Publication VIII

Ville Karavirta and Petri Ihantola. Automatic assessment of JavaScript exercises. In *roceedings of 1st Educators' Day on Web Engineering Curricula (WECU 2010)*, Volume 607 of *CEUR-WS*, Vienna, Austria, pages P9:1–P9:10, 2010.

Automatic Assessment of JavaScript Exercises

Ville Karavirta and Petri Ihantola

{vkaravir, petri}@cs.hut.fi
Aalto University, Department of Computer Science
P.O.Box 15400, FI-00076 Aalto, Finland

Abstract. In this paper, we present `js-assess` online playground for automatically assessed JavaScript programming exercises. We show how such automatic assessment systems can be implemented on top of JavaScript tools already used by the software industry. Furthermore, we demonstrate how automatically assessed assignments can be easily embedded to any web page without a server, as assessment runs inside the browser's JavaScript engine.

1 Introduction

Along with HTML and CSS, JavaScript (JS) is the language of the web. Thus, when web programming is part of the curricula, teaching JS is also important [1]. Several authors have argued that being able to produce something visible to the web is highly motivational [2–4]. It has also been suggested that scripting like languages are generally more suitable to introductory programming when compared to system programming languages like C or Java (e.g. [3, 5]). However, it should be noted that in practice, Python is more common first language, if a "scripting" language is preferred.

Despite all of the good reasons, JS has not become a mainstream teaching language. It is still a common misconception to consider JS simply a Java-like scripting language for flying banners as in the 90's. However, the language has functions as first class members, higher order functions, and closures, although it does also include the for statement and curly brackets borrowed from C/Java syntax. JS is often quoted to be "Lisp in C's clothing"[1]. Adding objects and prototype-based inheritance without a concept of classes contributes to JS being easily misunderstood among both learners and teachers. When we asked some of our our colleagues if they would consider teaching JS in CS1, the most common answer was that JS is too ugly and confusing.

Products like GMail and Facebook rely heavily on JS, there are many good tools for professionals (examples in Section 2), and well written JS books do exist (e.g. JavaScript: The Definitive Guide by David Flanagan and JavaScript: The Good Parts by Douglas Crockford). However, there are not that many educational tools supporting JS. For example, there are only few automatic assessment (AA) systems supporting JS [6–9]. Such systems give teachers more time to focus on improving other aspects of the course that cannot be automated [10].

[1] http://javascript.crockford.com/javascript.html

Automatic assessment helps both teachers and learners. Typically, the results can be used in summative assessment (i.e. grading). For students, automatic assessment provides valuable feedback (i.e. formative assessment) to guide learning while practicing his/her programming skills. Speed, availability, and consistency of grading are other typically mentioned benefits of AA [10]. The challenge in AA is to construct suitable feedback and measures for the selected learning goals. This means that the type of automatic assessment should be justified by pedagogical needs [11].

In this paper, we introduce a client-side learning environment with automatic assessment of JavaScript assignments. In Section 2, we highlight how different tools, already used by the software industry, can provide feedback from different features of JS programs. In Section 3, we describe how we built an automatic assessment environment/playground on top of these tools. Other systems capable to provide automatic feedback from web programming assignments are presented in Section 4. In section 5, we discuss the applicability and limitations of our solution and, finally, in Section 6 we conclude this paper and provide some thoughts on future directions.

2 Review of Industrial Tools Suitable for Assessment

In this section, we'll use Ala-Mutka's [11] categorization of automatically assessable features in programming exercises to present a selection of open source JS development tools. The features are divided between ones where the assessment is based on running the programs and ones built on top of static analysis. The aim is to underline how different tools support assessing different aspects of programming and, thus, different learning objectives. Furthermore, we focus on tools working inside a browser as we are building a client-side assessment system.

2.1 Dynamic analysis

Dynamic analysis refers to evaluating programs by executing them, possibly with different inputs. Ala-Mutka lists four aspects to assess using dynamic analysis: functionality, efficiency, testing skills, and special features.

