FACHHOCHSCHULE ERFURT UNIVERSITY OF APPLIED SCIENCES
Angewandte Informatik

Bachelorarbeit
in der Angewandten Informatik

AI-2022-BA-039

Konzeption und Umsetzung einer nativen Webkomponente

zur Darstellung großer, dynamischer Datenmengen im Web in tabellarischer Ansicht unter Berücksichtigung von Performance und Aufwand

Al Khodari, Molham

Abgabedatum: 14.03.2023

Prof. Dr. Marcel Spehr
Dipl. Thomas Dietzel

Kurzfassung

Das Ziel dieser Arbeit besteht darin, eine native Webkomponente mit TypeScript zu entwickeln, um große und dynamische Datenmengen in tabellarischer Ansicht im Web darzustellen. Die Anforderungen und Funktionen wurden an die Bedürfnisse des Praxispartners DAKO angepasst. Die DAKO GmbH ist ein deutsches Unternehmen, das sich auf die Entwicklung von Softwarelösungen für die Transport- und Logistikbranche spezialisiert hat. Die Komponente soll die Möglichkeit bieten, Inhalte zu suchen, zu filtern und die Kriterien zu speichern. Besonderes Augenmerk wurde darauf gelegt, dass die Komponente unkompliziert konfigurierbar, anpassbar und leistungsstark ist.

Die Arbeit ist in zwei Teile unterteilt. Im ersten Teil der Arbeit wird zunächst die Ausgangssituation bei der DAKO beschrieben, einschließlich der Hintergründe, Probleme und Ziele. Anschließend werden verschiedene Methoden und Tools vorgestellt, die für die Arbeit von Bedeutung sind, darunter native Webkomponenten, Stylesheets, Constructable Stylesheets, JavaScript und TypeScript. Dabei werden jeweils Vor- und Nachteile, eine Einführung, ein Überblick, die Funktionsweise sowie die Anforderungen erläutert. Der erste Teil endet mit einem Vergleich von TypeScript und JavaScript sowie einer näheren Erläuterung des TypeScript-Compilers.

Im zweiten Teil wird die Umsetzung des Prototyps dokumentiert. Dabei werden zunächst die Zielsetzung und Anforderungen erläutert und anhand von Bildern, Codebeispielen und Diagrammen dargestellt. Anschließend wird auf die Projektstruktur, Konfigurationen und den Aufbau des Datentabelles eingegangen. Es werden Methoden und Funktionen mit Codeausschnitten erklärt und schließlich der Workflow, die verwendeten Tools sowie Herausforderungen und Fazit.

Abstract

The aim of this project is to develop a native web component using TypeScript to display large and dynamic datasets in a tabular view on the web. The requirements and functionalities have been adapted to the needs of the practical partner DAKO. DAKO GmbH is a German company specializing in the development of software solutions for the transportation and logistics industry. The component should provide the ability to search, filter, and save criteria. Special attention has been paid to ensure that the component is easy to configure, customize, and powerful.

The project is divided into two parts. In the first part, the initial situation at DAKO is described, including the background, problems, and goals. Various methods and tools are then presented that are relevant to the project, including native web components, stylesheets, constructable stylesheets, JavaScript, and TypeScript. The advantages and disadvantages of each method, an introduction, an overview, functionality, and requirements are explained. The first part concludes with a comparison of TypeScript and JavaScript and a detailed explanation of the TypeScript compiler.

The second part documents the implementation of the prototype. The objectives and requirements are first explained and illustrated using images, code examples, and diagrams. The project structure, configurations, and construction of the DataTable are then discussed. Methods and functions are explained with code snippets, and finally, the workflow, tools used, challenges, and conclusions are presented.

Inhaltsverzeichnis

Abbildungsverzeichnis

Codebeispiel Verzeichnis

Glossar

API	Eine Schnittstelle, die Programmen ermöglicht, mit anderen Programmen oder Diensten zu interagieren
Changes	Änderungen an Programmcode
Compile Time	Die Zeit, die benötigt wird, um Quellcode in ausführbaren Code zu kompilieren.
CSS	Cascading Style Sheets, eine Technologie zur Formatierung von HTML-Dokumenten.
CustomElementRegistry	Das CustomElementRegistry ist eine Schnittstelle in der JavaScript-Programmiersprache, die es Entwicklern ermöglicht, benutzerdefinierte HTML-Elemente zu erstellen und zu registrieren. Diese benutzerdefinierten Elemente können dann wie herkömmliche HTML-Elemente in einer Webseite verwendet werden.
Custom elements	Benutzerdefinierte HTML-Elemente, die zur Erweiterung eines HTML-Dokuments verwendet werden.
dwDatatable	Eine dynamische, responsive Datentabelle, die eine Vielzahl von Funktionen und Optionen bietet.
jQuery	Eine JavaScript-Bibliothek, die zur einfachen Implementierung von Funktionen und Interaktionen verwendet
DOM (Document Object Model)	Ein hierarchisches System, das die Struktur von HTML-Dokumenten und deren Elemente beschreibt und es ermöglicht, diese zu manupulieren.
npm (Node Package Manager)	Ein Paketverwaltungssystem, das es Entwicklern ermöglicht, JavaScript-Bibliotheken, Module und Programme zu installieren und zu verwalten
Rendern	Vom englischen Begriff ‚to Render'. In der Entwicklung bedeutet es, das Prozess, indem den Code in visualisierten Inhalt für die Nutzer umgewandelt werden
Run Time	ist die Zeit, wenn der Code ausgeführt wurde
Spaghetti-Codes	Ein Programmierstil, der schlecht strukturiert ist und sehr schwer zu lesen und zu verstehen ist
TypeScript	Eine Programmiersprache, die von Microsoft entwickelt wurde und auf JavaScript basiert.

1 Einleitung

1.1 DAKO Im Überblick

Die DAKO GmbH ist ein deutsches Unternehmen mit Hauptsitz in Jena, das sich auf die Entwicklung von Softwarelösungen für die Transport- und Logistikbranche spezialisiert hat. Das Unternehmen wurde 1995 gegründet und ist seitdem zu einem führenden Anbieter von Telematik-Lösungen für Lkw und andere Fahrzeuge sowie Transportmanagement-Systemen herangewachsen. Die ist in Deutschland ein wichtiger Anbieter von Telematik-Lösungen und Transportmanagement-Systemen. Das Unternehmen ist jedoch auch international tätig und hat Tochtergesellschaften und Partner in verschiedenen Ländern.[1]

Die DAKO GmbH bietet eine breite Palette von Produkten und Dienstleistungen an, die auf die Bedürfnisse von Transport- und Logistikunternehmen zugeschnitten sind. Dazu gehören unter anderem GPS-Ortungssysteme, mobile Datenerfassungsgeräte, Frachtauftrags-Management-Software und Flottenmanagement-Systeme. Ein wichtiger Schwerpunkt der Arbeit von DAKO ist die Optimierung der Transportprozesse und die Effizienzsteigerung von Logistikunternehmen. Hierbei setzt das Unternehmen auf moderne Technologien wie mobile Datenerfassungsgeräte und GPS-Ortungssysteme, um eine genaue und Echtzeit-Überwachung der Transporte zu gewährleisten.[2]

Zu den Kunden der DAKO GmbH zählen zahlreiche namhafte Unternehmen aus der Transport- und Logistikbranche, darunter Speditionen, Fuhrunternehmen und Logistikdienstleister. Insgesamt ist die DAKO GmbH ein wichtiger Player in der Transport- und Logistikbranche und trägt durch seine innovativen Lösungen zur Optimierung der Branche bei.

1.2 Ausgangssituation

1.2.1 Hintergrund

DAKO verwendet das jQuery Framework, um große Mengen dynamischer Daten im Web in tabellarischer Form darzustellen. In Zukunft sollen jedoch eigene Webkomponenten diese Aufgaben übernehmen. Der Hauptgrund für den grundsätzlichen Verzicht auf jQuery ist, dass es in seiner Funktionalität veraltet und nicht mehr zeitgemäß ist. Es bietet zu viele Funktionen, die heutzutage nicht mehr benötigt werden. Früher war es sinnvoll, die verschiedenen Browser auf einen gemeinsamen Stand zu bringen und Darstellungen auf eine einheitliche Art und Weise in jedem Browser möglich zu machen. Heutzutage ist dies jedoch standardisiert, so dass jQuery an sich nicht mehr notwendig ist. Es gibt viele Möglichkeiten, Darstellungen mit anderen Technologien einfacher und vor allem performanter zu machen. Der Hauptgrund für den Verzicht auf jQuery ist, dass es schwer, langsam und zu 90% bis 95% überflüssig ist. Die Implementierung einer Datentabelle in jQuery war damals das einzige große Element, das gut verwendet werden konnte und halbwegs anpassungsfähig war. Sobald man jedoch ein vorgefertigtes Framework oder eine Datentabelle verwendet, kann man das in den seltensten Fällen genauso bearbeiten, wie es benötigt wird. Es gibt immer Elemente, die man vielleicht gar nicht benötigt, die zusätzlichen Overheads generieren und zu viel machen. Die Darstellungen machen nicht das, was benötigt wird, oder bieten nicht die Möglichkeit, sie so anzupassen, wie es erforderlich ist.

Das DAKO System basiert im Frontend und Backend darauf, dass das Frontend mithilfe der Datentabelle die Daten an das Backend übermittelt. Das Backend bearbeitet die Daten durch Filterung und Sortierung und gibt sie in einer Form zurück, die von der Datentabelle im Frontend verarbeitet und dargestellt werden kann. Wenn man auf ein anderes Framework oder eine andere Datentabelle umsteigen will, müsste man möglicherweise das

[1] (DAKO GmbH, 2023)
[2] (Unternehmensprofile - DAKO, 2023)

gesamte System - also Frontend, Zwischenschicht, Backend und alles, was dahinter liegt - komplett austauschen.

1.2.2 Probleme

DAKO plant zukünftig auf jQuery zu verzichten, da es mittlerweile durch neue JavaScript-Funktionen teilweise überflüssig geworden ist und Spaghetti-Code erzeugen kann. Im Vergleich zu CSS ist jQuery auch sehr langsam. Man hat jedoch begrenzte Möglichkeiten, den Code der Datentabelle zu beeinflussen und kann nur auf vorhandene Strukturen aufbauen. Leider können grundlegende Änderungen an der Datentabelle aufgrund von Einschränkungen im System nur schwer oder gar nicht vorgenommen werden. Wenn man direkt in den JavaScript-Code der Datentabelle eingreifen will, könnte dies dazu führen, dass die Tabelle nach einem Update nicht mehr funktioniert. Ein Upgrade auf eine neue Revision mit breaking changes würde erhebliche Anpassungen erfordern. Insgesamt ist das Framework der Datentabelle aus codetechnischer Sicht nicht optimal für unser System geeignet.

1.2.3 Ziel und Aufgabenstellung

Die Aufgabenstellung dieser Arbeit besteht darin, eine native Webkomponente mit TypeScript zu entwickeln, um große und dynamische Datenmengen in tabellarischer Ansicht im Web darzustellen. Die Anforderungen und Funktionen wurden an die Bedürfnisse des Praxispartners DAKO angepasst. Die Komponente soll die Möglichkeit bieten, Inhalte zu suchen, zu filtern und die Kriterien zu speichern. Besonderes Augenmerk wurde daraufgelegt, dass die Komponente unkompliziert konfigurierbar, anpassbar und leistungsstark ist.

Das Ziel, native Webkomponenten zu nutzen, besteht darin, dass man die Komponenten mühelos in all unseren zukünftigen Projekten verwenden könnte. Die Basis ist von DAKO selbst geschrieben und befindet sich in ihrem eigenen Projekt. Jedes neue Projekt kann diese Komponente integrieren und nutzen. Wenn Änderungen an der Komponente veröffentlicht können alle Projekte unkompliziert aktualisiert werden. Dies ist ein großer Vorteil, da keine manuelle Übertragung von Dateien zwischen Ordnern erforderlich ist und alles über "npm" funktioniert. Man kann Pakete teilen, einbinden und hat die volle Kontrolle darüber.

Als Webkomponente bietet es den zusätzlichen Vorteil, dass komplexe Strukturen unproblematischer erstellt werden können. Man kann einen HTML-Tag erstellen und Attribute hinzufügen, um Optionen anzupassen, anstatt komplizierte JavaScript-Objekte zu erstellen, in denen man Objekte erstellen und Klassen aufrufen muss. Selbst Benutzer, die nur wenig Erfahrung mit JavaScript haben, können die Datentabelle-Komponente unkompliziert integrieren. Auf diese Weise kann man viel mehr in unseren unterschiedlichen Projekten erreichen, ohne große, komplexe Dateien aufeinander aufzubauen, die voneinander erben. Dies erleichtert dem Benutzer die Verwendung erheblich und macht die Komponente als Webkomponente besonders wertvoll.

Die Webkomponente sollte auf TypeScript basieren, denn TypeScript hat den Vorteil, dass es eine Typsicherheit gibt und automatische Vervollständigung in VSC vorhanden ist. Wenn mehrere Leute an einer Komponente arbeiten, bekommt man viel leichter einen Überblick darüber, welche Typen gebraucht werden, ohne dass man irgendwo in einer Dokumentation oder in den anderen Quellen nachschauen muss. Das ist bei Strings und bei Zahlen immer noch relativ überschaubar, aber sobald man Konfigurationsobjekte irgendwo hineingibt, ist TypeScript im Team, wo mehrere Leute daran arbeiten, immer ein positiver Effekt.

