleveland Street
Toronto, Ontario, Canada
M4S 2W5

8 January 2017

Vincent Gaudet, chair
Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario
N2L 3G1

Dear Sir:

This report, entitled ‘‘Evaluation of common front-end JavaScript frameworks’’ was prepared as my 3A Work Report for the University of Waterloo. This report is in fulfillment of the course WKRPT 401. The purpose of this report is to investigate the current front-end JavaScript frameworks available and assess their difficulty and usefulness.

Autodesk, is a multinational software corporation that produces many software products designed for architecture, engineering, construction, media and entertainment industries. Autodesk products include AutoCad, 3ds Max, Maya, Revit, BIM 360, Pixlr, SketchBook and many others.

The team I was on was the Lynx application framework team, who were responsible for developing an application framework for future projects as well as converting existing applications to use the framework. The goal is to have multiple products, all using a unified design and underlying framework. This report was written for my Manager and team, as a assessment on front-end JavaScript frameworks in regards to the needs of our project.

I would like to thank my mentor John Tan, as well as all of the other members of the Application team. They were all very open and friendly whenever I needed help and allowed me to start doing meaningful work right away. I hereby confirm that I have received no further help other than what is mentioned above in writing this report. I also confirm this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

David James Lee
ID 20516557

Contributions

The team I worked with was relatively small. The Lynx application framework team consisted of 10 people and was part of the larger Autodesk product development group, otherwise known as PDG. PDG is a relatively large group consisting of hundreds of people at various locations all around the globe. The Lynx application framework team, worked with other teams in PDG located in California as well as Krakow, Poland.

The team's main goals were to flush out the up and coming application framework known as Lynx, by testing the boundaries and overcoming the potential shortcomings. Being a new internal framework, Lynx was not fully explored and the Lynx team was testing the limits through sample applications as well as adapting current projects. The team was responsible for identifying potential faults in the current framework and overcoming them. This could be adding additional functionality to the core code base to perform the required task or deciding on a proper practice to circumvent the issue. The team was tasked with evolving the framework to be everything it was hoped to be.

My tasks consisted of working on sample applications as well as adding functionality and fixing bugs in the Lynx application framework. The Lynx application framework team created and worked on several sample applications. The goal of the sample applications was to provide a relatively realistic use case of the framework to both confirm that the desired functionality was present and establish best practices such as how applications should be organized. One example of a task I had, was implementing image messages for a chat application. The application was a web application that allowed different users to connect to each other and send text messages to each other. I added functionality to send images back and forth between users, as an example of how to implement different message types properly. The other tasks were for working on the framework itself. One example of a task I did was a inspector. The inspector was a piece of code that could be implemented into any application and would be able to display all the underlying application data in a way that was useful to developers. This was desired since the underlying application data was stored in a obfuscated manner which made debugging Lynx applications difficult at times. The inspector I developed, fixed these problems and made developing for and troubleshooting the Lynx application framework easier.

The relationship between this report and my job is that for the various Lynx applications being created, they required some sort of front-end user interface, specifically a web based one. Since the team and the framework were in the early stages, the front-end JavaScript framework could be replaced much easier than after implementing a major application. This meant that the front-end framework, chosen by the Lynx application team, should be chosen carefully since it would be difficult to change later. This report is related to my work at

Autodesk, since it evaluates common frameworks to identify the benefits and shortcomings as well as when to use a given framework.

In the broader scheme of things, the work done on the Lynx application framework will help future Autodesk projects. As a result of this, this will establish a well made and easy to develop with application framework. New applications can be written as well as maintained quickly. This is due to the fact that acquired skills and knowledge obtained working on one project would be useful on another project since they have the same organizational scheme and underlying framework. In addition, having a unified platform for all projects will allow for great code reusability as well. Lynx applications will be able to use common application components, user interface pieces and other pieces of code. This makes production of applications quick and easy as well as well as provide a consistent and familiar experience across Autodesk applications.

