Diplomarbeit

Höhere Technische Bundeslehranstalt Leonding
Abteilung für Medientechnik

Erweiterung des WYSIWYG Texteditors "Trix" für Barrierefreiheit

Eingereicht von: **Halil Bahar, 5AHITM**
Sonja Cao, 5AHITM
Datum: **21. April 2021**
Betreuer: **Prof. Mag. Ing. Dr. Thomas Stütz**
Projektpartner: **Fabasoft Software GmbH**

Declaration of Academic Honesty

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.
This paper has neither been previously submitted to another authority nor has it been published yet.

Leonding, April 21, 2021

Halil Bahar, Sonja Cao

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorgelegte Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Gedanken, die aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.
Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Leonding, am 21. April 2021

Halil Bahar, Sonja Cao

Gender Declaration

In this diploma thesis the language form of the generic masculine is used for reasons of better readability. All personal designations are therefore to be understood as gender-neutral.

Gender Erklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Diplomarbeit die Sprachform des generischen Maskulinums verwendet. Alle personenbezogenen Bezeichnungen sind somit geschlechtsneutral zu verstehen.

Abstract

In the course of the diploma thesis of Halil Bahar and Sonja Cao an extension for the open source What You See Is What You Get (WYSIWYG) text editor Trix from the company Basecamp is being developed for the client Fabasoft Software GmbH. The previous text editor of the Fabasoft Cloud no longer meets the desired requirements and Trix is still unsuitable in terms of accessibility for people with disabilities. The extension is therefore intended to be used in products of Fabasoft and to meet the criteria of WCAG 2.1, making it accessible with screen readers and the keyboard to enable all users to write and format texts without barriers.

Zusammenfassung

Im Zuge der Diplomarbeit von Halil Bahar und Sonja Cao wird der Open Source What You See Is What You Get (WYSIWYG) Texteditor namens Trix vom Unternehmen Basecamp für den Auftraggeber Fabasoft Software GmbH erweitert. Der bisherige Texteditor der Fabasoft Cloud entspricht nicht mehr den gewünschten Anforderungen und Trix ist hinsichtlich der Accessibility für Menschen mit Beeinträchtigungen noch ungeeignet. Die Erweiterung soll einen Einsatz in Fabasoft Produkten ermöglichen sowie die Kriterien der WCAG 2.1 erfüllen und so mit Screenreadern sowie Tastatur zugänglich sein, um allen Benutzern die Möglichkeit zu geben, Texte barrierefrei zu verfassen und zu formatieren.

Danksagung

Ein besonderer Dank geht an die Firma Fabasoft, die die Diplomarbeit mit der nötigen Infrastruktur unterstützt hat und immer offen für neue Ideen gewesen ist. Insbesondere die Ratschläge und der Austausch von Ideen haben dabei geholfen die beste Lösung für das Unternehmen und das Projektteam zu finden.
Ein großer Dank gilt vor allem Thomas Bühringer, der uns als Projektleiter mit seiner kritischen Stellungnahme und seinen Feedbacks zur Seite gestanden ist.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Ausgangssituation

Die Fa. Fabasoft ist ein Softwareunternehmen im Zentralraum Oberösterreichs. Zum Geschäftsmodell zählen die Entwicklung und der Vertrieb von Softwareprodukten, die vor allem für die Bereiche Enterprise Search, Wissensmanagement, digitale Geschäftsprozesse und unternehmensübergreifende Zusammenarbeit in der Cloud konzipiert sind. Kunden sind private und öffentliche Auftraggeber, insbesondere des E-Governments, und Wirtschaftsunternehmen, die hohe Sicherheitsanforderungen stellen und hauptsächlich im deutschsprachigen Raum beheimatet sind. Ein Grundkonzept der Fabasoft Cloud ist Barrierefreiheit. Dieses Produkt wurde im Oktober 2019 als erste Web-Applikation von der Österreichischen Computer Gesellschaft mit dem WACA-Zertifikat (Web Accessibility Certificate Austria, vgl. [1], 18.08.2020) in der Stufe Silber ausgezeichnet.

1.2 Ist-Zustand

Ein Grundkonzept der Fabasoft Cloud ist Barrierefreiheit. Ziel ist es, dass die Benutzeroberfläche von allen Menschen, unabhängig ihrer Behinderungen, bedient werden kann. Oftmals wird in Softwareprodukten der Fa. Fabasoft die Eingabe von einfachen, formatierten Texten gefordert, die in einem WYSIWYG (What You See Is What You Get) Rich Text Editor erfasst werden. Übersetzt bedeutet "What You See Is What You Get" so viel wie "Was du siehst, ist das, was du bekommst". Der Text in einem WYSIWYG Editor wird bei der Ausgabe des Dokuments genau so angezeigt, wie er bei seiner Bearbeitung aussieht.
Ein Beispiel für solch einen Editor ist "Trix". Er ist ein Open Source WYSIWYG Texteditor vom Unternehmen Basecamp, den Machern von Ruby on Rails, und ist von den beiden Entwicklern Sam Stephenson und Javan Makhmali der Community auf GitHub (vgl. Makhmali und Stephenson 2013, [2], 09.08.2020) zugänglich gemacht worden. Mithilfe dieses Editors können Nachrichten, Kommentare, Artikel, Listen und vieles mehr verfasst werden. Es besitzt ein selbst entwickeltes Document Object Model (DOM), das

neben den gängigen JavaScript Events ebenso auf selbstdefinierte Events hört und eingebettete Anhänge unterstützt.

1.2.1 Systemarchitektur

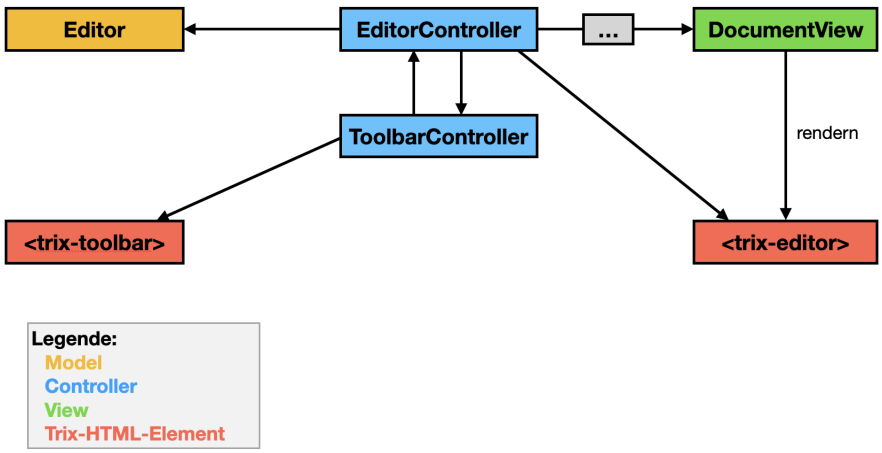

Abbildung 1.1: Vereinfachte Darstellung der Kommunikation zwischen den Klassen

Zur besseren Verständlichkeit wurden nur die Klassen dargestellt, die oberflächlich erkennbar sind. Die Funktionalitäten von Trix unterscheiden sich in ihrer Komplexität von denen der Konkurrenz. Außerdem verwendet er den Standard der Web Components (siehe Kapitel 3.2). Sobald die Trix Library eingebunden ist, werden ihre benutzerdefinierten HTML-Elemente (`<trix-editor>`, `<trix-toolbar>`) registriert. Beim Verwenden eines Trix Editors werden dementsprechend der Controller und in diesem alle weiteren wichtigen Klassen instanziiert, die nicht unbedingt in der Darstellung ersichtlich sind.

EditorController
Dieser Controller ist eine der Klassen, die umfangreichere Funktionalitäten beinhalten. Er kümmert sich um

- `InputEvents` bei jeder Tastatureingabe im Editor. Hier wird beispielsweise darauf geachtet, dass beim Einfügen eines Textes dessen Formatierung weiterhin bestehen bleibt.
- Formatierungen bei Bedarf und das Aktivieren und Deaktivieren dieser, was aufgrund der Kommunikation zwischen den Controllern auch über die Toolbar möglich ist.
- das Rendering, wobei er es im `DocumentView` auslöst.

ToolbarController
Wie auch im Absatz davor erwähnt, kümmert sich dieser Controller um das Aktivieren und Deaktivieren von Formatierungen über die Buttons in der Toolbar. Er hat einen eigenen `EventListener` auf sich selbst (`<trix-toolbar>`). Die Besonderheit besteht darin, dass beliebig viele Buttons verwendet werden können. Der Grund dafür ist, dass das Event das sogenannte "Bubbling" verwendet. Dies kann man sich wortwörtlich wie eine Blase unter Wasser vorstellen. Wenn ein HTML-Element ein Event besitzt, wird dieses Event auch von HTML-Elementen darunter (Child Elements) ausgelöst. Diese Blase steigt in der Struktur auf und löst das Event aus.
Die Buttons haben erst dann eine Funktion, wenn ein selbst definiertes HTML-Attribut hinzugefügt wird. Sobald das Event ausgelöst wird, überprüft der Controller, ob das Attribut vorhanden ist. Wenn das der Fall ist, handelt er anhand des Wertes und formatiert (kursiv) oder löst eine Funktion (rückgängig) aus.

DocumentView
Diese View ist für das gesamte Rendering im Editor zuständig. Das beinhaltet alle Textformatierungen und Textpositionierungen, die auch dementsprechend eine eigene View besitzen, wie etwa TextView für Texte oder AttachmentView für Bilder und Dokumente. Bei jedem Aufruf des Rederings wird das alte Dokument durch ein neues ersetzt.

1.3 Motivation

Der bisher integrierte Browser-UI-Control in Fabasoft Softwareprodukten weist einige Nachteile auf. Der sogenannte "CKEditor" in der Version 4 verliert seinen Long Term Support (LTS) im Jahr 2023 (vgl. CKSource, [3], 06.09.2020). Dieser wird von der Version 5 abgelöst. Da die neue Version 5 eine neue Lizenz mit sich bringt (vgl. CKSource, [4], 06.09.2020), ist dieser Texteditor in den Produkten nicht mehr weiterhin verwendbar. Der Grund dafür ist, dass die neue Lizenz unter den Bedingungen der GNU General Public License Version 2 or later (GPLv2) steht und der Sourcecode auf Nachfrage freigegeben werden muss. Dies entspricht nicht der Philosophie der Firma. Darum wird der Editor durch ein alternatives Produkt abgelöst. Durch eine erste Analyse hat sich herausgestellt, dass sich der Texteditor *Trix* als Ersatz eignet. Allerdings ist dieser nicht barrierefrei.

1.3.1 Gleichstellung von Menschen mit Behinderung

In der Gesellschaft und im Alltag machen Menschen mit Behinderung oftmals mit kleineren oder größeren Benachteiligungen Bekanntschaft. Global haben etwa 15 Prozent der Weltbevölkerung eine Art von Beeinträchtigung, wobei die rund eine Milliarden Kinder und Erwachsenen assistive Technologien, wie beispielsweise Rollstühle oder Hörgeräte, benötigen. International hat jedes Land eine eigene Definition für den Begriff, weshalb die WHO (World Health Organisation 2011, vgl. [5]) diesen nur grob beschreiben kann. Sie geht dabei immer von drei Punkten aus:

- **Impairment (dt. Schädigung)** bezeichnet jede Anomalie oder jeden Verlust der anatomischen, psychischen oder physiologischen Funktionen und Strukturen des menschlichen Körpers.
- **Disability (dt. Beeinträchtigung)** bezeichnet jeden Mangel oder jede Einschränkung der Fähigkeiten infolge einer Beeinträchtigung, wodurch eine von der Gesellschaft als normal oder in diesem Bereich betrachtete Tätigkeit nicht ausgeübt werden kann.
- **Handicap (dt. Behinderung)** bezeichnet den Nachteil einer Person aufgrund einer Schädigung oder Beeinträchtigung. Eine Behinderung steht im Zusammenhang mit einem Problem mit einer Struktur oder eines Organes des Körpers.

"In fact we have a moral duty to remove the barriers to participation, and to invest sufficient funding and expertise to unlock the vast potential of people with disabilities. Governments throughout the world can no longer overlook the hundreds of millions of people with disabilities who are denied access to health, rehabilitation, support, education and employment, and never get the chance to shine." (Hawking 2011, vgl. [6])

Grundsätzlich wird eine Behinderung in Österreich als dauerhafte Auswirkung einer körperlichen, geistigen oder psychischen Beeinträchtigung oder Beeinträchtigung der Sinnesfunktionen bezeichnet. Diese überschreitet einen Zeitraum von sechs Monaten und erschwert die Teilhabe in der Gesellschaft. Damit die Benachteiligung von Behinderten möglichst flächendeckend beseitigt wird, können diesbezügliche Gesetze auf der Webseite des Rechtsinformationssystems des Bundes (RIS) abgerufen werden:

- Das Bundesgesetz über die Gleichstellung von Menschen mit Behinderungen, auch als das Bundes-Behindertengleichstellungsgesetz (BGStG) bezeichnet, zur Regelung der Diskriminierung im täglichen Leben (vgl. RIS, [7], 30.08.2020).
- Das Behinderteneinstellungsgesetz (BEinstG) zur Regelung der Diskriminierung in der Arbeitswelt (vgl. RIS, [8], 30.08.2020).
- Das Bundesgesetz vom 17. Mai 1990 über die Beratung, Betreuung und besondere Hilfe für behinderte Menschen, bekannt als das Bundesbehindertengesetz (BBG) zur Regelung der Aufgaben und Befugnisse des Behindertenanwalts (vgl. RIS, [9], 30.08.2020).

Die Fassung des BGStG besagt, dass das Ziel die Beseitigung der Diskriminierung von Menschen mit Behinderungen sei, um diese gleichberechtigt am Leben in der Gesellschaft teilhaben zu lassen und eine selbstbestimmte Lebensführung zu gewährleisten. Der Diskriminierungsschutz gilt somit für diejenigen, die körperlich, geistig, psychisch behindert oder sinnesbehindert sind, ebenso für ihre Angehörigen. Es ist nicht erlaubt jemanden unmittelbar oder mittelbar zu diskriminieren.