2 Methoden und Tools

2.1 Web Components

2.1.1 Einleitung

Web Components sind eine leistungsstarke Möglichkeit, wiederverwendbare Komponenten für die Webentwicklung zu erstellen. Sie sind eine Sammlung von Web-Technologien, die es Entwicklern ermöglichen, benutzerdefinierte HTML-Elemente zu erstellen und zu verwenden, die in verschiedenen Webanwendungen verwendet werden können. Diese Komponenten können in einer Reihe von Anwendungsfällen eingesetzt werden, wie z.B. in der Entwicklung von UI-Komponenten, Widgets oder komplett benutzerdefinierten Webanwendungen.

Web Components bestehen aus drei Haupttechnologien: Custom Elements, Shadow DOM und HTML Templates. Custom Elements ermöglichen es Entwicklern, benutzerdefinierte HTML-Elemente zu erstellen und zu registrieren, während Shadow DOM eine Möglichkeit bietet, den Inhalt dieser Elemente zu isolieren und zu manipulieren. HTML Templates bieten eine Möglichkeit, das Markup einer Komponente zu definieren und es später zu instanziieren.

Web Components bieten viele Vorteile für die Webentwicklung, wie z.B. eine höhere Wiederverwendbarkeit von Code, eine bessere Trennung von Bedenken und eine bessere Skalierbarkeit von Webanwendungen. Sie können auch in verschiedenen Frameworks und Bibliotheken wie Angular, React oder Vue.js verwendet werden.[3]

2.1.2 Überblick

Web Components sind kleine Code-Blöcke, die den internen Aufbau von HTML-Elementen zusammen mit CSS und JavaScript vereinen und es ermöglichen, den Code an jeder gewünschten Stelle auf Websites und Web-Apps einzubinden. Die Idee wurde von einer Arbeitsgruppe des World Wide Web Consortium (W3C) entwickelt, welches 1994 vom Web-Erfinder Tim Berners-Lee ins Leben gerufen wurde, um eine Standardisierung aller grundlegenden Web-Technologien zu fördern. Das Web-Components-Modell, das im Jahr 2012 als Standard veröffentlicht wurde, umfasst vier Haupt-Spezifikationen:

- **Custom Elements**: Eine Reihe von JavaScript-APIs zur Definition von benutzerdefinierten Elementen.
- **Shadow DOM**: Eine Reihe von JavaScript-APIs zum Hinzufügen von DOM-Elementen.
- **HTML-Templates**: Mark-up-Vorlagen, die auf der dargestellten Seite nicht sichtbar sind und als Basis für benutzerdefinierte Elemente dienen.

Heutzutage unterstützen alle gängigen Browser den Web-Components-Standard. Es ist möglich, mit den gekapselten HTML-Codes zu arbeiten, indem man alle JavaScript-Frameworks und -Bibliotheken verwendet, die mit HTML arbeiten.[4]

2.1.3 Funktionsweise

Seit Jahren gehören Bibliotheken und Frameworks wie Angular oder jQuery zu den unverzichtbaren Tools für jeden Web-Entwickler. Diese Code-Grundgerüste sind äußerst praktisch und vielseitig und erleichtern die Arbeit bei der Entwicklung von Projekten erheblich. Allerdings erweisen sie sich oft als unflexibel, insbesondere wenn ein Framework-Wechsel ansteht. Aus diesem Grund führte das World Wide Web Consortium (W3C) Web Components ein, einen universellen Rahmen für die einfache und plattformübergreifende Wiederverwendung von HTML-, CSS- und JavaScript-Code. Der

[3] (Kinsta, 2023)
[4] (IONOS, 2023)

"3C-Standard" zeichnet sich durch eine leicht verständliche Syntax aus, von der auch Programmieranfänger profitieren können.

Webkomponente-Bestandteile

Das Web-Components-Modell stützt sich grundsätzlich auf vier Spezifikationen. Zudem gebe ich einige konkrete Beispiele für die Web-Components-Spezifikationen.[5]

Custom Elements

Custom Elements sind spezielle HTML-Tags, die dazu verwendet werden, den Inhalt von HTML, einschließlich CSS-Anweisungen und Skripten, in sich zu kapseln. Sie werden mithilfe der CustomElementRegistry deklariert und bieten eine Reihe von Funktionen, die sie von anderen HTML-Elementen unterscheiden.

Eine wichtige Eigenschaft von Custom Elements ist, dass sie immer mit einem schließenden Tag enden und ihr Name ein DOM-String ist, der immer einen Bindestrich enthält. Darüber hinaus darf der Name innerhalb der CustomElementRegistry nur einmal vorkommen.

Um ein Custom Element zu erstellen, benötigen Entwickler JavaScript und die Methode "define". Diese Methode erlaubt es, ein benutzerdefiniertes HTML-Element zu definieren und seine Funktionalität zu spezifizieren.

Ein Beispiel für eine Webkomponente, das mit Custom Elements erstellt wurde, ist ein individueller DwDatatable. Durch die Verwendung von Custom Elements kann der DwDatatable unkompliziert in HTML-Seiten eingebunden werden und bietet die Möglichkeit, individuelle Funktionalitäten zu implementieren.

Codebeispiel 1: definition ein customElement

```
customElements.define("dw-datatable", DwDatatable);
```

Um dieses Element nun in einer Webanwendung nutzen zu können, reicht folgender Code:

Codebeispiel 2: Nutzung ein customElement im DOM

```
    <dw-datatable
        src="https://jsonplaceholder.typicode.com/todos"
        columns="userId, id, title, completed"
        livefilter="id, title">
    </dw-datatable>
```

Shadow DOM

Web Components besitzen die Fähigkeit, HTML-Elemente abzuschotten und somit zu kapseln. Dies wird durch die Nutzung der Shadow-DOM-API erreicht, die es erlaubt, versteckte DOM-Bäume in den Dokumentenbaum einzufügen. Dabei ist ausschließlich das HTML-Tag des Shadow DOMs sichtbar, was es uns ermöglicht, das Shadow DOM um HTML-Elemente zu erweitern, ohne das Haupt-DOM jedes Mal anpassen zu müssen.

In den letzten Jahren ist die Komplexität von Webprojekten rapide gestiegen. Moderne Seiten sind häufig sehr umfangreich und beinhalten Inhalte aus verschiedenen Quellen wie sozialen Netzwerken, Streaming-Plattformen oder Content Delivery Networks. Wenn man diese Inhalte nicht in iFrames einbetten möchte, ist Shadow DOM eine exzellente Option, um Drittinhalte in das Webprojekt zu integrieren.

Shadow DOM ist eine Unterform des DOM und eine der vier grundlegenden Säulen der 2012 von dem "3C-Konsortium" standardisierten Web Components. Ein Schatten-DOM wird von gängigen Browsern automatisch aus dem HTML-Code generiert, gilt aber nur für

[5] (mdn web docs, 2023)

die spezifischen Projekt-Komponenten und nicht für das gesamte Webprojekt. Shadow DOMs grenzen zudem die enthaltenen Elemente von projektübergreifenden Design- und Strukturierungsvorgaben ab, wie beispielsweise bestimmten CSS-Anweisungen. Kurz gesagt sind Shadow DOMs eigenständige Code-Kapseln innerhalb des DOMs, die einen unabhängigen Gültigkeitsbereich besitzen.[6]

Aufbau Shadow DOM

Der Shadow DOM besteht aus einer oder mehreren sogenannten Shadow Roots, die jeweils ihre eigenen Elemente und Stylesheets enthalten. Jeder Shadow Root ist mit einem bestimmten Element im Haupt-DOM oder in einem anderen Shadow DOM verknüpft, was als Shadow Host bezeichnet wird.

Um einen Shadow DOM zu erstellen, muss man zunächst ein Element erstellen, dem man den Shadow DOM zuweisen möchte. Dann kann man den Shadow DOM mit der Methode element.attachShadow() erstellen und diesem Element zuweisen. Die Methode attachShadow() erwartet ein Objekt mit einer mode Eigenschaft, die den Typ des Shadow DOMs definiert. Es gibt zwei Typen: open und closed. Im open Modus kann man auf den Shadow DOM von außen zugreifen, während er im closed Modus vollständig von außen verborgen ist.

Sobald der Shadow DOM erstellt wurde, können Elemente und Stylesheets in den Shadow Root eingefügt werden. Dabei wird das Element oder der Style nur im Scope des Shadow Roots angewendet und hat keine Auswirkungen auf den Haupt-DOM oder andere Shadow DOMs. Der Übergang zwischen dem Haupt-DOM und dem Shadow DOM wird als Shadow Boundary bezeichnet.[7]

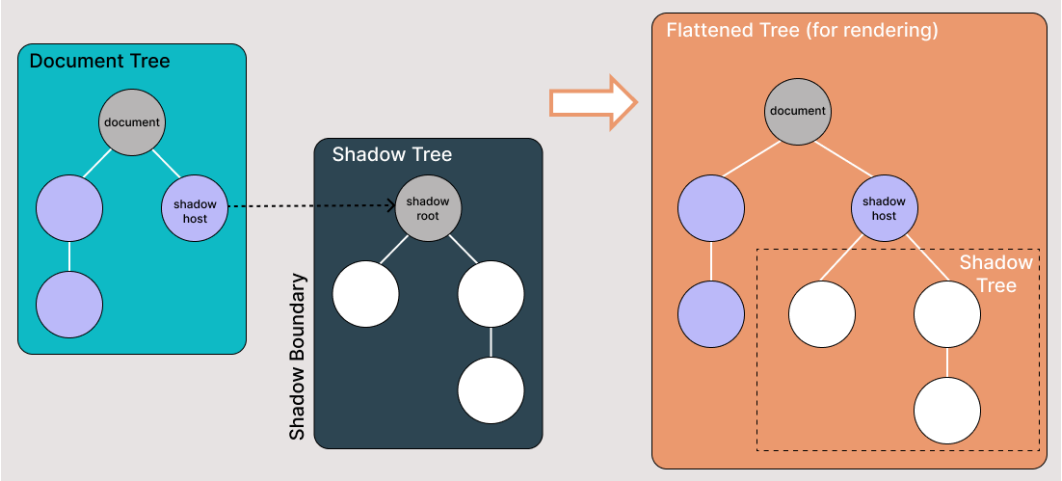


Abbildung 1: Shadow Host A1[8]

HTML-Templates

HTML-Templates sind strukturierte Vorlagen für HTML-Dateien, welche Elemente enthalten, die erst dann gerendert werden, wenn sie aufgerufen werden. Dies vermeidet unnötige Verzögerungen während der Ladezeit einer Webseite und macht HTML-Templates zu einer vorteilhaften Option im Vergleich zu herkömmlichen JavaScript-Methoden. Mithilfe des <template>-Tags kann man ein HTML-Template definieren.

[6] (IONOS, 2023) (IONOS, 2023)
[7] (mdn web docs, 2023)
[8] (lambdatest, 2023)

Effektive Integration von Daten und Inhalten

Es gibt verschiedene Möglichkeiten, Daten und Inhalte in Webkomponenten einzufügen.

1. **Verwendung von Eigenschaften (Properties):** Man kann Eigenschaften für Ihre Webkomponenten definieren, um Daten und Inhalte zu übergeben. Man kann diese Eigenschaften dann in der Komponente abrufen und anzeigen.
2. **Verwendung von Attributen:** Man kann auch Daten und Inhalte in Webkomponenten mithilfe von Attributen übergeben. In diesem Fall können Sie die `attributeChangedCallback()`-Methode verwenden, um die Änderungen an den Attributen zu überwachen und das HTML der Komponente entsprechend zu aktualisieren.
3. **Verwendung von Slots:** Slots sind ein Mechanismus zum Einfügen von HTML-Inhalten in eine Webkomponente. Man kann Slots verwenden, um Platzhalter für Inhalte zu definieren, die dann von anderen Elementen oder Komponenten ersetzt werden können.[9]

Lifecycle Hooks in Web Components

Vom Augenblick der Erstellung eines Custom Elements bis zu seiner Zerstörung können eine Vielzahl von Ereignissen eintreten. Zum Beispiel wird das Element in den DOM eingefügt, aktualisiert sich bei UI-Ereignissen oder kann aus dem DOM entfernt werden. Dieser Lebenszyklus des Elements wird durch eine Reihe von Callback-Funktionen, auch bekannt als "Custom Element Reactions", gesteuert, die uns ermöglichen, auf wichtige Ereignisse im Leben des Elements zu reagieren.

Die Custom Elements werden besonders sorgfältig aufgerufen, um sicherzustellen, dass der Benutzercode nicht mitten in einem heiklen Prozess ausgeführt wird. Sie werden verzögert aufgerufen, bis alle erforderlichen Schritte ausgeführt wurden. Um sicherzustellen, dass die Hooks in der richtigen Reihenfolge aufgerufen werden, hat jedes Custom Element eine eigene "Custom Element Reaction Queue".

Es gibt mehrere Lifecycle Hooks, die zur Verfügung stehen, um auf bestimmte Ereignisse zu reagieren. Dazu gehören der Constructor, Connected Callback, Disconnected Callback, Attribute Changed Callback und Adopted Callback.[10]

Codebeispiel 3: Code Beispiel LifecycleExample

```
class LifecycleExample extends HTMLElement {
    constructor() {
      super();
    }
    connectedCallback() {
      // called when the element is attached to the document
      // similar to React's "componentDidMount"
    }
    disconnectedCallback() {
      // called on removal
      // similar to React's „componentWillUnmount"
    }
    attributeChangedCallback(attribute, previousValue, currentValue) {
      // called when attributes are added, removed, or changed
    }
    adoptedCallback() {
    }
  }
```

[9] (mdn web docs, 2023)
[10] (Ultimate Courses, 2023)

constructor()

Ein Konstruktor kann genutzt werden, um eine Instanz des Shadow DOM zu erstellen, Ereignis-Listener einzurichten und den Zustand einer Komponente zu initialisieren. Allerdings ist es nicht empfehlenswert, Aufgaben wie Rendern oder Abrufen von Ressourcen im Konstruktor auszuführen. Stattdessen sollten diese Aufgaben im connectedCallback-Hook ausgeführt werden.