Functionality The most common way to use automatic assessment is to test the functionality and correctness of students' solutions. In educational context, this is typically done by testing the output or return values of the student's program or smaller units of the program. Unit testing frameworks can be applied, but many automatic assessment systems have implemented their own ways to specify tests and related feedback.

Tools for JavaScript: Numerous tools to assess functionality can be divided between unit testing (e.g. JSUnit[2]) and web testing frameworks focusing

[2] http://www.jsunit.net/

on integration and system testing (e.g. Selenium[3]). Whereas XUnit clones call JS functions under test directly with desired values, web testing frameworks are designed to test behavior of rendered HTML by comparing the results of simulated user operations (such as clicks) on the web page and the expected results.

Efficiency Although Ala-Mutka focuses on time efficiency in her survey, efficiency is a wider topic. It can be related to the use of resources like time, memory, network, etc. One JS specific aspect of the efficiency is load time of the code. To reduce load times, various optimization methods like minimization, compression, and load order changes have been investigated. However, how the code is written in the first place still matters. Thus, this could be assessed in addition of traditional efficiency metrics when assessing larger JS programs.

Tools for JavaScript: The `hrtimer`[4] project includes tools for profiling JS applications, though only in Firefox. In addition, Dromaeo[5] is a system that runs JavaScript efficiency tests developed for different JS engines in the browser. Basically, it runs a piece of code and counts the number of times it was run in a certain amount of time. This system can be used to compare a reference implementation with a student solution. Load times, on the other hand, can be estimated from the size of the JS code, which can first be minified using, for example, JSMin[6].

Testing skills While the automated tests take care of assessing students' work, students should also be introduced to testing their own programs. Assessing testing skills of students is typically done by requiring students to write their own tests or test data and then evaluating how good that test or test data is by examining the code coverage. Since code targeting specific browsers (or browser versions) is sometimes needed, coverage data should be interpreted carefully. Getting full line coverage might not be possible by running the tests in a single browser only. This should not be required unless exercises are designed to avoid browser-specific solutions or course requirements require programs to function in only a certain browser.

Tools for JavaScript: When it comes to tools for assessing code coverage in JS, there are no suitable tools available for cross-browser client-side assessment. Tools requiring a server or a specific browser are available, though. JSCoverage[7] is a tool that executes the tests in the browser on the client-side and shows the line coverage. However, it requires instrumenting the code on server-side. Another choice is `hrcov`, a Firefox extension that works entirely on client-side. The `hrcov` is part of the `hrtimer` project.

[3] http://seleniumhq.org/
[4] http://hrtimer.mozdev.org/
[5] https://wiki.mozilla.org/Dromaeo
[6] http://crockford.com/javascript/jsmin
[7] http://siliconforks.com/jscoverage/

Special features Compared to server-side software that runs in a single environment, the JavaScript application needs to work on all the platforms the users of the site have. This adds a level of complexity to the testing setup and is perhaps the most obvious special feature related to AA of JS programs. Exercises can be designed to avoid browser specific issues but sometimes these are desirable learning goals. In real projects, JS programs have to work across browsers. Thus, feedback on how their solutions work in other browsers can be beneficial for students.

Tools for JavaScript: Several tools have been designed to automate the task of running the same tests on multiple browsers and to collect results obtained from different browsers. Obviously, these require a server to collect the results from the multiple clients. TestSwarm[8] and JsTestDriver[9] are distributed test drivers designed to ease running tests (implemented by using the existing browser based XUnit clones) on multiple browsers. Both provide a server where different browsers can connect. Server distributes tests to clients and collects the results.

2.2 Static analysis

Using static analysis the programs can be assessed without executing the code. Ala-Mutka lists five different features to assess statically: style, programming errors, software metrics, design, and special features.

