HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

[Thema der Bachelorarbeit]

Sandro Tonon

Konstanz, 15.02.2016

BACHELORARBEIT

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

an der

Hochschule Konstanz

Technik, Wirtschaft und Gestaltung

Fakultät Informatik

Studiengang Angewandte Informatik

Thema: [**Thema der Bachelorarbeit**]

Bachelorkandidat: Sandro Tonon, Allemannenstraße 10, 78467 Konstanz

1. Prüfer: Prof. Dr. Marko Boger
2. Prüfer: Dipl. Ing. Andreas Maurer

Ausgabedatum: [Datum]
Abgabedatum: 15.02.2016

Zusammenfassung (Abstract)

Thema:	[Thema der Bachelorarbeit]
Bachelorkandidat:	Sandro Tonon
Firma:	Seitenbau GmbH
Betreuer:	Prof. Dr. Marko Boger Dipl. Ing. Andreas Maurer
Abgabedatum:	15.02.2016
Schlagworte:	[Platz, für, spezifische, Schlagworte, zur, Ausarbeitung]

[Text der Zusammenfassung etwa 150 Worte. Es soll der Lösungsweg beschrieben sein.]

Ehrenwörtliche Erklärung

Hiermit erkläre ich *Sandro Tonon, geboren am 02.07.1990 in Waldshut-Tiengen*, dass ich

(1) meine Bachelorarbeit mit dem Titel

[**Thema der Bachelorarbeit**]

bei der Seitenbau GmbH unter Anleitung von Prof. Dr. Marko Boger selbständig und ohne fremde Hilfe angefertigt und keine anderen als die angeführten Hilfen benutzt habe;

(2) die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Konstanz, 15.02.2016

(Unterschrift)

Inhaltsverzeichnis

Abkürzungsverzeichnis

HTTP Hypertext Transfer Protocol

HTTP Hypertext Transfer Protocol

LOL Laughing Out Loud

1 Einleitung

Der Begriff “Web Components” ist ein Dachbegriff für mehrere entstehende Standards [Schaffranek, 2014], die es für Webentwickler ermöglichen sollen, komplexe Anwendungsentwicklungen mit einer neuen Sammlung an Werkzeugen zu vereinfachen. Diese sollen die Wartbarkeit, Interoperabilität und Kapselung verbessern und somit ein Plugin-System für das Web schaffen. Durch die neuen Standards soll das Web zu einer Plattform werden, die es ermöglicht die Web-Sprache selbst, HTML, zu erweitern. Dies ist bisher nicht möglich, da die HTML-Technologie, und somit die Möglichkeiten HTML-Tags zu benutzen, vom W3C definiert und standardisiert wird. Unter den wichtigsten der neuen Standard sind die folgenden vier Technologien aufzuführen: Custom Elements, Shadow DOM, HTML Templates und HTML Imports. Custom Elements ermöglichen es einem Webentwickler eigene HTML-Tags und deren Verhalten zu definieren, oder bereits vorhandene oder native HTML-Tags zu erweitern. Das Shadow DOM stellt einen Sub-DOM in einem HTML Element bereit, welcher dem Element zugehöriges Markup, CSS und JavaScript kapselt. HTML Templates stellen, wie der Name impliziert, einen Template-Mechanismus für HTML bereit [Kröner, 2014b] und erlauben das Laden von HTML-Dokumenten in andere HTML-Dokumente [Kröner, 2014a].

Diese neuen Technologien werden allerdings noch nicht von den populärsten Browsern, zu welchen Google Chrome, Mozilla Firefox, Opera und der Internet Explorer, bzw. Edge, gehören, unterstützt. Des Weiteren ist das Implementieren einer Applikation, welche diese Technologien nativ benutzt, bisher sehr komplex und schwierig zu organisieren. Im Zuge dessen, entwickelt Google aktiv an einer Bibliothek namens Polymer, welche sich diesen Problemen annimmt. Polymer stellt dabei eine Reihe an unterschiedlichen Schichten dar, welche den Umgang mit Web Components vereinfachen sollen. So stellt Polymer eine Sammlung an Mechanismen bereit, die älteren Browsern die nötigen Features für den Einsatz von Web Components beibringen. Ebenso soll das Erstellen von eigenen HTML-Elementen mit der Polymer-Bibliothek, und der damit bereitgestellten API, für Entwickler komfortabel gemacht werden. Um nun bereits entwickelte Web Components einfach wiederverwenden zu können, bietet Polymer eine Sammlung von fertigen Polymer-Elementen an.