In ES6-Klassen ist das Definieren eines Konstruktors optional. Wenn es nicht definiert ist, wird von der JavaScript-Engine automatisch ein leerer Konstruktor erstellt. Bei der Erstellung von Custom Elements ist das Definieren eines Konstruktors jedoch obligatorisch. Dies liegt daran, dass Custom Elements vom Benutzer definiert werden und daher der Prototyp und der Konstruktor ebenfalls definiert werden müssen.

Es gibt bestimmte Regeln, die beim Definieren eines Konstruktors befolgt werden müssen. Die erste Anweisung muss ein parameterloser Aufruf von super() sein, um sicherzustellen, dass die Komponente die korrekte Prototypkette erbt und alle Eigenschaften und Methoden der erweiterten Klasse erhält. Es ist verboten, eine return-Anweisung zu verwenden, es sei denn, es handelt sich um eine unkomplizierte vorzeitige Rückgabe (return oder return this). Darüber hinaus können document.write() oder document.open() nicht verwendet werden. Es ist auch nicht ratsam, Attribute oder untergeordnete Elemente des Elements zu überprüfen, da sie noch nicht in das DOM eingefügt wurden und daher noch keine Attribute zur Überprüfung verfügbar sind.

connectedCallback()

Der connectedCallback() ist ein sogenannter Lifecycle-Hook, der aufgerufen wird, wenn ein Custom Element zum ersten Mal in das Dokument eingefügt wird. Es ist ein wichtiger Moment, da das Element jetzt im Dokument vorhanden ist und in der Lage ist, mit anderen Elementen und dem Benutzer zu interagieren. In diesem Hook können verschiedene Aufgaben ausgeführt werden, um das Element zu initialisieren und seine Funktionalität zu gewährleisten.

Einige der möglichen Aufgaben, die in connectedCallback() ausgeführt werden können, sind:

1. Initialisieren des Zustands: Der Zustand des Elements kann initialisiert werden, indem Eigenschaften gesetzt oder Daten geladen werden.
2. Binden von Ereignissen: Ereignis-Listener können auf das Element gebunden werden, um auf Benutzeraktionen zu reagieren.
3. Erstellen von Shadow DOM: Der Shadow DOM kann in connectedCallback() erstellt werden, wenn dies nicht bereits im Konstruktor geschehen ist.
4. Laden von Ressourcen: Benötigte Ressourcen, wie zum Beispiel Stylesheets oder Bilder, können in connectedCallback() geladen werden, um sicherzustellen, dass sie verfügbar sind, wenn das Element gerendert wird.
5. Interaktion mit anderen Elementen: Das Element kann mit anderen Elementen im Dokument interagieren und auf Änderungen in diesen Elementen reagieren.

disconnectedCallback()

Der disconnectedCallback() ist ein weiterer Lifecycle-Hook, der aufgerufen wird, wenn ein Custom Element aus dem Dokument entfernt wird. Es ist ein wichtiger Moment, da das Element jetzt nicht mehr im Dokument vorhanden ist und in der Lage ist, mit anderen Elementen und dem Benutzer zu interagieren. In diesem Hook können verschiedene Aufräumarbeiten durchgeführt werden, um sicherzustellen, dass das Element aufgeräumt und bereit ist, bei Bedarf wieder eingefügt zu werden.

Einige der möglichen Aufgaben, die in disconnectedCallback() ausgeführt werden können, sind:

1. Freigabe von Ressourcen: Benötigte Ressourcen, wie zum Beispiel Event-Listener, können in disconnectedCallback() freigegeben werden, um sicherzustellen, dass keine Speicherlecks oder unerwünschte Aktivitäten auftreten.
2. Speichern von Zustand: Der aktuelle Zustand des Elements kann gespeichert werden, damit er bei einer späteren Einfügung wiederhergestellt werden kann.
3. Interaktion mit anderen Elementen: Das Element kann mit anderen Elementen im Dokument interagieren und auf Änderungen in diesen Elementen reagieren, auch wenn es nicht im Dokument vorhanden ist.

attributeChangedCallback()

Der attributeChangedCallback() ist ein weiterer Lifecycle-Hook, der aufgerufen wird, wenn der Wert eines Attributs eines Custom Elements geändert wird. Es ist ein wichtiger Moment, da das Element jetzt aktualisiert werden muss, um die neuen Attributwerte widerzuspiegeln. In diesem Hook können verschiedene Aktionen ausgeführt werden, um das Element entsprechend zu aktualisieren.

Einige der möglichen Aufgaben, die in attributeChangedCallback() ausgeführt werden können, sind:

1. Aktualisierung des Zustands: Der Zustand des Elements kann aktualisiert werden, indem Eigenschaften oder Daten mit den neuen Attributwerten synchronisiert werden.
2. Aktualisierung des DOM: Das DOM-Rendering des Elements kann aktualisiert werden, um die neuen Attributwerte widerzuspiegeln. Dies kann durch das Hinzufügen, Entfernen oder Ändern von Elementen und Inhalten erfolgen.
3. Durchführung von Validierung: Die neuen Attributwerte können validiert werden, um sicherzustellen, dass sie gültig sind und keine unerwarteten Ergebnisse oder Fehler verursachen.

adoptedCallback()

Der adoptedCallback() ist ein Lifecycle-Hook, der aufgerufen wird, wenn ein Custom Element von einem Dokument in ein anderes Dokument verschoben wird. Es tritt auf, wenn das Element aus einem Dokument gelöscht wird und in ein neues Dokument eingefügt wird. In diesem Hook können verschiedene Aktionen ausgeführt werden, um sicherzustellen, dass das Element bereit ist, im neuen Dokument zu funktionieren.

Einige der möglichen Aufgaben, die in adoptedCallback() ausgeführt werden können, sind:

1. Neuinitialisierung: Das Element kann neu initialisiert werden, um sicherzustellen, dass es im neuen Dokument korrekt funktioniert. Dies kann durch das Zurücksetzen von Eigenschaften oder Daten, die mit dem alten Dokument verknüpft sind, oder durch das Ausführen von Schritten erfolgen, die beim ersten Mal ausgeführt wurden, als das Element erstellt wurde.
2. Aktualisierung von Referenzen: Referenzen auf andere Elemente oder Ressourcen können aktualisiert werden, um sicherzustellen, dass sie im neuen Dokument korrekt sind.
3. Durchführung von Validierung: Das Element und seine Attribute können validiert werden, um sicherzustellen, dass sie im neuen Dokument gültig sind und keine unerwarteten Ergebnisse oder Fehler verursachen.

2.1.4 Web Components Vor- und Nachteile

Vorteile:

- Standardisiert: Native Web Komponenten sind standardisierte HTML-Elemente, die von allen modernen Browsern unterstützt werden. Dies bedeutet, dass sie plattformübergreifend und zukunftssicher sind.
- Wiederverwendbarkeit: Native Web Komponenten sind modular und wiederverwendbar, was die Entwicklung beschleunigt und die Codequalität verbessert. Man kann unkompliziert in anderen Projekten oder Komponenten wiederverwendet werden
- Unabhängigkeit von Frameworks: Native Web Komponenten sind unabhängig von Frameworks und Bibliotheken, was bedeutet, dass sie leicht zu warten und zu aktualisieren sind.
- Leichtgewichtig: Native Web Komponenten sind leichtgewichtig und haben eine geringe Dateigröße, was die Ladezeit der Webseite verkürzt und die Leistung verbessert.

Nachteile:

- Kompatibilität: Obwohl native Webkomponenten in allen modernen Browsern unterstützt werden, kann es bei älteren Browsern zu Kompatibilitätsproblemen kommen.
- Limitierte Funktionalität: Native Webkomponenten bieten nur begrenzte Funktionalität im Vergleich zu benutzerdefinierten Komponenten, die mit JavaScript erstellt wurden.
- Mangel an Browserunterstützung für einige Funktionen: Einige Funktionen werden in einigen Browsern nicht vollständig unterstützt, was die Entwicklung erschweren kann.

2.2 Stylesheets

In der Vergangenheit war CSS die bevorzugte Wahl der Entwickler für die Erstellung von Websites. Doch seit der Entstehung von SASS hat sich dies deutlich geändert, da dessen Verwendung immer weniger wurde. Um genau zu sein, hat SCSS, als erweiterte Version von SASS, an Popularität gewonnen und wird heutzutage häufiger genutzt.

2.2.1 CSS

CSS, was für "Cascading Style Sheets" steht, ist eine Technologie, die zur Gestaltung von Webseiten verwendet wird. Mit CSS kann man das Aussehen einer Webseite definieren, indem man Elemente wie Texte, Bilder, Hintergründe und Layouts formatiert. Im Gegensatz zu HTML, das die Struktur der Webseite definiert, beschreibt CSS, wie die Webseite gestaltet wird.

CSS verwendet eine Vielzahl von Selektoren, um Elemente auf einer Webseite auszuwählen und ihnen Styles zuzuweisen. Es gibt verschiedene Arten von Selektoren, wie beispielsweise Elementselektoren, Klassen- und ID-Selektoren, Pseudoklassen und Pseudoelemente. CSS bietet auch eine breite Palette von Stil-Optionen, darunter Farben, Schriftarten, Größen, Abstände, Positionen und Animationen.

Um CSS auf einer Webseite anzuwenden, kann man es entweder direkt in die HTML-Datei einfügen oder in einer separaten CSS-Datei speichern und diese dann mit dem HTML-Dokument verknüpfen. Durch die Verwendung von CSS kann man das Erscheinungsbild einer Webseite schnell und unkompliziert ändern, ohne dabei den HTML-Code selbst verändern zu müssen. Das macht CSS zu einem unverzichtbaren Werkzeug für Webdesigner und Entwickler.

CSS hat sowohl Vor- als auch Nachteile, die im Folgenden näher erläutert werden:
Vorteile:

- Trennung von Inhalt und Layout: Durch die Verwendung von CSS wird das Aussehen der Webseite vom Inhalt getrennt, was die Wartung und Aktualisierung der Webseite erleichtert.
- Konsistenz: CSS ermöglicht es, das Erscheinungsbild der gesamten Webseite konsistent zu gestalten und zu wahren, was ein professionelles und ansprechendes Aussehen garantiert.
- Flexibilität: CSS bietet eine große Auswahl an Gestaltungsmöglichkeiten, um eine Webseite ansprechend und funktional zu gestalten. Mit CSS kann man schnell Änderungen an der Gestaltung vornehmen, ohne den HTML-Code zu ändern.
- Barrierefreiheit: Durch die Verwendung von CSS kann man eine Webseite barrierefreundlich gestalten, da es ermöglicht, den Inhalt der Webseite in einer logischen Reihenfolge zu präsentieren und ihn für Screenreader und andere Assistenztechnologien zugänglich zu machen.

Nachteile:

- Lernkurve: CSS kann anfangs schwierig zu erlernen sein und erfordert einige Zeit, um die Konzepte und Syntax zu verstehen.
- Browser-Kompatibilität: CSS wird von verschiedenen Browsern unterschiedlich interpretiert, was zu Darstellungsfehlern führen kann. Um sicherzustellen, dass eine Webseite in allen Browsern gleich aussieht, müssen spezifische Anpassungen vorgenommen werden.
- Zusätzliche Dateien: Da CSS in einer separaten Datei gespeichert wird, bedeutet dies zusätzliche Ladezeit und somit eine Verzögerung beim Aufrufen der Webseite.
- Abhängigkeit von JavaScript: Einige komplexe CSS-Funktionen erfordern die Verwendung von JavaScript, was die Webseite schwerer und langsamer machen kann.

2.2.2 SCSS

SCSS, was für "Sassy CSS" steht, ist eine erweiterte Version von CSS, die als Präprozessor fungiert. Ein Präprozessor ist eine Software, die den geschriebenen Code in eine andere Form umwandelt, bevor er von einem Browser interpretiert wird. SCSS bietet zusätzliche Funktionen, die CSS nicht hat, und erleichtert damit die Gestaltung von Webseiten.

Eine der Hauptfunktionen von SCSS ist die Möglichkeit, Variablen zu verwenden. Mit Variablen kann man Werte speichern, die in mehreren Stellen im Code wiederverwendet werden können. Das macht den Code übersichtlicher und erleichtert die Wartung der Webseite.

SCSS bietet auch sogenannte Mixins, die es ermöglichen, eine Gruppe von Styles zu definieren und diese dann auf mehrere Elemente anzuwenden. Das spart Zeit und reduziert die Menge an redundantem Code, was die Wartung der Webseite weiter erleichtert.

Eine weitere wichtige Funktion von SCSS sind verschachtelte Selektoren. Durch die Verwendung von verschachtelten Selektoren kann man den Code besser organisieren und eine klare Struktur schaffen. Das macht den Code leichter lesbar und reduziert die Fehleranfälligkeit.

SCSS bietet auch die Möglichkeit, mathematische Operationen durchzuführen, wie z.B. Addition, Subtraktion, Multiplikation und Division. Das erleichtert die Erstellung von dynamischen Webseiten, bei denen bestimmte Werte, wie z.B. Abstände und Größen, automatisch berechnet werden.