Style Many programming languages have de-facto standards on how to style the code. Students are taught these conventions and the assessment can give feedback on following them. In JavaScript, style discussion has not stabilized and there are different, sometimes contradictory, guidelines available. Easiest choice, when using some of the JS libraries such as JQuery or Prototype, is to follow the style conventions of the selected library.

Tools for JavaScript: JSLint[10] is a useful tool for checking many aspects of JavaScript code. It can be used for assessing style as well, for example to check things like indentation, line lengths, correct use of curly braces, etc. In addition to lint tools, many integrated development environments (e.g. Eclipse) support pretty printing of JavaScript.

Programming errors Some programming errors can be discovered without executing code. Such errors can be, for example, unreachable code, or use of uninitialized variables.

Tools for JavaScript: Two most heavily used applications of this category are JSLint (already mentioned above) and JavaScript Lint[11]. Both are able to

[8] http://www.testswarm.com
[9] http://code.google.com/p/js-test-driver/
[10] http://www.jslint.com
[11] http://www.javascriptlint.com/

catch many typical programming errors in JavaScript such as dead code, missing semicolons at the end of the line, etc. JavaScript Lint can be used as a command-line tool, through an IDE-plugin, or online. JSLint is implemented in JavaScript and is also available as a web service.

Software Metrics Different metrics are easy to calculate, but the use should be pedagogically justified. Statement count, branch count, cyclomatic complexity, lines of code, lines of comments, percentage of lines containing comments, and code depth are useless for students unless accompanied by a desired value or range for the measure.

Tools for JavaScript: There are not many tools in this category. However, JSMeter[12] supports all the metrics mentioned above, runs inside a browser, and is enough for most needs.

Design Assessment of design is challenging (for humans as well) but also an important feature for many advanced programming exercises. There are very few tools to give feedback about the high level design in any language. Examples of recent work that could be used are done by Dong et. al. [12] to recognize design patterns from programs, and work by Taherkhani [13] to recognize different sorting algorithms through static analysis.

Tools for JavaScript: We are not aware of tools that could be used to automatically assess design of JavaScript programs.

Special Features Special features that could be assessed by using static analysis are typically requirements/restraints to use some language constructs. For example, the use of some keywords or functions of the language can be restricted. Another example, also mentioned by Ala-Mutka, is plagiarism detection. This is often built on static analysis to remove the effect of variable renaming, adding comments, etc.

Tools for JavaScript: We are not aware of JS tools falling directly into this category. However, in JavaScript, restrictions not to use some functions can be assessed statically by using regular expressions. Furthermore, it is often easier to re-implement prohibited functions to do nothing or to throw exceptions. However, both of these approaches are easy to bypass by talented students.

3 Open Source Playground with Automatic Assessment

We implemented a system called `js-assess` to provide automatic feedback from small JS assignments. The system is open source[13] and built on top of the industry tools already presented in Section 2. We decided to use students' own browsers to do the assessment. This makes the setup simpler and also trivializes

[12] http://code.google.com/p/jsmeter/
[13] http://github.com/vkaravir/jsassess-demo

sandboxing needed in dynamic analysis of unknown programs. Similar approach of running the tests on a student's machine has been proposed before [6, 14, 15].

Our `js-assess` tool combines an on-line editor and automatic assessment of the following features: functionality (using various testing frameworks), style (using JSLint), programming errors (using JSLint), and various software metrics (using JSMeter). Functionality, style, and programming errors are the most common features assessed automatically. Thus, they were natural choices to support. Assessment of software metrics was added because we wanted to see how it could be of benefit in assignments requiring mathematical problem solving with a focus on the efficiency of assessment. This is a typical feature assessed in programming competitions (e.g. [16]).

Our exercises need no server and are implemented purely in HTML and JavaScript. Benefit of this is that it allows the exercises to be embedded into any web page. However, the lack of a server implies that the feedback is not stored. Thus, the tool is, for now, suitable for self-studying only.

