UNIVERSITY OF WATERLOO

Faculty of Mathematics

IMPLEMENTING MODERN CODING STRATEGIES THAN PRIMITIVE ONES

Universe Experiences Inc.,

A Division Of Live Nations Entertainment

Toronto, Ontario

Prepared By
Kritin Singhal
1B Computer Science
ID 20731264
June 2018

33 Wood Street
Toronto, ON
M4Y 2P8

25th June, 2018

Joshua Kelly
Chief Technology Officer
17 Phoebe St
Toronto, ON
M5T 1A8

Dear Joshua Kelly,

I have prepared the enclosed report, “Implementing modern coding strategy than primitive ones”, as my 1B work term report for Universe. This report, the first of four work reports that the Co-operative Education Program requires that I successfully complete as part of my BCS Co-op degree requirements, has not received academic credit.

As a Software Development Intern, I am working on developing components to add new services to the website, and part of my work involved working with Ember, an obsolete coding framework. This report will explore several issues faced by both the developers and the running system as a result of using obsolete frameworks.

The Faculty of Mathematics requests that you evaluate this report for command of topic and technical content/analysis. Following your assessment, the report, together with your evaluation, will be submitted to the Math Undergrad Office for evaluation on campus by qualified work report markers. The combined marks determine whether the report will receive credit and whether it will be considered for an award.

I would like to thank you for your assistance in preparing this report.

Sincerely,
Kritin Singhal

Table of Contents

List of Figures

Executive Summary

The report entitled “Implementing Modern Coding Strategy Rather Than Primitive Ones” focuses on using modern coding elements while building up applications for startups, new companies and other existing ones. The report focuses on the negative effects of using primitive coding strategies and its implication on the working development system of Universe.

A lot of information on the advantages and disadvantages of various libraries have been provided in the Analysis Section also including the details of a research in UC Davis that published a study of the effect of programming languages on software quality using a very large data set from GitHub. The data loading time and delays have been demonstrated in the same section.

A lot of live experiments have been performed and shown in the following section demonstrating the adverse effects of some libraries in the loading time of the page affecting Search Engine Recognitions. Overall, the report is an analysis of different libraries and frameworks that we use in our everyday life and how we can wisely select the most appropriate one.

1.0 Introduction

Since the advent of twenty first century, we have seen technology evolve in a drastic way. With this change, we have also noticed the shift of our Operating Systems from Linux to Mac OS and Microsoft. We also hear about thousands of frameworks evolving each day.

Developers often want to start writing an app but don't want to manually evaluate 300 different options in order to select something maintainable. There's currently however almost too much choice when it comes to what to use for structuring our JavaScript application. Part of this problem is fueled by how different JavaScript developers interpret how a scalable JavaScript application should be organized — everyone feels their way is the best. As a community we seem to prefer creating our own silos over improving existing solutions. (Colin, 2014).

Now the big question is, which platform and library stack should we use while building our application or enterprise? Answers can be diverse but this report is going to focus on how to select the right framework and library.

Founded in Toronto in 2011, Universe is a platform for people to discover and create events all over the world. Following unprecedented growth in 2014, Universe has expanded their offices from Toronto to San Francisco and doubled the size of our team that's dedicated to serving the needs of our organizers and fans. In 2015, Universe was acquired by Live Nation Entertainment and has continued its rapid growth, doubling the size of the team since the acquisition. In 2016, the company expanded to open offices in New York City and London England, to cover Europe.

Together, we've built a community of over 32,000 event organizers and hundreds of thousands of fans who are the heart of our company.

The Universe Experiences Inc, under Ticketmaster and Livenation is one of the many startups that have been suffering from the effects of using native coding languages which have now become obsolete. As a result of which, the coding experience of the developer is completely shattered as it takes hours to fix a bug because of poor abstraction and multiple platforms. The modern library stacks have a common arena for all the platforms meaning, code once written can be supported in all the systems which basically dumps the requirement of writing a similar set of code multiple times for different platforms.

To dive within the current system of Universe, Universe has all its code for the web written in EmberJS which has become obsolete as stronger frameworks like React JS have evolved with time. As a result, the need to shift the coding platform from one element to the other has become a necessity. For the mobile, Universe currently operates four different applications, two which runs on iOS and the other two which runs on Android to host two applications that are identical on both the platforms. With native code, the developers had to write separate lines of code to implement the same application under iOS and Android. While if they had used systems like React-Natives earlier, they could have worked it out with just a single set of code and could used it for all the systems, iOS, web, Android, etc.

Even though the issue seems quite much of a personal choice, it is extremely beneficial to be on the right track from the very start. And to help entrepreneurs and developers, I would like to address this issue for my work term report.