Web Components und die Polymer Bibliothek greifen stark in den Entwicklungsprozess von Webseiten ein und sollen diesen Verbessern und vereinfachen. Die Seitenbau GmbH interessiert sich stark für diese neue Technologie, da Wiederverwendbarkeit, Wartbarkeit und neue Technologien im Fokus des Frontend Engineerings des Unternehmens stehen. Die Seitenbau GmbH ist ein mittelständischer IT-Dienstleister und unterstützt seit 1996 Organisationen aus Privatwirtschaft und öffentlicher Verwaltung bei der Planung, Konzeption und Umsetzung hochwertiger Softwarelösungen für E-Business und E-Government. Zu den Kernkompetenzen der Seitenbau GmbH zählen dabei vor allem

das Frontend Engineering & Content Management, die Konzeption und Entwicklung von Individualsoftware sowie der Aufbau von personalisierten Intranet- & Portallösungen.

Im Rahmen dieser Bachelorarbeit sollen die verschiedenen Technologien unter dem Dachbegriff Web Components, sowie deren Funktionsweise ohne, als auch mit der Polymer Bibliothek, untersucht werden. Zur Veranschaulichung soll eine Web Component mit Hilfe von Polymer implementiert, und mit einer ähnlichen Implementierung mit AngularJS verglichen werden. Am Beispiel einer Web Component in Form einer Multi-Navigations-Applikation sollen die Vor- und Nachteile des Einsatzes von Polymer in Hinblick auf Implementierung und Performanz dargestellt werden.

1.1 Polyfills mit webcomponents.js

In den Abschnitten 2.2 bis 2.5 wurde gezeigt, wie die Web Component Technologien funktionieren und ob diese bereits in allen Browsern unterstützt werden. In diesem Abschnitt wird darauf genauer eingegangen, was Polyfills sind, sowie deren Browserunterstützung und Performanz, und wie sie die Browserunterstützung der Web Components Technologien verbessern.

1.1.1 Native Browserunterstützung von Web Components

In den einzelnen Unterkapiteln zu den Technologien wurde jeweils kurz gezeigt, ob sie von den Browsern unterstützt wird oder nicht. Es wurde deutlich, dass Chrome und Opera bisher die einzigen Vorreiter sind. Bis auf HTML Templates, welche von allen modernen Browsern unterstützt werden, unterstützen sie als einzige alle Technologien. [citeulike:13914379]

Chrome

Hat alle Spezifikationen der Web Component Standards ab Version 36 komplett implementiert.

Firefox

Unterstützt nativ HTML Templates, Custom Elements und Shadow DOM sind zwar implementiert, müssen aber manuell mit dem Flag `dom.webcomponents.enabled` in den Entwicklereinstellungen aktiviert werden. HTML Imports werden, wie in Kapitel erwähnt, bis auf weiteres nicht unterstützt (Siehe Kapitel 2.5).

Safari

HTML Templates werden ab Version 8 unterstützt, Custom Elements und Shadow DOM befinden sich in der Entwicklung (Stand Januar 2016), HTML Imports werden jedoch nicht unterstützt.

Internet Explorer

Als einziger Browser unterstützt der Internet Explorer keine der Web Components Technologien. Die Unterstützung wird, auf Grund der Einstellung der Entwicklung und des Wechsels zu Microsoft Edge, auch nicht nachträglich implementiert werden.

Microsoft Edge

Templates werden ab Version 13 unterstützt, über die Entwicklung der restlichen Technologien kann allerdings abgestimmt werden [citeulike:13914237].

Mobile Browser

Alle Technologien werden bisher nur auf Android in den Browsern, Chrome für Android, Opera und Android Browser unterstützt.

Die Unterstützung der modernen Browser ist also noch verhalten, wird sich aber stark verbessern. Das bedeutet jedoch nicht, dass die Web Components noch nicht verwendet werden können. Mittels JavaScript besteht die Möglichkeit deren Funktionalitäten den aktuellen Browsern, welche Web Components nicht unterstützen, sowie noch älteren

Browsern beizubringen. Das hierfür benutzte JavaScript wird Polyfill genannt, mit dessen Hilfe können die Funktionen auf alle relevanten Browser portiert werden.

