SwingJS: Giving Java applets new life as JavaScript equivalents, applications to education and science

Udo Borkowski

abego Software GmbH
Aachen, Germany
ub@abego-software.de

Robert M. Hanson, Tahir Ahsan, Nikesh Yadav, Nadia El Mouldi, Andrew Lee, Andreas Raduege

St. Olaf College, Northfield, MN, USA
[hansonr|ahsan1|yadav1|elmoul1|leeas]@stolaf.edu, ajraduege@gmail.com

Zhou Renjian

Software Developer
Shanghai, China
zhourenjian@gmail.com

Abstract: Java applets for web applications have become outdated and clumsy, requiring the user-install of Java, updated security certificates, and use of specific, generally nonstandard, browsers. Hundreds of web sites depending upon such applets are no longer functional. By converting Java applets to JavaScript, we can regain the functionality of these valuable educational resources, returning these web pages to active service to education and science. We created SwingJS, a system to convert Java applets to JavaScript without need for extensive rewriting of the Java code or starting de novo. The JavaScript "applets" are created in parallel with standard Eclipse-based Java development, along with their Java counterparts simultaneously, in real time. The result is that we have the original functional Java applet or stand-alone application, and we have its virtually identical JavaScript counterpart, with all the layout and event-driven functionality of the original. Several important applications relevant to Chemistry and Physics, such as Jmol, JSpecView, Physlets, and PhET, were successfully converted.

1 Background and Motivation

Java Applets, already introduced with the first version of Java in 1995, provide for more nearly 20 years an easy way to deploy interactive, visual applications using web browsers. In the early years, the functionality of applets exceeded the capabilities of web applications written in JavaScript and HTML, both in performance and feature richness.

In education and science, applets provided a way for more dynamic, interactive courseware and complex visualizations. As a consequence, a lot of applets were created, covering a wide range of domains and subjects, such as chemistry, physics, mathematics, and physiology. [3,4,5,6,7]

Since 2012, however, more and more web browser vendors have reduced or dropped their support for Java applets, and most mobile browsers never supported them at all. The lack of support for applets led to the deprecation of the Applet technology with Java 9 in 2017.

All this has been bad news for hundreds of web sites that have depended on applets and are now essentially non-functional. SwingJS [1] was created to conserve these valuable assets and avoid cost-intensive rewrites of the applet code. SwingJS is an open source project. Its goal is to allow Java Swing (and, to a lesser extent, pre-Swing) applets to be ported to JavaScript/HTML5 with a minimum of extra manual work.

2 Java2Script as the Foundation

SwingJS derives from the successful port of the Jmol Java Applet [3] to JavaScript/HTML5 (JSmol [4]) using Java2Script [2], a project going back to the year 2005. Java2Script (J2S) provides an Eclipse Java-to-JavaScript compiler plugin and additional JavaScript libraries to be used at runtime in the web browsers. The plugin translates Java source code to JavaScript code, which can then be executed in web browsers.

3 Swing for the Web Browsers

As Java2Script only implemented a very limited subset of the Java Runtime Environment (JRE), especially lacked the Swing GUI Toolkit [8], a major part of our initial work was the development of modified Swing components and “platform look and feel” classes designed specifically for use in JavaScript. This work, started in 2016 and dubbed “SwingJS”, was a major extension of the JSmol project, which does not use either Swing or the Java Abstract Window Toolkit (AWT) for its JavaScript user interface. The code is based on the OpenJDK [9] implementation of the Java standard libraries.

To the greatest extent possible, the standard Java class code was not modified and runs in JavaScript form exactly as it would in Java. For instance, the Java AWT event queuing system is reproduced identically in JavaScript. Window layout is handled via layout managers exactly as in Java. The most significant changes were made in the form of a new HTML5-compatible “platform look and feel” Java package, which, when compiled into JavaScript, leverages standard HTML5 objects, such as canvas, table, image, and textarea as implementations of Java-based equivalents.

The implementation of the libraries was carried out in an incremental way over the course of 19 months (June 2016 – December 2017). Applets with more and more capabilities were automatically converted to JavaScript, where they were validated against their

Java equivalents in real time. When problems occurred, such as missing implementations in the runtime library or systematic errors in transpiling or class processing, additional classes were added, the compiler was fixed, or the runtime "virtual machine" was revised. This afforded a steadily increasing coverage of Swing capabilities in JavaScript. This step-by-step approach allowed for efficient work, as individual applets typically only use a small subset of Swing, and extra work on 'unused features' could be avoided.