SCSS bietet viele Vorteile im Vergleich zu herkömmlichem CSS, aber es gibt auch einige Nachteile, die beachtet werden sollten:

- Komplexität: SCSS ist komplexer als reines CSS und erfordert mehr Lernaufwand. Entwickler müssen sich mit neuen Konzepten wie Variablen, Mixins und verschachtelten Selektoren vertraut machen, um sie effektiv nutzen zu können.
- Abhängigkeit von einer Compiler-Software: Da SCSS nicht direkt von Browsern interpretiert werden kann, benötigt man eine Compiler-Software, um den SCSS-Code in reguläres CSS umzuwandeln. Das bedeutet, dass man zusätzliche Schritte ausführen muss, um die SCSS-Datei zu kompilieren und die Ergebnisse zu überprüfen.
- Zusätzliche Dateien und Speicherplatz: Da man separate SCSS-Dateien erstellt, benötigt man mehr Speicherplatz auf dem Server, um die zusätzlichen Dateien zu speichern. Dies kann insbesondere bei größeren Webseiten zu Problemen führen.
- Browserkompatibilität: Obwohl SCSS eine erweiterte Version von CSS ist, kann es trotzdem zu Kompatibilitätsproblemen mit älteren Browsern kommen, die möglicherweise nicht alle SCSS-Funktionen unterstützen. In solchen Fällen müssen Entwickler alternative Lösungen finden, um sicherzustellen, dass die Webseite in allen Browsern korrekt angezeigt wird.
- Potenzielle Sicherheitsprobleme: Da SCSS zusätzliche Funktionen bietet, besteht ein höheres Risiko für Sicherheitsprobleme, insbesondere wenn nicht sorgfältig mit Variablen und Mixins umgegangen wird. Es ist wichtig sicherzustellen, dass der SCSS-Code sicher und frei von Sicherheitslücken ist.[11]

Insgesamt ist SCSS eine nützliche Ergänzung zu CSS, aber es erfordert zusätzlichen Aufwand und es gibt einige Nachteile, die beachtet werden sollten. Es ist wichtig, die Vor- und Nachteile von SCSS sorgfältig abzuwägen, um zu entscheiden, ob es die richtige Wahl für ein bestimmtes Projekt ist.[12]

2.3 Constructable Stylesheets

"Konstruierbare Stylesheets" oder "Constructable Stylesheets" sind eine praktische Möglichkeit, um wiederverwendbare Stilvorlagen zu erstellen und zu verteilen, insbesondere in Verbindung mit der Verwendung von Shadow DOM. Mit Constructable Stylesheets können Entwickler CSS-Regeln definieren und sie als eigenständige Module in andere Komponenten einbinden. Dies ermöglicht eine effizientere Gestaltung und Wartung von Webseiten, da sich Stilvorlagen problemlos auf verschiedene Elemente anwenden lassen, ohne dass sie miteinander interferieren.

Mit Constructable Stylesheets können Entwickler auch die Leistung ihrer Webseiten verbessern, da Stilvorlagen nur einmalig heruntergeladen und zwischengespeichert werden müssen, anstatt jedes Mal neu heruntergeladen zu werden, wenn ein neues Element auf der Seite erstellt wird. Dies reduziert die Latenz und sorgt für eine reibungslosere Benutzererfahrung.

Darüber hinaus ermöglicht die Verwendung von Constructable Stylesheets eine klare Trennung zwischen der Gestaltung und der Funktionalität einer Webseite, was die Wartung

[11] (javapoint, 2023)
[12] (nextgeneration, 2023)

und Skalierung von Code erleichtert. Entwickler können die Stilvorlagen auf unkomplizierte Weise aktualisieren oder ersetzen, ohne dass dadurch die Funktionalität der Webseite beeinträchtigt wird.[13]

2.3.1 Einleitung

Es war schon immer möglich, mit Hilfe von JavaScript Stylesheets zu erstellen. Allerdings gab es in der Vergangenheit einen Prozess, bei dem ein <style>-Element mit document.createElement('style') erstellt wurde und dann auf dessen Blatteigenschaft zugegriffen wurde, um eine Referenz auf die zugrunde liegende CSSStyleSheet-Instanz zu erhalten. Diese Methode hatte jedoch den Nachteil, dass sie doppelten CSS-Code und eine unnötige Aufblähung verursachen konnte, und das Anhängen des Elements führte zu einem unerwünschten Aufblitzen von ungestyltem Inhalt, unabhängig davon, ob eine Aufblähung vorlag oder nicht.

Die CSSStyleSheet-Schnittstelle bildet nun die Grundlage einer Sammlung von CSS-Darstellungsschnittstellen, die als CSSOM bezeichnet werden. Diese Schnittstellen ermöglichen eine programmgesteuerte Manipulation von Stylesheets und beseitigen die Probleme, die mit der alten Methode verbunden waren.

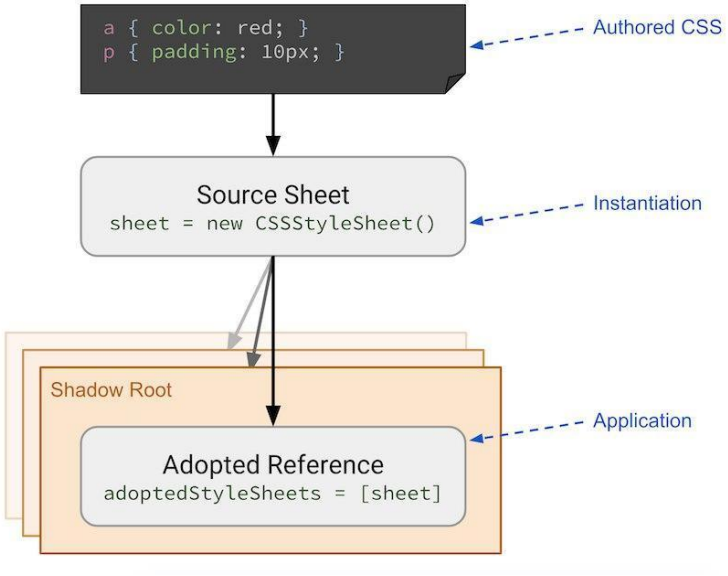


Abbildung 2: Einfügung CSS-Datei in Shadow Root

Constructable Stylesheets bieten die Möglichkeit, gemeinsame CSS-Stile vorzubereiten und dann ohne Duplizierung auf mehrere Schattenwurzeln oder das Dokument anzuwenden. Änderungen an einem gemeinsam genutzten CSSStyleSheet werden automatisch auf alle Roots angewendet, in die es integriert wurde. Diese Assoziation ist nützlich für eine Vielzahl von Anwendungen, wie die Bereitstellung eines zentralisierten Designs, die Verteilung von benutzerdefinierten CSS-Eigenschaftswerten an bestimmte DOM-Teilbäume und als direkte Schnittstelle zum CSS-Parser des Browsers. Dadurch können Stylesheets vorab geladen werden, ohne dass sie in das DOM eingefügt werden müssen.[14]

2.3.2 Überblick

Statt eine frische API einzuführen, um diese Funktionalität zu erreichen, bietet die Constructable Stylesheets-Spezifikation eine Möglichkeit, Stylesheets sicherer zu erstellen. Hierbei wird der CSSStyleSheet() Konstruktor aufgerufen, um Stylesheets zu erstellen. Das daraus resultierende CSSStyleSheet-Objekt beinhaltet zwei neue Methoden, welche das Hinzufügen und Aktualisieren von Stylesheet-Regeln auf eine Art und Weise ermöglichen, die Flash of Unstyled Content (FOUC) verhindert. Diese

[13] (mdn web docs, 2023)
[14] (web.dev, 2023)

Methoden heißen replace() und replaceSync(). Beide ersetzen das Stylesheet durch einen CSS-String, jedoch gibt replace() ein Promis zurück. Es ist wichtig zu erwähnen, dass externe Stylesheet-Referenzen nicht unterstützt werden - alle @import-Regeln werden ignoriert und lösen eine Warnung aus.

2.3.3 Funktionsweise

Eine zusätzliche Funktion, die durch Constructable StyleSheets hinzugefügt wird, ist die adoptedStyleSheets Eigenschaft, die sowohl für Shadow Roots als auch für Dokumente verfügbar ist. Diese Eigenschaft ermöglicht es uns, die durch ein CSSStyleSheet definierten Stile spezifisch auf einen bestimmten DOM-Unterbaum anzuwenden. Hierzu setzt man die Eigenschaft auf ein Array von einem oder mehreren Stylesheets, die auf dieses Element angewendet werden sollen.

Codebeispiel 4: Einfügung CSS-Datei ins DOM

```
// Create shared stylesheet:
const sheet = new CSSStyleSheet();
sheet.replaceSync('a { color: red; }');
// Applay the stylesheet to a document
document.adoptedStyleSheets = [sheet];
//Apply the stylesheet to a Shadow Root:
const node      = document.createElement('div');
const shadow = node.attachShadow({ mode: 'open' });
shadow.adoptedStyleSheets = [sheet];
```

Es ist wichtig zu beachten, dass man bei der Verwendung von adoptedStyleSheets den Wert überschreiben muss, anstatt das vorhandene Array zu ändern. Dies liegt daran, dass das Array eingefroren ist, und In-Place-Mutationen wie "push()" eine Ausnahme auslösen würden. Daher muss man ein neues Array zuweisen. Um alle bereits vorhandenen StyleSheets beizubehalten, die über adoptedStyleSheets hinzugefügt wurden, kann man die "concat"-Funktion verwenden. Dadurch wird ein neues Array erstellt, das sowohl die vorhandenen Sheets als auch die hinzuzufügenden Sheets enthält.

Codebeispiel 5: Einfügung weitere CSS-Datei ins DOM

```
const sheet = new CSSStyleSheet();
sheet.replaceSync( `a { color: red }` );

// Combine existing sheets with new one:
document.adoptedStyleSheets = [… document.adoptedStyleSheets, sheet];
```

2.4 JavaScript

2.4.1 Einleitung

JavaScript ist eine Programmiersprache, die hauptsächlich in Web-Browsern verwendet wird, um Webseiten dynamisch und interaktiv zu gestalten. Es wurde in den späten 1990er Jahren entwickelt und ist seitdem zu einer der beliebtesten Programmiersprachen geworden.

JavaScript kann verwendet werden, um HTML-Elemente zu manipulieren, Ereignisse zu handhaben, Formulare zu validieren, Animationen zu erstellen und vieles mehr. Es ist eine clientseitige Sprache, die direkt im Web-Browser des Benutzers ausgeführt wird, was bedeutet, dass der Benutzer nicht auf einen Server zugreifen muss, um auf dynamischen Inhalt zuzugreifen oder mit ihm zu interagieren.

Die Sprache wird kontinuierlich weiterentwickelt und aktualisiert, um neue Funktionen und Tools bereitzustellen. Es gibt viele Frameworks und Bibliotheken, die auf JavaScript aufbauen und es erleichtern, komplexe Aufgaben zu erledigen.[15]

[15] (TenMedia, 2023)

2.4.2 Überblick

JavaScript ist eine vielseitige Programmiersprache, die in zahlreichen Webanwendungen genutzt wird. Sie dient beispielsweise dazu, automatisch aktualisierte Newsfeeds, interaktive Formulare und Suchfunktionen zu erstellen. Praktisch alle interaktiven Funktionen, die man auf modernen Webseiten findet, werden mithilfe von JavaScript realisiert.

JavaScript wird von allen gängigen Webbrowsern unterstützt und ist deshalb aus der Frontend-Entwicklung nicht mehr wegzudenken. Doch damit nicht genug: JavaScript kommt auch in anderen Bereichen zum Einsatz. So nutzen Entwicklerinnen und Entwickler diese Sprache beispielsweise für die Erstellung von mobilen Anwendungen, für die Entwicklung von Videospielen und für Backend-Prozesse.

Bibliotheken und –Frameworks

Es gibt viele JS-Bibliotheken und -Frameworks, die die Programmierung erleichtern. Bibliotheken helfen Entwicklern, komplexere Funktionen schnell und effizient zu implementieren, indem sie vorgefertigte Code-Snippets zur Verfügung stellen. Dadurch können Entwickler Zeit sparen und wiederverwendbaren Code erstellen. Frameworks sind dagegen vorgefertigte Strukturen, die Entwicklern eine bestimmte Codestruktur vorgeben, die sie unkompliziert befolgen können. Dies kann die Entwicklung vereinfachen, aber auch bestimmte Einschränkungen mit sich bringen.

Es gibt viele bekannte JS-Bibliotheken, darunter jQuery, ReactJS, Dojo-Toolkit und Google Polymer. Auch JS-Frameworks wie AngularJS, Vue.js und Apache Royale sind weit verbreitet. Wenn JavaScript ohne Bibliotheken oder Frameworks geschrieben wird, spricht man von "Vanilla-JavaScript".

Die Verwendung von Bibliotheken und Frameworks ist eine effektive Möglichkeit, um die Entwicklung von Webanwendungen zu beschleunigen und Code-Wiederverwendbarkeit zu fördern. Allerdings sollte man bei der Auswahl einer Bibliothek oder eines Frameworks die Bedürfnisse des Projekts berücksichtigen und sicherstellen, dass sie die gewünschte Funktionalität bieten.

Clientseitiges vs. Serverseitiges

JavaScript ist eine Programmiersprache, die sowohl auf der Clientseite als auch auf der Serverseite eingesetzt werden kann. Auf der Clientseite ermöglicht JavaScript die Interaktion mit dem Browser und ermöglicht die Manipulation von Webseiten. Dies ist eine der häufigsten Verwendungen von JavaScript.

Serverseitiges JavaScript dagegen läuft auf dem Server und nicht im Browser. Es ist weniger verbreitet als clientseitiges JavaScript, bietet jedoch einige Vorteile. Beispielsweise kann serverseitiges JavaScript Daten auf dem Server verarbeiten, bevor sie an den Browser des Nutzers gesendet werden. Ein Beispiel für ein beliebtes serverseitiges JavaScript-Framework ist Node.js.

