



Disposition

Bachelorarbeit

Frameworkagnostic Design Systems

What restrictions do Web Components impose on an Angular & Tailwind Design System?

eingereicht zur Erlangung des akademischen Grades:

Bachelor of Science in Informatik

Referent: Fabian Gosebrink

vorgelegt von:

Remo Kessler
Hubstrasse 35
9535 Wilen TG

Tel.: +41 79 964 03 29
E-Mail: kalaidos@remok.dev

Datum der Abgabe: 01.07.2024

Inhaltsverzeichnis

1	Einleitung	2
1.1	Abgrenzung des Forschungsgebiets	2
2	Theorie	3
2.1	Was ist ein Design-System?	3
2.2	Was sind die Eigenschaften von Tailwind?	3
2.3	Was sind Web Components?	3
2.3.1	Shadow Dom mit Tailwind	4
2.3.2	Wie können Web Components erstellt werden?	4
2.4	Atomic Design	5
2.4.1	Implizierungen auf die Architektur	5
3	Methode	6
3.1	Mehrwert	7
3.2	Erhebung	7
3.2.1	Abgrenzung	7
3.3	Technische Einschränkungen	8
3.3.1	Technische Machbarkeit	8
3.3.2	Tailwind Integration	8
3.3.3	Angular integration	8
3.4	Contributions	9
3.4.1	Erhebung	9
3.5	Dev Experience	10
3.5.1	Erhebung	10
3.5.2	Abgrenzung	10
3.6	User Experience	11
3.6.1	Erhebung	11
3.6.2	Abgrenzung	11
4	Zeitplan	12
5	Ausblick und erwartete Resultate	13
6	Quellenverzeichnis	14
	Verzeichnis der Darstellungen	15
	Fremdwortverzeichnis	15
	Diagrammverzeichnis	17

1 Einleitung

Die Bühler AG (fortan Bühler) verfügt über ein zentrales Angular-Design-System, das konzernweit mit über 50 Komponenten in etwa 25 Applikationen eingesetzt wird. Derzeit basiert das Design-System ausschliesslich auf dem von Google stammenden Framework Angular, da die grossen digitalen Produkte allesamt Single Page Applications sind. Mit der wachsenden Grösse des Design-Systems und des Unternehmens treten jedoch vermehrt Anwendungsfälle auf, in denen das Design-System nicht genutzt werden kann, beispielsweise weil die Bühler-Website in WordPress geschrieben ist oder SAP kein Angular unterstützt. Zudem mangelt es an Software-Standardisierung, was dazu geführt hat, dass viele Prototypen ausserhalb der IT-Abteilung (Bühlers Informatikabteilung) entwickelt wurden. Diese Prototypen sind in der Regel nicht in Angular und C# geschrieben, sondern basieren auf den Frameworks, mit denen die jeweiligen Entwickler am besten vertraut sind.

Dieses Problem wurde 2020 mit der Implementierung des Web Components Standards in allen populären Browsern gelöst (Schoors & Deveria, 2024). Zum Zeitpunkt der Erstellung dieser These verwenden etwa 15% aller aufgerufenen Webseiten Web Components (Google Inc., 2024).

Trotz der langen Unterstützung von Web Components gibt es kaum Open Source Design-Systeme, die auf Web Components basieren. Libraries wie Bootstrap bieten ausschliesslich CSS-Klassen und JavaScript-Helfer an, während Libraries wie Angular Material spezifisch nur Angular-Komponenten anbieten. Das Ionic Framework erklärte 2019 in einem Whitepaper, dass der Technologie-Lock-In und die zu viel Flexibilität mit ausschliesslich HTML/CSS Komponenten der Grund seien, warum viele Design-Systeme nicht ihr volles Potential erreichen können (Ionic, 2019).

1.1 Abgrenzung des Forschungsgebiets

In der Praxis gestaltet es sich schwierig, ein bestehendes Softwareprodukt vollständig neu zu entwickeln. Daher soll im Rahmen dieser Bachelorarbeit ein Konzept erarbeitet werden, das einen Übergang der bestehenden Software-Architektur zu einem neuen System ermöglicht, das auf Web Components und Tailwind CSS (fortan Tailwind) basiert.

Neben der Entwicklung eines Konzepts sollen auch die Hindernisse und Grenzen von Web Components beleuchtet werden. Ein wesentlicher Aspekt ist dabei, dass das bestehende Bühler Design-System bereits sehr umfangreich ist und in Angular implementiert wurde. Angular bietet eine Vielzahl von integrierten Funktionen und Tools, die eine umfassende Entwicklung und Verwaltung von Komponenten ermöglichen. Im Vergleich dazu sind Web Components sehr leichtgewichtig, weshalb genau untersucht werden soll, worin dieses Delta besteht, ob und wie es abgefangen werden kann.

