Ausgangssituation

Softwareentwicklung für den Browser ist ohne Zweifel eine komplexe Angelegenheit. Die Gründe für diesen Umstand sind so vielfältig, dass darüber eine ganze Forschungsarbeit geschrieben werden könnte. Je nach Blickwinkel könnte man von der technischen Seite argumentieren und beispielsweise Designschwächen von JavaScript oder die unflexiblen Browserplattformen als Bremsklotz der Webentwicklung benennen.[1] Oder man könnte die organisatorische Seite betrachten und die vielen beteiligten Organisationen benennen, die für Standardisierungsprozesse im Internet zuständig sind.[2] Oder aber man nimmt die historische Entwicklung in Augenschein, die dazu geführt hat, dass die technische Entwicklung der rasanten Evolution im Internet nicht Schritt gehalten hat. Die Zeiten statischer HTML/CSS Seiten auf einfachen Desktopgeräten ist mithin noch gar nicht so lange her.

Dennoch tragen alle diese Facetten dazu bei, dass die Entwicklung und Wartung von komplexen Webapplikationen, wie sie heute Standard sind, mit enormen Zeit- und Geldaufwand verbunden sind. Die Anzahl der Frameworks, Werkzeuge und Bibliotheken mit Javascript als Zielsprache ist nahezu unüberblickbar und wandelt sich in einer Geschwindigkeit, die vielen Entwicklern Schwierigkeiten bereitet.[3] Auch die Konsumenten und Nutzern der Webdiensten bekommen diese Probleme zumindest teilweise spüren. Einerseits spürbar in Form von “Downtimes” der Services. Andererseits subtil wie etwa durch lange Ladezeiten wegen aufgeblähtem Quellcode. Frederic Filloux zeigte in einem Blogpost anschaulich, dass sich in einem Zeitungsartikel der britischen Zeitung “The Guardian” zu jedem lesbaren Buchstaben des Artikels über 100 Zeichen Code addieren.[4]

Aus diesen Grund wurde schon 2013 das “Extensible Web Manifesto” proklamiert. Darin wird, kurz gefasst, eine Öffnung der Webplattformen avisiert, um Webprogrammierung teilweise von Browserstandards zu entkoppeln.[1] Drei Jahre später sind diese Ziele in W3C Spezifikationen konkretisiert worden. Einige dieser neuen Standards werden bereits nativ in (einigen) Browsern unterstützt.

Zielsetzung

Viele populäre Frameworks für den Browser als Zielplattform versprechen dem Entwickler eine bessere Kontrolle über die Webseite. Der Grund dafür liegt unter anderem darin, dass es bisher nicht möglich war einzelne Browserelemente in ihrem Aussehen und Verhalten zu isolieren. Selbst kleinste Änderungen können daher die Funktionalität des komplexen Gesamtsystems beeinträchtigen. Mit der neuen W3C Spezifikation “Web Components”, die mehrere Substandards unter sich vereint, soll die Modularität nun auch nativ im Browser unterstützt werden.[2] Modularität als Designpattern ist in der IT schon länger Standard um komplexe Systeme beherrschbar zu gestalten und auch zukünftige Unsicherheiten abzuwägen.[5, p. 1] Die Unsicherheit, die Frameworks immer anhaftet, ist die Lebensdauer derselben.

Web Components ist der Entwickler in der Lage, eigene HTML Elemente zu definieren und diese per Templates zu exportieren oder zu importieren. Darüber hinaus können sie JavaScript und CSS Funktionalität enthalten und diese vom Rest der Webseite abzukapseln. Ziel dieser Arbeit ist die systematische Erfassung und Risikoabwägung der neuen Technologien sowie eine praktische Erläuterung möglicher Architekturmodelle. Bisher gibt es keine einheitlichen “best practices” wie denn die Komponenten umzusetzen sind, obwohl diese Technologien bereits länger durch Polyfills genutzt werden können. Manch ein Entwickler fürchtet schon eine Flut von schlecht gebauten Komponenten, die wiederum neue Probleme schaffen könnten.[6]

Vorgehensweise

Den Anfang dieser Arbeit soll eine Analyse der aktuellen Situation der Frontend Entwicklung aufzeigen. Dort soll auf die Problematik der bisherigen Architekturmodelle und die Probleme mit monolithischen Frameworks eingegangen werden. Außerdem soll dieser Teil vor Augen führen, warum es bisher nur mit zusätzlicher Abstraktionsebenen möglich war einen modularen Aufbau von Web Applikationen zu ermöglichen.

Im nächsten Abschnitt der Arbeit sollen die neuen Standards aufgeschlüsselt, zugänglich gemacht und auf Anwendungsmöglichkeiten untersucht werden. Explorativ sollen gute Bespiele offengelegt werden und mögliche Einbettungen ins Gesamtsystem diskutiert werden. Auch die Probleme, wie sie beispielsweise bei alten Browsern auftreten können, sollen hier behandelt werden.

Der letzte Abschnitt soll das Thema noch mit einer Metaperspektive abrunden, in der auch die neuen low-level CSS Standards Houdini als Teil des Gesamtsystems betrachtet werden sollen.[^h]

[^h]: https://drafts.css-houdini.org/

title: Microservices im Browser durch Web Components subtitle: Kombinierte W3C Spezifikationen für plattformübergreifende Browsermodule author: Jan Peteler, FHWS Würzburg-Schweinfurt, jan.peteler@student.fhws.de date:

[1] https://extensiblewebmanifesto.org/

[2] https://www.w3.org/standards/techs/components

November 2016 abstract: ...

[1] Y. Katz, "Extend the web forward," 2013.

[2] P. Walton, "Houdini: Maybe the most exciting development in css you've never heard of," 2016.

[3] D. Berner, "Not an imposter: Fighting front-end fatigue," 2016.

[4] F. Filloux, "Bloated html, the best and the worse," 2016.

[5] C. Y. Baldwin and K. B. Clark, "Modularity in the design of complex engineering systems," in *Complex engineered systems: Science meets technology*, D. Braha, A. A. Minai, and Y. Bar-Yam, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 175–205.

[6] J. Keith, "Web components," 2014.