Summary

The main purpose of this report, is to evaluate common front-end JavaScript frameworks as well as when they should be used. Currently, there are many constantly changing JavaScript frameworks available as well as more being created all the time. This report is meant to be a quick look into common options when choosing a front-end framework, to help give a general sense of direction to get started. With all the frameworks available it can seem like a daunting task, in which this report is mean to remedy by examining a few common ones, such as: AngularJS, React and Vue.js.

The major points documented/covered in this report are, what is it like to use a given framework such as; the benefits and shortcomings, as well as when the framework should be used. Each framework is evaluated based on the the experience for both the developer and the user. The factors that are evaluated include the learning curve to use it, the available resources such as documentation and online questions, performance and load times as well as other quirks that come with the framework.

The major conclusions in this report are while all frameworks accomplish a similar goal, they vary quite differently in implementation and overall experience. AngularJS has a reasonable learning curve, very good documentation and many online resources, and decent performance. React has a large learning curve with many quirks, very good documentation, and requires compilation for respectable performance. Vue.js has a very simple learning curve, decent documentation, little online resources and great performance.

The major recommendations in this report are to use AngularJS in projects with developers who are familiar with it, to use React in projects that already include a build process with developers who are familiar with JavaScript technologies and to use Vue.js on small and simple projects with newer developers.

Table of Contents

List of Figures

List of Tables

1 Introduction

Autodesk produces and maintains several software products and continues to do so. They offer software to a wide variety of industries ranging from construction and design to photo manipulation and computer animations used in movies/video games. Like most companies, Autodesk continues to expand to new areas and acquire businesses. The list of software they manage can become unwieldy. Having so many different products using different technologies and payment models as well as targeting different audiences can become difficult to manage and will sooner or later require reorganization of the products. Autodesk is investigating web technologies and cloud computing for this reason. Having product on the web are very beneficial for both Autodesk and the consumer. Autodesk can easily update the software since it is on their end, which means that the user is always using software that is up-to-date. Since it is on the Internet already, sharing and collaboration between users can be integrated easier. Online applications also removes a lot of the system requirements local software requires. This results in much lower end devices, being able to run heavy applications that would otherwise require a expensive and large computer. By moving to an online service, Autodesk can also implement a subscription model for payment instead of the traditional licenses it used to offer. Pricing is always a difficult problem for software that is marketed towards large multi-million-dollar companies as well as hobbyists. While the large companies have lots of money to spend and will use the same software until they need a newer feature, the software company must charge enough to support the product assuming that not every user continues to buy the next iteration. This deters hobbyists since they only intend on using the software for a short time and on little projects. The large cost is often too hard to justify so they choose a cheaper inferior alternatives. A subscription model allows Autodesk and other software companies to offer reasonable costs for both large companies as well as hobbyists. Hobbyists can get away with shorter subscriptions and large companies can get the latest iteration of the software cheaper than buying new licensing annually. For these reasons, Autodesk is interested in offering more web-based applications. This is one goal of the Lynx application framework. Since the Lynx framework is web-based, Autodesk and users can experience all of the benefits listed above for all Lynx applications.

1.1 Requirements

The desired functionality and requirements used in this report are based on a project given to the Lynx application framework team. The project is the commercialization of a current research effort. The current project up to this point was designed to experiment with a technology, that is new and somewhat confidential. This resulted in the production of an application that was geared towards the research staff working on it and not everyday

consumers. Since it was to be commercialized, the current design of the interface was unsuitable for the target market and needed to be overhauled. The Lynx application framework team would work with the current research team, to connect their underlying data and functionality with the vision the designers were creating. This means that the interface would be designed mostly from scratch, except for a few assets and components that could be salvaged from the existing code.

The Lynx application framework team had several requirements in mind going into this project. The Lynx team was composed of developers who were familiar with JavaScript but not familiar with many of the commonly used front-end frameworks. So the first requirement was that it should be easy to learn for an existing JavaScript developer, by not requiring lots of new conventions and knowledge. The team had lots of experience utilizing smaller new internal projects, which had little documentation and caused lots of wasted time and effort trying to figure things out. This lead to the second requirement that the framework had have lots of well written documentation and a community with lots of activity on question and answer websites such as Stack Overflow. Finally, the team wanted to ensure that the product they were producing was up to Autodesk standards in terms of quality and performance. The framework should be fast and efficient, to provide the user with an enjoyable experience. One common requirement of web software that could be ignored, was browser compatibility. Since this was a new project and had no previous requirements, the team could choose whatever the requirements they desired. Based on these requirements and lack there of, the three front-end frameworks investigated in this report are: AngularJS, React and Vue.js

1.2 Development and Testing

Due to the confidential nature of the project, the website djlee.xyz was used for this report.