1.1.2 Polyfill webcomponents.js

> A polyfill, or polyfiller, is a piece of code (or plugin) that provides the technology that you, the developer, expect the browser to provide natively. Flattening the API landscape if you will. [citeulike:13914241]

Mit Hilfe von JavaScript kann eine Technologie also auch in Browsern benutzt werden, welche die Technologie nicht unterstützen. Mit Hilfe von Polyfills können Technologie-Lücken in Browsern auf mehrere, unterschiedliche Arten ("Poly") gestopft ("fill") werden [citeulike:13914234]. Eine Sammlung an Polyfills für die Verschiedenen Technologien der Web Components bildet das JavaScript webcomponents.js. Es wurde von Google im Rahmen der Polymer Entwicklung entwickelt und wurde so weit verbreitet, dass entschlossen wurde, es auszugliedern, damit es auch unabhängig von der Benutzung von Polymer eingesetzt werden kann [citeulike:13914239].

1.1.3 Browserunterstüzung

Mit dem Einsatz der webcomponents.js Polyfills werden die Web Components auch auf den Internet Explorer, Firefox sowie Safari portiert. Eine detaillierte Matrix der Browserunterstützung der Web Components mit Einsatz der Polyfills ist in Abbildung X dargestellt.

images/6-webcomponentsjs-browserunterstuetzung.jpg

Abbildung 1.1: Bild: Browserunterstützung der Web Components Technologien mit webcomponents.js

Jedoch werden auch trotz Einsatz des Polyfills nur die aktuelleren Versionen des jeweiligen Browsers unterstützt. Darunter fallen weiterhin nicht beispielsweise der Internet Explorer in Version 8 und 9. Des weiteren werden einige Technologien auf Grund der Komplexität nicht komplett simuliert. Hier muss bei einigen Technologien auf folgende Punkte geachtet werden.

Custom Elements

Die CSS Pseudoklasse :unresolved wird nicht unterstützt.

Shadow DOM

Der Shadow DOM kann auf Grund der Kapselung nicht komplett künstlich simuliert werden, dennoch versucht das webcomponents.js Polyfill einige der Features zu simulieren. So sprechen definierte CSS Regeln alle Elemente in einem künstlichen Shadow Root an - Als würde man den >>> Selektor benutzen - auch die `::shadow` und `::content` Pseudoelemente verhalten sich so.

HTML Templates

Templates die mit einem Polyfill erzeugt werden sind nicht unsichtbar für den Browser, ihre enthaltenen Ressourcen werden also schon beim initialen Laden der Seite heruntergeladen.

HTML Imports

Die zu importierenden HTML Dateien werden mit einem XHR Request, und somit asynchron heruntergeladen, selbst wenn das `async`-Attribut (siehe Abschnitt 2.5.8 - Asynchrones Laden von Imports) nicht gesetzt ist.

1.1.4 Performance

Das webcomponents.js JavaScript bringt mit seiner Größe von 116KB [citeulike:13914238] einen großen Umfang mit, was sich negativ auf die Ladezeiten der Webseite auswirkt. Des Weiteren müssen die von den Browsern nicht unterstützten und ignorierten CSS Regeln, wie `::shadow` oder `::slotted`, mit Regular Expressions nachgebaut werden, was momentan 40 Stück sind. Das macht die Polyfills extrem komplex und träge. Die Funktionen um im DOM zu traversieren müssen angepasst werden, damit nur die richtigen Elemente angezeigt werden und eine Shadow Boundary simuliert wird. Diese werden mit 42 Wrappern umgesetzt, was wie die Regular Expressions zur Simulation der CSS Regeln, sehr aufwändig ist. Allerdings können einige Funktionen wie `window.document` schlichtweg nicht überschrieben werden. Im Allgemeinen wird die DOM API stark verlangsamt, wodurch die Performanz, speziell auf mobilen Browsern, drastisch sinkt und mitunter nicht tolerierbar ist [citeulike:13886251].

2 Grundlagen

Laughing Out Loud (LOL) Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem Hypertext Transfer Protocol (HTTP) ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

2.1 tbd

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

2.1.1 tbd

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

2.1.1.1 tbd

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

3 Codebeispiel

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

3.1 tbd

Referenz aufs Codebeispiel 3.1.

