Comparative Evaluation of JavaScript Frameworks

Andreas B. Gizas, Sotiris P. Christodoulou and Theodore S. Papatheodorou
HPCLab, Computer Engineering & Informatics Dept., Univ. of Patras, 26500 Rion, Patras
{gizas,spc,tsp}@hpclab.ceid.upatras.gr

ABSTRACT
For web programmers, it is important to choose the proper JavaScript framework that not only serves their current web project needs, but also provides code of high quality and good performance. The scope of this work is to provide a thorough quality and performance evaluation of the most popular JavaScript frameworks, taking into account well established software quality factors and performance tests. The major outcome is that we highlight the pros and cons of JavaScript frameworks in various areas of interest and signify which and where are the problematical points of their code, that probably need to be improved in the next versions.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *complexity measures, Performance measures.* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – *Corrections, Enhancement*.

General Terms
Measurement, Performance, Experimentation, Languages.

Keywords
JavaScript Frameworks, Metrics, Quality, Performance.

1. INTRODUCTION
The most popular programming language for the browser today is JavaScript [2]. Due to the plethora of applications that JavaScript serves and the variety of programming needs, **JavaScript Frameworks** (JFs) have been developed in order to facilitate the work of web programmers. For them, it is crucial to choose the framework that better matches their needs, and provides high quality code and good performance. There are several more factors that may influence their choice, like maintainability, validity, active supporting community, etc. Towards this direction, the scope of this work is to provide a thorough quality, validity and performance evaluation of the six most popular JFs.

There is a plethora of software quality metrics that have been proposed and evolved during time. **Choosing the right metrics** is crucial in order to assess accurately. The authors in [1] studied and evaluated several software quality metrics and found correlation among them, indicating that some of them seem to measure the same properties. Furthermore, other researchers [3] focused on the analysis of **performance and quality of web applications**. We based our own methodology in such outcomes, in order to choose the right metrics relevant to JavaScript programming language and frameworks.

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

As far as **JavaScript performance** testing is concerned, the authors in [4] provide us with an evaluation of benchmarks that should be used. There are some efforts regarding JavaScript frameworks comparison, but none is focused on the software quality factors of the frameworks.

2. ANATOMY OF JFs EVALUATED
JFs typically provide a library of classes or functions that will do a multitude of operations like managing DOM traversal and manipulations, insert visual effects, Ajax manipulations, managing layout, and impose an architecture that provides an unformed way to extend the framework (e.g. plug-ins, modules).

We chose to evaluate the six most popular JavaScript frameworks: ExtJS, Dojo, jQuery, MooTools, Prototype and YUI. Most of them come in 3 different run-time versions: **Base Version (uncompressed),** which includes a minified set of core functions; **Compact Version,** i.e. the base version with no comments and blank lines; **Full Compact Versions**, with all the available functions in a compact format. Additionally, JFs provide development versions (***Development Kits)*** in order to satisfy more advanced developer needs. Moreover, there are plenty of add-on functions or even ready additional programs (widgets), which come to enhance the basic functionalities of the core in each JFs (like jQuery UI and *Scriptaculous* for Prototype.

We define as **core version** the set of functions for DOM manipulation, selectors, Ajax functionalities, basic elements of forms, functions for base event handling and functions for compatibility support and loading utilities. In order to evaluate the code of similar functions for each JF, we needed to extract or create the core version for each JF.

3. THE EVALUATION METHODOLOGY
We conducted diverse quality, performance and validation tests. For performance tests we used the core compact versions while for the other tests we used the non-compact. All files with the detailed measurements of the tests described below are available here: http://150.140.142.212:100/JFmetrics/index.html

3.1 Quality Tests
As far as quality is concerned, we measure the size, the complexity and the maintainability of the code as a whole and per function. We were based on the following well-known software metrics. **Size Metrics:** *lines of code (LOC), number of statements*, the *comment lines* and the *ratio* between comment lines and lines of code. **Complexity Metrics:** *McCabe's cyclomatic complexity (CC), Branches* and *Depth*. **Maintainability Metrics:** Halstead metrics (*Program Volume* and *Program Level*) and *Maintainability Index (MI)*.

The following tools were used: JSmeter (jsmeter.info), Cloc (cloc.sourceforge.net) and Understand (scitools.com). Table 1 outlines the Size Metrics.

Issues revealed: Prototype is very poorly commented, YUI3 provides the larger set of functions and the largest sum of cyclomatic complexity, indicating a hard to maintain code.

Table 1. Size Metrics

Frame works	Statements	Comment Lines	Lines	%Comments per Lines
Dojo 1.7.2	7551	6470	13858	46.69%
ExtJS 4.0.7	6500	5239	11932	43.91%
jQuery 1.7.0	7252	1242	9301	13.35%
MooTools 1.4.4	6650	1338	6428	20.82%
Prototype 1.7	6265	42	6085	0.69%
YUI2 2.9.0	6624	5197	12592	41.27%
YUI3 3.4.1	12210	9624	24115	39.91%

Table 2 shows the functions' distribution of problematic CC, depth and MI.