This website was recreated using all the frameworks specified above and the recreated websites will be used as the performance benchmarks. Refer to Appendix A for the code. This website was chosen since the source code is accessible and it is small but contains enough code to test the framework properly. The performance testing and statistics will be acquired by loading the front-end framework specific version of the website as well as any JavaScript dependencies locally. This should remove most inconsistencies due to random nature of network delays and load times when viewing the home page. The report will also evaluate the recreated album gallery page which will show how the front-end framework responds to the use network calls, but may experience some of the network delay listed above.

Figure 1 . djlee.xyz, the website used to evaluate the front-end frameworks.

2 AngularJS

AngularJS is a front-end JavaScript framework that was first released in 2010 and is owned by Google. Since the release, Angular has been updated and now has a version 2. Version 2 has a lot of new changes that make it almost a new front-end framework, compared to version 1. Version 2 has many new and exciting changes but also has some that are somewhat undesirable. Angular 2 has a lot of TypeScript dependencies and that conflicts with the requirement of having a language that is easy to pickup for a JavaScript developer. For this reason, the still respectable Angular 1 is investigated instead of Angular 2.

2.1 Development Experience

Setting up AngularJS is as simple as implementing front-end frameworks can get. The process involves including the script into the HTML page of the application it is required on. There are CDN links as well as files to download for the required dependencies. Once the file has been added to the page, all the developer needs to do is add the AngularJS code and it will work.

Learning AngularJS is front-loaded, meaning all the difficulty is at the beginning. Once the developer understands the syntax, it is smooth sailing. AngularJS works by writing functionality in JavaScript and accessing it in the HTML[1]. The JavaScript contains controllers that can make network calls and modify/set data. Then special AngularJS code is added to the HTML page, which access the data from the JavaScript. The HTML code also has additional functionality that controls what is displayed. This means that certain content

can be hidden based on data from the controller, or lists can be automatically populated in HTML using data from the controller. This can allow for a lot of code to be reused, reducing the overall lines of code as well as simplifying the code for developers familiar with AngularJS.

2.2 Resources

AngularJS is very well documented[2]. There is a lot of documentation for all of the framework specific syntax including a description, examples and even links to the source code. AngularJS also has a lot of learning resources such as guides and tutorials, many of which are well done and very useful for getting over the initial learning curve. AngularJS also has a large community surrounding it. For example Stack Overflow has 214,713 questions tagged as AngularJS as of now.

2.3 Performance

Performance of AngularJS is very respectable. The production version of the dependency is only 156 kB. This means that the impact to load the dependency is small. The performance of the page loading and processing is worse but not by much. Table 1 contains load times for the local AngularJS version of djlee.xyz.

Table 1. A table of load times for AngularJS

Run Number	**Homepage (ms)**	**Album Gallery (ms)**
1	419	258
2	415	233
3	452	247
4	450	241
5	429	245
6	434	217
7	405	244
8	460	240
9	472	243
10	466	229
Average	440.2	239.7

As seen in the table, the values varied a bit between runs, but were relatively close to the average value of 440 ms. The homepage was a little slower than standard JavaScript, but the album gallery was rather quick. The Angular implementation for the network call to retrieve the album art is superior to the JQuery implementation used in the original web page.

2.4 Strengths and Weaknesses

AngularJS is a very good front-end framework. The main weaknesses of the framework is the load time. The processing of the new AngularJS and elements seems to add additional load time to the page in comparison to the original static HTML page. While this is unavoidable, since the Angular code requires more computation, the load time was increased quite a bit. The AngularJS syntax also required some getting used to but compared to other frameworks it was neither too easy or too difficult.

