# Beyond .*Script

**Implementing A Language For The Web**

Veit Heller

April 26, 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 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
April 26, 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.

All of this is for a good reason: having fewer technologies means that browsers and clients have to care for only a handful of things instead of fighting a hydra. However, there is undoubtably potential lost when we are focussing on what we already have instead of what could be. There is not a lot of research going on into real alternatives for the web, meaning embedded languages and technologies.(1) The author believes that experimenting with these alternatives is worth trying.

To substantiate his ideas, the author created a small runtime for a R5RS-like language with a foreign function interface into Javascript, some syntactical additions and a fairly extensive standard library. The language can be used within the context of webpages without being preprocessed, given that the runtime was already loaded, with all the usual comfort and discomfort of web pages.

# Contents

# Abbreviations

| | |
|---|---|
| **AST** | Abstract Syntax Tree. 3 |
| **IR** | Intermediate Representation. 3 |

# 1. Introduction

Controlling complexity is the essence of computer programming.

*(B. Kernighan)*

## 1.1. Motivation

There is a profound hatred of many programmers, who are mostly not related with web programming in the first place, for JavaScript. Its' design was critized since its' initial inception in 1993. Brendan Eich, its' designer, was mocked by traditional desktop programmers for a long time, and it is a fairly recent change for people to step forward and say that they like JavaScript. This is partially due to its' extremely vivid ecosystem and a plethora of frameworks, libraries and learning material. The other reason is that recent changes have made the language more manageable; not only are many syntactic additions such as classes and lambdas and interface reworks included in the latest standards ECMAscript 6 and 7, languages that compile to JavaScript have also risen in popularity.

This popularity has given way for a new dogma: JavaScript is the assembly of the web. But not all of it is good, particularly as an assembly language. This is the reason why the creator of the popular Emscripten transpiler located a subset of the language that he calls ASM.js, a highly optimizable dialect that is extremely hard to write for humans. All of this work comes at a cost, though, and that is transparency. One of the reasons the web works with an interpreted language and not bytecode in the first place is security - compiling and minifying it goes straight against that intuition.

Maybe it is too late to change all of this.

## 1.2. Purpose of this work

## 1.3. Structure of this work

# 2. Motivation

Practicality beats purity.

(T. Peters—*The Zen of Python*)

# 3. Implementation

If you look into the abyss for long enough, bytecode will look back at you.

*(*F. Nietzsche—*approximately)*

Intermediate Representation (IR) Abstract Syntax Tree (AST)

# 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

## 5.1. Summary of contributions

#### Some bollocks

I did some bollocks, and it was pretty interesting.

#### Some other bollocks

Really, if you think about it, I did a whole other lot of bollocks as well.

# A. An appendix

Put any appendices here—they are just like regular chapters, except they follow the `\appendix` directive.

## References

Aho, A. V., M. S. Lam, R. Sethi, and J. D. Ullman (2006). *Compilers: Principles, Techniques, and Tools*. Pearson Eudcation, Inc.

Fogus, M. (2013). *Functional JavaScript*. O'Reilly Media.

Parr, T. (2010). *Language Implementation Patterns*. The Pragmatic Programmers, LLC.

Queinnec, C. (2003). *Lisp in Small Pieces*. Cambridge University Press.

# List of Figures

# List of Tables