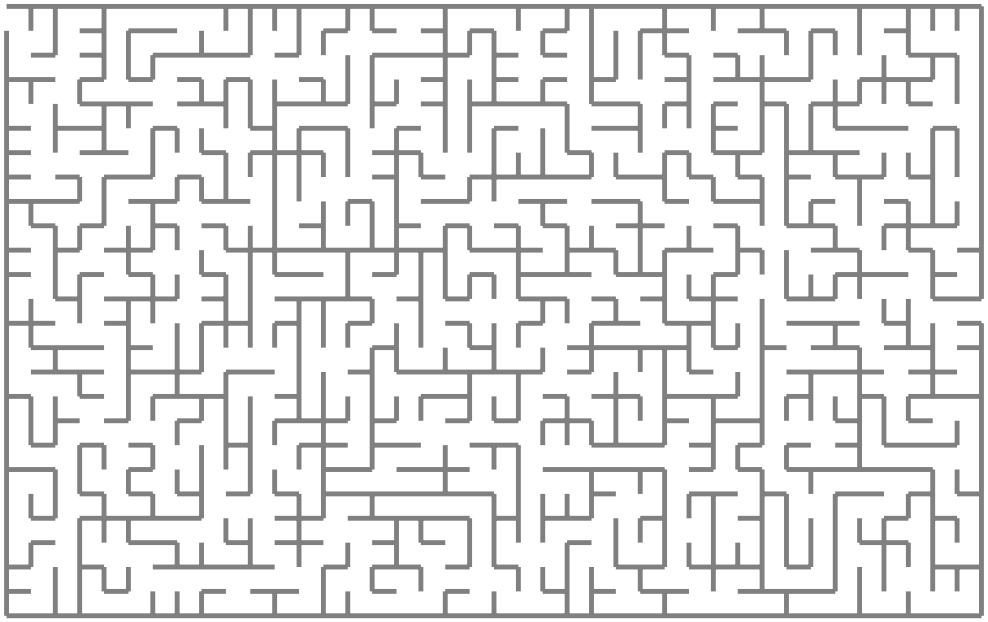
# Modernizing the WebDSL front-end: A case study in SDF3 and Statix

*Version of February 4, 2021*



Max Machiel de Krieger

# Modernizing the WebDSL front-end: A case study in SDF3 and Statix

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Max Machiel de Krieger
born in Delft, the Netherlands

Programming Languages Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Cover picture: Random maze.

# Modernizing the WebDSL front-end: A case study in SDF3 and Statix

Author: Max Machiel de Krieger
Student id: 4705483
Email: M.M.deKrieger@student.tudelft.nl

**Abstract**

WebDSL is a domain-specific language for web programming that is being used for over ten years. As web applications evolved over the past decade, so did WebDSL. A complete formal specification of WebDSL has been *TO-DO: check if missing or not updated* since its original development. With the introduction of Statix in the Spoofax language workbench, a declarative language that generates a typechecker, we made an elegant and practical formal semantics for WebDSL.

Thesis Committee:

| | |
|---|---|
| Chair: | Prof. dr. E. Visser, Faculty EEMCS, TU Delft |
| Committee Member: | Dr. A. Katsifodimos, Faculty EEMCS, TU Delft |
| University Supervisor: | Ir. D. M. Groenewegen, Faculty EEMCS, TU Delft |
| Expert: | Ir. A. Zwaan, Faculty EEMCS, TU Delft |

# Preface

Preface here.

Max Machiel de Krieger
Delft, the Netherlands
February 4, 2021

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Many different programming languages exist, with many different properties and advantages.

The implementation of a programmig languages can be seperated into two parts: the front-end and the back-end.

The front-end is the part of the programming language with which the user interacts (the syntax, early feedback using analysis results) and the back-end is the part that makes the programming language operational (optimization, code generation).

While the back-end of a programming langauge makes it work, the front-end defines how a user experiences the progrmaming language.

Programming languages are constantly evolving, requiring updates to its specification and implementation. One such language is WebDSL.

WebDSL is a domain-specific language for developing web applications, developed at the Delft University of Technology.

Because of its academic nature, many research projects added features to the language, all contributing to the success of existing WebDSL applications.

The downside of having many different contributors adding new features, is that the development experience that comes from the front-end leaves much to be desired. (too harsh?)

Currently, the WebDSL implementation is composed of multiple definitions in meta-DSLs supported by the Spoofax language workbench: the syntax is defined in SDF2 and the desugaring, typechecking, optimization and code generation is defined in the term transformation language Stratego.

The interleaving of all the latter processes in the same Stratego environment poses a threat to the readability and maintainability of the WebDSL language.

In this thesis, we will be focusing on modernizing the WebDSL front-end, by implementing the syntax definition in SDF3 and the static analyses in Statix.

In this process, we aim to answer the following questions:

- Are SDF3 and Statix capable of modeling a programming language used in practice?
- Is the resulting WebDSL front-end performant?
- Is the resulting WebDSL front-end user friendly?

## 1.1 Contributions

This thesis provides the following contributions:

- We present the biggest case study for SDF3 and Statix thusfar.
- We present a new, modernized WebDSL front-end.

- We present a formalization of the WebDSL static semantics in the form of a Statix specification.