Table 2. Complexity and Maintainability Metrics

Frameworks	CC:16-20	CC>20	Depth>5	MI<100	MI<85
Dojo 1.7.2	14	11	7	41	5
ExtJS 4.0.7	1	12	5	22	6
jQuery 1.7.0	10	18	6	68	14
MooTools 1.4.4	5	10	4	38	9
Prototype 1.7	5	11	2	42	21
YUI2 2.9.0	3	12	7	31	1
YUI3 3.4.1	11	28	13	54	11

Issues revealed: Functions with CC > 20 or Depth >5 or MI < 85 must be checked. In all frameworks there are many functions that need to be checked and probably their code needs to be improved. YUI3 and jQuery seem to be the hardest to maintain. This may influence the evolution of them negatively.

The full list of the worst functions for CC, Depth and MI metrics can be accessed from our server. Those functions that appear in all three lists are the most crucial ones for check. For example in jQuery the (Anonymous1).(Anonymous16).Sizzle.filter function (starts in line 4153) has CC=24, depth=7 and MI= 68.973.

3.2 Validation Tests

Validation tests conducted by using Yasca (sourceforge.net/ projects/yasca) software utility, in combination with JavaScript Lint (javascriptlint.com). Table 3 summarizes the errors found for every JF. Overall errors include critical, high, low severity errors and informational errors.

Table 3. Vulnerability and Conformance

Frameworks	Critical Errors	High Severity Errors	Overall
Dojo 1.7.2	41	16	999
ExtJS 4.0.7	21	17	1996
jQuery 1.7.0	29	28	1124
MooTools 1.4.4	46	31	1152
Prototype 1.7	58	23	1010
YUI2 2.9.0	16	13	1186
YUI3 3.4.1	33	18	1650

The detailed list of errors and details about them can be found in our web server. **Issues revealed:** jQuery v1.6.2 has 19 critical and 28 high errors, meaning that since the last release 10 more critical errors were introduced. Prototype and MooTools are the worst ranked JFs.

3.3 Performance Tests

In order to conduct the performance tests we used the SlickSpeed Selectors test framework (code.google.com/p/slickspeed/).We tested the performance of each JF under 7 different browsers and 4 different operating systems (Win7,WinXP, WinVista, Ubuntu). Figure 1 illustrates the total execution times (the average among five separate runs) for each JF, on 7 browsers, on Windows 7. The times displayed are in milliseconds. The results on the other OS are similar. Information about the test environment (hardware, OS, etc.) and the detailed test results can be found on our web server.

Issues revealed: (1) IE8 produced very big execution times for MooTools, YUI2 and YUI3. Based on statistics, IE8 is used (January 2012) by 15%-28% of web users. (2) Comparing with tests on previous versions of the JFs (accessible from our server), we observed clear performance improvements (10% to 60%) between JFs versions released during the last year.

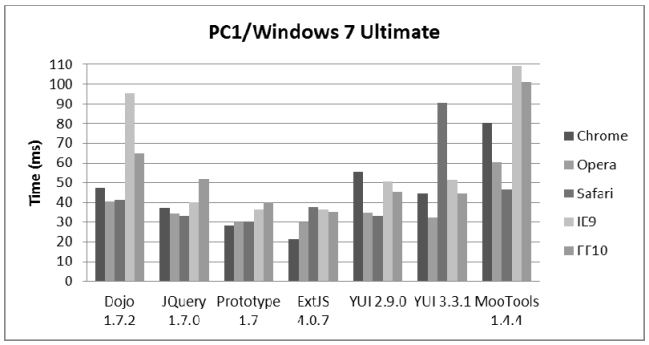

Figure 1. Performance of Frameworks per Browser

4. CONCLUSIONS

In this work, we tried to accomplish an overall quality, validity and performance assessment of the most commonly used JFs today. The results of the quality tests revealed some functions and points in code that probably need to be improved, while the validation tests revealed parts of code that must be modified in order to harmonize with browsers continuous evolvement.

Our intention was not to name the best framework, but to reveal to their supporting community their drawbacks and help them in producing high quality, full-featured JavaScript frameworks for the web developers. Regarding future work, we intend to use the same methodology and tools in order to evaluate the same JavaScript frameworks on mobile environments.

5. REFERENCES

[1] Barkmann, H., Lincke, R., Lowe, W., 2009. Quantitative evaluation of software quality metrics in open-source projects. In *Proceedings of The 2009 IEEE International Workshop on Quantitative Evaluation of large-scale Systems and Technologies (QuEST09)*, (2009)UK, Bradford

[2] Chuan, Y., Wang, H., 2009. Characterizing Insecure JavaScript Practices on the Web. In *Proceeding WWW '09 Proceedings of the 18th international conference on World Wide Web*. Madrid, Spain (April 2009), 964-965

[3] Olsina, L. and Rossi, G. 2002. Measuring Web Application Quality with WebQEM. *IEEE Multimedia*,9,4(2002), 20-29

[4] Ratanaworabhan, P., Livshits, B., Zorn, B.G., 2010. JSMeter: Comparing the behavior of JavaScript benchmarks with real web applications. In *USENIX Conference on Web Application Development (WebApps)*, (June 2010) 27–38.