The design is simple and allows different assessment tools to be configured for each exercise. Technically, the actual assessment is done by running the selected tools in their own `iframe` elements and then reporting the results to the top HTML page. Figure 1 shows an example of an exercise. The editor and feedback do not need to to be side by side as in the figure but any CSS and HTML can be used to build the layout. We are using EditArea[14] as the editor. EditArea has some limited IDE features so it is well-suited for this kind of online coding playground.

Detailed configuration of the selected tools is also possible. For JSLint, there is a wide range of options for things to report on from indentation to restrictions on variable introduction. All these can be configured if the default behavior is not desired. In addition, data provided by JSMeter can be processed to provide more meaningful feedback based on it. This is implemented through simple filter functions teachers can write by themselves. The functionality can be tested using a wide range of JS testing frameworks, and the prototype comes with examples in QUnit and JSSpec.

Listing 1.1 is an example of what needs to be added to HTML to embed an automatically assessed assignment into the document. The text inside the `textarea` element on Line 2 is the initial skeleton code given to students. Div on line 7 is where the results of the assessment are added. This can be anywhere on the page. Line 8 should be replaced with loading of the required JS libraries. Finally, the `script` element on lines 9-14 includes the exercise specific configurations. These can include the filter function for JSMetrics, the options for JSLint, and the file(s) that contain the unit tests. Options also specify path to the sources of the assessment framework.

[14] http://sourceforge.net/projects/editarea/

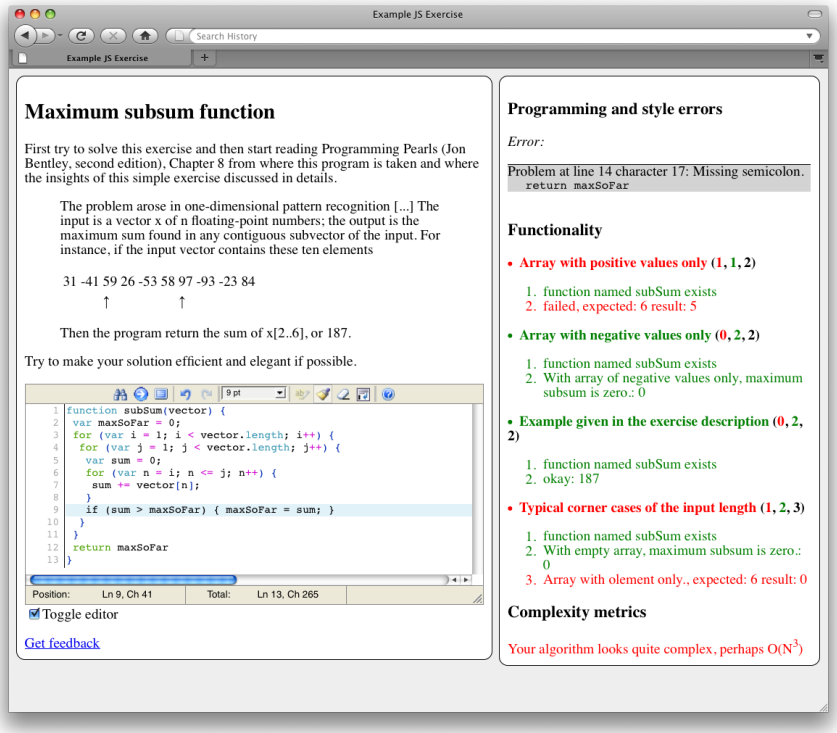

Fig. 1. Screenshot from the tool with the code on the left and feedback on the right.

```
<form>
  <textarea id="jsassess-editor">// initial code for students</textarea>
</form>
<div id="exerciseButtons">
  <a href="#" id="submitButton">Get feedback</a>
</div>
<div id="jsassess-feedback"></div>
<!-- script src... -->
<script>
  exerciseOptions = {
    'jsmeter': jsmeterFilters.defaultFilter,
    'jslint': true, 'tests': ['sub_sum_qunit.html'],
    'commonspath': '../javascript/'};
</script>
```

Listing 1.1. Embedding an exercise into HTML.