Listing 3.1: Codebeispiel Java

```
@SpringBootApplication
@EnableEurekaServer
public class Application {

  public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
  }
}
```

4 Bilder

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt

4.1 Ein Bild ganz breit

Verweis auf das Beispielbild 4.1

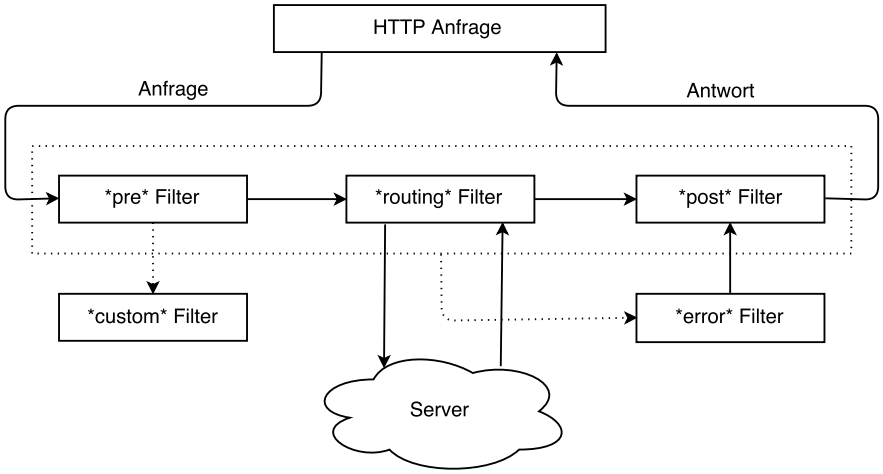
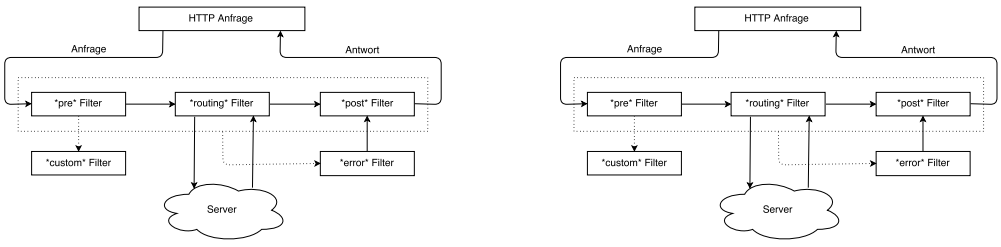


Abbildung 4.1: Beispielbild

4.2 Zwei Bilder nebeneinander



(a) Antwortzeiten

(b) prozentualer Anteil an fehlerhaften Anfragen

Abbildung 4.2: Lasttest Szenarien

5 Tabelle

Generierbar mit `http://www.tablesgenerator.com/`

5.1 tbd

Tabelle 5.1: Ergebnisse des Integartionstests in Sekunden

Testszenario	Start	Registrierung	1. Nachricht	**Gesamt**
Integration_01	14,598	10,079	45,449	70,026
Integration_02	14,485	10,108	54,382	79,088
Integration_03	14,498	10,055	55,125	79,678
Integration_04	14.598	5,361	36,655	56,614

6 Quellen verwenden / verwalten

Am besten http://www.citeulike.org/ verwenden und Expotieren als bibtex Datei.

6.1 Verwenden

Beispiel - Typ: apalike
  Ergebnis: [**?**]