2.0 Analysis

To begin with, a framework is a collection of libraries and a library is a collection of innate creative and usable classes. Frameworks are a great way of taming complexity. Rather than have to remember all the fiddly details of vendor prefixes and syntax, and rather than have to put up with the limitations inherent in the CSS syntax (still no variables!), one can load up a framework and let ‘er rip. One can even, depending on the framework, invoke a few simple classes to get precalculated layouts. Frameworks are popular for very good reasons. On the other hand, in many ways they’ve traded one form of complexity for another. It’s a veritable jungle of frameworks large and small out there, and figuring out how to navigate that jungle requires an expert guide to get one off to a good start. One needs that guide not to tells one the specific characteristics of every plant and animal in the underbrush, but to give one hard-won advice on how to approach various situations, what to look for and what to avoid, and thus how to thrive in a constantly shifting environment. (Meiert, 2015).

.The negative effects of using primitive coding strategy are plenty. With Universe, one of the worst effects was the debug system. It takes at least one hour to fix even the smallest of issue while working with the primitive code while with the translated ones, it takes just a few minutes.

Secondly, the page loading time increases with the age of the frameworks. The modern frameworks have the capability to load pages faster. Google Search is going to start to penalise websites that takes time to load from July 2018. Universe has most of its platform developed with the primitive code as a result the page loading time is 285384 ms which is huge. As a result,

the need to translate the code into a modern framework has become a matter that requires urgent action.

Third, it takes ages to test a piece of code. “I don’t want to sound like a broken record here but I’ve spent a significant portion today either waiting for my build(s) to start, or waiting for more than 15 minutes for a build to finish. Not only is this slowing my development workflow, it’s also affecting my concentration and focus. I cannot sit around for at least 15 minutes while my build finishes, so I switch context to a different feature or branch to make use of this time.” (Personal Communication, Ahmed Elhoussani, Data Scientist at Universe).

2.1 Various Programming Languages

Computer scientists at the University of California — Davis have published a study of the effect of programming languages on software quality using a very large data set from GitHub. They analyzed 729 projects with 80 million SLOC by 29,000 authors and 1.5 million commits in 17 languages. The large sample size allowed them to use a mixed-methods approach, combining multiple regression modeling with visualization and text analytics, to study the effect of language features such as static vs. dynamic typing, strong vs. weak typing on software quality. By triangulating findings from different methods, and controlling for confounding effects such as team size, project size, and project history, they report that language design does have a significant, but modest effect on software quality. (Devanbu, 2017).

Generic programming errors account for around 88.53% of all bug fix commits and occur in all the language classes. Possnet’s analysis draws a similar conclusion since programming errors

represent the majority of the studied fixes. All languages incur programming errors such as faulty error-handling, faulty object and variable definitions, incorrect data initialization, typos, etc.. Still, some programming errors are more language-specific. For example, we find 122 runtime errors in JavaScript that are not present in TypeScript. In contrast, TypeScript has more type related errors, since TypeScript compiler flags them during development. Thus, although generic programming errors occur in all languages, some kinds relate to specific language features. (Posnett, 2017).

2.2 Identifying top languages

The top languages in GitHub are measured by first finding the number of open source GitHub projects developed in each language, and then choosing the top languages with the maximum number of projects. However, since multiple languages are often used to develop a project, assigning a single language to a project is difficult. GitHub Linguist can measure such a language distribution of a GitHub project repository. Since languages can be identified by the extension of a project's source files, GitHub Linguist counts the number of source files with different extensions. (Linguist). The language with the maximum number of source files is assigned as primary language of the project. GitHub Archive stores this information. We aggregate projects based on their primary language. Then we select the top languages having maximum number of projects for further analysis as shown in Table 1.

Table I: Top three projects in each language

Language	Projects
C	linux, git, php-src
C++	node-webkit, phantomjs, mongo
C#	SignalR, SparkleShare, ServiceStack
Objective-C	AFNetworking, GPUImage, RestKit
Go	docker, lime, websocketd
Java	storm, elasticsearch, ActionBarSherlock
CoffeeScript	coffee-script, hubot, brunch
JavaScript	bootstrap, jquery, node
TypeScript	bitcoin, litecoin, qBittorrent
Ruby	rails, gitlabhq, homebrew
Php	laravel, CodeIgniter, symfony
Python	flask, django, reddit
Perl	gitolite, showdown, rails-dev-box
Clojure	LightTable, leiningen, clojurescript
Erlang	ChicagoBoss, cowboy, couchdb
Haskell	pandoc, yesod, git-annex
Scala	Play20, spark, scala

2.3 Prevent Ember

On doing a research on developers experience with Ember, I have formulated the common problems that developers face if they are coding in Ember. Firstly, Silent errors pop up which are extremely difficult to debug. Secondly, Ember forces deep linking which is highly unreliable. Based on developer names, we have identified in total 581,856 distinct developers who have contributed to the code bases of the projects with issues. Those issues were filed by 239,629 reporters. Bissyande, furthermore found that, for each project, one third of the developers (33%, the median value) have written some issue reports for the project. On the other hand, 42% of the issue reporters for a given project do not contribute to the code base. (Bissyande, 2007). Thirdly, Ember has fewer libraries. Duvall felt that he was re-inventing the wheel, and poorly because he had a deadline to hit. (Duvall, 2007). Lastly, slower development and complexity bugs plenty of developers.