The main strengths of AngularJS is the wealth of resources/large community and an established code base. AngularJS has a large amount of resources including well written documentation of the API and online question forums created by the many users using AngularJS. This makes development easy and efficient, since there are many ways to figure something out or get help. The other main strength is the established code base. The code is well written and bugs are ironed out. Since a lot of developers used AngularJS any bugs are identified and fixed quickly. This results in a clean and efficient code base, which makes the AngularJS code easy to follow if the developer understands the syntax.

Overall AngularJS is a great front-end framework with a lot going for it. While load times are increased, it can be overlooked due to the large community and great code base. Angular seems to be a great choice for larger teams and projects. This is due to the fact that the cleaner looking code is easy to follow, meaning developers have an easier time learning and understanding the code with little effort. Also with a larger community, there are more likely a lot a developers who are familiar with AngularJS. This means that hiring new developers is easy and existing developers may already to familiar with the framework. This makes the transition from the previous solution to AngularJS smooth and easy.

3 React

React is a front-end JavaScript framework that was first released in 2013 and is maintained mostly by Facebook and a few others. React is a different due to the code and requirements. React implements a technically optional new syntax known as JSX[3]. JSX is a syntax extension to JavaScript, which allows HTML-like code to be written alongside JavaScript in the JSX file. The goal of this is to allow a React component written in JavaScript-like code to express how it is to be displayed in HTML-like code. Browsers cannot read the new JSX code, so it must be compiled to common JavaScript. While it is optional, React is designed to be written in JSX and the pure JavaScript version is much harder to write and follow. JSX can also take advantage of the new JavaScript version called ES6 such as

modules. Unfortunately, ES6 JavaScript it is not supported on common browsers. Due to the remarks above, React will be investigated using JSX and without the ES6 JavaScript.

Since JSX is being used, the code must get compiled through Babel[4]. A problem that arises is that, Babel is best compiled beforehand and served when the page is loaded, rather than compiling the JSX at run-time. This is unfair to compare with the other front-end frameworks that compute at run-time. Consequently, two compilation strategies will be investigated when the performance is evaluated. One test for compiling the JSX in the browser at run-time as well as compiling it beforehand.

3.1 Development Experience

Setting up React is rather complicated. To have the React JSX code be compiled at run-time, Babel is added as a script to the HTML page, then the JSX code is added as a babel script. This is similar to other front-end frameworks and relatively easy unlike the environment to compile the JSX beforehand. Babel has many different way to run it, the selected method in this report was the task runner Gulp. Gulp would read a specific file in another syntax which compiled the JSX scripts using Babel and then stored the compiled scripts in a new folder to be referenced in the HTML page. The initial setup for this build strategy was much harder than most front-end framework. It required a lot a dependencies and knowledge. In addition, after the environment is setup, the JSX must be recompiled before changes will appear, this results in slightly more time and effort when developing.

Learning React can be challenging. React works using the idea of components. React components are written using special syntax and represent a piece of HTML with various functionality. React components have the ability to get and manipulate its data as well as specify how the data is meant to be displayed. This results in a component-based design which allows for a large amount of code reuse, one component can be used in many places reducing the amount of redundant code and keeps relevant code together. Getting started with React is not an easy task though. The build process will be ignored as far as learning React since there are many different ways to run Babel, the developer should be familiar with at least one method. React itself is a challenge due to the use of JSX and the organization of the code. While it is possible to get by without using JSX or building beforehand the experience is not as good. Taking the time to fully understand the React is a lot more work but at the same time is also more rewarding. Using JSX and compiling before hand is the proper way to write with React and offers the true React experience.

3.2 Resources

React is well documented. There is a lot of API documentation as well as beginner guides on some of the quirks it possess. Babel is also well implemented and contains many guides to help aid the in the build process with many different task runners. The community surrounding React is also decently sized. Currently there are 30,656 questions tagged as React on Stack Overflow. Being used and maintained by companies like Facebook also causes the code base to be well kept.