3.1 Assignment Collection

The assignments available in our current collection include: the always-required HelloWorld, mathematical problems demonstrating the feedback from code met-

rics and efficiency, JSON (JavaScript Object Notation) and DOM manipulation, augmenting native JavaScript objects, and generating HTML which requires understanding modularity in JavaScript. Exercises can be tested and sources are available online[15].

3.2 First Experiences with Students

We have tested our exercises on a really small special course on JavaScript programming. This experience helped us smooth out some of the problems especially with the usability of the system and the output of the various tools. On the course, the students were able to successfully solve the assignments currently implemented in our tool, as well as create new assignments for each other to solve.

4 Related Work

Google Code Playground[16] and W3Schools interactive demonstrations[17] both allow writing of JavaScript and execution of the programs. However, both are missing the automatic assessment aspects, i.e. the results of verification that the code does what it is supposed to do, style conventions are followed, etc. Furthermore, it is difficult to integrate these tools into other websites.

Most of the real AA systems designed to give feedback from web programming assignments have the focus on integration testing whereas our approach to give feedback from the functionality is based on unit testing. For example, AWAT [9] is an environment for web programming assignments where students submit an URL of a site they developed. Teacher defines the components that should exist on the web page and tests by using the Watir Ruby library. Testing is then performed by using Internet Explorer from the submission server. Authors are planning to publish AWAT as open source. Apogee [7] and Electronic Commerce Virtual Laboratory [6] are both similar to AWAT.

A different approach to train web programming skills is to present a ready-made web system or excerpts from such to students. In Stingray [17] students are then asked to do various small assignments related to the security of the given software. For example, students can reverse engineer given JS to find SQL injection vulnerabilities. Other sites similar to Stringray exist[18].

5 Discussion and Future Work

JavaScript is an important programming language to be taught, but there are only few tools for automatic assessment of students' programs. Such tools are

[15] http://www.cs.hut.fi/u/vkaravir/jsassess/
[16] http://code.google.com/apis/ajax/playground/
[17] http://www.w3schools.com/JS/
[18] http://www.hackthissite.org/ and http://www.hackerslab.org/

important to let teachers focus on other teaching activities that can not be automated. To fill this gap, we have surveyed several tools used in JavaScript development and explained how those could be adopted to provide feedback from JS programming exercises. We have presented `js-assess` – a tool that demonstrates how the industrial tools can be integrated to give feedback from functionality, style and programming errors, and various software metrics. Novelty of our prototype is that automatically assessed JS exercises can be embedded to any webpage with little effort.

Future work is still needed to improve `js-assess`. Not all the features listed in Section 2 are supported currently with support for efficiency, testing skills, design assessment, and special features like cross-browser compatibility missing. Some of them offer interesting challenges to add without a server. For example, feedback for students' testing skills can be provided by using JSCoverage, which needs a server.

In addition, AJAX topics are clearly of interest when teaching JavaScript. AJAX exercises would also be possible by writing a simple server simulator with simulated network delays etc. all in JS. In this way, calls to a server could be practiced without the need of a real server.

Furthermore, although the novelty of `js-assess` is the serverless approach, it could be beneficial to integrate it into existing submission/course management system such as CourseMaster [18] due to the ease of tampering with grading data on the browser-side. This would enable the use of the system as part of the course grading.

From a pedagogical perspective, future research is still needed to understand how JavaScript should actually be taught, what libraries should be used, and where in the curricula JS should be introduced. In real projects, JavaScript is rarely used as is – without libraries like Closure[19], jQuery[20], or Prototype[21]. Because of the dynamic nature of JS, libraries can modify the JS base objects, which implies that some code snippets can be interpreted differently based on the libraries loaded before the snippet. Thus, JavaScript libraries are more than libraries in most other languages as they affect what the language looks like. Although it is clear that after learning the very basics of the language, some libraries should be used, it is not clear how to select which to use. What are the pedagogical and other arguments behind this decision? For example, Protype emphasizes object-orientation more than jQuery but on the other hand, there there might be more project using JQuery than Prototype.