Obwohl serverseitiges JavaScript weniger verbreitet ist, kann es in bestimmten Anwendungsfällen sehr nützlich sein. Es kann beispielsweise bei der Entwicklung von skalierbaren Webanwendungen und APIs helfen, indem es asynchrone und ereignisgesteuerte Verarbeitungsmuster ermöglicht. Clientseitiges JavaScript ist jedoch nach wie vor die gängigste Form und wird für eine Vielzahl von Webanwendungen und Interaktionen eingesetzt.

2.4.3 JavaScript Vor- und Nachteile

JavaScript hat wie jede Programmiersprache Vor- und Nachteile. Im Folgenden sind einige der wichtigsten Vor- und Nachteile von JavaScript aufgeführt:

Vorteile:

- Breite Unterstützung: JavaScript wird von allen gängigen Browsern unterstützt, was es zu einer wichtigen Sprache für die Frontend-Entwicklung macht.
- Interaktivität: JavaScript ermöglicht es, Webseiten interaktiver zu gestalten und Benutzereingaben zu verarbeiten, wodurch die Nutzererfahrung verbessert wird.
- Unkomplizierte Einbindung: JavaScript-Code kann unkompliziert in eine HTML-Datei eingebettet werden, was die Einbindung von Skripten sehr unkompliziert macht.
- Reichhaltige Bibliotheken und Frameworks: Es gibt zahlreiche Bibliotheken und Frameworks, die Entwicklern helfen, schnell komplexe Anwendungen zu entwickeln.
- Vielseitigkeit: JavaScript kann sowohl clientseitig als auch serverseitig eingesetzt werden und eignet sich für die Entwicklung von Webanwendungen, mobilen Anwendungen, Desktopanwendungen und sogar für die Spieleentwicklung.

Nachteile:

- Browser-Abhängigkeit: JavaScript kann je nach Browser unterschiedlich interpretiert werden, was die Entwicklung komplizierter machen kann.
- Sicherheitsbedenken: JavaScript kann für Angriffe auf Benutzer verwendet werden, daher müssen Sicherheitsmaßnahmen ergriffen werden, um potenzielle Bedrohungen zu minimieren.
- Leistungseinschränkungen: JavaScript kann bei der Verarbeitung großer Datenmengen langsamer sein als andere Sprachen.
- Skalierbarkeit: Bei der Entwicklung größerer Anwendungen kann es schwierig sein, den Code in einer Weise zu organisieren, die Skalierbarkeit und Wartbarkeit gewährleistet.
- Lernkurve: JavaScript kann für Anfänger eine steile Lernkurve haben und erfordert Kenntnisse in HTML, CSS und anderen Technologien, um eine vollständige Webanwendung zu erstellen.

Trotz einiger Nachteile ist JavaScript eine sehr wichtige Sprache für die Frontend-Entwicklung und wird für eine Vielzahl von Webanwendungen eingesetzt. Mit den richtigen Kenntnissen und Tools können Entwickler qualitativ hochwertige Anwendungen erstellen, die den Anforderungen ihrer Kunden gerecht werden.

2.5 TypeScript

TypeScript ist eine von Microsoft entwickelte Programmiersprache, die auf JavaScript aufbaut und es um wichtige Funktionen erweitert. Im Gegensatz zu JavaScript ermöglicht TypeScript die explizite Angabe von Datentypen bei der Variablendefinition und unterstützt verschiedene Datentypen wie Booleans, Zahlen und Strings. Ein weiterer Vorteil von TypeScript ist, dass es auf objektorientierter Programmierung basiert und daher Konzepte wie Klassen, Vererbung und Module bietet. Im Gegensatz dazu ist JavaScript eine Dynamic-Interpreted Sprache, die den Typ zur Laufzeit und nicht zur Kompilierzeit überprüft. TypeScript hingegen ist eine Static-Compiled Sprache, was bedeutet, dass ein Typfehler bereits während der Kompilierung erkannt wird, bevor der Code überhaupt

ausgeführt wird. Um TypeScript-Code im Browser nutzen zu können, muss dieser durch einen Transcompiler angepasst werden.[16]

Codebeispiel 6: JavaScript Code Beispiel

```
// sample.js
function add(x, y) {
    return x + y;
}
Console.log(add(2, 2)); // 4
```

Codebeispiel 7: TypeScript Code Beispiel

```
// sample.ts
function add(x: number, y: number): number {
    return x + y;
}
Console.log(add(2, 2)); // 4
```

Die beiden Beispiele zeigen auf, wie die gleiche Funktion in beiden Sprachen geschrieben wird. In TypeScript wird der Typ der Variable und die Funktion in dem Vompiler-Time angegeben. Diese ist aber nicht der Fall für JavaScript, wo die Variable in dem Run-Time bestimmt werden kann.

2.5.1 TypeScript Compiler TSC

Ein weiterer potenzieller Vorteil von TypeScript besteht darin, dass keine neue Technologie auf dem Browser erforderlich ist, da der fertige TypeScript-Code in reguläres JavaScript kompiliert wird, das dann auf jedem Browser ausgeführt werden kann.

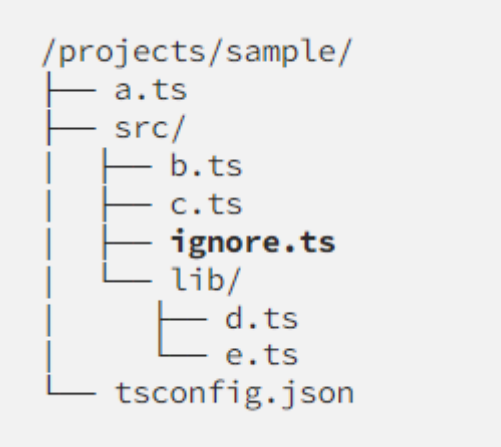


Abbildung 3: TS Projekt Struktur A2

Abbildung 4: tsconfig.json Datei A2

Wenn der Compiler von TypeScript ausgeführt wird, sucht er nach der tsconfig.json Datei. Diese Datei hat die Konfigurationen, die der Compiler benötigt z.B. wohin die kompilierten JavaScript Dateien abgelegt werden sollen.[17]

Die Abbildungen 04 und 05 zeigen ein Beispiel, wie die tsconfig.json Datei aussehen kann, um alle .ts Dateien außer ignore.ts zu Kompilieren.

[16] (typescriptlang, 2023)
[17] (medium, 2023)

2.6 Vergleich von TypeScript und JavaScript

TypeScript und JavaScript sind beide Programmiersprachen, die eng miteinander verwandt sind und von Microsoft entwickelt wurden. Obwohl sie viele Gemeinsamkeiten haben, gibt es auch einige Unterschiede zwischen den beiden Sprachen.

Einige der wichtigsten Unterschiede sind:

1. Typisierung: TypeScript unterstützt eine statische Typisierung, während JavaScript eine dynamische Typisierung hat. Das bedeutet, dass in TypeScript Variablen und Funktionen einen festen Datentyp haben, der zur Kompilierzeit überprüft wird, während JavaScript Variablen und Funktionen einen Datentyp haben, der zur Laufzeit bestimmt wird.
2. Skalierbarkeit: TypeScript wurde speziell für die Entwicklung großer Anwendungen entwickelt, während JavaScript eher für kleinere Projekte geeignet ist. TypeScript bietet durch seine statische Typisierung und seine erweiterte Objektorientierung eine bessere Skalierbarkeit und Wartbarkeit von Code.
3. Kompatibilität: Da TypeScript eine Übermenge von JavaScript ist, ist der TypeScript-Code in der Regel vollständig kompatibel mit JavaScript. Dies bedeutet, dass man problemlos von JavaScript zu TypeScript wechseln und umgekehrt wechseln kann.
4. Entwicklungswerkzeuge: TypeScript wird oft mit einer Vielzahl von Entwicklungswerkzeugen und -editoren geliefert, die speziell für TypeScript entwickelt wurden. Obwohl viele dieser Tools auch für JavaScript verfügbar sind, ist die Unterstützung für TypeScript oft umfangreicher.
5. Lernkurve: TypeScript kann für Anfänger eine steilere Lernkurve haben als JavaScript, da es zusätzliche Konzepte wie statische Typisierung und erweiterte Objektorientierung enthält.

Zusammenfassend lässt sich sagen, dass TypeScript speziell für die Entwicklung großer Anwendungen entwickelt wurde und durch seine statische Typisierung und erweiterte Objektorientierung eine bessere Skalierbarkeit und Wartbarkeit von Code bietet. JavaScript hingegen ist besser geeignet für kleinere Projekte, die weniger komplexe Code-Strukturen erfordern. Beide Sprachen haben ihre Stärken und Schwächen, und die Wahl der Sprache hängt oft von den Anforderungen des Projekts und den Kenntnissen des Entwicklers ab.[18]

[18] (K & C, 2023)

3 Zielbestimmung

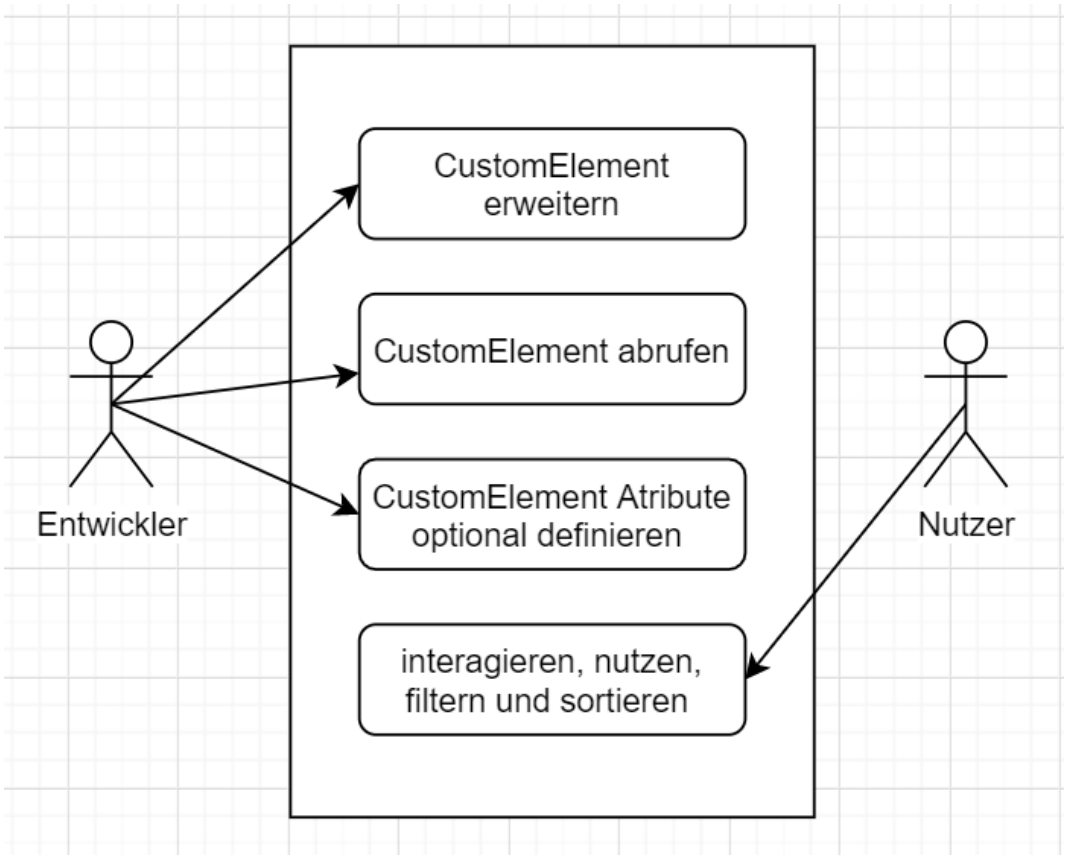


Abbildung 5: Stakeholder

Bei den Prototypen sind zwei Hauptakteure beteiligt:

- DAKO-Entwickler: Diese können das CustomElement updaten und weiter erweitern, optional Attribute definieren und beliebig abrufen und nutzen.
- Externe Nutzer: Diese können mit der Datentabelle interagieren, Daten anschauen, Daten sortieren und filtern.

3.1 Musskriterien

Anzeigen von Daten: Die Datentabelle muss in der Lage sein, Daten in einer Tabelle anzuzeigen, einschließlich der Möglichkeit, Daten zu sortieren, filtern und zu suchen.

Responsivität: Die Datentabelle sollte responsiv sein und sich an verschiedene Bildschirmgrößen anpassen, um eine optimale Benutzererfahrung auf verschiedenen Geräten zu gewährleisten.

Live-Filterung und -Suche: Die Datentabelle sollte eine Live-Filterung und -Suche ermöglichen, d.h. die Ergebnisse sollten sich in Echtzeit aktualisieren, während der Benutzer die Suchkriterien eingibt.

Fehlerbehandlung: Die Datentabelle sollte mit Fehlern und Problemen umgehen können, z.B. wenn die Daten nicht geladen werden können oder wenn es Probleme mit der Netzwerkverbindung gibt.

Datenmenge: Die Datentabelle sollte in der Lage sein, eine große Menge an Daten effizient zu verarbeiten, ohne dass es zu Leistungsproblemen kommt.

Datenformat: Die Datentabelle sollte in der Lage sein, mit verschiedenen Datenformaten umzugehen, wie z.B. Zahlen, Texte, Datumsangaben, und anderen Datentypen.

Interaktionsmöglichkeiten: Die Datentabelle sollte verschiedene Interaktionsmöglichkeiten für den Benutzer bereitstellen, wie z.B. Filterung, Sortierung, Suche, und Markierung.