3.3 Performance

Performance for React really depended on the build strategy. While Babel can be run at run-time, React/Facebook highly recommends it to be only used for quick prototyping rather than in production code. This results in extremely slow times when run at run-time rather than being built before hand. Table 2 contains the load times for the local React version of djlee.xyz when compiled at run time. Table 3 contains the load times when compiled before hand.

Table 2. A table of load times for React (Compiled at run-time)

Run Number	**Homepage (ms)**	**Album Gallery (ms)**
1	881	717
2	820	749
3	806	732
4	819	752
5	811	755
6	830	745
7	811	737
8	828	736
9	845	743
10	826	774
Average	820.7	774

As shown in Figure 2 , the load times are halved when the JSX is compiled beforehand. This significant difference shows that while possible, JSX is not meant to be compiled at run-time. This also means that React will only be considered using the building step, since the extra setup and effort is required to make React competitive.

Table 3. A table of load times for React (Compiled before test)

Run Number	Homepage (ms)	Album Gallery (ms)
1	406	314
2	405	312
3	426	307
4	427	309
5	429	317
6	425	312
7	413	317
8	406	319
9	440	315
10	402	313
Average	417.9	313.5

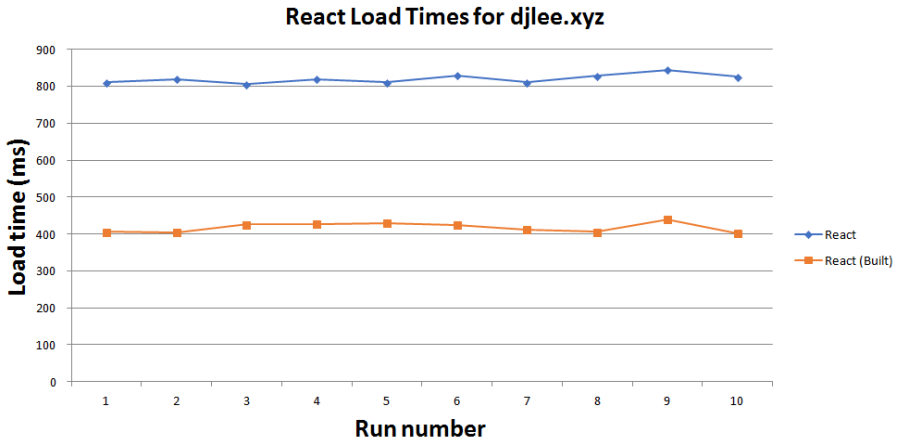

Figure 2 . A graph load times for the homepage of djlee.xyz using different build strategies

3.4 Strengths and Weaknesses

The main weaknesses of React are the learning curve and the environment setup. The learning curve for React is not only steep due to JSX but also the sheer amount of options available. While most front-end frameworks have one way to using it, React seems to have so many options it can be overwhelming. For example, the developer can choose to write using JSX or without, building the JSX or not and using ES6 JavaScript or not. There are also other functionality and options such as writing ES6 JavaScript and having Babel compile it into the older version of JavaScript. The amount of required knowledge to make certain decisions is rather high and daunting.

The main strength of React is the development. After all the setup, learning and decision-making the rest of the development is great. Since React has so much new code it has a lot of functionality and good practices. Once the developer has finally made it past the high

entry barrier, the result is very clean and efficient, because things like building the code beforehand optimizes the performance.

Overall React is a front-end framework with a lot of potential. It is great for teams with experienced developers who know React and already contain a build process. If the team is already building an web application that requires the technology to compile the JSX with Babel then most of the difficulty removed. Teams with developers that know React will also be much farther ahead, since it would nullify most of the weakness React faces. As a result, React is a great choice since the well designed code can be obtained without much extra time or effort.

4 Vue.js

Vue.js is a smaller front-end framework originally released in 2013. It is a small and lightweight framework that is one of the lesser used front-end frameworks. It is well made and does the job well, but may lack some of the more complex functionality of the larger more established frameworks.

4.1 Development Experience

Setting up Vue.js is as simple as front-end frameworks can get. The required JavaScript dependency is added to the page using either a CDN link or by referencing the Vue.js file. Once that is done, Vue.js code can be accessed right away.

