Masterarbeit

Statische Typsysteme für JavaScript

Entwicklung eines Transpilers von Flow nach TypeScript

Zur Erlangung des akademisches Grades eines
Master of Science

angefertigt von Jonathan Gruber (68341)

Fakultät Informatik, Mathematik und Naturwissenschaften
Studiengang: Master Informatik (16INM/VZ)

Erstprüfer	Prof. Dr. rer. nat. habil. Frank
Zweitprüfer	M. Sc. Michael Lückgen

ausgegeben am 11. Juni 2019
abgegeben am TBA 2019

Zusammenfassung

Die vorliegende Masterarbeit beschäftigt sich mit Transpilierung statischer Typsysteme für JavaScript. Dabei werden zwei derzeit populäre Systeme betrachtet:

Inhaltsverzeichnis

1 Motivation

1.1 JavaScripts Typsystem

Historische Entwicklung usw blabla. Warum ist das Typsystem eher nicht so gut? [2]

1.2 Sinnhaftigkeit statischer Typsysteme

Überleitung: genau deswegen brauchen wir statische Typsysteme. Da hätten wir zwei…

1.3 Zielsetzung und Aufbau der Arbeit

2 Grundlagen

… und diese werden nun sogleich näher beschrieben:

2.1 Statische Typsysteme für JavaScript

Es gibt noch viele weitere: https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS

2.1.1 Flow

Flow beschreiben (und zwar mit entsprechender Fachsprache)

Basistypen

Hilfstypen

Deklarationen

2.1.2 TypeScript

TS beschreiben (und zwar mit entsprechender Fachsprache)

2.2 Quelltext-Transformation durch Transpilierung

Was macht eigentlich so ein Transpiler? Hier Theorie (AST etc.)

2.2.1 Theoretische Grundlagen

Parser, AST etc.

2.2.2 Babel

...als populärer Vertreter eines JavaScript-Compilers

2.3 Evaluation bestehender Transpilierungs-Ansätze

Erläutern, dass es da schon eine Handvoll Ansätze auf GitHub gab, aber die alle nicht einsatzbereit waren.

3 Ziel- und Anforderungsanalyse

3.1 Beschreibung der Ausgangslage

Was geht so bei TeamShirts? Wie verwenden wir Flow? Wie ist die Typisierung bisher (eher implizit, explizit etc)?

3.2 Ziele der angestrebten Migration zu TypeScript

Hypothesen, Wünsche, Hoffnungen

3.2.1 Erkennung neuer Bugs und Typfehler

3.2.2 Unterstützung externer Bibliotheken und Frameworks

3.2.3 Stabilität und Geschwindigkeit des Typsystems

3.2.4 Zukunftssicherheit und Transparenz der Technologie

3.3 Anforderungen an den Transpiler

3.3.1 Korrekte Übersetzung der Flow-Typen nach TypeScript

3.3.2 Semantisch äquivalente Transpilierung des Quelltexts

3.3.3 Programmatische Verarbeitung eines gesamten Projekts

3.3.4 Beibehaltung der Quelltext-Formatierung

4 Umsetzung des Transpilers

4.1 Software-Architektur

4.2 Entwicklungsprozess

Irgendwo sollte wohl geschrieben werden, dass TypeScript, TDD usw. verwendet wurde, um das Plugin zu bauen…

4.3 Implementierung als Babel-Plugin

4.3.1 Transpilierung der Basistypen

4.3.2 Transpilierung der Hilfstypen

4.3.3 Transpilierung der Deklarationen

4.3.4 Weitere Optimierungen

Übersetzung gängiger Typimporte

Konvertierung von Class Decorators

Mapping der Importe (verschiedene Typnamen in Flow und TS), Umwandlung der Decorators usw.

4.4 Erweiterung als Kommandozeilenprogramm

4.5 Formatierung des Ausgabequelltexts

Prettier, synchronisieren der Leerzeilen und Kommentare beschreiben usw.

5 Durchführung der Migration

5.1 Transpilierung der Projekte

5.2 Manuelle Behebung neuer Typfehler

6 Ergebnisse

6.1 Semantische Äquivalenz des Ausgabeprogramms

6.2 Erkennung neuer Typ- und Programmfehler

6.3 Verfügbarkeit und Qualität externer Typdefinitionen

6.4 Performance der Typüberprüfungen mittels TypeScript

6.5 Formatierung des Ausgabequelltexts

7 Auswertung und Diskussion

7.1 Bewertung der Ergebnisse hinsichtlich der Zielvorgabe

7.2 Vergleich des Transpilers mit konkurrierenden Ansätzen

8 Schlussbetrachtung

8.1 Zusammenfassung

8.2 Ausblick

Literatur

[1] *MIT License*. Massachusetts Institute of Technology. URL: https://opensource.org/licenses/MIT (besucht am 01. 07. 2019) (siehe S. VI).

[2] Charles Severance. „JavaScript: Designing a Language in 10 Days“. In: *Computer* 45 (Feb. 2012), S. 7–8. ISSN: 0018-9162. DOI: 10.1109/MC.2012.57. URL: doi.ieeecomputersociety.org/10.1109/MC.2012.57 (siehe S. 1).

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcode-Listings

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig verfasst habe. Ich versichere, dass ich keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahren gewesen ist.

Leipzig, den 2. Juli 2019

_____

Jonathan Gruber

A Quelltexte

A.1 Transpiler (Reflow)

Der Quelltext des im Zuge dieser Arbeit entwickelten Transpilers ist unter folgendem GitHub-Repository vollständig einsehbar:

https://github.com/grubersjoe/reflow

Das Projekt wurde unter der MIT License [1] veröffentlicht.