JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Franz Miketta

Submitted at
Institute for System Software

Supervisor
tbd

Co-Supervisor
Dr. Christian Wirth

Co-Supervisor
Dr. Daniele Bonetta

March 20, 2021

CONTRIBUTING THE ECMASCRIPT PROPOSAL “MODULE BLOCKS” INTO GRAAL.JS

Bachelor Thesis

to obtain the academic degree of

Bachelor of Science

in the Bachelor’s Program

Informatik

JOHANNES KEPLER UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

JOHANNES KEPLER
UNIVERSITY LINZ

Sworn Declaration

I hereby declare under oath that the submitted thesis has been written solely by me without any third-party assistance, information other than provided sources or aids have not been used and those used have been fully documented. Sources for literal, paraphrased and cited quotes have been accurately credited.

The submitted document here present is identical to the electronically submitted text document.

Linz, March 20, 2021

JOHANNES KEPLER
UNIVERSITY LINZ

Abstract

The thesis aims at implementing an ECMAScript proposal from the TC39 proposals, the module blocks, into the existing Graal.js interpreter. Besides the initial task of changing the parser and the runtime and unit tests, the base framework for shipping code between processes is implemented and explicit benchmarks are shown. The final product contributes to the open source version of Graal.js on the platform github.com.

Zusammenfassung

Die vorliegende Arbeit verfolgt das Ziel einen ECMAScript-Antrag aus den TC39-Anträgen, die module blocks, im vorliegenden Graal.js-Interpreter zu implementieren. Nach der Implementierung im Parser und der Runtime und entsprechenden Tests wird weiters ein Framework zum Übertragen von Code zwischen Prozessen eingerichtet und dessen Performance mit ursprünglichen Implementierungen verglichen. Die finale Implementierung wird über die Plattform github.com der Open-Source-Version von Graal.js hinzugefügt.

JOHANNES KEPLER
UNIVERSITY LINZ

Contents

JOHANNES KEPLER
UNIVERSITY LINZ

List of Figures

JOHANNES KEPLER
UNIVERSITY LINZ

List of Tables

JOHANNES KEPLER
UNIVERSITY LINZ

1 Introduction

The following paragraphs serve as an introduction to the thesis by explaining the relevance both for *GraalVM* and *Graal.Js* in general and in the specific case of the ECMAScript proposal *module blocks*. The last paragraph explains the outline of the thesis.
In 2019 Oracle released the GraalVM with active development up to present days. GraalVM is a multilingual runtime with multiple core features such as certain compiler optimizations, ahead-of-time compilation, polyglot programming, LLVM runtime and the Truffle language implementation framework. With the amount of supported programming languages and the foregoing mentioned features the GraalVM can be implemented into a variety of production environments. One environment seems of particular interest: Microservices on server. The GraalVM native image was able to lower startup time and memory footprint by a significant amount as graph 1.1 shows. These savings won over the social media platform Twitter which Microservices run on GraalVM and GraalVM in return created savings for their CPU times which makes it have an environmental impact. The other core feature of GraalVM is the support of multiple languages via the Truffle framework. With this framework different languages can be implemented on top of GraalVM. The different language implementations are split up into single projects for each language. This setup leads to the project Graal.Js.
Graal.Js is the Truffle implementation of ECMAScript on the GraalVM. ECMAScript, with unofficial name JavaScript, is on spot three on the PYPL popularity index behind Java and Python indicating the programming language's relevance. From this relevance need arises to address the project's relevance. Graal.Js is the interface, on top of Truffle implemented on the GraalVM, between polyglot programs consisting of JavaScript, Java and other languages. This standalone feature lets the interpreter have a stand-out position on all the engines. Coming from other relevant engines like V8 or SpiderMonkey which cannot support polyglot programs GraalVM is on par with their feature support of ECMAScript 6 and newer standards. An important matter for relevancy of an engine is obviously the feature support. With new features being added to the standard these features have to be implemented into the engines which brings us to the new feature to be implemented module blocks.
Module blocks is an ECMAScript proposal by Daniel Ehrenberg and Surma based on inline modules. Inlining modules is a feature which is missing from the current ECMAScript. The absence has resulted into workarounds with various issues. ECMAScript cannot share code between processes thus residing on a single thread. Every module, worker and worklet needs a separate file cluttering project folders. Tasks short of stringification cannot be shared across agents. The multi-

tude of problems cited can be addressed by module blocks. Module blocks is a stage 2 proposal on ECMAScript 262 and is most likely to be implemented in the 2022 version of ECMAScript. Since it has a high relevance for polyglot programs its implementation is key to staying on top of the technology stack which brings us to the purpose of this thesis.
The main objective of this thesis is to implement the new ECMAScript stage 2 proposal module blocks into the Graal.Js engine which is implemented via the Truffle framework. Furthermore testing for this implementation will be conducted. When the general workings are set a code shipping framework and benchmarks will be included.

2 Background

3 Implementation

4 Evaluation

5 Future Work

6 Conclusions

JOHANNES KEPLER
UNIVERSITY LINZ

Bibliography

[1] Daniel Taupin. *MusiXTEX*. 1.30. 2020.