References

1. Wang, X.: A practical way to teach web programming in computer science. Journal of Computing Sciences in Colleges **22**(1) (2006) 211–220
2. Mahmoud, Q.H., Dobosiewicz, W., Swayne, D.: Redesigning introductory computer programming with html, javascript, and java. In: Proceedings of the 35th

[19] http://code.google.com/p/closure
[20] http://jquery.com
[21] http://prototypejs.org

SIGCSE technical symposium on Computer science education, New York, NY, USA, ACM (2004) 120–124
3. Mercuri, R., Herrmann, N., Popyack, J.: Using html and javascript in introductory programming courses. In: Proceedings of the 29th SIGCSE technical symposium on Computer science education, New York, NY, USA, ACM (1998) 176–180
4. Wu, P.: Teaching basic game programming using javascript. J. Comput. Small Coll. **24**(4) (2009) 211–220
5. Warren, P.: Teaching programming using scripting languages. J. Comput. Small Coll. **17**(2) (2001) 205–216
6. Coffman, J., Weaver, A.C.: Electronic commerce virtual laboratory. In: SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education, New York, NY, USA, ACM (2010) 92–96
7. Fu, X., Peltsverger, B., Qian, K., Tao, L., Liu, J.: Apogee: automated project grading and instant feedback system for web based computing. In: SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education, New York, NY, USA, ACM (2008) 77–81
8. Hwang, W.Y., Wang, C.Y., Hwang, G.J., Huang, Y.M., Huang, S.: A web-based programming learning environment to support cognitive development. Interacting with Computers **20**(6) (2008) 524 – 534
9. Sztipanovits, M., Qian, K., Fu, X.: The automated web application testing (awat) system. In: ACM-SE 46: Proceedings of the 46th Annual Southeast Regional Conference, New York, NY, USA, ACM (2008) 88–93
10. Carter, J., English, J., Ala-Mutka, K., Dick, M., Fone, W., Fuller, U., Sheard, J.: ITICSE working group report: How shall we assess this? SIGCSE Bulletin **35**(4) (2003) 107–123
11. Ala-Mutka, K.: A survey of automated assessment approaches for programming assignments. Computer Science Education **15**(2) (2005) 83–102
12. Dong, J., Sun, Y., Zhao, Y.: Design pattern detection by template matching. In: Proceedings of the 2008 ACM symposium on Applied computing, New York, NY, USA, ACM (2008) 765–769
13. Taherkhani, A., Malmi, L., Korhonen, A.: Algorithm recognition by static analysis and its application in students' submissions assessment. In Pears, A., Malmi, L., eds.: Proceedings of 8th Koli Calling International Conference on Computing Education Research (Koli Calling 2008), Uppsala University (2009) 88–91
14. Sant, J.A.: "mailing it in": email-centric automated assessment. In: ITiCSE '09: Proceedings of the 14th annual ACM SIGCSE Conf. on Innovation and technology in computer science education, New York, NY, USA, ACM (2009) 308–312
15. Wang, H., Yin, B., Li, W.: Development and exploration of chinese national olympiad in informatics (cnoi). Olympiads in Informatics **1** (2007) 165–174
16. Forišek, M.: On the suitability of programming tasks for automated evaluation. Informatics in education **5**(1) (2006) 63–76
17. Gutierrez, F.: Stingray: a hands-on approach to learning information security. In: SIGITE '06: Proceedings of the 7th conference on Information technology education, New York, NY, USA, ACM (2006) 53–58
18. Higgins, C., Symeonidis, P., Tsintsifas, A.: The marking system for CourseMaster. In: Proceedings of the 7th annual conference on Innovation and Technology in Computer Science Education, ACM Press (2002) 46–50