2 Theorie

Das Bühler Design-System nutzt Angular und Tailwind zur Erstellung von Komponenten, welche gemäss dem Atomic Design Prinzip gegliedert sind. Im Zuge dieser Arbeit sollen neue Web Components bereitgestellt werden, um eine frameworkunabhängige Bereitstellung der Komponenten zu ermöglichen. In diesem Kapitel werden Begriffe und Prinzipien im Zusammenhang mit der Arbeit erläutert.

2.1 Was ist ein Design-System?

„There isn't a standard definition of 'design system' within the web community, and people use the term in different ways" (Kholmatova, 2017). Jedes Design-System ist sowohl in der Erstellung als auch im Gebrauch und im Funktionsumfang einzigartig. Im Fall von Bühler ist das Design-System speziell für die Industrie optimiert. Es umfasst Tailwind als CSS-Framework, eine Vielzahl von Angular-Komponenten, UX- und Development-Patterns sowie Konfigurationen für Third-Party Libraries wie beispielsweise amCharts¹. Gleichzeitig fungiert das Design-System als treibende Kraft hinter einem UI-Harmonisierungsprojekt, in dem Bühler die visuelle Harmonisierung aller Softwareprodukte anstrebt.

2.2 Was sind die Eigenschaften von Tailwind?

Die Strukturierung von CSS stellt seit der Einführung des Webs eine erhebliche Herausforderung für Frontend-Entwickler dar. Trotz der Existenz von Best Practices erweist sich deren praktische Anwendung häufig als schwierig. Dieses Problem wird durch das CSS-Framework Tailwind adressiert. Im Gegensatz zum Trend der Shadow DOM-Encapsulation von Styles verfolgt Tailwind einen anderen Ansatz: Es bietet eine Vielzahl von Utility-Klassen an. Diese Utility-Klassen sind global verfügbar und besitzen alle die Specificity 1 (Relevanzberechnung der CSS-Regeln). (Rappin, 2022)

Diese Klassen ermöglichen eine hohe Wiederverwendbarkeit und eliminieren viele gängige Herausforderungen bei der Nutzung von CSS. Beispielsweise ist das Naming-Konzept von Tailwind bewusst vorgegeben, sodass Entwickler keine Zeit mehr in die Benennung nach dem BEM-Konzept (Block, Element, Modifier) investieren müssen. Gedanken zur Wiederverwendbarkeit werden ebenfalls minimiert, da diese im modernen Frontend-Framework über Komponenten und nicht über CSS gelöst werden sollen. Zudem unterstützt das Tooling von Tailwind die Entwickler bei der Verwendung der Klassen durch Syntax-Vorschläge basierend auf der Tailwind-Konfiguration (Tailwind Labs Inc., 2024).

2.3 Was sind Web Components?

Web Components sind eine Technologie zur Erstellung von frameworkagnostischen Komponenten. Das konzeptionelle Prinzip dieser Komponenten findet sich in den gängigen modernen JavaScript-Frameworks wie Angular, React, Vue.js und anderen wieder. Im Gegensatz zu diesen Frameworks bieten Web Components jedoch einen interoperablen Standard zwischen verschiedenen Frameworks. Dies ermöglicht es beispielsweise, einen Button mit dem gleichen Code sowohl in einer Angular- als auch in einer React-Anwendung zu verwenden (Rylan, 2021).

¹ <https://www.amcharts.com/>

2.3.1 Shadow Dom mit Tailwind

Alle Web Components sind technisch gesehen mittels des sogenannten Shadow DOM vom Rest der Webseite isoliert. Dieser Ansatz beruht darauf, dass Web Components überall einheitlich aussehen sollen und daher CSS-Styles von ausserhalb der Web Component nicht berücksichtigt werden. (Rylan, 2021)

Im Bühler Design-System führt diese Vorgehensweise jedoch zu einem Problem, da Tailwind ausschliesslich globale Styles unterstützt und keine gekapselten Styles bereitstellt. Dieser Zielkonflikt wird in dieser Bachelorarbeit eingehend untersucht. Zudem werden Möglichkeiten erörtert, wie die Design-Tokens aus der Tailwind-Konfiguration dennoch effektiv eingesetzt werden können.

2.3.2 Wie können Web Components erstellt werden?

Web Components können entweder nativ geschrieben oder mithilfe verschiedener Frameworks erstellt werden. (Rylan, 2021) In Rahmen dieser Arbeit wird die Erstellung mittels des LitElement Framework untersucht.