Definition: Diskriminierung
Wird eine Person aufgrund ihrer Behinderung in bestimmten Situationen oder gegenüber anderen anders behandelt, so liegt eine unmittelbare Diskriminierung vor. Eine mittelbare Diskriminierung hingegen liegt genau dann vor, wenn neutrale Vorschriften, Kriterien, Verfahren oder Merkmale gestalteter Lebensbereiche den Anschein erwecken, Menschen mit Behinderungen gegenüber anderen auf jegliche Art und Weise zu benachteiligen.
Zu diesen Benachteiligungen gehören:

- Barrieren
- eine weniger günstige Behandlung
- Diskriminierungsaufforderungen
- und Belästigung aufgrund einer Behinderung

Arten von Behinderungen
Grundsätzlich lassen sich Behinderungen in sechs Kategorien einteilen:

- **Körperliche Behinderungen** kommen am häufigsten vor und nehmen nicht selten mit dem Alter zu. Bezeichnet werden damit starke physische Einschränkungen, die auf eine Dysfunktion oder Schädigung der Stütz- und Bewegungsorgane zurückgeführt werden.
- **Geistige Behinderungen** sind fortwährende, deutlich über dem Durchschnitt liegende Einschränkungen der kognitiven Fähigkeiten und kommen am zweithäufigsten vor. Zu diesen kognitiven Fähigkeiten zählen die Wahrnehmung, die Aufmerksamkeit, das Denken und das Lernen ebenso wie die Erinnerung, die Motivation und die Konzentration.
- **Sinnesbehinderungen** umfassen alle Seh- sowie Hörbehinderungen wie Fehlsichtigkeit, Blindheit, Schwerhörigkeit, Gehörlosigkeit und Taubblindheit.
- **Sprachbehinderungen** oder anders formuliert Störungen des Redeflusses, des Spracherwerbs, des Sprechens und der Stimme bezeichnet all jene Menschen, die nicht fähig sind ihre Muttersprache im entsprechenden Alter schriftlich oder mündlich zu beherrschen.
- **Psychische Behinderungen** umfassen die Beeinflussung des Denkens, Fühlens und Handelns von Menschen. Das heißt wiederum, dass es Abgrenzungen zwischen Verhalten und Erleben gibt.

1.4 Barrierefreiheit im Web

1.4.1 Definition: Web-Zugänglichkeit

Web-Zugänglichkeit, oder auch Web Accessibility, bezeichnet, dass alle Menschen auf dem bestmöglichen Weg einen Zugang zu Information oder Dienstleistung im Inter-

net haben. Ziel ist es, Websites und jegliche mobile Anwendungen für alle Nutzerinnen und Nutzer, vor allem auch diejenigen mit Behinderungen, besser zugänglich und somit barrierefrei zu gestalten. In der heutigen Gesellschaft ist das Internet mit all seinen verfügbaren Leistungen kaum noch wegzudenken. Zahlreiche Dienstleistungen wie etwa Onlinebanking, elektronische Dienste von Instituten und Institutionen, Onlineshop und viele mehr gehören dazu.
Das Web-Zugänglichkeits-Gesetz (WZG) umfasst alle rechtlichen Grundlagen zur Erreichung dieses Ziels (vgl. RIS, [10], 30.08.2020). Insbesondere der Bund und alle mit ihm in Zusammenhang stehende Einrichtungen sind dazu verpflichtet barrierefreie Dienstleistungen bereitzustellen. Hierzu gibt es bestimmte Anforderungen:

- Wahrnehmbarkeit
- Bedienbarkeit
- Verständlichkeit
- Robustheit

Des Weiteren wurde ein zeitlicher Anwendungsbereich festgelegt, in dem alle Websites und mobile Anwendungen des Bundes barrierefrei sein müssen:

- Neue Websites nach dem 23. September 2018 ab dem 23. September 2019
- Alte Websites vor dem 23. September 2018 ab dem 23. September 2020
- Alle mobilen Anwendungen ab dem 23. Juni 2021

Damit diese Dienstleistungen fortlaufend zugänglich sind, müssen die oben genannten Anforderungen regelmäßig überprüft werden. Grundsätzlich hat jede Nutzerin und jeder Nutzer das Recht dazu, sich über Mängel bei Nichteinhaltung der Barrierefreiheit zu beschweren, die dann, wenn sie berechtigt sind, beseitigt werden müssen.

1.5 Aufgabenstellung

Im Rahmen dieser Diplomarbeit wurde eine Erweiterung für den Texteditor *Trix* geschrieben, sodass dieser den Ansprüchen für die Produkte der Fa. Fabasoft gerecht wird. Insbesondere wurden die Kriterien des WCAG 2.1 (Web Content Accessibility Guidelines, siehe Kapitel 3.1) erfüllt sowie eine Bedienung mit Screenreadern und vollständig mit Tastatur ermöglicht worden.

1.6 Zielsetzung

Ziel dieser Arbeit war es, den Mitarbeiten und Kunden von der Fa. Fabasoft die Verwendung des Systems weiterhin zu ermöglichen. Der Texteditor sollte weitestgehend von allen Personen einfach und ohne Probleme bedient werden können.

1.7 Sollzustand

1.7.1 Erweiterte Systemarchitektur

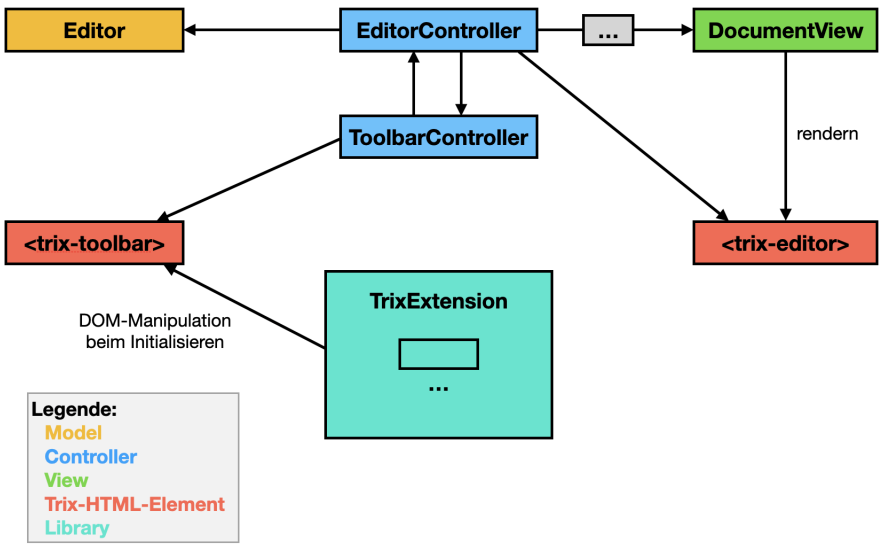

Abbildung 1.2: Vereinfachte Darstellung der Kommunikation zwischen den Klassen mit der Erweiterung

Mit der Erweiterung "Trix Extension" ist es möglich, die Toolbar in wenigen Codezeilen robust und sicher zu ersetzen. Hierbei wird der gesamte Inhalt von dem `<trix-toolbar>` überschrieben. Die Toolbar bestand zuvor aus HTML-Elementen, die statisch in einer JavaScript-Datei definiert waren. Die Struktur der neu ersetzten Toolbar bleibt dennoch dieselbe für Backwards Compatibility. Zusätzlich zu der neuen Anordnungs- und Designmöglichkeit der Buttons werden ARIA-Rollen (Accessible Rich Internet Applications, auch WAI-ARIA) zugeordnet. Des Weiteren werden `KeyEvents` an `<trix-editor>` und `<trix-toolbar>` gehängt. Das Event für den Editor ist sehr simpel, denn er hört auf nur eine Tastenkombination, die es ermöglicht den Fokus auf die Toolbar zu legen. Bei der Toolbar reagiert das Event nur bei Tasten, wie

- Pfeiltasten zum Navigieren zwischen den Buttons,

- Pfeiltasten mit Steuerung-Taste zum Navigieren zwischen den Button-Gruppen,
- Escape-Taste zum Verlassen der Toolbar und zurück Fokussieren des Editors,
- Pos1- und Ende-Taste zum Navigieren zum jeweiligen ersten und letzten Button,
- und Tab-Taste ignorieren, damit die Nutzer nicht verwirrt werden und die Orientierung verlieren.

1.7.2 Funktionale Anforderungen

Für diese Diplomarbeit sind keine funktionalen Anforderungen definiert. Der Grund ist der, dass die Autoren der Meinung sind, dass es keine neuen funktionalen Anforderungen gibt, weshalb die Arbeit schwerwiegend auf nichtfunktionalen Anforderungen basiert.

1.7.3 Nichtfunktionale Anforderungen

- Aufsetzen eines npm-basierten Build-Prozesses mit allen notwendigen Artefakten in das Fabasoft npm-Repository.
- Aufbau einer Unit-Test-Infrastruktur mit einer Line-Coverage von mindestens 70 Prozent.
- Einpflegen der entsprechenden Änderungen zur Erfüllung der Kriterien des WCAG 2.1.
- Vollständige Bedienung über eine Tastatur und einen Screenreader.
- Einpflegen eines High-Level-APIs, um die Instanzierung und Parametrierung aus dem Fabasoft UI möglichst simpel zu gestalten.

1.8 Projektablauf und Produkt

1.8.1 Projektablauf

Im ersten Schritt ist das GitHub Repository des Texteditors Trix geforkt worden, um die notwendigen Änderungen zur Erreichung der Barrierefreiheit einzupflegen. Allerdings sind die Entwickler, die auch die gleichzeitigen Betreuer des Editors sind, bezüglich der Issues und der Pull Requests, sehr inaktiv. Daraufhin wurde mit dem Team der Fabasoft Cloud entschieden, dass eine Erweiterung für den Texteditor sinnvoller ist.

Im nächsten Schritt wurde der korrekte Export der bestehenden formatierten Textinhalte in der Fabasoft Cloud ermöglicht. Damit keine zusätzlichen Zeilenumbrüche hinzugefügt und gewisse HTML-Elemente unterstützt und nicht übersprungen werden, sind eine Reihe von Funktionalitäten in Trix selbst und mithilfe von JavaScript Prototypes überschrieben worden. Bei Bedarf können diese Überschreibungen ignoriert werden.

Während der Entwicklungsphase dieser Diplomarbeit redigierte ein blinder Mitarbeiter regelmäßig die Funktionen des Texteditors. Zusätzlich zu den Verbesserungsvorschlägen teilte er in seinen Feedbacks mit, welche Kriterien des WCAG 2.1 gut umgesetzt wurden und welche noch erforderlich waren.

1.8.2 Produkt

Mithilfe der Erweiterung ist es möglich, den Texteditor über die Tastatur zu bedienen und somit visuelle Elemente, wie etwa Buttons zum Formatieren des Textes, zu erreichen.

Kapitel 2

Ausgewählte Technologien

2.1 Git und GitLab

2.1.1 Versionskontrollsystem

Oftmals arbeitet ein Entwickler oder ein Entwicklerteam an einem Softwareprojekt, um entweder Fehler zu beheben oder neue Funktionalitäten hinzuzufügen. Hierbei werden Änderungen am Quellcode durchgeführt, die regelmäßig gesichert werden müssen. Diese Änderungen können in manchen Fällen auch dazu führen, dass etwas anderes im Code nicht mehr so funktioniert wie es sollte. Dann werden weitere Änderungen gemacht, um wieder einen lauffähigen Stand herzustellen, jedoch kann es passieren, dass plötzlich gar nichts mehr funktioniert.

Abhilfe für dieses Problem bietet ein Versionskontrollsystem, auch als VCS (Version Control System) bezeichnet, das Änderungen in der Projektentwicklung festhält (Dirk 2019, vgl. [11], 15.11.2020). Dadurch ist es möglich zu einem späteren beliebigen Zeitpunkt auf ältere Versionen des Systems zurückzugreifen, den aktuellen Code mit früheren Versionen zu vergleichen oder Bugfixes zu implementieren. Arbeiten mehrere Entwickler an denselben Dateien im Quellcode, so kann mitverfolgt werden, welche Person welche Änderungen gemacht hat. Ein Projekt, das mit einem VCS verwaltet wird, wird Repository genannt. Ein Repository kann als eine Art Datenbank mit allen Änderungen betrachtet werden, die die History repräsentieren. Diese Änderungen sind ein markierter Stand, ein sogenannter Commit, in der Projektentwicklung und werden als Working Copy gespeichert, die einer Kopie des gesamten Projektes entspricht. Ein Commit kann als Schnappschuss oder als kleiner Meilenstein betrachtet werden. Er beinhaltet neben der Working Copy einen Zeitstempel und eine Nachricht, die aussagt, was sich seit dem letzten Commit geändert hat. Die Entwicklungsarbeiten können dabei in mehreren Entwicklungszweigen (engl. Branches) erfolgen. Eine Abspaltung eines Projektes wird hingegen als Fork bezeichnet.

Es gibt drei Arten von Version Control Systems:

- **Lokal:** Lokale Versionskontrollsysteme funktionieren nur auf einem Rechner und versionieren eine einzige Datei. Insbesondere in Büroanwendungen kommen Tools wie Revision Control System (RCS) oder Source Code Control System (SCCS) zum Einsatz. Das Dokument speichert dabei jede Version in seiner eigenen Datei.

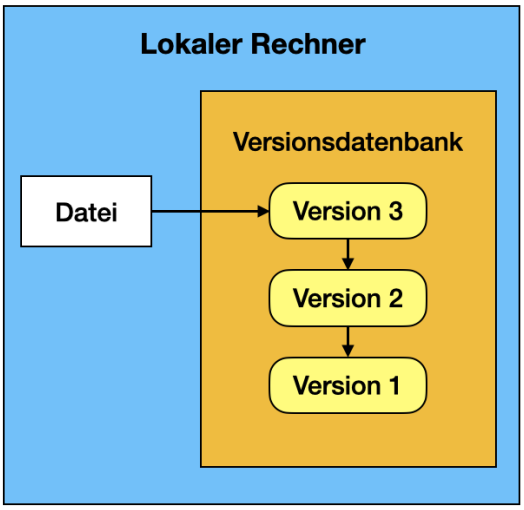

Abbildung 2.1: Lokales Versionskontrollsystem

- **Zentral:** Bei einem zentralisierten Versionskontrollsystem gibt es ein Respository, das sich mehrere Entwickler in einem Netzwerk miteinander teilen. Es handelt sich hierbei um ein Client-Server-System, das von zahlreichen kommerziellen Anbietern verwendet wird. Durch das Concurrent Versions System (CVS) wurde dieses Konzept berühmt, aber durch Subversion (SVN) neu implementiert.