Learning Vue.js is back-loaded, meaning it is relatively easy to get started but can be more challenging later on with more complex tasks. There are a couple getting started guides, however, but the overall syntax and structure is not overly complicated. HTML elements are set as a vue which allows it to run Vue.js code specified in the JavaScript[5]. The newly set vues can display data from the Vue.js code and display it in any way. While getting started is quite easy, the more advanced tasks are more challenging. This is due to to the fact that Vue.js is a smaller framework, it simply does not possess some of the complex functionality of other frameworks which makes complex tasks harder to perform.

4.2 Resources

Vue.js is a smaller and lesser used framework, so the available resources leave something to be desired. It does have decent API documentation[6] as well as some helpful guides

to get started, but it is not much more than that. On stack overflow there are only 3,168 questions that are tagged as Vue.js. Since it is still in more early stages, there are major updates stilling rolling out. A problem with this is that the few online question posts can becomes useless if the help provided become out of date. An example of this is that the keyword "ready" was used on vues to specify code that is to be run when it is loaded. The word was changed to mounted in a newer version, making any question posts that included the work "ready" obsolete and misleading to new readers.

4.3 Performance

The performance of Vue.js is really good. Since it is a smaller front-end framework the dependency is small and efficient. The required JavaScript dependency is only 71 kB which causes very little overhead. Table 4 contains the load times for the Vue.js version of djlee.xyz. Vue.js proves to be a very lightweight and efficient front-end framework.

Table 4. A table of load times for Vue.js

Run Number	**Homepage (ms)**	**Album Gallery (ms)**
1	428	285
2	414	287
3	442	308
4	384	297
5	392	286
6	399	294
7	408	289
8	374	291
9	381	299
10	423	296
Average	404.5	293.2

4.4 Strengths and Weaknesses

Vue.js is a great small framework. The main weaknesses are the lack of functionality and resources. Vue.js is a small framework that cannot do everything the bigger frameworks can do. Also, the complex tasks it can handle can be hard to implement due to the lack of community and help available.

The strengths of Vue.js is the initial learning curve as well as the speed and performance. Getting started with Vue.js is as easy as it can be. After importing a small script it can be used right away with little effort. The lacking of complex functionality, as expressed above, means that the framework can be very efficient. Since there is very little extra code and

overhead, pages can be loaded quickly.

Overall, Vue.js is an interesting front-end framework. It is a very small and simple framework that would be suited for smaller teams working on simple projects. These sorts of projects require something that is fast to load as well as to learn. Vue.js is perfect for such applications but may struggle for larger complex projects. Since there is a smaller community and resources, the best practices are not always specified. This can result in messy code bases for large projects and may in turn make future changes difficult.

5 Evaluation

All of the front-end frameworks investigated in the report have their merits and applicable scenarios where they succeed. This section will look at the frameworks based on the situation of the Lynx application framework team at Autodesk.

5.1 Developer Experience

In regards to setup, almost all the frameworks are equal. Both AngularJS and Vue.js have a very similar setup. The project already contained a build step, so implementing the Babel compilation step was rather easy, meaning that using React was not much more effort.

As far as learning, React was the best choice. While the team did not have knowledge of any of the frameworks, React was the best one to learn. This is because there are other Autodesk products that use React. Learning React now may provide transferable skills that may help with future projects. While AngularJS and Vue.js are easier to learn, React could prove to be more useful in the long run. This is to avoid learning AngularJS or Vue.js now and having to learn React later since a new project may already use React.

5.2 Resources

Figure 3 is a graph comparing the number of Stack Overflow questions for each framework. Angular is definitely the best choice as far as Resources. With an overwhelming amount of online question posts and well written documentation, learning is almost effortless. React is still respectable in regards of documentation. It is well kept by Facebook and lots of online activity and community albeit less than AngularJS. Vue.js is definitely the worst in this category. It is actually difficult to find help, and the available help may be outdated.