2.4 Atomic Design

Das Bühler Design-System folgt derzeit dem Atomic Design Aufbau. Diese Strukturierungsmethode ordnet jede Komponente als ein Atom, Molekül, Organismus, Template oder Page. Ist es klein und ohne Abhängigkeiten ist es ein Atom wie z.B. Buttons oder Icons. Verwendet eine Komponente mindestens ein Atom, so wird diese ein Molekül wie z.B. ein Akkordeon, oder eine Suchbar. Verwendet eine Komponente Moleküle und Atome so wird sie zu einem Organismus wie z.B. eine Tabelle.

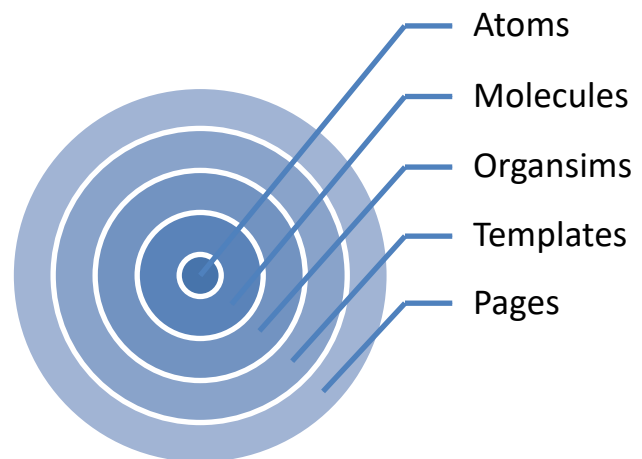


Diagram 1 - Atomic Design

Nebst den chemischen Organisationselementen gibt es ebenfalls Templates, welche grössere wiederverwendbare UI-Elemente bilden, die vor allem Layouting der Komponenten übernehmen. So kann z.B. eine spezifische UI-Karte ein Template sein, oder aber auch das Layout mit Header, Sidebar und Content. Zuletzt gibt es Pages, die die Integration anhand eines Beispiels zeigen. Zudem können so z.B. standardisierte 404-not-found-Seiten, oder User-settings Seiten harmonisiert werden.

Zusammengefasst lässt sich sagen, dass das Atomic Design eine aufsteigende Hierarchie ist. Je weiter aussen der Platz in der Hierarchie ist, umso spezifischer sind die Komponenten. So hat ein Button eine hohe Wiederverwendbarkeit (unspezifisch) und eine 404 Seite weniger (spezifisch). (Frost, 2016)

2.4.1 Implizierungen auf die Architektur

Die Atomic Design Struktur ist bereits gelebt, jedoch nicht technisch forciert im Bühler Design System. Durch die Nähe am Design war die Atomic-Design Struktur der einfachste Weg, die technischen Komponenten zu strukturieren. Da dieses Konzept jedoch nicht vorsieht mehrere Technologien zu Mischen muss erarbeitet werden, wie die Web Components in diese Architektur einfließen können, ohne die Entwicklung zu komplex zu gestalten.

3 Methode

Die dargelegte Arbeit befasst sich mit der Frage, welche Restriktionen durch Web Components im Design-System auftreten können. Die Restriktionen werden dabei Anhand der vier Variablen «Contributions», «technische Einschränkungen», «Dev Experience» und «User Experience» gemessen. Anschliessend kann dies dem Projekt Mehrwert gegenübergestellt werden um eine Handlungsempfehlung abzugeben.