## 1.2 Outline

The rest of this thesis is structured as follows. In chapter 2 we describe WebDSL and its features. Next, in chapter 3 we will give an overview of what a modernized front-end is, and what our solution looks like. Chapter 4 and 5 go in detail about the implementation of the WebDSL front-end in SDF3 and Statix respectively. The result of this implementation is evaluated in chapter 6 and compared with related work in chapter 7. Finally, chapter 8 concludes this thesis.

# Chapter 2

# WebDSL

In this chapter, we describe WebDSL. WebDSL is a domain-specific language for developing web applications. The language incorporates ideas from various web programming frameworks and produces code for all tiers in a web application (Groenewegen, Chastelet, and Visser 2020). Ever since its introduction over 10 years ago (Visser 2007), WebDSL has been the subject of many published papers (cite some papers here) and on top of that, is the programming language underpinning several applications used daily by thousands of users. Examples of WebDSL applications include but are not limited to:

- **WebLab**: An online learning management system, used by the Delft University of Technology.
- **conf.researchr.org**: A domain-specific content management system for conferences, used by all ACM SIGPLAN and SIGSOFT conferences.
- **researchr.org**: A platform for finding, collecting, sharing, and reviewing scientific computer science related publications.

The rest of this chapter showcases the different aspects of WebDSL and zooms in on its non-trivial features. First, in section () we will describe how WebDSL offers functionality for creating web user interfaces. Next, in section () we illustrate how the langauge manages data models. Thirdly, section () contains information about WebDSL's solution for access control and in section () we highlight interesting aspects of its general-purpose object oriented function code. We concluse this chapter by going in detail about WebDSL's current implementation in section ().

## 2.1 User Interfaces

- Syntax
- Data binding and action code
- Template overriding
- Template overloading
- Dynamically scoped redefines

## 2.2 Data Model

- Syntax

- Inheritance
- Extending entities

## 2.3 Access Control

- Syntax
- Inferred visibility
- Nested rules
- Pointcuts

## 2.4 Functions

- Syntax
- Entities as classes
- Hooks for entity setters
- Extending functions

# Chapter 3

# Modernization

## 3.1 Spoofax language workbench

- History
- Goal
- Achievements

## 3.2 Current implementation of WebDSL

- Large Stratego specification where desugaring, static analysis, optimization and code-generation are interleaved (exaggeration?)
- Side effects using dynamic rules.
- Unexpected consequences of changes due to limited static analysis in untyped setting.

Go over some interesting WebDSL features and how they are implemented:

- Access control
- Template overloading and overriding
- Entity extension

## 3.3 Goal

- A complete and maintainable SDF3 and Statix specification of WebDSL.
- Gather insight into the capabilities, elegance and performance of SDF3 and Statix.
- (Incrementalization for free leveraging the parallel Statix solver)

# Chapter 4

# WebDSL in SDF3

- Introduce SDF3

# Chapter 5

# WebDSL in Statix

- Introduce Statix

(Hemel et al. 2011)
Covered by current implementation:

- Action code
- Entities
- Entity extension
- *Access control* (*TO-DO: name and type resolution*)
- *UI* (*TO-DO: everything except definitions*)

TO-DO:

- Template elements
- Access control name resolution
- Module system
- Native Java Classes
- HQL
- Form submit action code such as returns, goto, replace, etc

# Chapter 6

# Evaluation

## 6.1 Correctness

- Defining correctness in absence of a formal specification
- How correct is the implementation WebDSL
- Explain correctness
- Edge cases

## 6.2 Validation

- How elegant in the Statix implementation?

## 6.3 Performance

- Explain metrics and methods
- Results
- Discuss results

## 6.4 Evaluating Statix

- Repeat reasons for using Statix
- What worked out as intended?
- What did not work as intended?
- What are the workarounds?
- Recommendations for improving Statix

# Chapter 7

# Related work

- Papers about WebDSL
- Papers about modern Spoofax
- Papers about the definition of web programming languages
- Papers about modernizing (web) programming languages

# Chapter 8

# Conclusion

## 8.1 Future work

- Use analysis results for back-end
- Incrementalization in back-end?

# Bibliography

Groenewegen, Danny M., Elmer van Chastelet, and Eelco Visser (2020). "Evolution of the WebDSL Runtime: Reliability Engineering of the WebDSL Web Programming Language". In: *Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming*. '20. Porto, Portugal: Association for Computing Machinery, pp. 77–83. ISBN: 9781450375078. DOI: 10.1145/3397537.3397553. URL: https://doi.org/10.1145/3397537.3397553.

Hemel, Zef et al. (2011). "Static consistency checking of web applications with WebDSL". In: *Journal of Symbolic Computation* 46.2. Automated Specification and Verification of Web Systems, pp. 150–182. ISSN: 0747-7171. DOI: https://doi.org/10.1016/j.jsc.2010.08.006. URL: http://www.sciencedirect.com/science/article/pii/S0747717110001367.

Visser, Eelco (2007). "WebDSL: A Case Study in Domain-Specific Language Engineering". In: *Generative and Transformational Techniques in Software Engineering II, International Summer School, GTTSE 2007*. Ed. by Ralf Lämmel, Joost Visser, and João Saraiva. Vol. 5235. Lecture Notes in Computer Science. Braga, Portugal: Springer, pp. 291–373. ISBN: 978-3-540-88642-6. DOI: 10.1007/978-3-540-88643-3_7. URL: http://dx.doi.org/10.1007/978-3-540-88643-3_7.

# Acronyms

**AST** abstract syntax tree

**DSL** domain-specific language

# Appendix A

# A