Performance: Die Datentabelle sollte schnell und reaktionsfreudig sein, um ein gutes Benutzererlebnis zu gewährleisten.

3.2 Wunschkriterien

Kompatibilität: Die Datentabelle sollte auf verschiedenen Geräten und in verschiedenen Browsern einheitlich und zuverlässig funktionieren.

Anpassbare Spaltenbreite: Die Möglichkeit, die Breite der Spalten anzupassen, um die Anzeige von Daten zu optimieren.

Exportfunktion: Die Möglichkeit, die Daten in verschiedenen Formaten (z.B. CSV, Excel) zu exportieren.

Drag-and-Drop-Funktionalität: Die Möglichkeit, Daten per Drag-and-Drop in der Tabelle zu verschieben und zu sortieren.

Echtzeitaktualisierungen: Die Möglichkeit, Daten in Echtzeit zu aktualisieren, ohne dass der Benutzer die Seite neu laden muss.

3.3 Abgrenzungskriterien

Datenquellen: Die Datentabelle sollte in der Lage sein, Daten aus verschiedenen Datenquellen zu beziehen, wie z.B. aus einer Datenbank, einem JSON-Feed oder einer REST API.

Anzeigeoptionen: Die Datentabelle sollte in der Lage sein, Daten in verschiedenen Formaten anzuzeigen, wie z.B. als Tabelle, Kacheln oder Karten.

Interaktionsmöglichkeiten: Die Datentabelle sollte verschiedene Interaktionsmöglichkeiten für den Benutzer bereitstellen, wie z.B. Paginierung und Gruppierung von Daten.

Anpassbarkeit: Die Datentabelle sollte anpassbar sein, so dass sie den Anforderungen des Projekts oder der Anwendung entspricht.

Barrierefreiheit: Die Datentabelle sollte für alle Benutzer zugänglich sein, auch für Menschen mit Behinderungen. Dazu gehören z.B. die Unterstützung von Screenreadern und die Verwendung von barrierefreien Farbkombinationen.

4 Software Requirements Specification SRS

4.1 Funktionale Anforderungen

Spalten Sortierung

- Die Benutzer sollten in der Lage sein, Tabellen oder Listen nach einer oder mehreren Spalten zu sortieren.
- Die Spalten-Sortierung sollte sowohl aufsteigend als auch absteigend erfolgen können.
- Wenn die Benutzer eine Spalte sortieren, sollten die Daten in allen anderen Spalten der Tabelle oder Liste entsprechend sortiert werden.
- Die Sortierung sollte in Echtzeit aktualisiert werden, wenn Daten in der Tabelle oder Liste hinzugefügt, gelöscht oder geändert werden.

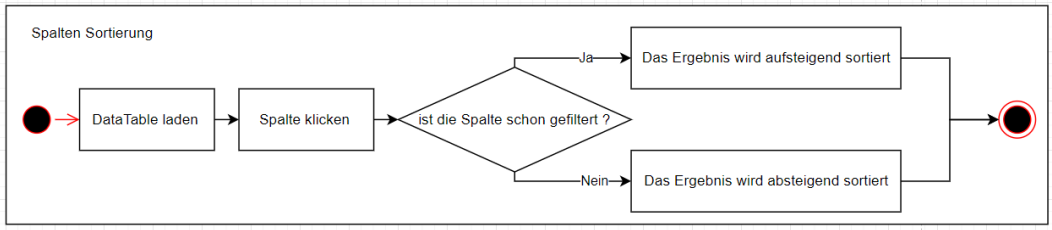


Abbildung 6: Spalten Sortierung Aktivitätsdiagramm

Live Search Filter definieren

- Echtzeit-Updates: Das System muss in der Lage sein, die Ergebnisse in Echtzeit zu aktualisieren, wenn der Benutzer Filterkriterien oder Suchbegriffe ändert.
- Filteroptionen: Das System muss dem Benutzer verschiedene Filteroptionen zur Verfügung stellen, damit dieser die Suche nach seinen Bedürfnissen anpassen kann. Hierbei können Filterkriterien wie Preis, Kategorie, Datum oder andere Eigenschaften verwendet werden.
- Suchbegriff-Unterstützung: Das System sollte in der Lage sein, die Eingabe des Benutzers zu erkennen und Vorschläge für passende Suchbegriffe oder Autovervollständigung anbieten.
- Mehrfachfilterung: Das System sollte es dem Benutzer ermöglichen, mehrere Filterkriterien gleichzeitig anzuwenden, um die Suche weiter einzugrenzen.

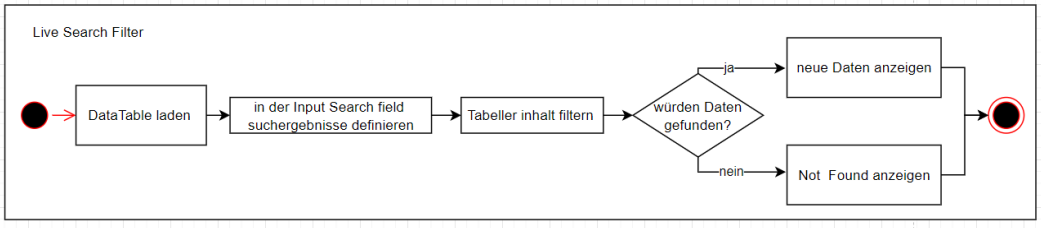


Abbildung 7: Live Search Filter Aktivitätsdiagramm

Zeilen Markieren

- Benutzerinteraktion: Das System sollte es dem Benutzer ermöglichen, eine oder mehrere Zeilen in einer Tabelle auszuwählen und zu markieren.
- Mehrfachmarkierung: Das System sollte dem Benutzer die Möglichkeit bieten, mehrere Zeilen gleichzeitig zu markieren, um die Verarbeitung von Daten zu erleichtern.

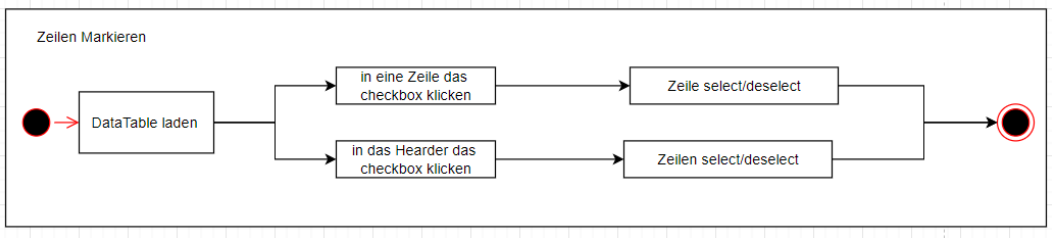


Abbildung 8: Zeilen Markieren Aktivitätsdiagramm

Spalten auswählen

- Das System muss in der Lage sein, Benutzern die Möglichkeit zu geben, bestimmte Spalten anzuzeigen oder auszublenden, um die Sichtbarkeit und Übersichtlichkeit von Daten zu verbessern

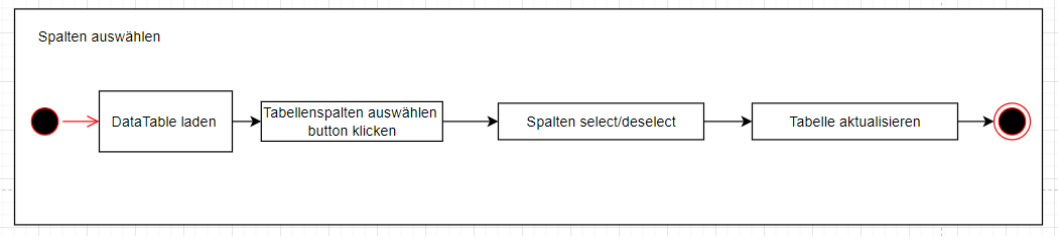


Abbildung 9: Spalten auswählen Aktivitätsdiagramm

Scrollen

- Lazy Loading: Das System sollte in der Lage sein, Daten nur bei Bedarf zu laden, um die Ladezeiten zu minimieren und die Performance zu verbessern.
- Benutzerinteraktion: Das System sollte es dem Benutzer ermöglichen, in der Tabelle zu scrollen, um Daten anzuzeigen, die nicht auf dem Bildschirm sichtbar sind.

4.2 Nicht funktionale Anforderungen

- Skalierbarkeit: Die Datentabelle sollte in der Lage sein, mitwachsenden Datenmengen und Benutzeranforderungen zu skalieren, ohne dass dies zu Einbußen bei der Leistung oder Benutzererfahrung führt.
- Barrierefreiheit: Die Datentabelle sollte für Benutzer mit Einschränkungen wie Sehbehinderungen oder motorischen Beeinträchtigungen zugänglich und nutzbar sein. Hierbei müssen spezielle Techniken wie beispielsweise Screenreader-Unterstützung und Tastaturzugriff berücksichtigt werden.
- Sicherheit: Die Datentabelle Webkomponente sollte sicher sein, indem sie die Privatsphäre und Sicherheit der Benutzerdaten gewährleistet und gegen Sicherheitsbedrohungen wie Cross-Site-Scripting (XSS) und SQL Injection geschützt ist.

5 Umsetzung

5.1 Projektstruktur

Die Projektstruktur besteht aus verschiedenen Unterordnern:

- Im Ordner "demo" befindet sich eine index.html-Datei, die zur Abruf und zum Testen der Webkomponente dient.
- Die TypeScript-Dateien im Projekt wurden mit TSC in JavaScript umgewandelt und im Ordner "dist" gespeichert.
- Im Ordner "src" befindet sich die Implementierung der Webkomponente in index.html, index.scss und index.ts.
- Weitere wichtige Dateien, um das Projekt auszuführen, sind tsconfig.json, package.json und declarations.d.ts.
- "employees.json" ist eine Testdatei, die aus mehr als 327.000 Code-Zeilen besteht und Objekte, verschachtelte Objekte und verschiedene Variablen mit unterschiedlichen Datentypen enthält. Damit wurde die Datentabelle mit Daten gefüllt.

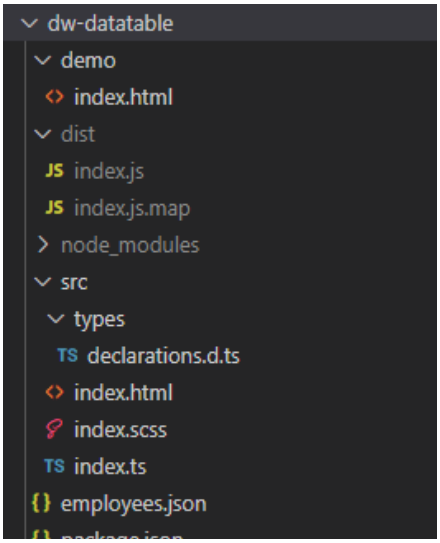


Abbildung 11: Projektstruktur

Abbildung 10: employees.json

5.2 Konfiguration

Die "tsconfig"-Datei dient dazu, das Projekt zu konfigurieren. In ihr wird festgelegt, welche TypeScript-Dateien in JavaScript-Dateien umgewandelt werden sollen, wo die erzeugten JavaScript-Dateien gespeichert werden sollen, welche Compiler-Optionen für das Projekt gelten und weitere Einstellungen.

Codebeispiel 8: tsconfig.ts

```
{
  "extends": "../../tsconfig.base.json",
  "include": [
    "src/**/*.d.ts",
    "src/**/*.ts"
  ],
  "compilerOptions": {
    "baseUrl": "."
  }
}
```

In der "package.json"-Datei werden allgemeine Informationen zum Projekt definiert, wie der Projektname, der Autor, die Version usw. Außerdem werden in ihr Skripte und Abhängigkeiten (Dependencies) konfiguriert.

Codebeispiel 9: package.json

```
{
...
  "scripts": {
    "prepublish": "npm run lint && npm run build && npm run analyze",
    "build": "wcb --scss --ts --loader=html",
    "lint": "eslint ./src",
    "watch": "wcb --scss --ts --loader=html --watch",
    "start": "web-dev-server --node-resolve --open demo/ --watch",
    "dev": "concurrently \"npm:watch\" \"npm:start\"",
    "analyze": "custom-elements-manifest analyze",
    "analyse": "npm run analyze"
  },
  "devDependencies": {
...
}
```

5.3 Installation

1. Klonen des Repositories (nur für DAKO-Mitarbeiter zugänglich):
   HTTPS-Variante:
   git clone https://git.ai.fh-erfurt.de/ba_thesis/ba_thesis_ws2223_molham_alkhodari.git

2. Installation der Pakete:
   PNPM-Variante:
   pnpm install

3. Navigation zum Projekt
   cd .\packages
   cd .\dw-datatable

4. Starten des Servers:
   pnpm run dev (oder demo > index.html **Open with Live Server**)

5.4 Datenbank/ API

Im Projekt wurde mit verschiedenen Fake-APIs und lokalen Test-APIs gearbeitet:

5. https://jsonplaceholder.typicode.com/users: Die "jsonplaceholder users" haben zehn Datensätze und sind relativ übersichtlich. Sie beinhalten verschachtelte Objekte.
6. https://jsonplaceholder.typicode.com/todos: Die "jsonplaceholder todos" haben 200 Datensätze und eignen sich gut zum Testen der Scroll-Funktion.
7. https://fakeapi.andreaspabst.com/api/users: Die "fakeapi users" beinhalten Datums-Spalten und sind daher gut geeignet, um die "DateFormat()" Funktion zu testen.
8. "../employees.json" (lokal im Projekt gespeichert): Die "employees"-Datei ist relativ groß und enthält viele unterschiedliche Spalten, Datentypen und verschachtelte Objekte.