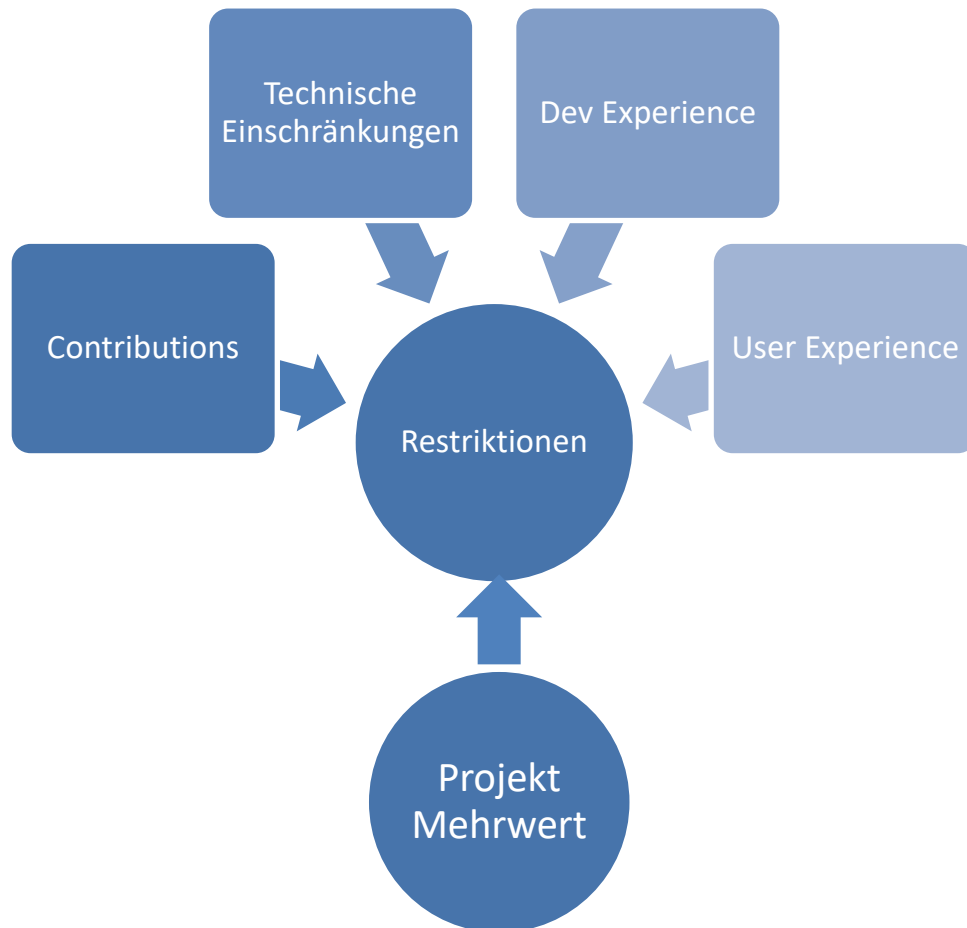


Diagram 2 - Variablen

Aus den Variablen lassen sich folgende Hypothesen bilden.

- **H1:** Umso mehr Contributions (manifest) desto weniger ist das Design-System durch Web Components eingeschränkt (latent).
- **H2:** Umso weniger technische Einschränkungen (manifest) es gibt desto weniger ist das Design-System durch Web Components eingeschränkt (latent).
- **H3:** Umso besser die Dev Experience (manifest) desto weniger ist das Design-System durch Web Components eingeschränkt (latent).
- **H4:** Umso besser die User Experience (manifest) desto weniger ist das Design-System durch Web Components eingeschränkt (latent).
- **H5:** Umso mehr Mehrwert im Projekt (manifest) generiert wird, desto mehr Restriktionen (latent) können in Kauf genommen werden.

Die genaue Definition und Erhebung der Variablen ist in den nachfolgenden Unterkapiteln niedergeschrieben.

3.1 Mehrwert

Das OneView Projekt ist die Frontend-Harmonisierung der Bühler-UIs. Der Mehrwert der Lösung hängt vom Mehrwert des Projekts ab. Dabei sollen durch standardisierte Patterns die Entwicklung vereinfacht und beschleunigt, eine einheitliche Benutzeroberfläche für Kunden geschaffen werden und nicht zuletzt die Kosten durch hohe Wiederverwendbarkeit gesenkt werden.

3.2 Erhebung

Um den Mehrwert zu ermitteln, werden in einem ersten Schritt alle Projekte ermittelt, welche von der Endlösung profitieren können. Dies geschieht anhand des folgenden Fragebogens, der an alle Mitarbeiter in Bühler versendet wird. Er soll offenlegen welche Technologien verwendet wurden und auch welche Projekte es gibt.

Anschliessend werden die Projekte genauer durchleuchtet auf die technischen Anforderungen. Solange sie JavaScript basiert sind, können sie von den Web Components profitieren. Diese wird anhand der folgenden Messkriterien bewertet.

- K1: Ist die Applikation für Kunden oder für Intern?
 - Extern (2 Punkte) / Intern (1 Punkt)
- K2: Wie gross ist die Benutzerbasis für die Applikation?
 - 0-5, klein bis gross.
- K3: Kann die Applikation Web Components verwenden?
 - Ja (1 Punkt) / Nein (0 Punkt)
- K4: Hat die Applikation Budget um Web Components zu verwenden?
 - Ja (2 Punkte) / Nein (1 Punkt)
- K5: Ist das Management der Applikation offen dafür?
 - Ja (2 Punkte) / Nein (1 Punkt)

Anhand dieses Bewertungsraster und der folgenden Formel « $K1 * K2 * K3 * K4 * K5$ » ergibt sich eine Zahl von 0 bis 40. Diese Kennzahl gilt als Indikation für den Mehrwert für das Projekt. Dieser kann verglichen werden mit dem Wert ermittelt der bisherigen Applikationen.

K1 bis K5 werden entweder als Fragebogen, oder zusammen mit den Teams erhoben. Je nach Anzahl neuer Projekte.