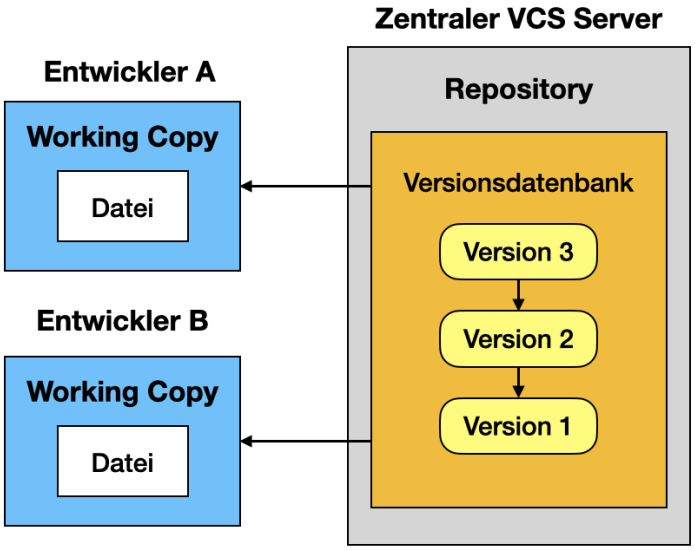

Abbildung 2.2: Zentralisiertes Versionskontrollsystem

- **Verteiltes VCS:** Im Gegensatz zum zentralisierten VCS existiert beim verteilten Versionskontrollsystem kein zentrales Repository. Jeder am Projekt tätige Entwickler verfügt über ein eigenes Repository, das er mit anderen Repositories abgleichen kann. Auch hier ist die History klar ersichtlich. Der einzige Unterschied zu den beiden anderen Versionskontrollsystemen ist der, dass Änderungen hier am lokalen Rechner erfolgen können. Eine Verbindung mit dem Server ist nicht notwendig.

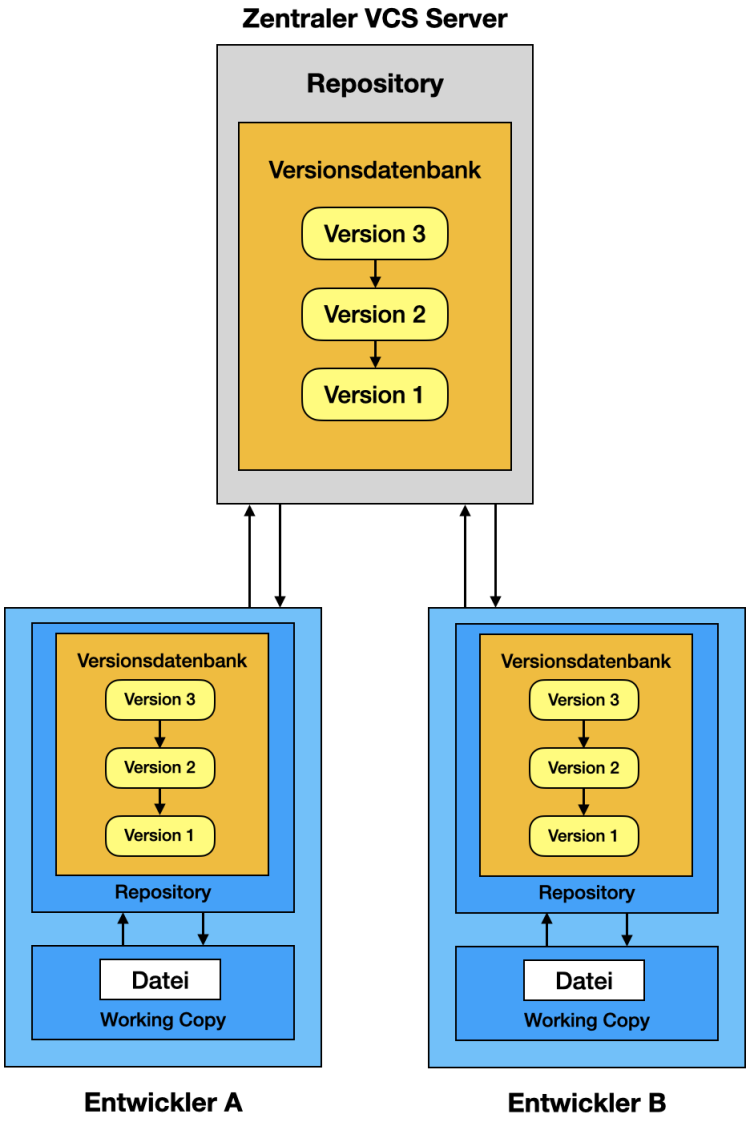

Abbildung 2.3: Verteiltes Versionskontrollsystem

2.1.2 Git

Git ist ein Open-Source-Tool, das von Linus Torvalds, dem Entwickler des Betriebssystems Linux, entwickelt wurde und stammt aus dem Jahr 2005 (Atlassian, vgl. [12], 15.11.2020). Heutzutage zählt es zu eines der weitverbreitesten Versionsverwaltungssysteme weltweit. Die Softwareentwickler kommen sowohl aus dem kommerziellen als auch aus dem öffentlichen Bereich. Aufgrund seiner Architektur ist Git ein Distributed VCS (DVCS, dt. verteiltes VCS) und funktioniert in zahlreichen Plattformen und Entwick-

lungsumgebungen. Das bedeutet, dass auf die gesamte History mit allen Entwicklungsarbeiten von jedem Standort aus zugegriffen werden kann. Neben seinem verteilten System ist Git unter anderem auf Performance, Sicherheit und Flexibilität fokussiert.

Snapshots
Bei den meisten VCS werden Informationen als eine Liste mit den Änderungen innerhalb der Dateien gespeichert. Das bedeutet, dass bei jeder Version nur die Dateien festgehalten werden, deren Inhalt sich während den Entwicklungsarbeiten geändert haben.

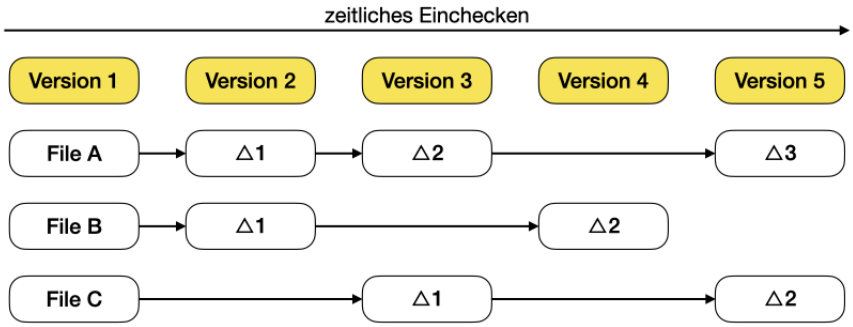

Abbildung 2.4: Speichern von Daten bei anderen VCS

Im Gegensatz dazu werden die Daten als eine Art Folge von Schnappschüssen (engl. snapshots) betrachtet. Nach jedem Commit wird der aktuelle Stand des Repositories gespeichert. Dieser beinhaltet die gesamte Kopie des Projekts zu diesem Zeitpunkt und erhält eine Referenz zu diesem Schnappschuss in Form eines einzigartigen Hashwertes. Wenn das Entwicklerteam weiter am Projekt arbeitet und die Änderungen committet, erhalten die neuen und geänderten Dateien eine neuen Commit-Hash, während die Dateien, die sich nicht geändert haben, nur auf dieselbe Datei im vorherigen Schnappschuss verlinkt werden.

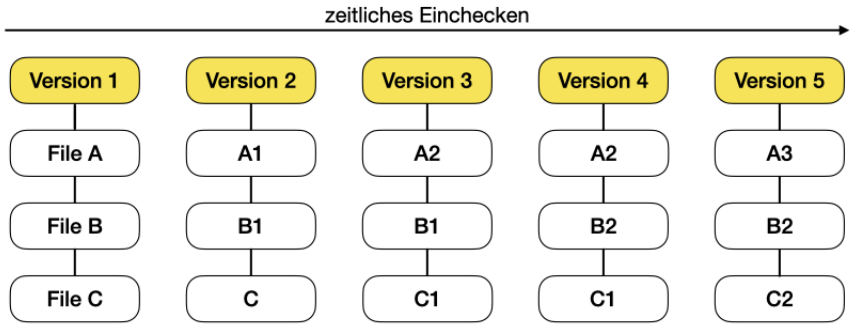

Abbildung 2.5: Speichern von Daten bei Git

Um eine Sicherheit garantieren zu können, werden die Dateiinhalte sowie ihre Beziehungen zu anderen Dateien, Verzeichnissen, Versionen, Commits und Tags mit dem Hashing-Algorithmus SHA1 gesichert. Dadurch kann der Quellcode des Entwicklers vor ungewollten Änderungen geschützt und die Historie vollständig mitverfolgt werden. In der Datenbank werden keine Dateinamen sondern nur Hashwerte gespeichert. Ein Beispiel für so einen Wert ist `fe215ed7198155ca796fbb8ef81683137c210492`.

Workflow
In einem Git-Projekt sind die drei wichtigsten Stufen das Working Directory, die Staging Area und das Repository, wobei dieses in das `local` und das `remote` Repository gegliedert werden kann. Mit `add` wird eine Kopie des Working Directory erstellt, wobei die Änderungen an den Dateien mit `new`, `modified` oder `deleted` gekennzeichnet sind. Die Arbeitskopie befindet sich nun in der Staging Area und noch nicht in der Datenbank. Nun haben die Dateien den Status `staged` und können im nächsten Schnappschuss eingepflegt werden. Nach einem `commit` wird ein Schnappschuss im lokalen Repository erstellt, der mit einem `push` in der Datenbank und somit im entfernten Repository gesichert wird.

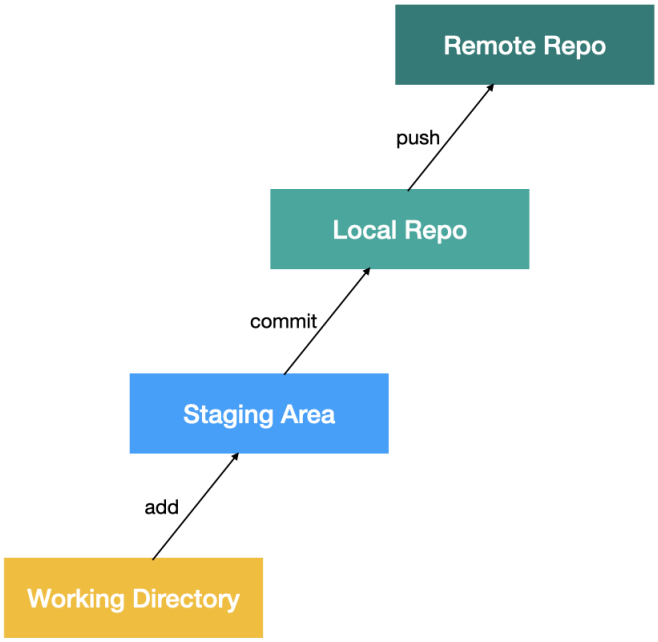

Abbildung 2.6: Die drei Hauptstufen von Git

Merge
Oftmals werden Entwicklungszweige erstellt um Bugs zu beheben oder an neuen Funktionalitäten einer Software zu arbeiten. Diese Änderungen sollen zu einem späteren Zeitpunkt in einen einzigen Branch integriert werden, der in den meisten Fällen der Hauptzweig `master` ist. Dies wird durch einen Merge ermöglicht.

Mehrere Commits aus zwei Entwicklungszweigen werden in einen einheitlichen Verlauf zusammengeführt und die Branches somit vereint. Im Prinzip sucht sich der Pointer der Commits einen gemeinsamen Commit als Basis des Merges. Genau dann, wenn diese Basis gefunden wird, wird ein zusätzlicher Commit für den Merge erstellt. Dadurch werden alle seither vorgenommenen Änderungen in einen gemeinsamen Verlauf vereint.

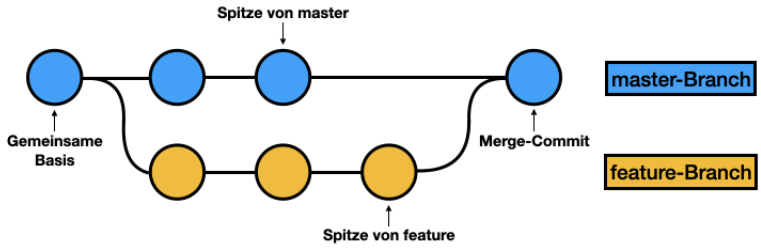

Abbildung 2.7: Mergen von zwei Branches mit Merge-Commit

Im Gegensatz zu einfachen Commits während der Entwicklung, haben Merge-Commits zwei übergeordnete Commits. Sind Änderungen in beiden Branches vorgenommen worden, können von Git nicht automatisch zusammengeführt werden. Dies führt oftmals zu einem Merge-Konflikt, der von dem Entwickler manuell beseitigt werden muss. Im Normalfall kann Git die Commit-Historie automatischen zusammenführen, es sei denn es gibt Änderungen im aktuellen Branch und im Ziel-Branch, die zu Konflikten im Projekt führen.

Grundsätzlich gibt es zwei Arten von Merges (Atlassian, vgl. [13], 09.04.2020):

- **Fast-Forward-Merge:**
  Wenn der Verlauf vom aktuellen Entwicklungszweig zum Ziel-Branch linear ist, findet ein Fast-Forward-Merge statt. Linear bedeutet, dass nur an einem Entwicklungszweig Änderungen vorgenommen worden sind und der andere genauso geblieben ist, wie zu dem Zeitpunkt, an dem der andere Branch erstellt worden ist.
  Bei diesem Merge wird ausschließlich der Pointer des Ziel-Branches auf die Commit-Spitze des aktuellen Branches gelegt. Somit werden die beiden Entwicklungszweige kombiniert und enthalten dieselbe Commit-Historie. Da lediglich die Position des Pointers verändert wird, erstellt Git keinen Merge-Commit. Für den Fall, dass kein linearer Pfad zum Ziel-Branch existiert, erfolgt das Zusammenführen der Branches mit einem 3-Way-Merge.