5.5 Aufbau Datentabelle

5.5.1 HTML Markup

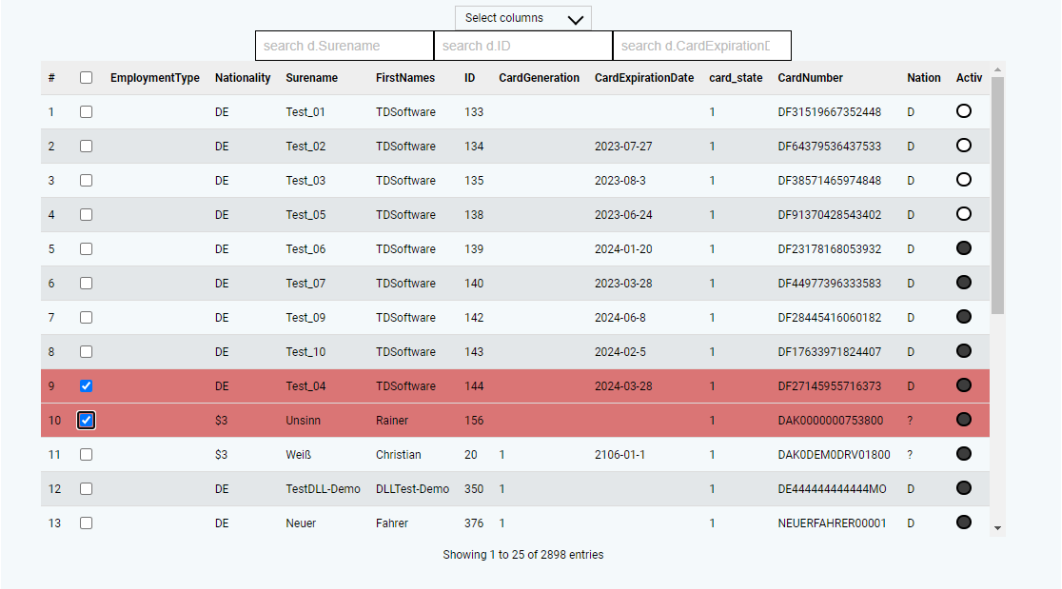
Die Datentabelle besteht aus grundlegendem HTML-Markup. Im Body befindet sich die DW-DataTable, die aus zwei Abschnitten besteht: dem Filter-Tool-Wrapper und dem DataTables-Wrapper. Im Filter-Tool-Wrapper befinden sich verschiedene dynamische Filter-Tools, während sich unter dem DataTables-Wrapper dynamische Buttons, Sucheingaben und die Tabelle selbst befinden. Die Tabelle besteht aus einem dynamischen thead und tbody, und enthält auch ein animiertes Lade-Symbol. Am Ende der Tabelle befindet sich eine dynamische Footer-Toolbar.

Codebeispiel 10: DataTable HTML Markup

```
<div class="main">
    <div class="dwDataTableWrapper">
        <div class="filterToolWrapper">
            <div class="columns"></div>
        </div>
        <div class="dataTablesWrapper">
            <div class="buttons"></div>
            <div class="inputs"></div>
            <div class="dataTables_scroll">
                <table>
                    <thead></thead>
                    <tbody></tbody>
                    <div class="ring"></div>
                </table>
            </div>
            <div class="fg-toolbar"></div>
        </div>
    </div>
</div>
```

5.5.2 CSS

Die Datentabelle hat eine spezifische Höhe und Breite. Gerade und ungerade Zeilen sind durch unterschiedliche Hintergrundfarben voneinander zu unterscheiden. Die Filter-Tools und das Table-Header sind stets sichtbar, auch wenn die Tabelle gescrollt wird.



#		EmploymentType	Nationality	Surename	FirstNames	ID	CardGeneration	CardExpirationDate	card_state	CardNumber	Nation	Activ
1	☐		DE	Test_01	TDSoftware	133			1	DF31519667352448	D	○
2	☐		DE	Test_02	TDSoftware	134		2023-07-27	1	DF64379536437533	D	○
3	☐		DE	Test_03	TDSoftware	135		2023-08-3	1	DF38571465974848	D	○
4	☐		DE	Test_05	TDSoftware	138		2023-06-24	1	DF91370428543402	D	○
5	☐		DE	Test_06	TDSoftware	139		2024-01-20	1	DF23178168053932	D	●
6	☐		DE	Test_07	TDSoftware	140		2023-03-28	1	DF44977396333583	D	●
7	☐		DE	Test_09	TDSoftware	142		2024-06-8	1	DF28445416060182	D	●
8	☐		DE	Test_10	TDSoftware	143		2024-02-5	1	DF17633971824407	D	●
9	☑		DE	Test_04	TDSoftware	144		2024-03-28	1	DF27145955716373	D	●
10	☑		$3	Unsinn	Rainer	156			1	DAK0000000753800	?	●
11	☐		$3	Weiß	Christian	20	1	2106-01-1	1	DAK0DEM0DRV01800	?	●
12	☐		DE	TestDLL-Demo	DLLTest-Demo	350	1		1	DE444444444444MO	D	●
13	☐		DE	Neuer	Fahrer	376	1		1	NEUERFAHRER00001	D	●

Abbildung 12: DataTable Desktop Layout

Beim Scrollen wird kurzzeitig ein Ring ausgeblendet, um den Benutzern anzuzeigen, dass die Datentabelle aktualisiert wurde. Dieser Ring wurde mittels SCSS selbst erstellt und animiert.

Abbildung 13: Loading Icon

Die Checkbox zur Mehrfachauswahl wird dynamisch dargestellt und ermöglicht das Ein- und Ausblenden von Spalten in den Filteroptionen.

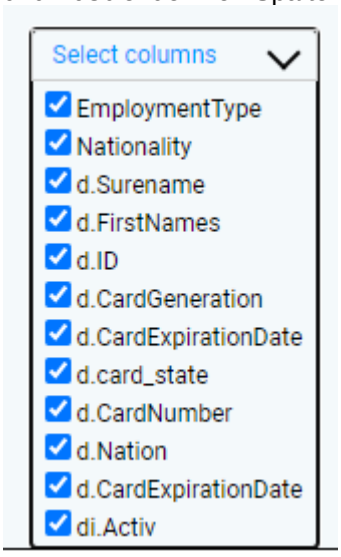


Abbildung 14: Select columns

5.5.3 TS

Zunächst wurden die HTML- und SCSS-Dateien importiert. Anschließend wurde ein leeres Template erstellt, das mit dem Basis-HTML überschrieben wurde. Daraufhin wurde eine Klasse namens "DWDatatable" erstellt, die von der Klasse "HTMLElement" erbt. In dieser Klasse wurden verschiedene Variablen definiert.

Im Konstruktor wurde das HTML-Template und das SCSS-Styling zum ShadowRoot hinzugefügt.

Codebeispiel 11: DataTable constructor

```
    constructor() {
        super();
        this.attachShadow({ mode: "open" });

        // render HTML template
                    (this.shadowRoot as
       ShadowRoot).appendChild(template.content.cloneNode(true));
        // render CSS styling
        this.sheet = new CSSStyleSheet();
        this.sheet.replaceSync(css);
        (this.shadowRoot as ShadowRoot).adoptedStyleSheets = [this.sheet];
        // bind this in Events
        this.sort = this.sort.bind(this);
        this.search = this.search.bind(this);
    }
```

Anschließend wird automatisch die Methode attributeChangedCallback aufgerufen. Hier wird überprüft, welche Attribute definiert wurden, und diese werden in das passende Datenformat umgewandelt. Das "src"-Attribut wird beispielsweise in einer String-Variablen gespeichert, während "columns" und "livefilter" in zwei separate Arrays überführt werden.

Codebeispiel 12: DataTable attributeChangedCallback

```
    static get observedAttributes() {
        return ["src", "columns", "livefilter"];
    }
                    attributeChangedCallback(_name: string, _oldValue:
       string, _newValue:  string) {
        if (_name === "src") {
            this.src = _newValue;
        }
        if (_name === "columns") {
            this.columns = _newValue.split(",").map(i => i.trim());
        }
        if (_name === "livefilter") {
            this.livefilter = _newValue.split(",").map(i => i.trim());
        }
    }
```

Nach dem Aufruf des Konstruktors wird die Methode connectedCallback() ausgeführt. Dabei wird zunächst die Datenerfassung durchgeführt und das Ergebnis in einem Objekt gespeichert. Anschließend werden die Sucheingabefelder erstellt, die vom Live-Filter abhängig sind, und im shadowRoot gerendert. Für jedes Eingabefeld wird ein Event-Listener für das "search"-Ereignis hinzugefügt. Das Mehrfachauswahl-Kontrollkästchen wird erstellt und mit den Spaltennamen ausgefüllt. Für jede Checkbox wird auch ein Event-Listener hinzugefügt, um die Anzeige der Spalten ein- oder auszublenden. Dann wird dem Datentabelle ein Scroll-Ereignis hinzugefügt sowie ein Event-Listener zum Markieren von Zeilen. Die thead und tbody werden dynamisch basierend auf den Daten und Spalten gerendert. Jede Zeile wird nummeriert und mit einem Kontrollkästchen versehen.

Das tbody wird durch eine Schleife ausgefüllt, wobei nur 25 Einträge gerendert werden. Sobald das Ende erreicht ist, werden weitere 25 Datensätze gerendert.

5.6 Methoden & Funktionen

Hier werden lediglich bedeutende Methoden dargestellt.

5.6.1 fetchData()

Die Methode fetchData ist eine asynchrone Funktion, die eine HTTP-Anfrage an einen bestimmten Endpunkt im Netzwerk sendet und eine Antwort zurückgibt. Die Funktion erwartet einen Parameter src, der die URL des Endpunkts als String enthält.

Innerhalb der Funktion wird eine try-catch-Struktur verwendet, um Fehler beim Aufruf von fetch und beim Parsen der Antwort als JSON zu behandeln. Wenn der HTTP-Antwortcode des Endpunkts 200 OK lautet, wird die Antwort als JSON interpretiert und zurückgegeben.

Wenn der HTTP-Antwortcode des Endpunkts jedoch nicht 200 OK ist, wird eine Fehlermeldung generiert und als Error-Objekt ausgelöst.

Wenn ein anderer Fehler auftritt, wird der Fehler in eine lesbare Fehlermeldung umgewandelt und als Error-Objekt ausgelöst.

Codebeispiel 13: fetchData

```
async fetchData(src: string): Promise<unknown> {
    try {
        const response = await fetch(src);

        if (response.ok) {
            const data = await response.json();
            return data;
        }

        const errorMessage = `An error has occurred: ${response.status}`;
        throw new Error(errorMessage);
    } catch (error) {
        const errorMessage = `Fetch error: ${error.message}`;
        throw new Error(errorMessage);
    }
}
```

5.6.2 checkDate()

Die Methode checkDate() überprüft, ob ein übergebenes String-Objekt ein gültiges Datumsformat entspricht. Dazu verwendet es einen regulären Ausdruck, der das Datumsformat YYYY-MM-DDTHH:mm:ss oder YYYY-MM-DDTHH:mm:ss.sss entspricht. Es kann auch ein Zeitzonenoffset angegeben werden, z.B. +01:00 oder -02:30. Die Methode gibt true zurück, wenn das Eingabeobjekt das erwartete Datumsformat erfüllt, oder false, wenn es nicht erfüllt wird.

Codebeispiel 14: checkDate

```
checkDate(input: string) {
    // eslint-disable-next-line no-useless-escape
                const regex = /[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-
    9]{2}:[0-9]{2}(\.[0-9]+)?([Zz]|([\+-])([01]\d|2[0-3]):?([0-5]\d)?)?/i;
    return regex.test(input);
}
```

5.6.3 dateFormat()

Diese Methode nimmt ein Datum als String entgegen und konvertiert es in ein Format mit Tag, Monat und Jahr. Es gibt das Datum als String im Format "Tag.Monat.Jahr" zurück.

Codebeispiel 15: dateFormat

```
dateFormat(date: string) {
    const timestamp = new Date(date).getTime();
    const day = new Date(timestamp).getDate();
    const month = new Date(timestamp).getMonth();
    const year = new Date(timestamp).getFullYear();

    return `${day}.${month}.${year}`;
}
```

5.6.4 filterColumns()

Diese Methode filtert Spalten in einer Tabelle. Es wird ein Event übergeben, welches das Target-Element enthält. Es wird überprüft, ob die Spalte schon vorhanden ist, und diese dann entfernt oder hinzugefügt. Anschließend wird der Header neu gerendert und die Tabelle neu geladen.

Codebeispiel 16: filterColumns

```
filterColumns(e: Event) {
    const column = (<HTMLTextAreaElement>e.target).name;
    const index = this.columns.indexOf(column);
    if (index !== -1) {
        this.columns.splice(index, 1);
    } else {
        this.columns.push(column);
    }
    this.renderHeader(this.columns);
    this.reloadTable(this.newData);
}
```

5.6.5 search()

Diese Methode sucht nach einem bestimmten Eintrag in einer Tabelle anhand des Inhalts eines HTML-Input-Elements. Es wird das Element mit der angegebenen ID ermittelt und dessen Wert für die Suche verwendet. Anschließend werden die Daten anhand des Suchtextes gefiltert. Wenn Ergebnisse gefunden wurden, wird die Tabelle mit diesen Daten neu geladen. Wenn keine Ergebnisse gefunden werden, wird eine Meldung angezeigt.