3.2.1 Abgrenzung

Da es sich bei der These um eine Informatik-These handelt wird keine Forschung im Bereich der Kosten angesetzt. Der Mehrwert ist hierbei eine reine Indikator-Zahl für das potentielle Wachstum der technischen Endlösung.

3.3 Technische Einschränkungen

Anhand eines Proof of Concept (PoC) werden verschiedene Typen von Web Components erstellt. Diese sollen dann für die Integrationstests verwendet werden. Dabei wird in einer ersten Phase die technische Machbarkeit ermittelt, in einer zweiten Phase die Tailwind Integration und Tooling Integration sichergestellt und in der dritten Phase die Angular Integration getestet.

3.3.1 Technische Machbarkeit

In diesem Schritt soll ein Minimum Viable Product MVP auf dem internen Package-Feed publiziert werden, um festzustellen, ob die Tooling-Integration das Publizieren von Web Components zulässt. Dieser Prototyp kann gleichzeitig für die Dev Experience Analyse verwendet werden. Dazu sollen die folgenden Komponenten nachgebaut werden.

- **Primary Button:** eine einfache und die meist genutzte Komponente um etwas zu publizieren, das überall verwendet wird.
- **Accordion:** eine komplexere Komponente, die State halten und diesen nach aussen Propagieren muss.
- **Form-Field:** als einfache Komponente, die Komplex in der Einbindung ist aufgrund ihrer vielen Variationen.

3.3.2 Tailwind Integration

Nachdem die Umgebung bereitgestellt wurde soll das Zusammenspiel zwischen Tailwind und Web Components genauer untersucht werden. In diesem zweiten PoC ist die Frage zu klären, wie Tailwind in Web Components verwendet werden kann. Dazu gibt es die folgenden Ansätze:

- Auflösung der Web Components Shadow DOM Encapsulation um die globalen Tailwind Klassen wiederverwenden zu können
- Die nötigen Tailwind Klassen kompilieren lassen und in die Web Component mitkompilieren.
- Tailwind so konfigurieren, dass es auf Styling Variables aufsetzt, die via CSS-Variables durch die Shadow DOM Encapsulation geschleust werden können.

3.3.3 Angular integration

Um die Einschränkungen zu erheben wird der PoC im dritten Schritt erweitert mit der Angular Integration. Innerhalb dieses Schritts werden einige Komponenten selektiert, die Web Components innerhalb des Design Systems konsumieren sollen. Anhand dieses PoC soll eine Aussage getroffen werden können ob die verschiedenen Patterns von Angular weiterhin genutzt werden können. Einige dieser Patterns sind:

- Dependency Injection
- Directives
- Internationalization
- Localization

(Rappin, 2022)

3.4 Contributions

Das Bühler Design-System hat einen Innersource Ansatz (ähnlich Opensource jedoch nur innerhalb der Firma) und lebt einen merklichen Teil von dem, dass jeder Bühler-Entwickler Code mitschreiben kann. Dies soll auch nach der Einführung der Web Components immer noch der Fall sein. Da diese Contributions je nach Features und Kapazitäten stark fluktuieren, kann kein Durchschnittswert verglichen werden.

3.4.1 Erhebung

Es gilt hier zwei Zielgruppen zu erfassen, die bisherigen Contributors und die potentiellen neuen Contributors. Dabei soll via Selbsteinschätzung mittels eines Fragebogens ermittelt werden, wie bereitwillig bestehende Contributors waren etwas zum Code des Design-Systems beizutragen, und wie wahrscheinlich es ist, dass zukünftige Contributions Web Components involvieren. Für neue Contributors soll erhoben werden, wie wahrscheinlich es ist, dass sie etwas zum Code von Web Components beitragen im Vergleich zu ihren täglich eingesetzten Frameworks.

So soll versucht werden festzustellen, wie Contributions in Zukunft aussehen können. In diesem Fragebogen soll auch die Art der Contributions festgehalten werden (Bugfixes, Style Anpassungen, neues Feature, neue Komponenten, Reviews, Tests, Dokumentation, Konzeption, ...)

Der genaue Fragebogen soll innerhalb der These basierend auf den verwendeten Technologien erarbeitet werden. Einflussfaktoren können sein:

- Wahl des Web Component Frameworks
- Erfahrungslevel des Entwicklers
- Vorerfahrungen mit Web Components
- Grösse der Contribution
- Art der Contribution

3.5 Dev Experience

Die Dev Experience innerhalb des Design-Systems ist ein Sammelbegriff für alle Einflüsse auf das Entwickeln der technischen Library selbst. Dabei sind verschiedene Aspekte wie IDE (VS-Code und Webstorm), Tooling-Integration, Automatisierung (z.B. CI/CD) aber auch Dokumentation entscheidend.