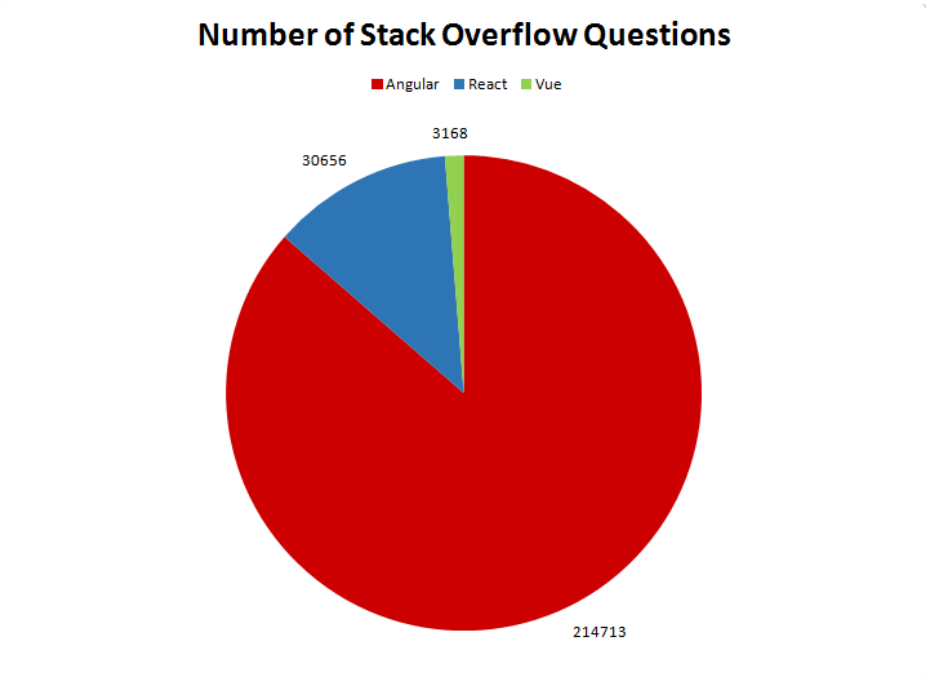

Figure 3 . Graph showing the relative number of Stack Overflow questions for each framework

5.3 Performance

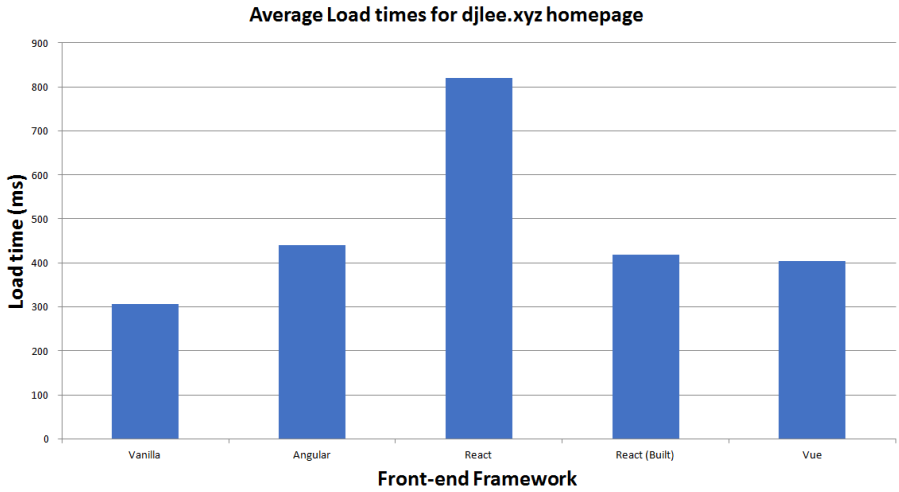

Figure 4 . Graph comparing the load times of djlee.xyz homepage with different frameworks

Figure 4 show the difference in load times between all of the frameworks, with static HTML and plain JavaScript labelled as Vanilla. The most notable thing Figure 4 demonstrates is the poor performance of React when compiled at run-time. It is clear that only the built option is competitive, but as stated above the build process is easy to implement. While the frameworks are all competitive, Angular is slower than react by about 30 ms and react is

about 20 ms slower than Vue.js. Vue.js is has the best performance out of all the front-end frameworks, but the other frameworks are not that far behind. 20 ms and 30 ms are not that big considering the difference from the original website is anywhere from 100 ms to 150 ms. This means that all front-end frameworks are still viable, but Vue.js is the fastest followed by React, then AngularJS.