Codebeispiel 17: search

```
search(event: Event) {
   const thisInput = (event.target as HTMLInputElement).id;
                const searchElement = (this.shadowRoot as
    ShadowRoot).getElementById(`${thisInput}`) as HTMLInputElement;
   const searchText = searchElement.value;
   const tbody = (this.shadowRoot as ShadowRoot).querySelector("tbody");

                this.newData = this.data.filter(v =>
    this.objectByString(v, thisInput)?.toString().includes(searchText));

    if (tbody) {
        if (this.newData.length) {
            this.reloadTable(this.newData);
        } else {
            tbody.innerHTML = "<span>Not Found</span>";
        }
    }
}
```

5.6.6 sort()

Diese Methode sortiert ein unbekanntes Dataset mithilfe einer eingegebenen Spalte (this.sortCol). Der Sortieralgorithmus vergleicht die Werte in den jeweiligen Spalten (this.objectByString) und sortiert sie in absteigender Reihenfolge, wenn this.sortAsc auf

false gesetzt ist, und in aufsteigender Reihenfolge, wenn this.sortAsc auf true gesetzt ist. Nach erfolgreicher Sortierung wird die Tabelle mit den sortierten Daten neu geladen.

Codebeispiel 18: sort

```
    sort(e: { target: { dataset: unknown[]; }; }) {
        const thisSort = e.target.dataset.sort;
        this.sortAsc = this.sortCol === thisSort ? !this.sortAsc : true;
        this.sortCol = thisSort;

        this.newData.sort((a, b) => {
            let comparison = 0;
            if ((this.objectByString(a, this.sortCol) as any) <
(this.objectByString(b, this.sortCol) as any)) {
                comparison = this.sortAsc ? -1 : 1;
            } else if ((this.objectByString(a, this.sortCol) as any) >
(this.objectByString(b, this.sortCol) as any)) {
                comparison = this.sortAsc ? 1 : -1;
            }
            return comparison;
        });

        this.reloadTable(this.newData);
    }
```

5.6.7 objectByString()

objectByString ist eine Methode, die ein Objekt und eine Zeichenfolge als Parameter akzeptiert. Die Zeichenfolge enthält eine Liste von Schlüsseln, die in das Objekt eingehängt werden. Die Methode iteriert durch die Liste der Schlüssel, um den gesuchten Wert im Objekt zu finden. Wenn der angegebene Schlüssel im Objekt vorhanden ist, wird das aktuelle Objekt aktualisiert und die Iteration setzt sich fort, bis der gesuchte Wert gefunden wurde. Wenn der angegebene Schlüssel nicht im Objekt vorhanden ist, wird die Methode abgebrochen und kein Wert zurückgegeben.

Codebeispiel 19: objectByString

```
    objectByString(obj: { [x: string]: any; }, str: string): any {
        const keys = str.split(".");
        let currentObj = obj;

        for (const key of keys) {
            if (key in currentObj) {
                currentObj = currentObj[key];
            } else {
                return;
            }
        }
        return currentObj;
    }
```

5.7 Workflow

1. Recherche anstellen
2. Einen Branch erstellen
3. Aufgaben implementieren
4. Branch hochladen
5. Code-Überprüfung
6. Branch mergen und löschen
7. Tägliche Besprechungen (Dailys)
8. Vorstellung der erfolgten Arbeiten
9. Wöchentliche Meetings (Entwicklerrunden)
10. Ideen und Vorschläge besprechen

5.8 Tools

- Git
- Git Extensions
- Visual Studio Code
- Visual Studio
- Rocket.chat
- Google Chrome

6 Schwierigkeiten

- **Literaturquellen für Web Komponente in TypeScript:** Obwohl es möglicherweise weniger wissenschaftliche Quellen zu Webkomponenten in TypeScript gibt im Vergleich zu JavaScript, bietet die Dokumentation der Webkomponenten viele Informationen und begleitet die Entwickler von Anfang bis zur Fertigstellung einer vollständigen Komponente.

- **Dynamisches Umgang mit allen Faktoren:** Die Datentabelle -Komponente muss dynamisch Spalten, Datensätze und Spaltenfilter basierend auf den Attributen "src", "columns" und "lifefilter" rendern. Wenn Daten gefiltert, sortiert oder gescrollt werden, müssen die gefilterten und sortierten Daten dynamisch gerendert werden. Auch wenn Spalten ein- oder ausgeblendet werden, müssen die gefilterten Daten weiterhin verfügbar sein.

- **Dynamisches Umgang mit verschiedenen verschachtelten Objekten und Datentypen:** Die Herausforderung bei der Implementierung der Datentabelle-Komponente besteht darin, mit verschiedenen verschachtelten Objekten und Datentypen umzugehen. Die Komponente muss in der Lage sein, Daten aus verschiedenen Ebenen zu extrahieren, z. B. mit "objectKey.value" oder "objectKey[value]". Wenn es um tiefere Verschachtelungen geht, wie "objectKey.objectKey.value" oder "objectKey[objectKey][value]", muss die Komponente in der Lage sein, diese Daten zu extrahieren und richtig zu rendern.

- **Filterung und Sortierung:** Eine Datentabelle-Komponente muss auch in der Lage sein, Daten effektiv zu filtern und zu sortieren. Dies kann sehr komplex sein, da die Anforderungen an die Filterung und Sortierung je nach Datenmenge und -typ unterschiedlich sein können.

- **Performance:** Wenn eine Datentabelle große Mengen an Daten oder komplexe Filter- und Sortierfunktionen enthält, kann dies die Performance beeinträchtigen. Es ist wichtig, die Datentabelle-Komponente so zu implementieren, dass sie schnell und effektiv arbeitet und keine unnötige Last auf den Servern oder im Browser erzeugt.

7 Zusammenfassung

Eine angemessene Lösung wurde dank des Prototyps bereitgestellt. Die dw-DataTable erfüllt die Anforderungen vollständig, indem sie Daten dynamisch und mit hoher Leistung und Geschwindigkeit anzeigt, filtert und durchsucht.

Native Web Komponenten, SCSS und TypeScript sind alle gute Werkzeuge, um eine DataTabel-Webkomponente zu erstellen, die sowohl in Bezug auf den Aufwand als auch auf die Leistung optimiert ist.

Native Web Komponenten sind standardisierte Elemente, die in allen modernen Browsern unterstützt werden. Sie sind unkompliziert zu erstellen und können mit jedem Framework oder jeder Bibliothek verwendet werden. Da sie nativ sind, sind sie schnell und ressourcenschonend.

SCSS ist eine erweiterte Version von CSS, die Funktionen wie Variablen, Mixins und Nesting unterstützt. Diese Funktionen machen die Gestaltung der Komponente einfacher und schneller. Außerdem kann SCSS den CSS-Code komprimieren, um die Größe der CSS-Datei zu reduzieren und dadurch die Ladezeit der Webseite zu verkürzen.

Typescript ist eine statisch typisierte Erweiterung von JavaScript. Es bietet mehr Sicherheit und Fehlervermeidung als JavaScript, da es während der Entwicklung typgeprüft wird. Es ist auch einfacher zu warten, da es bessere Werkzeuge für die Codeanalyse und -navigation bietet. Durch die Verwendung von Typescript kann man auch die Leistung der Komponente verbessern, da sie schnellere Ausführungszeiten hat als JavaScript.

Zusammenfassend eignen sich Native Web Komponenten, SCSS und Typescript gut zur Erstellung einer DataTabel-Webkomponente, da sie die Entwicklung vereinfachen und die Leistung der Komponente verbessern. Native Web Komponenten sind schnell und ressourcenschonend, SCSS erleichtert die Gestaltung und verkürzt die Ladezeit, während Typescript für mehr Sicherheit und bessere Leistung sorgt.

Literaturverzeichnis

1 DAKO im Überblick unter: https://www.dako.de/ (aufgerufen am 16.02.2023)
2 DAKO im Überblick unter: https://www.dako.de/unternehmensprofil (aufgerufen am 16.02.2023)
3 Kinsta Einleitung zu Webkomponentenn unter: https://kinsta.com/de/blog/web-komponenten/ (aufgerufen am 17.02.2023)
4 IONOS Web Components unter: https://www.ionos.de/digitalguide/websites/web-entwicklung/web-components/ (aufgerufen am 18.02.2023)
5 mdn web docs Web Components unter: https://developer.mozilla.org/en-US/docs/Web/Web_Components (aufgerufen am 20.02.2023)
6 IONOS Shadow DOM unter: https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-der-shadow-dom/ (aufgerufen am 21.02.2023)
7 mdn web docs Using shadow DOM unter: https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM (aufgerufen am 22.02.2023)
8 Lambdatest Shadow DOM unter: https://www.lambdatest.com/blog/shadow-dom-in-selenium/ (aufgerufen am 21.02.2023)
9 mdn web docs Using templates and slots unter: https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_templates_and_slots (aufgerufen am 22.02.2023)
10 ultimatecourses lifecycle hook unter: https://ultimatecourses.com/blog/lifecycle-hooks-in-web-components (aufgerufen am 22.02.2023)
11 javatpoint Difference between CSS und SCSS unter: https://www.javatpoint.com/css-vs-scss (aufgerufen am 23.02.2023)
12 nextgeneration SCSS / SASS unter: https://nextgeneration.mysign.ch/de/themen/wissen/frontend-entwicklung/scss-sass.html (aufgerufen am 23.02.2023)
13 mdn web docs CSSStyleSheet unter: https://developer.mozilla.org/en-US/docs/Web/API/CSSStyleSheet/CSSStyleSheet#browser_compatibility (aufgerufen am 24.02.2023)
14 web.dev Constructable Stylesheets unter: https://web.dev/constructable-stylesheets/ (aufgerufen am 25.02.2023)
15 TenMedia JavaScript unter: https://www.tenmedia.de/de/glossar/javascript (aufgerufen am 26.02.2023)
16 TS TypeScript unter: https://www.typescriptlang.org/ (aufgerufen am 27.02.2023)
18 Medium TypeScript Compiler unter: https://medium.com/jspoint/typescript-compilation-the-typescript-compiler-4cb15f7244bc (aufgerufen am 28.02.2023)
19 K&C TypeScript vs JavaScript unter: https://kruschecompany.com/de/typescript-vs-javascript/ (aufgerufen am 01.03.2023)

Weitere Quellen:

- Add advanced interactions controls to your HTML tables unter: https://datatables.net (aufgerufen am 16.02.2023)
- Web Components – Das Endev om Javascript Framework Dschungel? Unter: https://innfactory.de/softwareentwicklung/web-components-das-ende-vom-javascript-framework-dschungel (aufgerufen am 17.02.2023)
- TypeScript
  Unter: https://www.typescriptlang.org (aufgerufen am 18.02.2023)
  Unter: https://de.wikipedia.org/wiki/TypeScript (aufgerufen am 18.02.2023)
  Unter: https://unsere-schule.org/programmieren/javascript/typescript (aufgerufen am 18.02.2023)

- Drei Gründe warum TypeScript das bessere JavaScript ist Unter: https://www.communardo.de/blog/drei-gruende-warum-typescript-das-bessere-javascript-ist (aufgerufen am 18.02.2023)
- jQuery Unter: https://www.dev-insider.de/was-ist-jquery-a-782237 (aufgerufen am 19.02.2023)
- Javascript Vs jQuery: What's The Difference? [2022] Unter: https://www.interviewbit.com/blog/javascript-vs-jquery (aufgerufen am 19.02.2023)
- Braucht man wirklich jQuery? Unter: https://www.novacapta.de/details/braucht-man-wirklich-jquery (aufgerufen am 20.02.2023)
- JavaScript
  Unter: https://www.seobility.net/de/wiki/JavaScript (aufgerufen am 20.02.2023)
- Grundlagen von JavaScript-SEO
  Unter: https://developers.google.com/search/docs/crawling-indexing/javascript/javascript-seo-basics?hl=de (aufgerufen am 20.02.2023)
- TypeScript vs JavaScript – the state of the struggle in 2021
  Unter: https://tsh.io/blog/typescript-vs-javascript-comparison (aufgerufen am 23.02.2023)
- Custom date picker with web component
  Unter: https://codepen.io/beforesemicolon/pen/jOMgZrY (aufgerufen am 24.02.2023)
- Web Components Crash Course (Custom Elements)
  Unter: https://javascript.plainenglish.io/web-components-crash-course-7c0df961a8b7 (aufgerufen am 25.02.2023)
- datatable using jquery.datatable in angularjs
  Unter: https://codepen.io/kalaiselvan/pen/RRBzda (aufgerufen am 26.02.2023)
- Building Table Sorting and Pagination in Web Component
  Unter: https://www.raymondcamden.com/2022/05/23/building-table-sorting-and-pagination-in-a-web-component (aufgerufen am 27.02.2023)
- Building Table Sorting and Pagination in Vue.js
  Unter: https://www.raymondcamden.com/2018/02/08/building-table-sorting-and-pagination-in-vuejs (aufgerufen am 27.02.2023)
- Building Table Sorting and Pagination in JavaScript
  Unter: https://www.raymondcamden.com/2022/03/14/building-table-sorting-and-pagination-in-javascript/ (aufgerufen am 28.02.2023)
- Building Table Sorting and Pagination in Alpine.js
  Unter: https://www.raymondcamden.com/2022/05/02/building-table-sorting-and-pagination-in-alpinejs (aufgerufen am 01.03.2023)
- Web Components
  Unter: https://en.wikipedia.org/wiki/Web_Components (aufgerufen am 02.03.2023)

Selbstständigkeitserklärung

Ich, Al-Khodari,Molham, erkläre hiermit, dass ich die vorliegende Bachelorarbeit mit dem Thema

> *Konzeption und Umsetzung einer nativen Webkomponente zur Darstellung großer, dynamischer Datenmengen im Web in tabellarischer Ansicht unter Berücksichtigung von Performance und Aufwand*

selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Erfurt, 14.03.2023