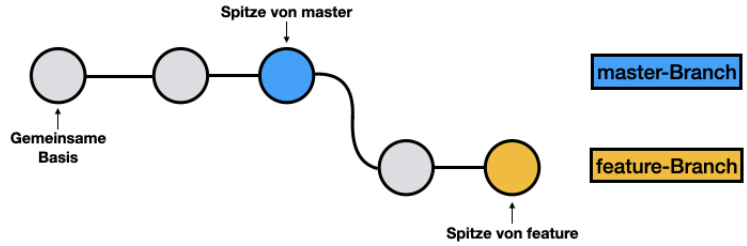

Abbildung 2.8: Es werden Änderungen am `feature`-Branch vorgenommen, der `master`-Branch bleibt gleich.

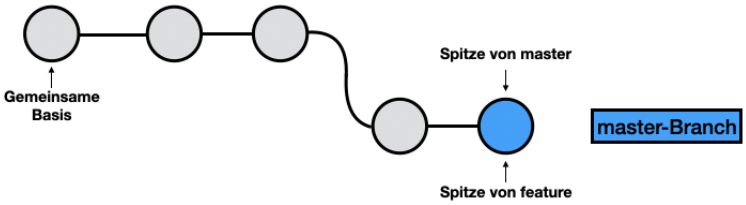

Abbildung 2.9: Die Änderungen des `feature`-Branches werden in den `master`-Branch eingepflegt.

- **3-Way-Merge:**
  Die Bezeichnung 3-Way-Merge kommt daher, dass für das Zusammenführen von zwei Entwicklungszweigen drei Commits benötigt werden. Zwei Commits sind die Spitzen der beiden Branches und ein Commit ist ihr gemeinsamer Basis-Commit. Hierbei werden seit der Erstellung eines neuen Entwicklungszweiges sowohl am Ziel-Branch als auch am aktuellen Branch neue Änderungen gemacht, die wieder zusammengeführt werden sollen. Vor allem beim Entwickeln neuer umfangreicher Funktionalitäten oder beim zeitgleichen Arbeiten am selben Projekt mit mehreren Entwicklern, ist ein 3-Way-Merge erforderlich.
  Allerdings kann es hierbei zu Konflikten während eines Merges kommen, die von dem Entwickler selbst gelöst werden muss.

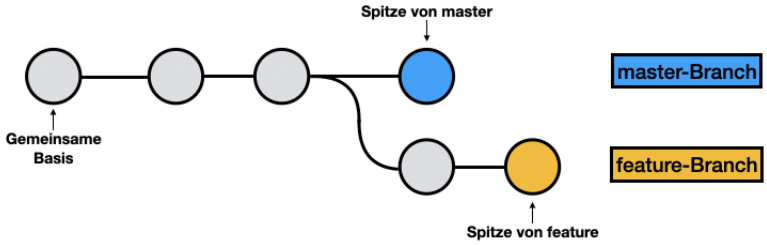

Abbildung 2.10: Es werden Änderungen am `feature`-Branch am `master`-Branch vorgenommen.

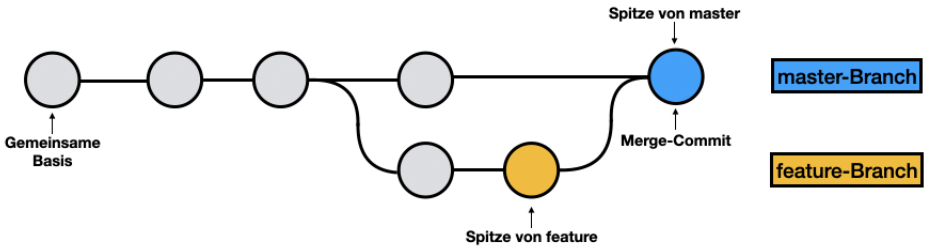

Abbildung 2.11: Die Änderungen des `feature`-Branches werden mit einem Merge-Commit in den `master`-Branch eingepflegt.

2.1.3 GitLab

Im Jahr 2011 ist GitLab von den ukrainischen Entwicklern Dmitri Saparoschez und Valery Sizov ursprünglich in der Programmiersprache Ruby on Rails und mittlerweile noch zusätzlich in Go und Vue.js geschrieben worden. Es ist eine Webapplikation zur Versionskontrolle, das auf Git basiert, in mehr als 100.000 Unternehmen zum Einsatz kommt und von etwa 30 Millionen registrierten Benutzern genutzt wird (GitLab, vgl. [14], 30.11.2020). Neben dem Arbeiten mit Git stellt diese Anwendung einige weitere Funktionen zur Verfügung. Diese sind beispielsweise eine integrierte und kostenlose Continuous Integration/Delivery (CI/CD), das Issue Tracking, die Wiki-Funktion und vieles mehr. Zusätzlich können Rollen zugewiesen werden, die die Berechtigungen von den jeweiligen Nutzern festlegen.

Seit 2013 gibt es zwei Linzenzmodelle von GitLab. Die Community Edition (CE) wird unter der MIT-Lizenz als Open-Source-Software entwickelt und ist auf gitlab.com als Software as a Service (SaaS) erreichbar. Hingegen wird die Enterprise Edition unter einer proprietären Lizenz entwickelt und beinhaltet mehr Funktionen, die für Unternehmen relevanter sind.

2.2 Hypertext Markup Language

Hypertext Markup Language, kurz HTML, ist eine Auszeichnungssprache, welche den Inhalt und die Struktur einer Webseite definiert. Die Strukturierung kann dabei in Paragraphen, Listen, Bildern, Tabellen und einigen anderen Elemente erfolgen (MDN Web Docs, vgl. [15], 10.04.2021).

"Hypertext" bezeichnet die Links, die entweder auf HTML-Elemente innerhalb einer Webseite verlinkt oder mehrere Webseiten miteinander verbindet. Grundsätzlich werden mithilfe von Links Inhalte in das Internet hochgeladen und mit Seiten verlinkt.
"Markup" dient dazu Texte, Bilder sowie weitere Inhalte zur Darstellung in einem Webbrowser zu annotieren.
Ein HTML-Element wird in den sogenannten **"Tags"** auf einer Webseite platziert. Dabei wird der Name des Elementes zwischen die Klammern `<` und `>` gegeben. Diese Tags sind case-insensitive, weshalb sie in Groß- oder Kleinbuchstaben oder in einer Mischung dieser geschrieben werden können. Beispielsweise kann ein `<label>` Tag auch als `<LABEL>` oder `<Label>` definiert werden.

Zusätzlich zu HTML kommen andere Technologien wie Cascading Style Sheets, auch CSS, und JavaScript zum Einsatz. Dabei beschreibt CSS das Aussehen der Webseite während JavaScript die Funktionalität und das Verhalten festlegt.

2.2.1 HTML-Elemente

Aufbau
Ein HTML-Element ist vorwiegend zusammengesetzt aus einem öffnenden und einem schließenden Tag sowie der Inhalt zwischen diesen Tags. Es ist eine Art Container oder Behälter für die Informationen, die innerhalb von ihm definiert sind.

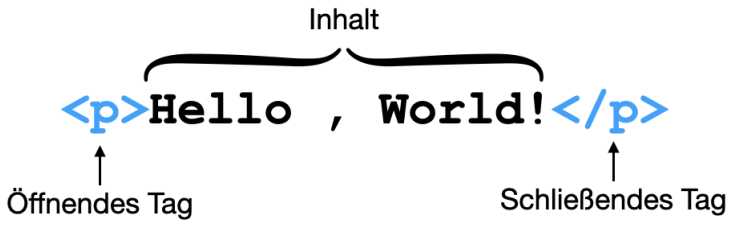

Abbildung 2.12: Beispiel eines HTML-Elementes

- Das **öffnende Tag** ist zusammengesetzt aus spitzen Klammern `< >`, die den Namen des HTML-Elementes umschließen. Es gibt die Stelle an, an der das Element beginnt.
- Das **schließende Tag** ist ähnlich dem öffnenden Tag nur, dass zusätzlich ein

Schrägstrich / vor dem Namen des HTML-Elementes steht. Es gibt die Stelle an, an der das Element endet.

- Der **Inhalt** des HTML-Elementes steht zwischen diesen beiden Tags. In der Abbildung handelt es sich hierbei um einen simplen Text.

Des Weiteren können HTML-Elemente Attribute beinhalten, die in den spitzen Klammern und nach dem Namen des öffnenden Tags geschrieben werden. Attribute geben Auskunft über zusätzliche Informationen über das Element, der nicht im Inhalt ersichtlich sein soll. Dabei müssen folgende Voraussetzungen erfüllt sein:

- Zwischen dem HTML-Elementname und dem Attribut oder mehreren Attributen muss ein Leerzeichen sein.
- Nach dem Attributnamen folgt ein Gleichheitszeichen `=`.
- Der Attributwert muss von Anführungszeichen `"` umschlossen sein.

Verschachtelte Elemente
HTML-Elemente, die sich innerhalb eines Elementes befinden, werden als Verschachtelung bezeichnet. Dabei muss stets darauf geachtet werden, dass die Tags wieder korrekt geschlossen werden. Befindet sich zum Beispiel ein `<strong>` Tag innerhalb eines `<p>` Tag, muss darauf geachtet werden, dass der innere Tag vor dem äußeren Tag geschlossen wird.

Leere Elemente
Obwohl die meisten HTML-Elemente einen Inhalt haben, gibt es dennoch einige, die keinen besitzen. Dazu zählt beispielsweise der `<img>` Tag. Bei diesem Element gibt es keinen schließenden Tag und es umhüllt auch keinen Inhalt. Der Zweck hinter diesen Tags ist es, einzelne Gestaltungselemente einzubinden.

```
<img src="images/statue.png" alt="Art Statue">
<img src="images/statue.png" alt="Art Statue" />
```

Abbildung 2.13: Beispiel von zwei leeren HTML-Elementen ohne und mit geschlossenem Ende

Damals ist mit Extensible HTML, kurz XHTML am Ende eines leeren HTML-Tags ein Schrägstrich gesetzt worden. Mit HTML5 kann dieser weggelassen werden.

2.2.2 Aufbau eines HTML-Dokumentes

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/statue.png" alt="Art Statue">
  </body>
</html>
```

Abbildung 2.14: Aufbau eines HTML-Dokumentes

Die folgenden HTML-Tags bilden zusammen eine HTML-Seite:

HTML-Element	Beschreibung
`<!DOCTYPE html>`	Früher hat dieses Tag dazu gedient, um mit Prüfung von Fehlern und anderen Überprüfungen sicherzustellen, dass die HTML-Seite gut dargestellt wird und funktioniert. Heute dient dieses Tag lediglich dazu, um sicherzugehen, dass das Verhalten des Dokumentes korrekt ist.
`<html></html>`	Diese Element schließt den gesamten Inhalt einer HTML-Seite ein und wird auch als Root- Element bezeichnet.
`<head></head>`	Dieses Element ist eine Art Container und umfasst alles, was auf der Webseite beinhaltet, aber nicht für den Betrachter sichtbar sein soll. Dazu zählen zum Beispiel Gestaltungen mit CSS, Zeichensatzdeklarationen und vieles mehr.
`<title></title>`	Damit wird der Titel der Webseite beschrieben, der in der Registerkarte des Browsers angezeigt wird.
`<body></body>`	Dieser HTML-Tag beinhaltet den Inhalt, der für die Benutzer der Webseite angezeigt wird.

Tabelle 2.1: Unterstützte Funktionalitäten der Tastatur

2.3 JavaScript

JavaScript, kurz JS, basiert auf dem ECMAScript Standard und ist eine dynamische, funktionale und objektorientierte Skriptsprache, die im Mai 1995 innerhalb von 10 Tagen von Brendan Eich entwickelt worden ist (MDN Web Docs, vgl. [16], 11.02.2021). Erstmals ist sie bei Netscape für dynamisches HTML in Webbrowsern zum Einsatz gekommen. Mittlerweile wird JS auch auf Servern oder Mikrocontrollern verwendet.

2.3.1 ECMAScript

ECMAScript ist von Brendan Eich bei Netscape entwickelt worden und gibt vor welche JavaScript Funktionalitäten von Webbrowsern implementiert werden müssen (Harband und Smith 2020, vgl. [17], 10.04.2021). Somit kann erreicht werden, dass Webseiten ohne Bedenken in jedem beliebigen Browser funktionsfähig sind. Gäbe es diesen Standard nicht, müssten Polyfills eingesetzt werden. Diese würden Funktionalitäten in JavaScript implementieren, die jeder Browser verstehen sollte. Diese Notlösung führt dazu, dass zusätzlich mehr programmiert wird, der Nutzer viel größere Dateien herunterladet, die Lösung nicht nativ ist und die Performance darunter leidet.

2.3.2 Dynamische Typisierung

JavaScript ist schwach typisierte und dynamische Programmiersprache. Wenn Variablen erstellt werden, haben sie keinen expliziten Datentyp. Anhand des Wertes einer Variable ändert sich intern auch ihr Datentyp. Der aktuellste ECMAScript Standard definiert neun Typen, die in zwei Kategorien unterteilt werden können.

Primitive Datentypen:

- `undefined`: Die Variable hat keinen Datentyp oder wurde explizit als `undefined` angegeben.
- `Boolean` ist ein logischer Datentyp der einen Wahrheitswert `true` oder `false` annehmen kann.
- `Number` ist ein numerischer Datentyp, der dem IEEE 754 konform ist. Das heißt, dass fast alle Zahlen, inklusive ganze Zahlen als Gleitkommazahl gespeichert werden.
- `String` ist eine Sequenz von Buchstaben.
- `BigInt` ist ein numerischer Datentyp, der ganze Zahlen in der Langzahlarithmetik darstellt.
- `Symbol` ist ein eindeutiger String, der nicht mehrmals erstellt werden kann. Es wird als Objekteigenschaft verwendet. Somit können JavaScript-Objekte wiederholt denselben Eigenschaftsnamen verwenden.
- `null` gibt an, das absichtlich ein Wert fehlt und die Variable auf kein Objekt zeigt.

Strukturierte Typen:

- `Object`: Alle Variablen, die mit `new` angelegt werden, sind Objekte. Dazu zählt auch das JavaScript-Objekt { ... }.
- `Function` gibt den Datentypen an, die als JavaScript-Funktion mit dem Schlüssenwort `function` deklariert worden ist.

Unterschied zwischen == und ===
== überprüft auf Gleichheit der beiden Variablen. Sollten diese nicht vom gleichen Typ sein, werden diese auf denselben Datentyp konvertiert. Referenzen gelten gleich, wenn sie auf dasselbe Objekt zeigen.

- `1 == 1`
- `'1'== 1`
- `1 == true`
- `0 == false`

=== achtet genauso wie oben genannt auf den Inhalt mit dem Unterschied, dass davor überprüft wird, ob beide Variablen denselben Typ haben.

Dasselbe gilt für die Überprüfung der Ungleichheit `!=` und `!==`.

2.4 TypeScript

TypeScript (TS) ist eine Erweiterung von JavaScript und basiert ebenfalls auf den ECMAScript Standards (Malcher, Hoppe und Koppenhagen 2020, vgl. [**?**]). Im Gegensatz zu JavaScript besitzt TypeScript ein statisches Typsystem. An und für sich wird TypeScript nicht von Browsern unterstützt, denn es handelt sich um eine Programmiersprache, die mittels eines Kompilierers diese Sprache auf JavaScript umwandelt. Man spricht hierbei von einem Transpiler. Mit einem Typsystem ist es möglich, bereits zur Entwicklungszeit Fehler und Probleme ausfindig zu machen.
TypeScript ermöglicht auch die Verwendung von neuen ECMAScript-Features, weshalb Projekte mit den neuesten Features erstellt werden können. Der Kompilierer ist dafür zuständig, das Browser mit älteren Versionen des ECMAScriptes den transpilierten Code ausführen können.

2.4.1 Typen

Wie zuvor erwähnt worden ist, ist eine der Stärken von TypeScript die Typisierung. Es werden einige primitive Datentypen mitgeliefert. Zu denen gehören `number`, `string` und `boolean`.

- `number` umfasst alle Gleitkommazahlen sowie ganze Zahlen.
- `string` definiert eine Zeichenkette.
- `boolean` kann die Wahrheitswerte `true` oder `false` annehmen.

Variablen, die mit diesen jeweiligen Typen definiert sind, dürfen keinen anderen Wert annehmen, der nicht ihrem Datentyp entspricht. Ansonsten wird ein Fehler vom Kompilierer geworfen. Einige Entwicklungsumgebung besitzen ebenfalls die Funktion diese Fehler erkennen zu können.