4 Transpiler and Runtime Core

While migrating applets and implementing the runtime libraries, several deficits of the original Java2Script transpiler and runtime core appeared. Substantial rewriting of Java code was required to get around these problems. While this was practiced initially in 2016, it quickly became obvious that it was not a sustainable approach. Therefore, a major rewrite of central parts of the Java-to-JavaScript transpiler and runtime core was undertaken in June through December of 2017. The rewritten Java2Script transpiler and runtime solves all of the issues seen in previous versions. As a bonus, the resulting code is faster and completely modular.

As the new implementation has proven to be superior over the original Java2Script one, the SwingJS code was merged into the Java2Script project in 2018, thus becoming the current Java2Script transpiler and runtime.

5 Development Environment

SwingJS provides several features that simplify the development workflow. Within the Eclipse framework, the compiler automatically supplements the .class file output with a matching set of .js files along with the automatic generation of HTML test files for every applet or class containing a public static void main(String[] args) method.

As the transpiler is part of the incremental Eclipse Java compiler, if a project is set to build automatically, every saved change in the Java code is simultaneously reflected in the corresponding JavaScript code, which is immediately testable in an external browser. A reload of the test page or any page the user has developed shows the changes directly and allows rapid testing, debugging, and validation.

To support automatic build pipelines, JavaScript compilation and compression can also be used from the command line.

6 Relation to Other Systems

Other systems that allow the translation of Java (source or byte code) to JavaScript [10], such as Google Web Toolkit (GWT) [11] are designed for a completely different purpose. The goal of SwingJS is to enable *concurrent nondivergent Java and JavaScript development* and the migrating of existing Java applet code written using AWT or Swing to JavaScript. These other systems utilize their own specialized GUI widgets for JavaScript-only applications. In addition, they are quite limited in terms of JRE coverage. For example, GWT supports only 256 Java classes. A notable omission is the dynamic loading of classes using Class.forName(). In contrast, SwingJS loads classes only as needed and currently supports over 1400 classes, including over half of the 2000 classes in the java and javax.swing packages.

7 Limitations

SwingJS does not support multithreading, as web browsers used to run all JavaScript code in a single thread. This means that java.lang.Thread.sleep(), Thread.wait(), and Thread.notify() have no JavaScript counterpart. As a result, a certain amount of reworking of the Java code for a project must be done by hand for applets that use these Java features. Our experience, though, is that, with a little clever coding, this has always been possible, and the result is indistinguishable from a fully threaded user experience.

The successful migration of over 100 Java applets in less than two years has been accomplished by a handful of developers, several of whom, with a little help from the SwingJS team, have been working independently.

References

[1] https://github.com/SwingJS/SwingJS (accessed 4/18/2018)

[2] https://java2script.github.io/java2script/, http://j2s.sourceforge.net (accessed 4/18/2018)

[3] Hanson, R. M. J. Appl. Cryst., 43, 1250-1260 (2010); https://sourceforge.net/projects/jmol/ (accessed 4/18/2018)

[4] Hanson, R. M., Prilusky, J. , Renjian, Z. , Nakane, T. and Sussman, J. L. Isr. J. Chem., 53, 207-216 (2013); https://sourceforge.net/projects/jsmol/ (accessed 4/18/2018)

[5] Moore, E. B., Chamberlain, J. M., Parson, R., Perkins, K. K., J. Chem. Educ, 91, 1191-1197 (2014); https://phet.colorado.edu (accessed 4/18/2018; mix of HTML5 and Java applets)

[6] http://www.falstad.com/mathphysics.html (accessed 4/18/2018; SwingJS-converted applets)

[7] http://thevirtualheart.org/CAPindex.html (accessed 4/18/2018; nonfunctional Java applets)

[8] https://docs.oracle.com/javase/6/docs/api/javax/swing/package-summary.html (accessed 4/18/2018)

[9] http://openjdk.java.net (accessed 4/18/2018)

[10] https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS#javajvm (accessed 4/18/2018)

[11] http://www.gwtproject.org (accessed 4/18/2018)