Browsernative Microservices durch Web Components

Jan Peteler, FH Würzburg-Schweinfurt, jan.peteler@student.fhws.de

Ausgangssituation

Softwareentwicklung für den Browser ist ohne Zweifel eine komplexe und fehleranfällige 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 die Designschwächen der Browserplattform erläutern.[1] Oder man könnte die organisatorische Seite betrachten und die Schwerfälligkeit von Standardisierungsprozessen anprangern.[2] Oder aber man nimmt die historische Entwicklung in Augenschein. Die Zeiten statischer HTML/CSS Seiten auf Desktopgeräten ist mithin noch nicht lange her.

Dennoch tragen alle diese Facetten dazu bei, dass die Entwicklung und Wartung von komplexen Webapplikationen mit enormen Zeit- und Geldaufwand verbunden ist. Offensichtlich wird das Problem, wenn man die schiere Anzahl der Frameworks und Bibliotheken mit Javascript als Zielsprache betrachtet, die vielen Entwicklern Schwierigkeiten bereitet.[3]

Um die inhärenten Designschwächen von Browsern aus der Welt zu schaffen, wurde schon im Jahr 2013 das *Extensible Web Manifesto* proklamiert.[1] Darin wird, kurz gefasst, eine Öffnung der Webplattformen avisiert, um Webprogrammierung teilweise von Browserstandards zu entkoppeln. Drei Jahre später sind diese Ziele in W3C Standards konkretisiert worden und teilweise schon umgesetzt worden.

Zielsetzung

Viele populäre Frameworks für den Browser als Zielplattform versprechen dem Entwickler eine bessere Kontrolle über seinen Webservice. Die Notwendigkeit rührt unter anderem daher, dass es bisher nicht möglich war einzelne Browserelemente in ihrem Aussehen und Verhalten zu isolieren. Selbst kleinste Änderungen können daher die Funktionalität des Gesamtsystems beeinträchtigen. Mit der neuen W3C Spezifikation *Web Components*,[2] die mehrere Substandards unter sich vereint, soll die Modularität nun auch nativ im Browser unterstützt werden. Modularität als Designpattern ist in der IT schon länger Standard um komplexe Systeme beherrschbar zu gestalten und auch zukünftige Unsicherheiten abzuwägen.[4, p. 1] Die Unsicherheit, die Frameworks immer anhaftet, ist die Lebensdauer derselben.

Mit *Web Components* ist der Entwickler in der Lage, eigene HTML Elemente zu definieren und diese per Templates zu ex- & 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 einheitliche Bestpractice, 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.[5]

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 Bestpractices 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 mit einer Metaebene abrunden, in der auch die neuen CSS Standards *Houdini* als Teil des Gesamtsystems betrachtet werden sollen.[3]

[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] 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.

[5] J. Keith, “Web components,” 2014.

[1] https://extensiblewebmanifesto.org/

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

[3] https://drafts.css-houdini.org/