```
let age: number = 19;
let name: string = "Sonja";
let isAdult: boolean = true;
let height: number = 1.51;
```

Abbildung 2.15: Variablen mit primitiven Datentypen

Neben den primitiven Datentypen gibt es die beliebigen Werte `any` und `unknown`. Diese Datentypen sind dynamisch und können jeden möglichen Wert annehmen, ohne dass sich der Kompilierer beschwert.

2.4.2 Union Types

Union Types beschreiben Typen, die aus mehren Typen zusammengesetzt sind. Ein geeignetes Beispiel hierfür ist die Hausnummer. Sie kann entweder eine Zahl oder eine Zahl mit Buchstaben sein.

```
let houseNumber: string | number = 28;
houseNumber = "117a";
```

Abbildung 2.16: Variable mit mehreren Typen

2.4.3 Type Definition

Angenommen es gibt eine JavaScript Bibliothek, die oft zum Einsatz kommt und unter anderem wird diese auch für TypeScript Projekte verwendet. Allerdings kann TypeScript nichts mit dieser Bibliothek anfangen, da die Typen nicht existent sind. Genau aus diesem Grund werden Type Definitions verwendet. Sie werden mit den jeweiligen JavaScript-Dateien geliefert und beinhalten keinen Quellcode, sondern lediglich nur Informationen zu diesem. Dazu gehören Informationen wie Datentypen, Konstanten und Funktionen mit Parameter- und Rückgabetypen.

Eine Funktion wird vom TypeScript-Compiler in JavaScript umgewandelt und verliert somit jegliche Informationen zu den vorher festgelegten Datentypen. Daher können Dateien für Type Definitions generiert oder erstellt werden.

```
function add(a: number, b: number): number {
    return a + b;
}
```

Abbildung 2.17: Funktion zum Addieren von zwei Zahlen in TypeScript

```
function add(a, b) {
    return a + b;
}
```

Abbildung 2.18: Funktion zum Addieren von zwei Zahlen in JavaScript

```
declare function add(a: number, b: number): number;
```

Abbildung 2.19: Type Definition für die Funktion zum Addieren von zwei Zahlen

2.4.4 TypeScript Compiler

Der TypeScript Compiler gibt dem Entwickler einen Spielraum, gewisse Einstellungen zu definieren. Dazu zählen strikte Null-Überprüfungen, wo ein Fehler geworfen wird, wenn die Variable `null` sein kann und nicht überprüft wird. Außerdem kann der Datentyp `any` ausgeschaltet werden für eine genauere Datentypisierung. Variablen, die nicht verwendet werden, wie beispielsweise lokale Variablen oder Parametervariablen, können so eingestellt werden, dass ein Fehler geworfen wird. Für eine einfachere und angenehmere Entwicklung können Source-Maps aktiviert werden. Diese dienen zur Anzeige des

Quellcodes in TypeScript, wenn Fehler im Browser geworfen werden oder ein Debug durchgeführt wird.

2.5 Node.js und npm

2.5.1 Node.js

Node.js ist eine JavaScript-Laufzeitumgebung mit der JavaScript-Code außerhalb eines Webbrowsers ausgeführt werden kann (OpenJS Foundation, vgl.[20], 10.04.2021). Somit kann der Code nicht nur clientseitig, sondern auch serverseitig verwendet werden. Node.js hat eine eventbasierte Architektur und wird auf der im Jahr 2009 von Google entwickelten Open-Source-JavaScript-Engine V8 ausgeführt, der ursprünglich nur für Browser verwendet worden ist.
Zu den Einsatzgebieten gehören die Entwicklung von Webservern, die Erstellung von Skripten oder Tools im Terminal sowie seit neuestem die Entwicklung von Desktopanwendungen. Neben der Laufzeitumgebung stellt Node.js seine Core-API zur Verfügung. Diese reicht von simplen Netzwerkzugriffen bis hin zu Zugriffen auf Dateien.

Vorteile und Gründe zur Verwendung von Node.js
Mit seiner eventbasierten Architektur führt Node.js nur einen einzigen Thread aus, in dem ein Event Loop läuft. Im Gegensatz zu Node.js verwenden traditionelle Webframeworks für jede Anfrage auf den Server einen eigenen Thread. Näher betrachtet kann das Starten eines Threads einige Nachteile mit sich bringen. Angenommen ein Thread, der lediglich die Anfrage bearbeitet, benötigt 2 MB im Hauptspeicher, beginnt die Überlastung eines Servers mit 8 GB Hauptspeicher und 4000 gleichzeitigen Anfragen. Ein weiteres Beispiel ist in Blogpost von caustik's blog zu finden (caustik's blog 2012, vgl. [21], 13.02.2021).
Mit der Verwendung eines einzigen Threads, muss auch dementsprechend mit Gefahren gerechnet werden.

Rechenintensive Anfragen könnten den einzigen Thread überlasten und alle anderen Anfragen verzögern. Diese Verlangsamung würde solange andauern bis dieser rechenintensive Prozess fertig ausgeführt worden ist. Im Normalfall wird bei einer Exception oder bei einem Fehler der Thread gestoppt. Ist allerdings nur ein einzelner Thread vorhanden, kann ein Fehler bereits zum Absturz des Programmes führen. Um das Abstürzen aufgrund von Exceptions zu vermeiden können diese Fehler mit Callbacks verarbeitet werden.

Konkurrenz
Deno ist genauso wie Node.js eine JavaScript-Laufzeitumgebung und ist von demselben Entwickler Ryan Dahl (Deno, vgl. [22], 13.02.2021). Node.js hat viele Probleme gehabt und die Benutzerfreundlichkeit ist ebenfalls nicht zufriedenstellend gewesen. Aus diesem Grund ist Deno erstmals im Jahr 2018 veröffentlicht worden. Neben JavaScript wird auch

TypeScript nativ unterstützt und ein großer Wert auf die Sicherheit mit Erlaubnissen gelegt. Ein Modulsystem wie npm kommt nicht mehr zum Einsatz. Externe Module beziehungsweise Bibliotheken werden mit absoluten Adressen importiert. Dieses Konzept ist von Golang inspiriert.

2.5.2 npm

Node Package Manager, kurz npm, ist ein Paketmanager, der die Verwaltung und Bereitstellung der Module und Abhängigkeiten pflegt (vgl. [23], 12.02.2021). Diese und jegliche andere Informationen stehen in der `package.json` Datei. npm setzt sich aus zwei Komponenten zusammen:

Registry
Die Registry beziehungsweise das Repository ist eine Plattform von npm selber, welche Entwickler die Möglichkeit gibt, Open-Source Software zu veröffentlichen, die als Packages bekannt sind (vgl. [24], 09.04.2021). Der Schwerpunkt liegt hier bei JavaScript Libraries und Frameworks. Außerdem dürfen auch andere Dateien, wie zum Beispiel CSS oder TypeScript Deklarationen, hochgeladen werden.
Für Unternehmen oder Personen, die ihren Code nicht veröffentlichen wollen, besteht die Option sich einer npm Mitgliedschaft anzuschließen. Mit der Mitgliedschaft dürfen die Benutzer private packages hochladen. Die Zugriffsverteilung ist je nach Abonnement unterschiedlich und auf Einzelpersonen oder Mitglieder und Teams eines Unternehmens eingeschränkt.

Alternativen für die npm Registry sind:

Verdaccio ist eine simple und leichtgewichtige Registry zum selber Hosten. Die Open-Source Software eignet sich gut für Personen die sich kaum mit dem Einrichten von Produkten auseinandergesetzt haben, da keine Konfigurationen erforderlich sind (vgl. [25], 09.04.2021).

Jfrog Artifactory ist eine sogenannte One-Stop Solution Software. Jfrog Artifactory hat nicht nur die npm Registry integriert, sondern auch andere wie Maven, Rubygems, Docker, Go und etliche andere. Sie wird als die perfekte Lösung für große Unternehmen, die sich mit verschiedenen Technologien beschäftigen, bezeichnet (JFrog Ltd., vgl. [26], 09.04.2021). Die Zugriffe auf Packages oder zahlreiche andere Repository-Formate erfolgt mit einer Benutzer- und Rechteverwaltung. Jfrog Artifactory kann selbst gehostet oder auf der Cloud von Jfrog bereitgestellt werden.

npm CLI
Die größte Eigenschaft von der CLI, ist die Installation von einer npm Registry. Als Standard wird die Registry von npm herangezogen. Mit `npm config set registry https://my-registry.com` kann die Registry geändert werden. Um Abhängigkeiten

oder Module zu installieren, wird der Befehl `npm install my-package` verwendet. Die installierten Packages kommen in den Ordner `/node_modules` und werden auch automatisch in die `package.json` eingepflegt. Somit muss der Entwickler diese Aufgabe nicht händisch übernehmen. Des Weiteren können Packages global installiert werden. Die am weitesten verbreitete Anwendungsfälle sind CLIs die in JavaScript programmiert worden sind. Ein Beispiel dafür wäre das beliebte Web-Framework Angular mit seiner Angular-CLI. Nach der globalen Installation kann man auf das CLI-Programm `ng` direkt vom Terminal aus zugreifen.
Für das Erstellen von npm Projekten wird der Befehl `npm init` benötigt. Hier fragt die npm CLI nach Informationen wie Projektname, Beschreibung, Author etc.

2.6 Jest

Jest ist ein Testing Framework für die JavaScript-Laufzeitumgebung NodeJS (Facebook, vgl. [27], 11.04.2021). Da Jest zuerst die Tests durchführt, die beim letzten Mal fehlgeschlagen sind und anschließend die Tests durchführt, die mehr Zeit beanspruchen, ist das parallele Testen sehr schnell. Jest bietet dem Entwickler die Möglichkeit, die Test Coverage zu evaluieren. Mit der Mocking API kann jedes beliebige Objekt gemockt werden.

2.7 Webpack

Webpack ist ein statischer Modul-Bundler für JavaScript-Applikationen (vgl. [28], 11.02.2021). Mehrere JavaScript-Dateien werden zur Nutzung in einem Webbrowser zusammengeführt und zu einer oder mehreren Dateien gebündelt, abhängig von der Anzahl an Modulen. Diese Module werden intern in einem Abhängigkeitsgraphen dargestellt. Rekursiv beginnt dieser vom Eingangspunkt und fügt alle Abhängigkeiten in diesen Graphen ein. Somit wird nur der Code zusammengebündelt, der auch benötigt wird.
Der Vorteil besteht darin, dass schnell eine Datei heruntergeladen werden kann, die verkleinert und gebündelt ist.

Das Konzept von Webpack kann in sechs Kernpunkte unterteilt werden:

2.7.1 Entry

Der Eingangspunkt gibt an wo Webpack beginnen soll das Bundle zusammenzusetzen. Anhand dieses Punktes kann herausgefunden werden welche Abhängigkeiten direkt oder indirekt benötigt werden, die anschließend in den Abhängigkeitsgraphen eingefügt werden.

```
module.exports = {
  entry: './src/main.js',
};
```

Abbildung 2.20: webpack.config.js

2.7.2 Output

Der Output definiert wie Webpack die zusammengebündelten benennen oder wo sie gespeichert werden sollen.

```
const path = require('path');

module.exports = {
  entry: './src/main.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'my-webpack.bundle.js'
  }
};
```

Abbildung 2.21: webpack.config.js

2.7.3 Loaders

Webpack unterstützt nur JavaScript und JSON Dateien als Import im Code. Mit Loadern is es möglich Webpack so zu erweitern, dass andere Dateien ebenfalls importiert und in den Abhängigkeitsgraphen eingefügt werden können. Diese Loader besitzen zwei wichtige Eigenschaften:

1. `test` gibt an welche Dateien von diesem Loader verwendet werden. Hierbei handelt es sich um einen regulären Ausdruck.

2. `use` legt fest welcher Loader beziehungsweise welche Bibliothek für die Dateien verwendet werden sollen.