7 Zukunftsprognose

Die Standards der Web Component Technologien sind noch nicht fertiggestellt und befinden sich momentan noch in Entwicklung. Um allgemeingültige Standards für alle Browserhersteller zu gewährleisten, werden sich diese vermutlich noch verändern. Jedoch werden sie dadurch ein breites Spektrum an Zustimmung und Implementierung bekommen, da sie eine zentrale Rolle spielen, wenn darum geht auch das Web zu einer Plattform zu machen, für welche die Paradigmen von höheren Programmiersprachen gelten. So soll es auch im Web möglich sein Applikationen aus mehreren verschiedenen Komponenten, die gekapselt, wartbar und interoperabel sind, zu entwickeln. Polymer wird dabei weiterhin eine wichtige Rolle innehaben, da die Bibliothek das Arbeiten mit den Web Components vereinfacht und zusätzliche Funktionalitäten dafür leistet. Dabei könnte sogar ein Vergleich der Polymer Bibliothek und der jQuery Bibliothek vorstellbar sein. Indem jQuery wurde zu einem de facto Standard für das Arbeiten mit DOM Elementen, so könnte Polymer ein Standard für das Arbeiten mit Web Components werden. Da Google maßgeblich die Entwicklung der Standards der Web Component Technologien vorantreibt und sich die anderen Browserhersteller teilweise daran orientieren, könnte es sogar sein, dass zumindest Teile von Polymer zum offiziellen Standard der Web Technologien wird. Wenn die Standards von allen Browsern akzeptiert und implementiert wurden, sinkt auch die Komplexität von Polymer um ein Vielfaches, da die komplette Schicht der Polyfills wegfällt. Neben den Polyfills kann Polymer dann auf die Nutzung des Bibliothek-internen Shady DOM verzichten und stattdessen mit einer standardisierten, schnellen und leichtgewichtigen Version des Shadow DOM arbeiten. Dadurch wird Polymer um einiges schneller arbeiten und auch auf mobilen Geräten effizienten Einzug erhalten. Neben Polymer basiert auch Angular ab Version 2.0 auf den Web Component Standards und dessen Technologien benutzen, jedoch verfolgt Angular einen anderen Ansatz als Polymer. Angular implementiert hierfür eigene Schicht, welche auf die nativen Technologien zugreift und das Framework komplexer macht. Da in Zukunft die Polyfills und der Shady DOM der Polymer Bibliothek wegfallen, könnte Angular Polymer in sich integrieren, statt die Web Components Standards selbst zu implementieren. Hierfür würde beispielsweise die Micro Schicht in Frage kommen, da diese nur die Grundfunktionalitäten für den Umgang mit den Web Components leistet. Jedoch werden beide Bibliotheken weiterhin koexistieren und weder Polymer Angular ersetzen oder andersrum, da Polymer nur eine erweiterbare Bibliothek und Angular ein vollständiges Framework ist. Beide Plattformen verfolgen daher unterschiedliche Ansätze und können unterschiedliche Probleme lösen. Jedoch kann durch die Entwicklung der Carbon Elemente die Polymer Bibliothek sukzessive als Framework erweitert werden, da diese eine neue Möglichkeit bieten, wie Applikationen strukturiert werden können. Durch sie können komplexe Applikationen realisiert werden, da die Elemente sehr nah

an der Plattform selbst sind, was einige Vorteile mit sich bringt. So haben sie wenige Abhängigkeiten, eine bessere Performanz, können leicht ausgetauscht werden und bieten ein vereinfachtes Debugging an, da sie die Tools der Plattform, statt die eines Frameworks benutzen. Polymer kann also als eine Bibliothek mit einem optionalen Framework Plugin verstanden werden, wobei das Framework nur die Meinung Googles' widerspiegelt wie ein Framework funktionieren sollte. Dieses bietet, je nach Anforderungen, Vor- und Nachteile wie jedes Andere Framework auch. Es liegt somit weiterhin im Ermessen der Entwickler ob Angular oder Polymer eingesetzt werden soll, oder ob sogar, bis zu einem gewissen Grad, beides verwendet werden soll. Jedoch beschränkt sich Polymer nicht nur auf den Einsatz in Angular, sondern kann mit jedem beliebigem Framework kombiniert werden. Wie genau die Entwicklung von Polymer weiter verläuft und ob sich diese auf Angular auswirken wird, liegt jedoch ganz im Ermessen von Google.

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

TODOs

Literaturverzeichnis

[Kröner, 2014a] Kröner, P. (2014a). Das Web der Zukunft. http://webkrauts.de/artikel/2014/das-web-der-zukunft.

[Kröner, 2014b] Kröner, P. (2014b). Web Components erklärt, Teil 1: Was sind Web Components? http://www.peterkroener.de/web-components-erklaert-teil-1-was-sind-web-components.

[Schaffranek, 2014] Schaffranek, R. (2014). Web Components – eine Einführung. https://blog.selfhtml.org/2014/12/09/web-components-eine-einfuehrung/.