6 Conclusions

From the analysis in the report body, it was concluded that all frameworks have benefits and shortcomings. Choosing the correct front-end framework really depends on the needs of the team, the requirements and the project itself.

For big projects with big teams AngularJS is an excellent choice. This is because of the large community surrounding the framework. The large community means that finding developers is easy and getting help is as well. For complex projects with experienced developers and a build process, React is a great choice. In these circumstances the learning curve and setup is less drastic and yields a very clean and robust result. For small teams working on simple projects, Vue.js is a good choice. Projects that are either quick prototypes or simple applications are great with Vue.js, since it is fast to setup and code with. These project also will not experience the problem of lacking complex functionality.

7 Recommendations

Based on the analysis and conclusions in this report, it is recommended that, the Lynx application team and others in a similar situation use React as their front-end framework. This is because of the lack of additional overhead, transferable skills and robust final product.

Since the Lynx application team already had a build system in place, adding the capability of compiling the React JSX code with Babel could be implemented easily with already existing knowledge. Building the code before hand has additional performance benefits which can be had at next to no additional overhead. Since Autodesk uses React in other projects, learning React could be a valuable skill that will come in handy later. Newer projects the Lynx team gets, may already be using React, so the team will already have the skills required and avoid having to learn yet another front-end framework. The final reason is React's code style fits the Lynx application framework. The heavy use of components is common in both, so the overall organization is more consistent.

Glossary

AngularJS: A front-end JavaScript framework owned by Google.

API: Aplication program interface; a set of routines, protocols and tools used for software development.

Babel: A JavaScript transpiler. It is used can be used to compile JSX to JavaScript.

CDN: Content delivery network; a system of servers that host and deliver web content.

djlee.xyz: A personal website for David Lee, a Univerity of Waterloo Computer engineering student.

HTML: Hypertext Markup Language; a standardized language to define layout and content for web pages.

JavaScript: A programming language commonly used in web development.

JQuery: A JavaScript library created by John Resig in 2006. The motto is: write less, do more.

JSX: A preprocessor for React. It adds XML syntax to JavaScript.

Lynx: An application framework used by Autodesk.

React: A front-end JavaScript framework by Facebook.

Stack Overflow: A question and answer website designed so professional and enthusiast programmers can share their knowledge.

Syntax: The specific words that create the language.

TypeScript: A super-set of JavaScript. It is a programming language that adds functionality not present in JavaScript. It is owned by Microsoft.

Vue: A Vue.js keyword. It is the term used to describe a HTML element that has access to Vue.js functionality.

Vue.js: A front-end JavaScript framework.

References

[1] H. Rowland, *AngularJS: Easy Guide on Web Application Development*. CreateSpace Independent Publishing Platform, 2016.

[2] P. Darwin, *Guide to angular 1 documentation*. [Online]. Available: `https://docs.angularjs.org/guide`.

[3] D. Abramov, *Introducing jsx*. [Online]. Available: `https://facebook.github.io/react/docs/introducing-jsx.html`.

[4] S. Tamizhvendan, *Setting up a react.js environment using npm, babel 6 and webpack*. [Online]. Available: `https://www.codementor.io/tamizhvendan/tutorials/beginner-guide-setup-reactjs-environment-npm-babel-6-webpack-du107r9zr`.

[5] O. Filipova, *Learning Vue.js 2*. Birmingham: Packt Publishing Limited, 2016.

[6] E. Oliveria, *Introduction to vue.js*. [Online]. Available: `https://vuejs.org/v2/guide/`.

Appendix A djlee.xyz codebase

All of the code for the report can be found at: https://github.com/davidjameslee/WKRPT401. The repository contains the original code (vanilla) for the website as well as the front-end framework specific recreations, each placed in a separate folder. The common resources such as pictures, styling and common scripts are placed into the Resources folder.