3.5.1 Erhebung

Innerhalb der These soll mittels des erstellten Prototyps herausgefunden werden, wie sich Web Components in das bestehende Monorepo integrieren lassen. Dabei soll der Minimalumfang von:

- Funktioniert Hot-Reloading zwischen Angular und LitElements?
- Gibt es Syntax-Highlighting für die bestehenden IDEs?
- Funktioniert Inteli-Sense Support?
- Funktionieren Imports via Module-Boundaries?
- Funktionieren das bestehende Testing Frameworks Jest?
- Können Web-Components von aussen getestet werden mittels End to End Testing Tools wie Playwright?
- Funktioniert die Integration ins Dokumentationstool Storybook?
- Kann das Code-Formatierungstool Prettier weiterhin verwendet werden?
- Kann das Code-Analysetool ES-Lint weiterhin verwendet werden?

Umso mehr dieser Tools und Konzepte wiederverwendet werden können, umso besser ist die Dev Experience.

3.5.2 Abgrenzung

Auch die Programmiersprache zur Erstellung der Web-Components gehört dazu. Es wird nur LitElements beleuchtet. Aufgrund der vorgegebenen Zeit kann nicht jede Untersuchung für jede Möglichkeit zur Erstellung von Web-Components genau durchleuchtet werden.

Es wurde sich aufgrund der folgenden Vorteile für LitElements entschieden:

- Es gibt eine bestehende Nx Integration
- Ähnliche Syntax zu Angular
- Google Support
- Sehr Lightweight
- TypeScript Support
- Alle LitElements sind HTMLElements

Ebenfalls wurden auch die folgenden Methoden in Betracht gezogen:

- **Angular-Elements:** Zu gross für Vue.js / React C\$\$onsumers da zone.js immer noch ins Bundle kommt.
- **Stencil.js:** Gut, aber Syntax ist ähnlicher zu React mit dem JSX-Format, als zu Angular. Und kein Nx Support
- **Native Web-Components:** Kein Nx Support,
- **Vue3 / React:** Angular ist das Haupt-Framework, ein weiteres grosses Framework zu lernen ist sehr Zeitintensiv. Dazu kommt eine grosse Bundle Size der Library.

3.6 User Experience

Die User Experience der technischen Library befasst sich mit der Frage, wie einfach und verständlich es ist, die Library als Entwickler zu verwenden. So soll in diesem Punkt via Experten-Interviews und kleinen Prototypen eine Aussage getroffen werden können, wie einfach es ist die erstellten Web-Components in den jeweiligen Frameworks einzusetzen.

3.6.1 Erhebung

Basierend auf den Experten, die sich aus der Mehrwerterhebung ergeben werden Tiefeninterviews geführt. Es soll ein Beta-Release mit verschiedenen Web-Components geben, welche von den Experten eingebaut werden sollen. Da der zeitliche Rahmen es nicht erlaubt, jedes Framework bis in die Tiefe zu lernen, wird hier auf das Wissen und die Meinung von Experten innerhalb von Bühler gesetzt.

Um die Basis zu setzen können die Komponenten aus dem PoC angeschaut werden. Die genaue Liste soll jedoch mit den jeweiligen Experten definiert werden.

Gleichzeitig kann damit auch eine Integration der Library getestet und das technische Konzept validiert werden.

Anschliessend wird das Feedback gesammelt und zusammen mit den technischen Limitationen ausgewertet.

Die Erhebung dabei erfolgt mit den gleichen Kriterien wie bei der Dev Experience. Eingesetzte Tools und Patterns sollen validiert werden pro Produkt. Die genauen Punkte werden im Rahmen der These als Teil der Tiefeninterviews erstellt.

3.6.2 Abgrenzung

Die User Experience für den Endkunden der Software erstellt mit dem Design-System ist von vielen Faktoren ausserhalb der Library geprägt. Deshalb wird dieser Bereich nicht untersucht innerhalb dieser Arbeit.

4 Zeitplan

Projekt Mehrwert (Juli)

- Erstellung Fragebogen
- Senden Fragebogen

Auswerten Mehrwert (August)

- 2 Wochen Urlaub
- Auswertung des Mehrwert

PoC Phase 1 technische Machbarkeit (September)

- Erstellung des PoC
- Dokumentation Findings für Dev-Experience / technische Einschränkungen
- Auswertung des PoC 1 für Dev Experience

PoC Phase 2 Tailwind Integration (Oktober - November)

- Integration Tailwind
- Dokumentation Findings

PoC Phase 3 Angular Integration (Dezember)

- Integration Angular
- Dokumentation der Findings
- 2 Wochen Urlaub

Umfragen Contributions und User Experience (Januar)

- Erstellung Fragebögen Contributions und User Experience
- Senden Fragebögen
- Auswerten

Fertigstellung Dokumentation (Februar)

- Rechtschreibprüfung
- Dokumentation polieren

Abgabe 1. März

- Vorbereitung Präsentation

Diagram 3 – Zeitplan

Pro Woche wird für die These 12h gerechnet. Alles ausser der Dokumentation im Februar Abzüglich der Ferien sind es 30 Wochen. Dies kommt zu einem Totalaufwand von 360h.

