A Limited JavaScript Compiler for Mozilla Firefox 3

Aarlo Stone Fish
CPSC 490

Introduction

Web browsers generally compile JavaScript into bytecode and then interpret the bytecode. I will write a module for the Mozilla Firefox 3 web browser to compile JavaScript code into native machine code. The module will also run the code. This will be for a subset of JavaScript.

JavaScript

JavaScript is an implementation of EcmaScript for web browsers. EcmaScript is “a dynamic, weakly typed, prototype-based language with first-class functions”\(^1\). The web browser downloads JavaScript code along with HTML code and executes the code after it renders the page. Using JavaScript, a programmer can manipulate the Document Object Model, the tree-like structure of the web page.

How Web Browsers Run JavaScript

Web browsers can run JavaScript code in one of several ways:

- Interpreting the JavaScript code line-by-line, after some possible pre-processing. This is the simplest and slowest way. It is used by Safari's WebKit before SquirrelFish.

- Compiling the JavaScript code into bytecode and then executing the bytecode – much like how Java is run. This is faster than interpreting line-by-line. It is currently used by Safari's SquirrelFish and Firefox 3's SpiderMonkey.

- Compiling the JavaScript code into native machine code and then executing the machine code. If done correctly, this can be the fastest way. There are many complications, however. For example, compilation speed, usually not important, is now crucial.

This method is used by Google Chrome's V8 JavaScript engine and Firefox 3.1's TraceMonkey (which uses another optimization, described

\(^1\) http://en.wikipedia.org/wiki/JavaScript
below, called “trace trees”). I am bringing a limited version of this method to Firefox 3.

Mozilla Firefox 3

Mozilla Firefox is an open source web browser. It is a product that has gone through many iterations, from version 1.1 in 2005 to version 3.0, released on June 17, 2008. Firefox 3 uses the Gecko layout and rendering engine, which uses SpiderMonkey to compile JavaScript into bytecode and then run the bytecode. The Gecko rendering engine alone has more than two million lines of code.

TraceMonkey

The version of Firefox currently under development, Firefox 3.1, uses TraceMonkey. TraceMonkey itself is also still under development. TraceMonkey compiles JavaScript code into machine code and then executes it. Initial tests show that TraceMonkey can be between two and thirty-eight times as fast as SpiderMonkey, the Firefox 3 rendering engine.

I may use TraceMonkey as a model. My code will be much simpler than TraceMonkey, however, for two reasons. First, as described below, my compiler will only be for a subset of JavaScript. Second, TraceMonkey includes lots of optimizations that are outside of our scope. Its main improvement is called tracing. “In simple terms tracing works by watching for commonly-repeated actions (such as loops, function calls, or type checking) and tries to optimize their resulting native code into the lowest number of actions.”

This project

In this project, I will write a limited JavaScript compiler and then integrate it with Firefox 3 code. My code, to both compile and run JavaScript, will replace SpiderMonkey, the module that currently handles JavaScript in Firefox 3.

A full JavaScript compiler, implementing all of the features of JavaScript, and interfacing exactly with Firefox, is too much work for a one-semester project. The baseline goal, then, is to have some JavaScript code compile and run inside of Firefox 3. In the best-case scenario, it would implement many JavaScript features.

The compiler may leave out some of the following features of JavaScript: interfacing with the Document Object Model of the page; garbage collection;
function closures; the “eval” statement, which allow the programmer to evaluate arbitrary JavaScript expressions on the fly; some built-in JavaScript objects like Date or Math; the JavaScript standard library with built-in JavaScript functions; other features that seem out of scope.

Timeline

- Replace SpiderMonkey with dummy code so that Firefox uses this code to handle JavaScript: 2/11
- Run a dummy executable from this code: 2/16
- Standalone Ecamscript compiler: 4/14
- Integrate compiler into Firefox: 4/24
- Finish loose ends and make final abstract, written report, and web pages: 5/1

Deliverables

- A build of Firefox that compiles and runs some version of JavaScript.
- Final abstract, written report, and web pages.