```
const path = require('path');

module.exports = {
  entry: './src/main.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'my-webpack.bundle.js'
  },
  module: {
    rules: [
      {
        test: /\.css$/i,
        use: ['style-loader', 'css-loader']
      }
    ]
  }
};
```

Abbildung 2.22: webpack.config.js

In der obigen Abbildung ist die Eigenschaft `rules` definiert mit den beiden erforderlichen Eigenschaften `test` und `use`. `rules` sagt dem Webpack Compiler, dass dieser die Dateien mit der Endung `.css` innerhalb eines `require()` oder `import` Statements mit einem `css-loader` umwandeln soll. Anschließend werden die transformierten Dateien zum Bundle hinzugefügt.

2.7.4 Plugins

Mithilfe von Plugins besteht die Möglichkeit Webpack so zu erweitern, dass gewisse Funktionalitäten optimiert, automatisiert oder neu hinzugefügt werden. Die Verwaltung von Assets und das Injizieren von Umgebungsvariablen ist ebenso möglich.
Plugins können verwendet werden, indem das `require()` Statement verwendet und zum `plugins`-Array hinzugefügt wird. Mittels Optionen können diese angepasst werden. Ein Plugin kann mehrmals in einer Konfiguration definiert werden, weshalb mit dem `new`-Operator eine Instanz erstellt werden muss.

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const webpack = require('webpack');

module.exports = {
  entry: './src/main.js',
  output: {
    path: path.resolve(__dirname, 'dist')
  },
  plugins: [new HtmlWebpackPlugin({ template: './src/index.html' })]
};
```

Abbildung 2.23: webpack.config.js

In diesem Beispiel wird das `html-webpack-plugin` verwendet, um anhand einer selbstgeschriebenen HTML-Vorlage ein Dokument zu generieren, das die JavaScript und Asset Bundles bereits importiert beziehungsweise referenziert und schließlich im Output-Ordner `dist` ablegt.

2.7.5 Mode

Webpack gibt die Möglichkeit zu definieren in welcher Umgebung das Bundle laufen wird. Hierbei können drei verschiedene Modi gesetzt werden: `development`, `production` oder `none`. Dabei versucht Webpack intern, je nach Modus, die bestmögliche Bundle-Optimierung einzusetzen.

```
module.exports = {
  mode: 'production'
};
```

Abbildung 2.24: webpack.config.js

2.7.6 Browserkompatibilität

Es ist wichtig zu wissen, dass Webpack alle Browser unterstützt, die ES5-konform sind. Dazu zählen alle moderne und mobile Browser. Ein nichtkompatibler Webbrowser ist beispielsweise der Internet Explorer der Version 8 oder darunter. Falls diese Browser jedoch unterstützt werden sollen, wird ein Polyfill für den JavaScript Standard ES5 benötigt.

Kapitel 3

Ausgewählte Aspekte

3.1 WCAG 2.1

Die Web Content Accessibility Guidelines 2.1 (WCAG 2.1) sind im Dezember 2008 veröffentlicht worden und sind ein Standard, der weltweit verwendet wird, um Dienstleistungen im Web so barrierefrei wie möglich zu gestalten (W3C 2018, vgl. [29], 18.08.2020). Somit können auch Menschen mit Behinderungen, wie etwa Sehschwäche, Blindheit, Hörverlust, Taubheit, körperliche oder kognitive Einschränkung, Sprachbehinderung, Lichtempfindlichkeit, Lernbehinderung oder Kombinationen dieser, das World Wide Web nutzen. Diese Richtlinien sind vom World Wide Web Consortium (W3C) am 5. Juni 2018 für Webanwendungen empfohlen worden.

Durch die Befolgung der Empfehlungen des W3C können Webdesigner und -entwickler, politische Entscheidungsträger, Käufer, Lehrer und Studenten beziehungsweise Schüler das WWW nahezu problemlos nutzen. Die WCAG 2.1 setzen sich aus allgemeinen Prinzipien, allgemeinen Richtlinien, prüfbaren Erfolgskriterien sowie einer Vielzahl an Techniken zusammen:

- Es gibt vier **Prinzipien**, die die Grundlagen für die Web-Zugänglichkeit bilden: wahrnehmbar, bedienbar, verständlich und robust (englisch: perceivable, operable, understandable and robust).
- Die Prinzipien beinhalten insgesamt 13 **Richtlinien**, an die sich Entwickler halten sollen, um das Web zugänglicher zu machen. Sie sind zwar nicht prüfbar, helfen allerdings dabei die Erfolgskriterien besser zu verstehen und die Techniken besser umzusetzen.
- **Prüfbare Erfolgskriterien** werden dort eingesetzt, wo bestimmte Anforderungen und Tests für die Konformität erforderlich sind, wie etwa beim Entwurf, beim Kauf, bei Regelungen und bei vertraglichen Vereinbarungen. Es gibt drei Ebenen der Konformität, wobei die höchste AAA, die mittlere AA und die niedrigste A ist.

- **Ausreichende und beratende Techniken** dienen zur Erfüllung der Erfolgskriterien und als Beratung. Häufig vorkommende Misserfolge sind ebenfalls dokumentiert.

Bei Erreichung der höchsten Ebene ist eine hundertprozentige Web-Zugänglichkeit jedoch nicht garantiert. Die WCAG 2.1 dienen ausschließlich als Anleitung, um das Web so barrierefrei wie möglich zu gestalten. Deshalb wird vom W3C empfohlen sich regelmäßig über den neuesten Stand zu informieren und sich mit anderen zu beraten. Folgende Bemerkung ist vom W3C veröffentlicht worden:

"[...] Authors are encouraged to consider the full range of techniques [...] as well as to seek relevant advice about current best practice to ensure that Web content is accessible, as far as possible, to this community. [...]" (W3C 2018, vgl. [29], 18.08.2020)

3.2 Web Components

Web Components sind eine Web-Technologie, die es den Entwicklern ermöglicht, selbstdefinierte HTML-Elemente zu erstellen und wiederzuverwenden (MDN Web Docs, vgl. [30], 03.10.2020). Diese sind mit ihrem CSS und JavaScript gekapselt und sind somit vollständig von anderem Code getrennt. Mit einigen JavaScript Frameworks wie etwa Angular war es zwar bereits möglich wiederverwendbare HTML-Elemente zu definieren, allerdings nutzt jedes der Frameworks einen anderen Standard. Dies bedeutet, dass der Code in anderen Projekten in den meisten Fällen nicht verwendbar ist. Genau dieses Problem ist durch Web Components gelöst worden. Die Web Components bestehen aus drei Hauptbestandteilen:

- **Customs Elements:** Ein Satz von JavaScript APIs zur Definition von benutzerdefinierten Elementen.
- **Shadow DOM:** Ein Satz von JavaScript APIs zum Hinzufügen eines DOM-Elementes mit gekapselten Shadow-DOM-Elementen, welches separat vom Hauptdokument DOM gerendert wird. Dadurch ist es möglich jegliche Funktionalitäten isoliert zu definieren, sodass der Programmierer keine Rücksicht auf Kollisionen mit anderen Dokumenten nehmen muss.
- **HTML Templates:** Markup-Vorlagen, die innerhalb der Elemente `<template>` und `<slot>` geschrieben werden, werden nicht auf der dargestellten Seite abgebildet. Diese sind dafür da, um sie mehrmals als Vorlage eines benutzerdefinierten Elements zu verwenden.

3.3 Factory Method Pattern

3.3.1 Was sind Design Patterns?

Um das Factory Method Pattern zu verstehen, muss man zunächst einmal wissen, was ein Design Pattern (dt. Entwurfsmuster) ist. In der Softwareentwicklung treten beim Entwurf oftmals dieselben Probleme auf. Abhilfe hierfür schaffen Design Patterns, die eine allgemeine wiederholbare Lösung dieser Designprobleme sind, den Entwicklungsprozess beschleunigen und programmiersprachenunabhängig sind (Source Making, vgl. [31], 30.10.2020). Sie repräsentieren somit eine Idee, und keine spezielle Implementierung, durch deren Verwendung der Quellcode flexibler, wiederverwendbar und einfacher für die Entwickler zu warten ist. Allerdings sind sie nicht in jedem Projekt zwingend erforderlich, da sie lediglich für die Problemlösung und nicht für die Projektentwicklung gedacht sind.

Arten von Design Patterns

- **Creational Pattern (dt. Erzeugungsmuster):** Sie dienen dazu Objekte zu erzeugen. Dieser Prozess wird gekapselt und ausgelagert, sodass die Objekterzeugung von der Implementierung des Objektes klar getrennt wird.
- **Structural Pattern (dt. Strukturmuster):** Sie stellen vorgefertigte Vorlagen für die Beziehungen zwischen den Klassen zur Verfügung und erleichtern somit den Softwareentwurf.
- **Behavioral Pattern (dt. Verhaltensmuster):** Sie führen dazu, dass die Software flexibler wird, indem das komplexe Verhalten dieser Software modelliert wird.

3.3.2 Begriffserklärung und Verwendung

Das Factory Method Pattern, auch bekannt als Factory Pattern oder Fabrikmethode, definiert ein Interface oder eine abstrakte Klasse zur Erstellung von Objekten (Source Making, vgl. [32], 30.10.2020). Die Objekterstellung wird hierbei von den Subklassen übernommen, die entscheiden welche Klasse instanziiert wird.

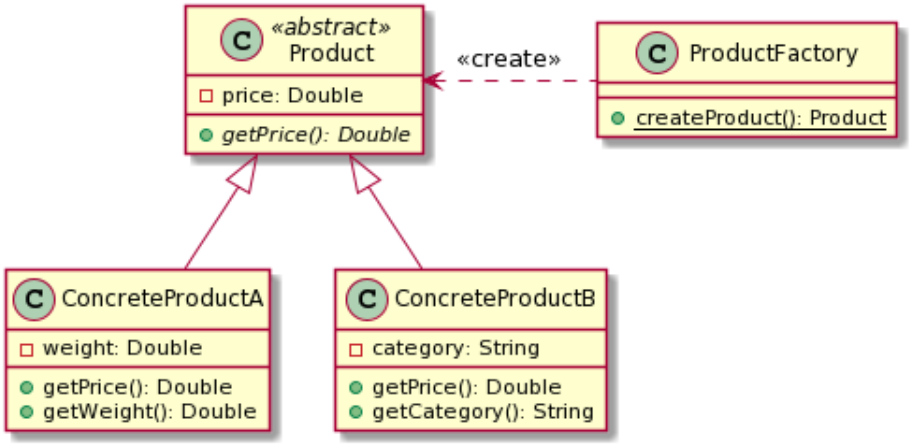

Abbildung 3.1: Veranschaulichung des Factory Method Patterns in einem Klassendiagramm

3.3.3 Vorteile und Nachteile

Vorteile

- Mit Hilfe des Factory Method Patterns können Unterklassen die Art, wie Objekte erstellt werden sollen, selbst wählen.
- Dieses Design Pattern fördert die lose Kopplung. Das heißt im Allgemeinen, dass anwendungsspezifische Klassen nicht mehr in den Code eingebunden werden müssen.
- Die Kommunikation erfolgt ausschließlich über die Schnittstelle oder die abstrakte Klasse, so dass der Code mit allen Klassen funktioniert, die entweder die Schnittstelle implementieren oder die abstrakte Klasse erweitern.

Nachteile

- Falls die verwendeten Klassen nicht abstrakt sind, muss für jede individuelle Klasse eine Factory geschrieben werden.

3.4 JavaScript Events und EventListener

trix-initialize
Sobald der `<trix-editor>` im DOM registriert wird und sein internes `Editor`-Objekt zur Benutzung bereit ist, wird das Event `trix-initialize` gefeuert.
Damit nicht die standardmäßig definierte Toolbar, sondern die erweiterte barrierefreie Toolbar, die diese ersetzen soll, angezeigt wird, ist es möglich auf dieses Event zu hören

und dementsprechend darauf zu reagieren. Somit kann mit der Trix-Extension die Toolbar ersetzt werden, ohne dass der Endbenutzer etwas von diesem
DOM-Manipulationsprozess mitbekommt.

mousedown
Das `mousedown` Event wird genau zu dem Zeitpunkt gefeuert, an dem sich der Mauszeiger auf einem HTML-Element befindet und dieses drückt.
Damit ein HTML-Button in der `<trix-toolbar>` aktiv ist und geklickt werden kann, hört dieser standardmäßig auf ein `mousedown` Event. Um diese Buttons mit einer Tastatur bedienen zu können, müsste der Trix-Editor im Quellcode auf Tastatureingaben reagieren und diese dementsprechend erneut validieren. In der Erweiterung Trix-Extension wird dieses Event künstlich ausgelöst mit `EventTarget.dispatchEvent()` und somit sind die Buttons in der Toolbar auch über die Tastatur bedienbar, ohne dass der Quellcode mit Wiederholung der Validation ergänzt werden muss.

keydown
Wenn eine beliebige Taste gedrückt ist, wird das `keydown` Event gefeuert und liefert einen Code, der aussagt, welche Taste im Moment gedrückt wird.
Um den WYSIWYG Texteditor mittels einer Tastatur bedienen zu können, wird auf das `keydown` Event gelauscht. Je nachdem, welche Taste gedrückt wird, wird entweder der nächste oder der vorherige Button oder die Button-Gruppe fokussiert, geklickt oder der Fokus zur weiteren Texteingabe in den Editor gesetzt.

Folgende Tasten und Tastenkombinationen gelten in der Toolbar und im Editor:

Tasten	Fokusbereich	Beschreibung
`ALT + F10`	`<trix-editor>`	Befindet sich der Fokus im Editor, dann gelangt man mit dieser Tastenkombination in die Toolbar und der erste vorkommende Button wird fokussiert.
`→ oder ↓`	`<trix-toolbar>`	Der nächste bzw. rechte Button in der Toolbar wird fokussiert.
`← oder ↑`	`<trix-toolbar>`	Der vorherige bzw. linke Button in der Toolbar wird fokussiert.
`STRG + →/↓`	`<trix-toolbar>`	Der erste Button der nächsten bzw. rechten Gruppe wird fokussiert.
`STRG + ←/↑`	`<trix-toolbar>`	Der erste Button der vorherigen bzw. linken Gruppe wird fokussiert.
`ENTER oder LEERTASTE`	`<trix-toolbar>`	Der aktuell fokussierte Button wird geklickt.
`ESC`	`<trix-toolbar>`	Befindet sich der Fokus in der Toolbar und wird die Taste gedrückt, so wird wieder der Editor fokussiert und der Cursor befindet sich an der zuletzt verwendeten Position.
`POS1`	`<trix-toolbar>`	Der erste in der Toolbar vorkommende Button wird fokussiert.
`ENDE`	`<trix-toolbar>`	Der letzte in der Toolbar vorkommende Button wird fokussiert.

