Browser Performance of JavaScript Framework, SAPUI5 & jQuery

Jaime Raigoza
Computer Science Department
California State University, Chico
Chico, USA

Rushi Thakkar
Computer Science Department
California State University, Chico
Chico, USA

***Abstract*— The research and commerce communities have a need to understand and use the Single-Page Application (SPA) framework written in JavaScript for knowledge discovery and global sharing of information. It is important to develop user-friendly web-based tools which efficiently process, analyze, and visualize scientific data. The efficient use of languages and tools such as HTML5, CSS3, and JavaScript Libraries is important. The need to asynchronously fetch new data and update the web page, without refreshing the entire page is essential to better visualize the data in real-time. Without any scientific methodology for comparing SPA frameworks written in JavaScript, it is troublesome to choose which one to use. Our work studies a method to better understand the web client technology with an interactive application. We build upon an existing open source web application JavaScript framework tool to measure the time distribution of client components. We study interactive web-based activity.**

Keywords- JavaScript; jQuery; Performance comparison; SAPUI5; User Interface.

I. INTRODUCTION

jQuery plugins and widgets are amongst the most implemented components within the front-end developer orbit. The OpenUI5 framework helps software application developers with many user interface controls and with other aspects such as the Model-View-Controller (MVC) architectural pattern. To develop highly interactive web applications, modern web designs are being influenced by the JavaScript Library/Framework. JavaScript technologies keeps on evolving but there continues to be a need to process large amounts of data and more efficiently render web pages and enhance the user experience. In this paper, we have exploited JavaScript frameworks to visualize and measure the performance of the specific features and components on the browser. JavaScript libraries are regularly evolving and becoming diverse. Our work presents a demonstration to visualize the performance of the advanced WorkLog application on the Web Server with two frameworks, jQuery and SAPUI5. Our goal is to help understand the differences in selecting a better JavaScript framework and components under the conditions tested. We used JavaScript and applied coding techniques to pursue reliable communication and efficient data storage by minimizing the computational complexity.

Efficient browser technology and its implementation is important for the success of a web application. JavaScript is executed on the user's computer when accessing a web page. The benefit is that less server resources are needed to allow a larger number of clients accessing the same server without incurring much server cost. This also results in less communication between the client and server which helps applications having better real-time performance. We have written code in two different languages using two different libraries. In our application, there are four functions and mostly everything is based on the bowser. The code structure for both applications are based on JavaScript. Our results serve to study the performance of these frameworks and can be a step forward to compare the JavaScript framework using the same jQuery library. SAPUI5 is based on the jQuery library so a better understanding of the performance will help justify its use.

II. PRELIMINARIES

It is generally agreed that there is no exact definition of the SPA concept. Mesbah and van Deursen [4] define it as: "the single-page web interface composed of individual components which can be updated/replaced independently, so that the entire page does not need to be reloaded on each user action". To achieve a better understanding of SPA, we define additional attributes.

Fernández-Villamor et al. [2] propose a method on how to compare agile web frameworks. They define a "blueprint architecture" containing criteria that are important for a web framework. The criteria is defined using a set of questions that summarize the general features of an agile web framework. They provide percentages and weight metrics of frameworks, and thereafter discuss the strengths and weaknesses.

Malmström [3] proposes a method on how to compare the SPA framework. He uses seven requirements taken from interviews. After defining the requirements, he discusses how the frameworks individually fulfill the requirements. Malmström [3] also performs a performance test on two frameworks.

The method proposed in this paper is closely related to the practical comparison by Fernández-Villamor et al. [2] and it also has a resemblance to Malmström's [3] way of comparing SPA frameworks. By combining these theoretical and practical approaches, it possible to carry out a scientific comparison of SPA frameworks.

We also leverage on a method for performance testing created by Microsoft Research which is applied to a web application to obtain structure testing [1]. The method consists of seven steps:

978-1-5090-5510-4/16 $31.00 © 2016 IEEE
DOI 10.1109/CSCI.2016.277

1. Identify test environment
2. Identify performance acceptance criteria
3. Plan and design tests
4. Configure test environment
5. Implement the test design
6. Execute the test
7. Analyze results, report and retest

In traditional web applications, the testing is mostly performed on the back-end service. However, the scope of this paper only covers the client side of SPAs. The crucial part of an SPA's performance is its data bindings, loading time and resource allocation.

