TECHNISCHE HOCHSCHULE MITTELHESSEN
CAMPUS GIESSEN
MNI Mathematik, Naturwissenschaften und Informatik

Bachelorarbeit

Vorlage für eine Abschlussarbeit mit LaTeX

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

vorgelegt dem
Fachbereich Mathematik, Naturwissenschaften und Informatik
der Technischen Hochschule Mittelhessen
von

Burkhardt Renz

im Januar 2018

Referent: Prof. Dr. Donald E. Knuth
Korreferent: Prof. Dr. Leslie Lamport

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Gießen, 12. September 2016

Abstract

Diese Arbeit befasst sich mit WebAssembly (abgekürzt Wasm). WebAssembly ist eine neue low-level Sprache die nahezu native Laufzeitperformanz bietet und von allen großen Browsern unterstützt wird. Nach einer Einführung, die einen ersten Überblick bietet, wird WebAssembly erläutert. Hierbei wird hauptsächlich auf die Architektur, die Konzepte und Unterstützte Programmiersprachen eingegangen.

Im Anschluss geht es um die Verwendung im Browser und zukünftige Features von WebAssembly. Nach dem ein größeres Verständnis zu der Technologie vorliegt, werden die Vorteile und Anwendungsmöglichkeiten erläutert. Im darauffolgenden Kapitel geht es um die Verbreitung dieser Technologie.

Der Ausblick zeigt welche Möglichkeiten WebAssembly in der Zukunft hat und argumentiert die Erfolgsaussichten. In der Diskussion wird ein kritisches Fazit gezogen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einführung

Für sehr lange Zeit war JavaScript die einzige Möglichkeit für Interaktive Anwendungen im Browser. Rechenintensive Aufgaben, wie z. B. Spiele, wurden von der schlechten Perfomance JavaScripts zurückgehalten. Trotz der unaufhöhrlichen Optimierung von JavaScript-Engines genügt die Performance nicht immer den hohen Anforderungen für das Web [10]. Ein erster Versuch dieses Problem anzugehen war asm.js. In der Theorie konnte mit dieser Technologie nahezu native Performance erzielt werden [14], jedoch war asm.js nie in allen Browsern konsistent schnell [11]. Der Hauptgrund hierfür war, dass asm.js kein offizieller Web-Standard war. Bei WebAssembly ist dies anders, es ist seit 2017 offiziel von allen großen Browsern (Chrome, Edge, Firefox und Webkit) unterstützt [13] und seit 2019 offizieler Standard für das Web [12]. Weitere Gründe warum WebAssembly schneller als asm.js ist werden unter 3.2 im Detail erläutert.

WebAssembly ist ein Binärinstruktionsformat für eine stackbasierte virtuelle Maschine. Es ist designed um als Kompilierziel von low-level Programmiersprachen wie C/C++/Rust zu dienen, hierdurch wird die Auslieferung im Web für Client- und Server-Anwendungen ermöglicht [1]. Die Nutzungsmöglichkeiten beschränken sich nicht nur ausschließlich auf das Web, jedoch liegt das Hauptaugenmerk auf diesem Gebiet. Das Ziel von WebAssembly ist Clientseitige Applikationen, auf verschiedenen Plattformen, in der Ausführungszeit so nah wie möglich an native Applikationen zu bringen und dabei effizient zu sein. Außerdem bietet WebAssembly neben der low-level Assemblersprache ein Menschenlesbares Textformat (WAT - WebAssembly Text Format) um Entwicklern zu helfen den Code zu lesen und die Fehlersuche zu vereinfachen. Dieses Textformat kann auch genutzt werden um Code zu schreiben und diesen in das Binärformat zu kompilieren.

