A Comparative Evaluation of the Execution Behavior of JavaScript Benchmarks and Real-World Web Applications

J.K. Martinsen and H. Grahn
School of Computing, Blekinge Institute of Technology, Karlskrona, Sweden
`jkm@bth.se, hgr@bth.se`

Abstract: In this study, we argue that the execution behavior of the current JavaScript benchmarks differs from the behavior of real-world Web Applications. We have compared the first 100 Web Applications in the Alexa top-sites list and 5 Facebook use cases, against three established JavaScript benchmark suites, i.e., Dromaeo, SunSpider, and V8. Our measurements indicate that the JavaScript `eval` function is important in most Web Applications, the scripts associated with Web Applications may change between page visits to the same page, anonymous functions are used more extensively for in Web Applications, and the workload of Web Applications often differ significantly from the workload found in many of the benchmarks.

Keywords: Web Applications JavaScript Workload Characterization

1. Introduction

JavaScript is a dynamically typed, object-based scripting language with run-time evaluation often used to add interactivity to Web Applications. The execution of a JavaScript program is done in a JavaScript engine, i.e., an interpreter/virtual machine that parses and executes the JavaScript program. Several techniques have been proposed to increase the performance of the JavaScript engine along with a number of benchmark suites [1, 6, 7] to evaluate their performance. Some of these benchmarks have been ported from domains other than Web Applications, and we suspect that some of these tasks are rarely performed in Web Applications. Therefore, it might also exist functionalities and execution behaviors in Web Applications that might not be present in the benchmark suites.

We have used a profiler to evaluate and compare the execution behavior of real-world Web Applications against the established JavaScript benchmarks. We have profiled the performance of the first 100 entries in the Alexa top-sites list and for a set of selected use cases for some Web Applications. We have then compared these measurements with three established JavaScript benchmark suites [1, 6, 7]. Related work indicate that significant differences exist [2, 3, 4].

2. Experimental Methodology

We want to compare the execution behavior of a set of real-world Web Applications against a set of benchmarks. All our experiments are done on a Microsoft Windows XP platform, and the experiments are executed through Firefox 3.6. To extract information from the execution, we have used the FireBug 1.5 profiler [5].

Web Applications and the benchmarks differ since executing benchmarks require no user-interaction. The user interaction pattern might differ from time to time, both when performed automatically and manually. To solve this we use the AutoIt automation scripting language to repeat a certain task multiple times. We execute each application 10 times.

The second critical issue in this type of study is which benchmarks and Web Applications that can be considered as representative. We have identified three main established benchmark suites that are frequently used for evaluating the performance of JavaScript engines, i.e., Dromaeo [1] from Mozilla, V8 [6] from Google, and SunSpider [7] from WebKit.

We have selected the 100 most visited sites from the Alexa top-list as representatives of popular Web Applications, and profiled the start page of each. In addition to evaluating the JavaScript behavior of the first page, we have profiled the behavior of a set of predefined use cases for Facebook, e.g., login, searching for friends, sending messages to friends, and posted news.

3. Experimental Results

3.1 Usage of the `eval` function

We have measured the number of the `eval` function calls relative to the total number of function calls. Our results show that `eval` is used only in 4 out of 35 benchmark applications. However, in these four applications are, on average, 31% of the total number of function calls invocations of the `eval` function. For the Alexa top sites list, we find that `eval` is used more frequently. 44 out of the top 100 sites use the `eval` function. In average, 11% of all function calls are `eval` calls. Further, we have found that for some Web Applications, e.g., `sina.com.cn`, up to 55% of all function calls are `eval` calls.

3.2 Changing JavaScript functions and code between reloads

By reloading a Web Application we have discovered that the executed code might change between successive reloads. As JavaScript has a function such as `eval`, scripts can dynamically generate JavaScript code. We have found that several function names are unique for a page reload, suggesting that changes occur between reloads. We have observed this in 6 of the benchmarks. However, if we do not count the `eval` function calls, the function names remain static in all the benchmarks.

For the Alexa top 100 web-sites there were functions that changed between reloads. For some applications there were significant differences, e.g., for `deviantart.com` 74% of all function calls had unique function names. For 4 Web Applications, the relative number of unique function names were more than 0.5, and 23 out of 100 Web Applications had unique function names after 10 reloads. Function names such as `adOnload_`970558 (and similar) suggest that the function name is indeed unique and that many functions probably are created dynamically.

3.3 Anonymous function calls

An anonymous function call is a call to a function that does not have a name. We have discovered that some of the anonymous function calls in the benchmarks are instrumentation codes, e.g., to start a certain benchmark. In our results, we have removed such instrumentation calls.

Our results show that 18 of the 35 benchmark applications use anonymous function calls, and 74 out of 100 real-world Web Applications use anonymous function calls. If we calculate the average relative number of anonymous function calls, we find that the benchmarks use anonymous function calls more frequently. On average are 16% of all function calls in the benchmarks anonymous. For the real-world Web Applications, our results show that only 4.7% of the function calls are anonymous.

3.4 Distribution of function calls

Our results indicate that both the benchmarks and the Web Applications have a large number of short-running functions. However, most of the benchmarks have a few functions that account for most of the execution time., i.e., a “hot-spot” function. For the Web Applications, the workload seems to be more evenly distributed. In our selected Web Applications, *no* JavaScript function contributes to more than at most 39% of the total execution time.

3.5 Facebook use cases

We have created 5 use cases for Facebook and execute each of these use cases 10 times. For the Facebook use cases, we have measured the difference between the largest number of unique functions against the lowest number of unique functions for each of the 5 cases. We found a different number of unique function calls for 3 out 5 use cases, and for 2 of the use cases there were no unique functions after 10 reloads. The use case where we added multiple strings to a news feed, had the highest number of unique function calls.

Our results on the Facebook use cases confirm that the workload is more distributed among the functions than in the benchmarks in general, with no clear “hot-spots” found. This indicates that the execution behavior of the use cases, and also the Web Applications, is different from execution behavior of the benchmarks. While benchmark applications sought to solve a problem with a clear start and end, Web Applications often would perform multiple tasks rather than addressing one single task or problem.

4. Conclusions

In this study we have done a comparative evaluation of the execution behavior of established JavaScript benchmarks, i.e., Dromaeo [1], V8 [6], and SunSpider [7], and Web Applications from the 100 most used web sites on the Alexa top-list.

Our results indicate that the execution behavior of the Web Applications from the Alexa top sites differs from the benchmarks on several points. Important differences are the use of the `eval` function, code that changes between reloads of the same page, and the lack of functions in the Alexa top sites list that could clearly be categorized as performance “hot-spots”.

References

[1] Dromaeo. Dromaeo: JavaScript performance testing, 2010. `http://dromaeo.com/`.

[2] A. Nazir et al. Unveiling Facebook: a measurement study of social network based applications. In *IMC'08*, pages 43–56, 2008.

[3] G. Richards et al. An analysis of the dynamic behavior of javascript programs. In *Programming Language Design and Implementation (PLDI)*, pages 1–12, 2010.

[4] P. Ratanaworabhan et al. JSMeter: Comparing the behavior of javascript benchmarks with realweb applications. In *Webapps'10*, pages 27–38, 2010.

[5] FireBug. Firebug, javascript profiler, 2010. http://getfirebug.com.

[6] Google. V8 benchmark suite - version 5, 2010. `http://v8.googlecode.com/svn/data/benchmarks/v5/run.html`.

[7] WebKit. SunSpider JavaScript Benchmark, 2010. `http://www2.webkit.org/perf/sunspider-0.9/sunspider.html`.