’’

Beyond Ṣcript

Implementing A Language For The Web

Veit Heller

June 17, 2016

A thesis submitted for the degree of
B.Sc. of Applied Computer Science of
The University of Applied Sciences Berlin

For Meredith, Tobias and all the people who cope with me. Your undying support will not be forgotten.

Except where otherwise indicated, this thesis is my own original work.

Veit Heller
June 17, 2016

Abstract

The modern web is comprised of an abundance of very different beasts. Technologies that powered the first versions of the World Wide Web, such as HTML, CSS and JavaScript, and relatively new conceptions like TypeScript, CoffeScript, PureScript, ClojureScript, Elm, LASS, SCSS, Jade and Emscripten - to name but a few - are shaping the internet as we know it. There is a flaw that many of the new technologies have in common, as different as they may look and feel - they are mere preprocessors. In the end, it all boils down to the classic technologies again and we are left with the same programming we have been doing for the last twenty years.

This thesis presents a port of the zepto programming language to the web. It aims to work as seamlessly with existing technologies as possible.

Contents

1. Introduction

> Controlling complexity is the essence of computer programming.
>
> *(B. Kernighan)*

1.1. Motivation

JavaScript has, since its inception, attracted a lot of controversy. This is rooted in various aspects of its design, from prototypal inheritance to operator precedence. Prototypal inheritance has the reputation of being counter-intuitive, though it is older than JavaScript, the first commonly known programming language that implements prototypal objects being Self.

* things get better
* es 6 and es7 thank god
* a lot of research funneled into it
* still a fundamental rethinking might be necessary

1.2. Purpose of this work

1.3. Structure of this work

2. Motivation

> Practicality beats purity.
>
> *(T. Peters—The Zen of Python)*

A common saying among programming language designers is that every programmer has written their own implementation of Lisp. There are a lot of different implementations of Lisp in the wild, even ones that compile to JavaScript[1].

The main reason for that is often cited to be the simplicity of the language on a parsing level. A simple Lisp can be implemented in less than one hundred lines of code, if no intermediate representation is generated. This is made possible by the unique property of Lisp of enclosing every statement in parentheses, where the first element within those parentheses is the statement and the other elements are the arguments. It can be evaluated straight from a textual level, because things such as operator precedence and statement amiguity do not exist. In regular Lisp as specified in the initial paper by John McCarthy(**GCM**) only six special forms exist to allow not only for Turing-completeness, but also for expressiveness.

2.1. zepto

Zepto is a new Scheme implementation that aims to be as small as possible, to be able to target a lot of different backends. Currently, LLVM and Erlang Core[2] bindings are under development, the reference implementation is a simple interpreter that interprets code directly from the Abstract Syntax Tree (AST). This is slow but ensures a small interpreter sizeThe entire codebase is about 4000 lines of Haskell code. The compilers are written directly in zepto itself.

* chosen because small
* makes a good counter-example to JavaScript
* a functional language as opposed to an object-oriented one

[1] such as ClojureScript.

[2] Erlang Core is the Intermediate Representation (IR) of Erlang code before it is complied. Resources and documentation about it are sparse, it mostly seems to exist inside the BEAM's implementation.

3. Implementation

> It always takes longer than you expect, even when you take into account Hofstadter's Law.
>
> *(Hofstadter's Law)*

4. Outlook

When I'm working on a problem, I never think about beauty. I think only how to solve the problem. But when I have finished, if the solution is not beautiful, I know it is wrong.

(R. Buckminster Fuller)

5. Conclusion