Mit WebAssembly wird nicht versucht JavaScript zu ersetzen. JavaScript ist und bleibt für die meisten Anwendungsfälle die dominante Sprache im Web. WebAssembly und JavaScript sollen jedoch auf verschiedene Weise zusammen arbeiten. Zum Beispiel können Internetseiten wie üblich aufgebaut bleiben aber um schnelle WebAssembly-Module ergänzt werden. Diese Module können dann rechenintensive Aufgaben wie, Simulationen, Bild-/Ton-/Video-Verarbeitung, Visualisierung, Animationen, Kompression und Verschlüsselung übernehmen [2].

Ziel dieser Arbeit ist es, WebAssembly mit allen Vor- und Nachteilen zu untersuchen und dem Leser einen Einblick in diese noch neue Technologie zu geben.

2 WebAssembly

Seit dem Entwicklungsstart im Jahr 2015 hat WebAssembly

2.1 Architektur

2.2 Konzepte

2.3 Unterstützte Sprachen

2.4 Verwendung im Browser

2.5 Zukünftige Features von WebAssembly

3 Vorteile und Anwendungsmöglichkeiten

3.1 Stärken

3.2 WebAssembly vs. asm.js

4 Verbreitung

4.1 Nutzungszwecke

5 Ausblick

6 Diskussion

Literaturverzeichnis

[1] WebAssembly Webiste, https://webassembly.org/, 17. Mai, 2020.

[2] WebAssembly FAQ, https://webassembly.org/docs/faq/, 17. Mai, 2020.

[3] Conrad Watt *Mechanising and Verifying the WebAssembly Specification*, https://www.cl.cam.ac.uk/~caw77/papers/mechanising-and-verifying-the-webassembly-specification.pdf, 2018.

[4] Micha Reiser und Luc Bläser. *Accelerate JavaScript Applications by Cross-Compiling to WebAssembly*, https://dl.acm.org/doi/pdf/10.1145/3141871.3141873, 2017.

[5] Abhinav Jangda, Bobby Powers, Emery D. Berger und Arjun Guha. emphNot So Fast: Analyzing the Performance of WebAssembly vs. Native Code, https://www.usenix.org/system/files/atc19-jangda.pdf, 2019.

[6] Andreas Haas, Andreas Rossberg, Derek l. Schuff, Ben L. Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai und JF Bastien. *Bringing the Web up to Speed with WebAssembly*, https://people.mpi-sws.org/~rossberg/papers/Haas,Rossberg,Schuff,Titzer,Gohman,Wagner,Zakai,Bastien,Holman-BringingtheWebuptoSpeedwithWebAssembly.pdf, 2017.

[7] Conrad Watt, Andreas Rossberg und Jean Pichon-Pharabod. *Weakening WebAssembly*, https://www.cl.cam.ac.uk/~jp622/weakening_webassembly.pdf, 2019.

[8] Marius Musch, Christian Wresnegger, Martin Johns und Konrad Rieck. *New Kid on the Web: A Study on the Prevelance of WebAssembly in the Wild*, https://www.sec.cs.tu-bs.de/pubs/2019a-dimva.pdf, 2019.

[9] Christoph Alladoum. *Understanding WebAssembly*, https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/understanding-web-assembly.pdf, 2018.

[10] Mozilla. *ARE WE FAST YET?*, https://arewefastyet.com, 2017

[11] Alon Zakai. *Why WebAssembly is Faster Than asm.js*, https://hacks.mozilla.org/2017/03/why-webassembly-is-faster-than-asm-js/, 2017.

[12] W3 Website, *World Wide Web Consortium (W3C) brings a new language to the Web as WebAssembly becomes a W3C Recommendation*, https://www.w3.org/2019/12/pressrelease-wasm-rec.html.en, 2019.

[13] W3 Lists Website, *WebAssembly consensus and end of Browser Preview*, https://lists.w3.org/Archives/Public/public-webassembly/2017Feb/0002.html, 2017.

[14] Alon Zakai und Robert Nyman. *Gap between asm.js and native performance gets even narrower with float32 optimizations*, https://hacks.mozilla.org/2013/12/gap-between-asm-js-and-native-performance-gets-even-narrower-with-float32-opti 2013.