Alles ausser Dokumentation kann innerhalb des Pensums des Anstellungsverhältnis bei der Bühler AG gemacht werden. Für die Dokumentation wurde der Freitagnachmittag mit 4h Reserviert.

5 Ausblick und erwartete Resultate

Da Web Components ein weit verbreiteter Standard sind, ist zu erwarten, dass sie technisch eingesetzt werden können. Jedoch ist auch zu erwarten, dass die Contributions der bestehenden Entwickler sinken werden. Zu hoffen ist, dass aufgrund der höheren Anzahl möglicher Contributors, die Contributions nur leicht sinken.

Des Weiteren wird damit gerechnet, dass die User-Experience vor allem für Anfänger in der Library etwas schlechter wird. Mit genügend Dokumentation sollte dies abgefangen werden können.

Die Dev Experience wird voraussichtlich Tooling-Mässig nicht darunter leiden, jedoch steigt die Komplexität aufgrund des zweiten Frameworks / Technologie.

Der technische PoC wird einiges an technischen Hindernissen aufwerfen und ist derzeit noch nicht einschätzbar. Es wird jedoch erwartet, dass die meisten Angular und Tailwind Probleme gelöst werden können oder ein Konzept für eine Lösung erarbeitet werden kann.

Letztens ist die Gegenüberstellung des Mehrwerts zu den Restriktionen. Hierbei wird mit einem grossen Mehrwert gerechnet und einigen Restriktionen vor allem im Bereich der Contributions und der Dev Experience. Die erwartete Handlungsempfehlung ist Web Components zu verwenden, jedoch mehr Leute ins Entwicklungs-Team des Design-Systems einzustellen aufgrund der höheren Komplexität.

6 Quellenverzeichnis

- Google Inc. (2024, 05 19). *Chrome Platform Status*. Retrieved from Chrome Platform Status: Custom Element Registry Define: <https://chromestatus.com/metrics/feature/timeline/popularity/1689>
- Yablonski, J. (2020). *Laws of UX*. 69123 Heidelberg: dpunk.verlag GmbH.
- Schoors, L., & Deveria, A. (2024, 05 19). "web components" | Can I use. Retrieved from Can I Use: <https://caniuse.com/?search=web%20components>
- Rylan, C. (2021). *Web Component Essentials*. Leanpub.
- Kholmatova, A. (2017). *Design Systems A practical guide to creating design languages for digital products*. Freiburg: Smashing Media AG.
- Frost, B. (2016). *Atomic Design*. Pittsburgh: Brad Frost.
- Fronteers. (2024, 05 19). *Web Components and Model Driven Views by Alex Russels - Fronteers*. Retrieved from Fronteers: <https://fronteers.nl/congres/2011/sessions/web-components-and-model-driven-views-alex-russell>
- Tailwind Labs Inc. (2024, 06 06). *Utility-First Fundamentals - Tailwind CSS*. Retrieved from Utility-First Fundamentals: <https://tailwindcss.com/docs/utility-first>
- Ionic. (2019, 05 19). *Guide to Building Design Systems with Web Components*. Madison: Ionic. Retrieved from https://cdn2.hubspot.net/hubfs/3776657/Guide%20to%20Building%20Design%20Systems%20With%20Web%20Components_August%202019.pdf
- Rappin, N. (2022). *Modern CSS with Tailwind*,. The Pragmatic Programmers, LLC.