Our prototype is an extension of the work created by TodoMVC whose code was taken from their GitHub repository. It was chosen because it is a minimal example of a web application with basic functionality including the create, read, update and delete data task operations. Currently, it has one or more implementations of each of the most popular SPA frameworks. The front-end user interface screen is shown in Figure 1.

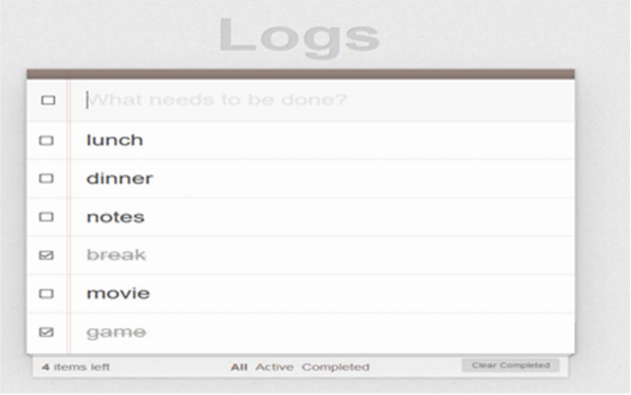

Figure 1. Our prototype UI.

III. RESULTS

We implemented two different modified small single-web application using a jQuery and a SAPUI5 client. With the modifications. Figures 2. And 3. show the distribution time duration for the listed components, given similar data.

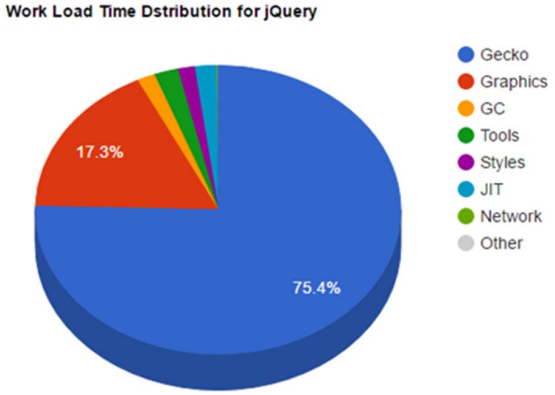

Figure 2. jQuery work load time distribution.

IV. CONCLUSION

Our results show that jQuery takes a longer time to fetch data from the browser where SAPUI5 takes a longer for graphics and images to load on the page. This paper tries to identify a set of criteria for comparison however, we do not promote any specific way of using weights in either questions or criterion.

For future work, security is a major issue for us. Many web applications store user credentials or credit card numbers in a database connected to the Internet. For example, given a security breach in the web application, all the user's data can be stolen and the user's trust could be lost. To better select any JavaScript framework, some top security concerns are i) does the framework have a security policy? ii) does the framework have a promotion for finding bugs?

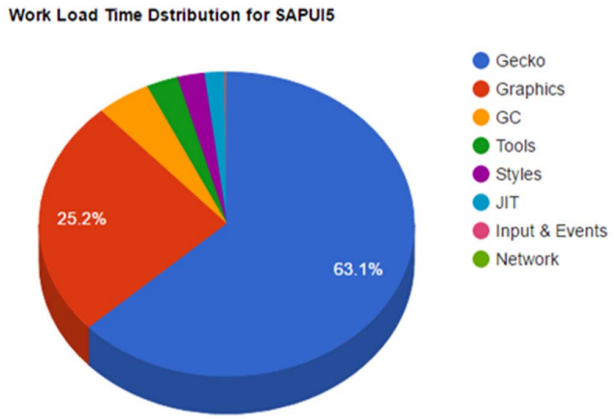

Figure 3. SAPUI5 work load time distribution.

REFERENCES

[1] Performance Testing Guidance for Web Applications. Microsoft Patterns & Practices. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb924375.aspx

[2] J. I. Fernández-Villamor, L. Díaz-Casillas, and C. Iglesias. A Comparison Model for Agile Web Frameworks. In ACM Proceedings of the Euro American Conference on Telematics and Information Systems, EATIS '08, pp. 14:1–14:8, New York, NY, 2008.

[3] T. J. Malmström. Structuring modern web applications: A study of how to structure web clients to achieve modular, maintainable and long-lived applications. Master's thesis, KTH, School of Computer Science and Communication (CSC), 2014.

[4] A. Mesbah and A. van Deursen. Migrating Multi-page Web Applications to Single-page AJAX Interfaces. Proceedings of the 11th European Conference on Software Maintenance and Reengineering pp. 181-190, CSMR 2007.