Tabelle 3.1: Tastenkombinationen zur Verwendung der Toolbar mit einer Tastatur

focusout
Sobald ein HTML-Element im Begriff dabei ist den Fokus zu verlieren, wird das `focusout` Event gefeuert.
Beim Initialisieren wird für jeden HTML-Button, der ein `DropdownButton` 3.8.4 ist, ein Dropdown Menü erstellt. Dieses Menü besteht aus einem HTML `<div>` Element, in dem sich weitere HTML-Buttons befinden. Damit es für den Benutzer zu Beginn noch nicht sichtbar ist, erhält es das Attribut `hidden`. Sobald der `DropdownButton` geklickt wird, wird auch das Menü sichtbar und der Fokus auf den ersten Button darin gelegt. Wenn das Menü allerdings nicht mehr fokussiert ist, also das Event `focusout` gefeuert wird, erhält das Dropdown Menü erneut das Attribut `hidden` und wird somit für den Benutzer nicht mehr sichtbar sein.

3.5 MutationObserver

Der `MutationObserver` ist ein Interface, der Veränderungen in der Baumstruktur des DOMs beobachtet und wurde konzipiert, um die Mutation Events aus der DOM3 Events Spezifikation abzulösen.
Sobald die Toolbar geladen wird bzw. bestimmte Buttons geklickt werden, werden einige andere Buttons deaktiviert und erhalten das Attribut `disabled`. Dieses verhindert allerdings das Fokussieren eines Buttons, was mit dem Attribut `aria-disabled` problemlos funktioniert. Abhilfe verschafft deshalb ein `MutationObserver`. Erhält ein Button nun das Attribut `disabled`, beobachtet der `MutationObserver` diese Veränderung und entfernt es. Stattdessen fügt es das Attribut `aria-disabled` hinzu und setzt es auf `true`. Bei allen anderen Buttons, die nicht deaktiviert sind, ist `aria-disabled=false`.

3.6 Delegation

Die Delegation ist eine Alternative für die Vererbung. Vereinfacht wird die Aufgabe eines Objektes auf ein anderes Objekt verlagert (GeeksforGeeks 2018, vgl. [33], 11.04.2021). Das kann so aufgefasst werden, dass die Methode des ursprünglichen Objektes die Methode des Zielobjektes aufruft. Ein Beispiel hierfür ist eine Basisklasse `Printer`, die die Klassen `RealPrinter` und `Computer` erbt. Somit ist diese Basisklasse in der Lage die Methode `print()` aufzurufen. Vererbung ist an und für sich eine gute Strategie, um Probleme zu lösen, sollte aber einen logischen Zusammenhang zwischen `parent` und `child` aufweisen können. Ein Beispiel für einen schlechten Zusammenhang ist, dass ein Auto einen Motor erbt. Stattdessen kann eine Assoziation solch eine Beziehung besser beschreiben als eine Vererbung (TU Wien 2013, vgl. [34], 11.04.2021).

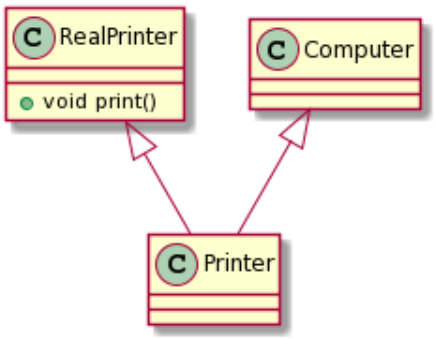

Abbildung 3.2: Ohne Delegation

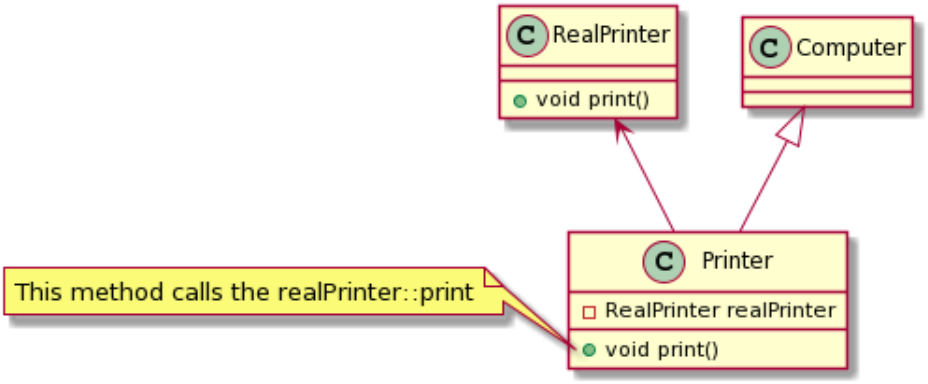

Abbildung 3.3: Mit Delegation

Im Rahmen der praktischen Arbeit dieser Diplomarbeit kann jedem `AttributeButton` kann die Identifikation eines HTML-Elements mitgegeben werden, das einen Dialog repräsentiert und bereits im HTML-Code erstellt worden ist. Die Delegation ermöglicht eine Kommunikation zwischen der Trix-Toolbar und dem Dialog. Sie dient ausschließlich dazu den Dialog zu öffnen und zu schließen, wobei zwei zusätzliche Funktionalitäten das Erstellen und das Entfernen von Links sind. Alles, was während oder nach dem Dialog geschieht, wird vom Entwickler selbst bestimmt.

3.7 Barrierefreie Toolbar

3.7.1 Bedienbarkeit und Navigation

Damit eine Toolbar für den Benutzer so barrierefrei wie möglich sein soll, gibt es einige bewährte Verfahrensweisen, die vom W3 Consortium empfohlen werden (W3C 2019, vgl. [35], 10.04.2021).

Tastaturanforderung

- Sobald in eine Toolbar tabuliert wird, soll das HTML-Element fokussiert werden, dass zuletzt im Fokus gewesen ist. Wenn der Fokus auf anderes HTML-Element in der Toolbar gelegt wird, soll diese Änderung mit dem HTML-Attribut `tabindex` gekennzeichnet sein. Das Element, das fokussiert wird oder ist, erhält den Wert 0 und alle anderen erhalten den Wert -1.
- Damit deaktivierte `<button>` Elemente zum Ausschneiden, Kopieren und Einfügen eines Textes in den Texteditor von Screenreadern weiterhin erkannt werden, sollen sie fokussierbar bleiben. Damit weiß dem Benutzer, dass er diese Funktionen zu einem späteren Zeitpunkt nutzen kann.

- Innerhalb der Toolbar soll die Navigation mit den Pfeiltasten möglich sein.

Gestaltung des Fokus

- Liegt der Fokus in der Toolbar, soll die Toolbar selbst anhand eines Rahmens hervorgehoben werden. Dies kennzeichnet dem Benutzer, dass eine Navigation mittels Pfeiltasten möglich ist.
- Wenn ein Element in der Toolbar selbst fokussiert ist, wird dieses nicht nur durch einen Rahmen hervorgehoben sondern auch durch eine andere Hintergrundfarbe.

Kennzeichnung von Buttons mit Pop-ups Gewisse Buttons, wie Textformatierungen, können über eine Kennzeichnung verfügen, die den Benutzern die Funktionalitäten der Toolbar näher beschreibt. Dazu gehören Textformatierungen, wie fettgedruckt, kursiv, unterstrichen, rechtsbündig etc.

- Wenn ein Button eine Kennzeichnung besitzt, soll diese durch einen Fokus oder wenn sich der Mauszeiger über diesem Element befindet, wird die Beschreibung aufgedeckt.
- Die Beschreibung bleibt so lange sichtbar bis der Mauszeiger oder der Fokus nicht mehr auf dem Element ist.
- Falls der Mauszeiger vom Element auf dessen Beschreibung wandert, bleibt diese aufgedeckt.

Unterstütze Funktionalitäten über die Tastatur

Taste	Funktionalität
`Tab`	• Legt den Fokus auf und entfernt ihn aus der Toolbar • Wenn die Toolbar nach dem Laden der Webseite zum ersten Mal fokussiert wird, erhält dessen erstes Element den Fokus. Ansonsten wird der Fokus auf das zuletzt fokussierte Element gelegt.
`rechte Pfeiltaste`	• Bewegt den Fokus nach rechts auf das nächste Element. • Ist das letzte Element am Ende der Toolbar fokussiert, erhält nun das erste Element den Fokus. • Wenn ein Element in einem Pop-up Menü fokussiert ist, ist die Funktionalität dieser Taste aufgehoben.
`linke Pfeiltaste`	• Bewegt den Fokus nach links auf das vorherige Element. • Ist das erste Element am Anfang der Toolbar fokussiert, erhält nun das letzte Element den Fokus. • Wie bei der rechten Pfeiltaste, ist auch diese Funktionalität der Taste aufgehoben, sobald der Fokus in einem Pop-up Menü liegt.
`Pos1`	Legt den Fokus auf das erste Element in der Toolbar
`Ende`	Legt den Fokus auf das letzte Element in der Toolbar
`Esc`	Falls die Beschreibung eines Buttons als Pop-up sichtbar ist, wird dieses wieder versteckt.

Tabelle 3.2: Unterstützte Funktionalitäten der Tastatur

Schalten zwischen Buttons
Damit eine Textformatierung auf den Text im Texteditor angewendet wird, kann mit der `Enter`-Taste oder mit der `Leertaste` zwischen den Buttons in der Toolbar gewechselt werden. Dieser ist dann entweder aktiv oder nicht aktiv.

Dropdown Menü
Bei einem Dropdown Menü gibt es verschiedene Tasten, die verwendet werden können, um das Menü zu öffnen und einen Button darin auszuwählen.

Taste	Funktionalität
`Enter`, `Leertaste`, `Pfeil nach oben`, `Pfeil nach unten`	Öffnet das Dropdown Menü und legt den Fokus auf einen Button in diesem Menü. Das ist entweder der aktuell aktive Button oder der erste Button, der darin zu finden ist.
`Pfeil nach unten`	• Fokusiert den nächsten Button im Menü • Wenn es sich um den letzten Button im Menü handelt, wird der erste Button fokusiert.
`Pfeil nach oben`	• Fokusiert den vorherigen Button im Menü • Wenn es sich um den ersten Button im Menü handelt, wird der letzte Button fokusiert.
`Escape`	Schließt das Drowdown Menü und fokusiert den Button des Menüs in der Toolbar.

Tabelle 3.3: Unterstützte Funktionalitäten der Tastatur

Links
Empfohlen wird, dass sich ein Link mit `Enter` oder der `Leertaste` öffnen lässt.

ARIA Attribute
Um eine Toolbar bestmöglich so zu gestalten, sodass diese für die meisten Menschen mit und ohne Einschränkungen zugänglich ist, können Accessible Rich Internet Applications (WAI-ARIA) Attribute zum Einsatz kommen (W3C 2006, vgl. [36], 10.09.2020). Diese sind bei den meisten Screenreadern und Browsern implementiert und helfen insbesondere bei dynamischen Benutzeroberflächen. Beispiele hierfür sind Aktualisierung von Inhalten in Echtzeit, Hinweise oder Meldungen bei Formularen oder auf der Webseite sowie die Navigation durch Webapplikationen.
Die wichtigsten ARIA Attribute für eine Toolbar sind in den folgenden Tabellen gelistet:

Toolbar

Rolle / ARIA Attribut	Beschreibung
`role='toolbar'` `<div>`	Dieses Attribut kennzeichnet das HTML-Element, das die Toolbar repräsentieren soll als solche. Der Container selbst ist nicht fokussierbar, sondern nur dessen Inhalt über das Attribut `tabindex`
`aria-label='Textformatierung'` `<div>`	Die Toolbar wird mit einem ansprechbaren Namen versehen.
`aria-controls='IDREF'` `<div>`	Dieses Attribut erhält den Wert, der auf das Textfeld verweist. Somit wird angegeben, dass dieses von der Toolbar kontrolliert wird.

`tabindex='-1'`
`<button>`,
`<div>`,
`<input type='checkbox'>`,
`<a>`
	Es wird auf alle HTML-Elemente der Toolbar angewendet, außer auf die Element, die mit `TAB` navigiert werden können.

`tabindex='0'`
`<button>`,
`<div>`,
`<input type='checkbox'>`,
`<a>`
	• Mit dem Wert 0 wird indiziert, dass sich dieses HTML-Element mit der `TAB`-Taste navigieren lässt.
	• Dieses Attribut mit diesem Wert wird nur auf ein HTML-Element in der Toolbar gesetzt.
	• Sobald eine Webseite geladen wird, erhält standardmäßig das erste Element in der Toolbar dieses Attribut.
	• Wenn der Fokus auf ein andere Toolbar-Element gelegt wird, erhält dieses den Wert 0 und das zuvor fokussierte Element den Wert -1.

`aria-pressed='true'` `<button>`	Gibt an, dass sich ein Button aktivieren und deaktivieren lässt sowie, dass die Textformatierung angewendet worden ist.
`aria-pressed='false'` `<button>`	Gibt an, dass sich ein Button aktivieren und deaktivieren lässt sowie, dass die Textformatierung nicht angewendet worden ist.
`aria-hidden='true'` `<button>`	Blendet das im zugänglichen Namen enthaltene Symbol aus
`aria-disabled='true'` `<button>`	Ein HTML-Button erhält dieses Attribute mit dem Wert `true`, wenn dessen Funktionalität nicht verfügbar werden kann.
`aria-disabled='false'` `<button>`	Ein HTML-Button erhält dieses Attribut mit dem Wert `false`, wenn dessen Funktonalität verfügbar ist.

`aria-label='Font: FONT FAMILY NAME'`
 	
`<button>`
	Definiert einen zugänglichen Namen für den `<button>` im Menü und ändert seinen Wert auf den Wert zum `aria-label` des Buttons, der im Menü ausgewählt wird.