Verzeichnis der Darstellungen

Fremdwortverzeichnis

Fremdwort	Bedeutung
Bootstrap	Ein populäres CSS-Framework
CSS-Klassen	Eine Gruppe von Styles die auf verschiedene Elemente auf einer Webseite angewendet werden können
Ionic Framework	https://ionicframework.com/ Ein populäres Design System
Third Party	Eine dritte involvierte Person, Team oder Firma. Hier Source-Code, von einer dritten Person
Library	Eine Ansammlung von Source-Code, welche wiederverwendbar geschrieben wurde für andere Software.
amCharts	https://www.amcharts.com/ Eine Charting-Library
Shadow DOM	Ein virtueller Baum von HTML-Knoten, der dem eigentlichen DOM angehängt werden kann.
Shadow DOM Encapsulation	Das abtrennen von CSS Styles zwischen dem virtuellem und dem eigentlichen DOM
Styles	CSS-Anweisungen, die z.B. die Text-Farbe eines HTML-Elements ändern.
Utility Klassen	Eine Klasse, die nur eine oder wenige CSS-Anweisungen enthält.
Specificity	https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity Das Modell zur Berechnung der Priorität von CSS-Regeln.
Global Styles	CSS-Regeln, die für alle HTML-Elemente gelten.
Scoped Styles	CSS-Regeln, die nur für einen spezifizierten Teil der HTML-Elemente gelten (Meist nur für eine Komponente).
LitElement Framework	https://lit.dev/ ein Framework zur Erstellung von Web Components
Contribution	Das erweitern des Bühler Design-Systems durch Entwickler ausserhalb des Core-Teams.
OneView	Das Projekt, das die Bühler UI-Harmonisierung vorantreiben soll.
Proof of Concept / PoC	Eine einfache Implementation um ein technisches Konzept zu Validieren.
MVP	Minimum Viable Product, das kleinste Featureset, das benötigt wird um eine Anforderung abzudecken.

Package-Feed	Eine Bibliothek mit verschiedenen Libraries und Frameworks, welche installiert werden können.
State	Der Status einer Komponente. Z.B. ist ein Accordion auf oder zugeklappt
Dependency Injection	Ein Softwareentwicklungs Pattern.
Directives	Eine Technik um Funktionalität auf verschiedene HTML-Elemente legen zu können.
Internationalisierung	Übersetzungen von Applikationen.
Localization	Das Übersetzen von z.B. Datum oder Nummern.
Hot-Reloading	Ein Prozess bei dem eine Source-Code Änderung den geänderten Teil der Applikation neu lädt im laufenden Zustand.
Syntax Highlighting	Das Einfärben von Source-Code um ihn einfacher zu lesen.
IDE	Die Entwicklungsumgebung/Code-Editor z.B. VS-Code oder Webstorm
Intelli-Sense	Vorschläge von gültigen Operationen innerhalb der IDE
Module Boundaries	Die Grenzen eines Stück Software
Jest	https://jestjs.io/ Ein populäres JavaScript Testing Framework.
End to End Testing	Ein Test, der die komplette Integration der Software testet.
Playwright	https://playwright.dev/ Ein E2E Testing tool
Storybook	https://storybook.js.org/ Ein Dokumentationstool für Design Systems
Prettier	https://prettier.io/ Ein Tool zur Formatierung des Source-Codes
ES-Lint	https://eslint.org/ Ein Tool zur statischen Code-Analyse.
Nx	https://nx.dev/ Ein Tool zur Verwaltung von grossen Source-Code Sammlungen
Monorepo	Eine Organisationsmöglichkeit für Source-Code bei der alles in einem einzigen Repository liegt.
Syntax	Die Grammatik einer Programmiersprache
Lightweight	Kleines (Anzahl Bytes), schnell und minimal

Diagrammverzeichnis

Diagram 1 - Atomic Design	5
Diagram 2 - Variablen	6
Diagram 3 – Zeitplan	12

Eigenständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst habe. Verwendete Literatur und Quellen habe ich vollständig aufgeführt und ich habe diese gemäss wissenschaftlichen Zitierregeln zitiert. Die Verwendung von KI-Hilfsmitteln habe ich transparent dokumentiert.

Die vorliegende Arbeit oder Teile davon habe ich nicht bereits an anderer Stelle als Leistungsnachweis verwendet, ausser es sei dies ausdrücklich mit dem zuständigen Referenten / der zuständigen Referentin vereinbart worden.

Ich bin mir bewusst, dass die vorliegende Arbeit auf Plagiate und KI-generierte Textstellen und Erzeugnisse – auch unter Verwendung entsprechender Software – überprüft werden kann. Ich ermächtige hiermit die Kalaidos Fachhochschule ausdrücklich zur Vornahme einer solchen Überprüfung.

Die Arbeit enthält die folgende Anzahl Zeichen: 20'756


Zeichen zählen: Textteil, exklusive Titelblatt, Inhaltsverzeichnis, Vorwort, Abstract/Management Summary, Abbildungs-, Diagramm-, Tabellen-, Abkürzungs-, Quellenverzeichnis, Anhänge und Eigenständigkeitserklärung. Textfelder, Fuss- und Endnoten werden nicht berücksichtigt.

Wilen 30.6.24

Ort, Datum

Kessler Remo

Name, Vorname



Unterschrift

Wurde die Arbeit als Gruppenarbeit erstellt, ist die Erklärung in Wir-Form abzugeben und die Daten und Unterschriften sämtlicher AutorInnen festzuhalten.