Table II: Bug Report

# Label	# Labeled Issues	% of Labeled Issues
bug/error	45,123	20.65%
feature/enhancement	46,402	21.23%

The developer should argue for how the solution is better than the existing options and why the user should care about it. Demonstrate how it can make their lives easier and why it's worth them investing their time in it. A lot of the repeated effort we set in the community doesn't justify itself or try to set itself apart. For example, by promoting our project as AngularJS or Ember in 50 lines of vanilla JavaScript, we're discounting the underlying complexity and completeness of those solutions. We're also discounting the efforts that went into architecting and optimizing them without going into detail of how one's solution compares performance wise to the exact same set of use cases. It's more fair to both our audience and other framework authors to be balanced and honest in our claims. (Addy, 2014).

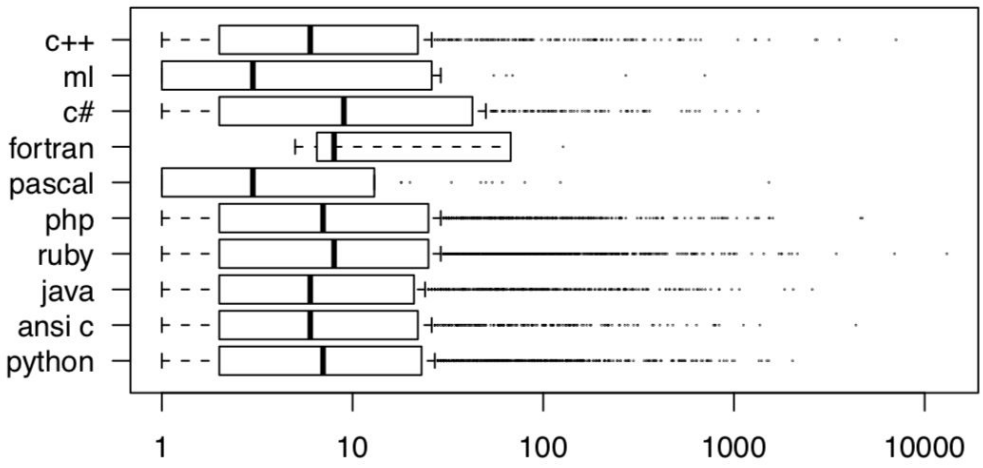

Figure 1 - Issues due to old programming languages

When we started translating the code into a modern framework, we noticed that the loading time has dropped to 2614 ms which was ten times faster than before. The page loaded faster and on top of that, the dynamic quality of the page magnified. Thus, we should stick to modern methods of coding with modern frameworks.

3.0 Conclusion

There are numerous libraries and frameworks in the web stack today. We are preoccupied by adding more to list. Instead, we should build the existing ones and make them strong and efficient. The primitive libraries have not been able to evolve because of the barrier which the new ones create for them. As a result, it forces the developers to both update the current library stack and learn the new framework. A developer should choose one framework wisely and stick to that till the library reaches its critical point. Choosing frameworks wisely can be a difficult task but with right research and prior knowledge, decision-making becomes easy. Ember should be prevented from being included in the new library stacks of the new projects. In fact, a developer should prevent all the primitive library stacks that have become obsolete.

References

Prem Devanbu (2017). “Developer Onboarding in GitHub: the Role of Prior Social Links and Language Experience”.

T. F. Bissyande, D. Lo, L. Jiang, L. Reveillere, J. Klein, and Y. Le Traon. “Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub.” In ISSRE, pages 188–197. IEEE, 2013.

P. M. Duvall, S. Matyas, and A. Glover. “Continuous integration: improving software quality and reducing risk.” Pearson Education, 2007.

Y. Fu, M. Yan, X. Zhang, L. Xu, D. Yang, and J. D. Kymer. “Automated classification of software change messages by semi-supervised Latent Dirichlet Allocation.” Information and Software Technology, 57:369–377, 2015.

Addy, Sindre, Pascal, Stephen and Colin. “Yet another framework syndrome.” February 3, 2014.

Jens Oliver Meiert. “The Little Book of HTML/CSS Frameworks.” Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Baishakhi Ray, Daryl Posnett, Vladimir Filkov, Premkumar Devanbu. “A Large Scale Study of Programming Languages and Code Quality in Github” (2017).

GitHub. Linguist: https://github.com/github/linguist.

Acknowledgement

I would like to thank Ahmed Elhoussani, Data Scientist at Universe, for providing me all the research data from within the company. I would also like to thank Evgeny Li, Senior Software Engineer at Universe, for carrying out the task of improving the Ember page load and demonstrate the difference between the loading time of the landing pages. I would also like to Wantong Li, Yu Fan Deng and Taylia Gabrielle Henderson for proving me a valuable peer review feedback.