`aria-haspopup='true'`
 	
`<button>`
	Mit diesem Attribut wird angegeben, dass sich nach dem Klick auf den HTML-Button ein Pop-up-Menü öffnet.

`aria-controls='IDREF'`
 	
`<button>`
	Der Wert dieses Attributes ist die Referenz auf das HTML-Element mit dem Attribut `role='menu'` und gibt an, dass sich ein Menü öffnen oder schließen lässt.

`aria-expanded='true'`
 	
`<button>`
	Wird erst dem `<button>` hinzugefügt, wenn das Menü offen ist und wieder entfernt, wenn sich das Menü wieder schließt.
	
	Gibt an, dass das Menü sichtbar ist und sich erst nach dem Aktivieren des Menü-Buttons wieder schließt.
	
	Wird verwendet, damit Screenreader auf Touch-Geräten den Benutzern mitteilen kann, dass der HTML-Button berührt werden kann, wenn das Menü offen ist. Tastaturbenutzer können diesen Button hingegen nicht fokussieren, wenn das Menü offen ist.

`role='menu'`
 	
`<ul>`
	Identifiziert das Menü, das weitere HTML-Buttons enthält.

`aria-label='Font Family'` `<ul>`	Definiert den zugänglichen Namen des Menüs
`role='menuitemradio'` `<li>`	Identifiziert das HTML-Listenelement, wobei der Textinhalt des HTML-Elementes zugleich der zugängliche Name ist.
`aria-checked='true'` `<li>`	Besitzt ein `<button>` die Rolle `menuitemradio` wird mit diesem ARIA Attribut beim `<li>` definiert, dass das ein HTML-Element ausgewählt ist und kann nur auf ein HTML-Element im jeweiligen Menü angewendet werden.
`aria-checked='true'` `<li>`	Besitzt ein `<button>` die Rolle `menuitemradio` wird mit diesem ARIA Attribut beim `<li>` Element definiert, dass das ein HTML-Element nicht ausgewählt ist und wird auf alle HTML-Elemente im jeweiligen Menü angewendet, die nicht ausgewählt sind.
`tabindex='-1'` `<li>`	Besitzt ein `<button>` die Rolle `menuitemradio` wird mit diesem ARIA Attribut beim `<li>` Element definiert, dass dieses fokussierbar ist. Im Normalfall erfolgt dies dynamisch mit JavaScript.

Tabelle 3.4: Wichtige Rollen und ARIA Attribute für die Webzugänglichkeit

3.7.2 Workarounds

Unterstützung aller HTML-Tags
Eines der ersten Probleme, die im Rahmen dieser Diplomarbeit aufgetreten sind, ist die Tatsache, dass der `<trix-editor>` nur die HTML-Tags unterstützt, die in der Tabelle 3.5 festgelegt sind. Aus diesem Grund ist der Code für das Rendering geändert und dafür ein `TextAttribute` 3.8.2 erstellt worden, das ein Objekt zurückgibt. Im Normalfall würde der Texteditor dieses Objekt nicht erkennen, doch nach dem Ändern der Ansicht wird es das. Der überschriebene Code wird dann ausgeführt, wenn die Klasse `HTMLParser` aufgerufen wird. Dies geschieht beim Einfügen von HTML im Programm oder beim Initialisieren des Dokuments mit HTML. Die Funktion `sanitize()` aus der Klasse `HTMLSanitizer` musste ebenfalls überschrieben werden, da ansonsten nur fest

kodierte HTML-Elemente zugelassen und das JavaScript-Protokoll verweigert werden.

Unterstützung von Tabellen und horizontalen Linien
Ein HTML `<table>` Element wird von Trix nicht unterstützt. Sobald eine Tabelle eingefügt wird, wird diese in Umbrüche und Pipes umgewandelt je nach Tabellenzeile und deren Inhalt. Um dieses Problem zu lösen, werden Tabellen in Trix Extension als einen Anhang betrachtet, da Trix mit Anhängen nichts anstellt und sie so belässt wie sie sind. Bei einem `<hr>` Element gab es eine größere Herausforderung. Als erste Lösung wurde dieses HTML- Element als `BlockAttribute` 3.8.2 registriert. Es stellte sich allerdings heraus, dass leere HTML-Elemente nicht unterstützt werden, weshalb `<hr>` zu geworden `<hr></hr>` ist. Deswegen wird auch hier das `<hr>` Element als Anhang betrachtet, wobei es in ein `<div>`-Tag eingepackt in `<trix-editor>` eingefügt wird. Alle anderen Anhänge werden hingegen in ein `<p>`-Tag gepackt.

Textformatierung beim Parsen eines Textes beibehalten
Beim Parsen von Absätzen in den `<trix-editor>` werden die Inhalte in ein `<div>` Element gegeben. Es hat sich herausgestellt, dass Trix eine Funktion besitzt, mit der Seitenränder überprüft werden. Da ein `<p>`- Tag einen größeren Seitenrand hat, ist in der Erweiterung des Texteditors Trix die Funktion überschrieben worden und ist nun leer.

```
Trix.HTMLSanitizer.prototype.sanitizeElement = function () { };
Trix.HTMLParser.prototype.translateBlockElementMarginsToNewlines = function () { };
Trix.PieceView.prototype.createElement = TrixUtils.createElementUpdated();
Trix.Composition.prototype.withTargetDOMRange = TrixUtils.withTargetDOMRangeUpdated();
```

Abbildung 3.4: Überschriebene Funktionen von Trix

3.7.3 Toolbar Replacer und Toolbar Replacer Factory

Mit der Klasse `ToolbarReplacer` werden die Buttons in `<trix-toolbar>` ersetzt. Hierfür muss zunächst eine Factory erstellt werden. Erst dann kann eine `<trix-toolbar>` mit den neuen gruppierten Buttons erstellt werden.

```
let factory = TrixExtensions.ToolbarReplacerFactory.create();
```

Abbildung 3.5: Erstellen einer Factory

Nachdem die Buttons fertig definiert worden sind kann mit einem Objekt der Klasse `ToolbarReplacerFactory` die Methode `build()` aufgerufen werden, die ein Objekt der Klasse `ToolbarReplacer` zurückgibt. Es wird für jeden Button einen HTML-Button erstellt. Mit der Methode `attachToTrix()` können diese HTML-Buttons in

`<trix-toolbar>` eingefügt werden, die mit ihrem `id`-Attribut oder als HTML-Element als Parameter übergeben wird.

3.8 Button Elemente

In dieser Erweiterung des Trix Texteditors wird zwischen vier verschiedenen Arten von Buttons unterschieden, wobei die letzten beiden Buttons zusätzliche HTML-Buttons in der `<trix-toolbar>` darstellen und keine Funktionalitäten von Trix haben. Diese können mit `new TrixExtensions.<button class>({...})` erstellt werden.

- **Action Button** - `new TrixExtensions.ActionButton({...})`
- **Attribute Button** - `new TrixExtensions.AttributeButton({...})`
- **Clickable Button** - `new TrixExtensions.ClickableButton({...})`
- **Dropdown Button** - - `new TrixExtensions.DropdownButton({...})`

Da es einige Attribute gibt, die in jedem HTML-Button von Trix zusätzlich zu dem Attribut `data-trix-attribute` oder `data-trix-action` enthalten sind, wurde im Rahmen dieser Diplomarbeit eine abstrakte Basisklasse `BaseButton` erstellt. Beim Erstellen eines HTML-Buttons können die folgenden Parameter definiert werden:

```
{
   trixKey: 'd',
   content: 'Underline',
   title: 'Underline',
   ariaDescription: 'Press ctrl + d to underline the text'
   isHidden: boolean
}
```

Abbildung 3.6: Standardparameter eines `<button>` Elementes

- Mit `trixKey` wird ein Tastenkürzel festgelegt
- `content` ist das `innerHTML` des Buttons
- `title` spezifiziert das gleichnamige HTML-Attribut des Buttons
- Die Beschreibung, die in `ariaDescription` angegeben wird, kann von einem Screenreader gelesen werden und trägt zur Barrierefreiheit des Texteditors bei.
- `isHidden` ist implementiert worden, da ein Dropdown Menü aus mehreren Buttons besteht und diese für den Benutzer nicht sichtbar sein sollen, sofern der dazugehörige HTML-Button zum Einblenden des Menüs nicht geklickt worden ist.

Standardmäßig hat `isHidden` den Wert `false` und somit ist der HTML-Button in der `<trix-toolbar>` sichtbar. Soll dieser nicht mehr sichtbar sein, wird der Wert auf `true` gesetzt.

Der Texteditor Trix hat bereits einige `data-trix-attribute` und `data-trix-action` vordefiniert. Mit Hilfe von Trix Extension kann über die Klasse `StandardButtonManager` folgenderweise auf diese zugegriffen und optional der Wert verändert werden:

```
let boldButton = TrixExtensions.BaseButton.standard.bold();

// statt STRG + B wird nun STRG + K als Tastenkürzel verwendet
let boldButton = TrixExtensions.BaseButton.standard.bold({ trixKey: 'k' });
```

Abbildung 3.7: Beispiel zum Erstellen eines Standard-Attribut-Buttons von Trix für einen fettgedruckten Text

In der folgenden Tabelle sind alle vordefinierten Trix Attribute und Actions sowie alle HTML-Elemente, in denen der Text gewrappt wird oder Aktionen, die auf diesen ausgeführt werden, gelistet:

Trix Attribut/Action	HTML-Element	Beschreibung
`bold`	`<strong>`	fettgedruckter Text
`italic`	`<em>`	kursiver Text
`strike`	`<del>`	durchgestrichener Text
`href`	`<a>`	`<button>`, der ein Dialog für die Eingabe eines Links öffnet
`heading1`	`<h1>`	Überschrift der Ebene 1
`quote`	`<blockquote>`	Hervorheben eines Zitates
`code`	`<pre>`	Hervorheben eines Codeblocks
`bullet`	`<ul>`	Liste mit Aufzählungspunkten
`number`	`<ol>`	nummerierte Liste
`decreaseNestingLevel`		Einrückung nach links
`increaseNestingLevel`		Einrückung nach rechts
`attachFiles`		Einfügen von Dateien, z. B. Bilder, Dokumente etc.
`undo`		Formatierung oder Text rückgängig machen
`redo`		Formatierung oder Text wiederherstellen

Tabelle 3.5: In Trix vordefinierte Buttons

3.8.1 Action Button

Bei einem Klick auf einen `<button>` mit dem Attribut `data-trix-action` wird eine bestimmte Aktion ausgeführt, die bereits von Trix definiert worden ist. Aus diesem Grund kann keine weitere Aktion von dem Benutzer für einen Action Button erstellt werden. Beim Klick auf diesen HTML-Button mit einer selbstdefinierten `trixAction` würde die Aktion nicht funktionieren.
Stattdessen kann mit Trix Extension ein Clickable Button 3.8.3 erstellt werden, der übergebene Funktionen ausführt.

Parameter eines Action Buttons

```
{
   trixAction: 'undo'
   trixKey: 'u',
   content: 'Undo',
   title: 'Undo',
   ariaDescription: 'Press ctrl + u to undo the changes',
   isHidden: boolean
}
```

Abbildung 3.8: Beispiel eines Action Buttons, der die Formatierung oder den Text rückgängig macht

Zusätzlich zu den Parametern, die in `BaseButton` definiert sind, gibt es `trixAction`. Hier ist der Wert einer der von Trix vordefinierten Aktionen. Der Action Button findet ausschließlich im `StandardButtonManager` Verwendung und es können keine selbstdefinierte Werte für `trixAction` festgelegt werden, da diese für den Texteditor unbekannt sind.

3.8.2 Attribute Button

Bei Trix gibt es zwei Typen zum Formatieren des Textes: `textAttributes` und `blockAttributes`. Wenn die gesamte Zeile gestaltet werden soll, werden `blockAttributes` verwendet. Dazu zählen beispielsweise Überschriften (`<h1>` bis `<h6>`), Listen (`<ol>` und `<ul>`) etc.
Wenn stattdessen nur ein bestimmter Bereich des Textes formatiert werden soll, werden `textAttributes` verwendet. Das betrifft zum Beispiel kurisven Text, Schriftgrößen etc. Diese Parameter beschreiben die standardmäßige Arbeitsweise der Button Elemente des Texteditors, die vor dem `<trix-editor>` erstellt werden.

Text Attribute
In der folgenden Abbildungen sind alle Eigenschaften für Textattribute ersichtlich:

```
{
    style: { ... },
    tagName: "tag",
    inheritable: true,
    parser: function(element) {
       return element.style.cssKey === "cssValue"
    }
}
```

Abbildung 3.9: Eigenschaften eines `textAttributes`

- In `style` kann ein beliebiger JavaScript Code zum Formatieren des Textes übergeben werden.
- Mit `tagName` wird festgelegt, in welches HTML-Tag der gestaltete Text gewrappt wird.
- `inheritable` spezifiziert, ob die Formatierung so lange verwendet werden soll bis der Attribute Button wieder deaktiviert wird (Wert: `true`) oder nur für ein Zeichen gilt (Wert: `false`).
- Der `parser` muss nur gesetzt werden, wenn die Eigenschaft `style` ebenfalls festgelegt wurde. Dadurch behält ein Text, der mit dieser Formatierung kopiert und eingefügt wird, diese bei.

```
Trix.config.textAttributes.myRedBackground = {
    style: {
        backgroundColor: "rgb(255, 0, 0)"
    },
    inheritable: true,
    parser: function(element) {
       return element.style.backgroundColor === "rgb(255, 0, 0)"
    }
}
```

Abbildung 3.10: Beispiel eines `textAttributes`, der den Text im `<trix-editor>` einen roten Hintergrund gibt

Block Attribute
Beim Texteditor Trix wird der Text immer in Blöcke gewrappt. Wenn ein dreizeiliger Text verfasst wurde, bedeutet das somit, dass diese drei Zeilen einen gemeinsamen Block bilden, der nach Belieben formatiert werden kann. Um ein Blockattribut anzuwenden, muss nicht unbedingt die gesamte Zeile markiert werden. Die Formatierung wird automatisch auf die Zeile angewendet, in dem sich der Cursor befindet.
Angenommen aus der ersten Zeile soll eine Überschrift werden. Durch das Styling wird der Block mit den drei Zeilen nun in einen Block mit dem HTML-Heading-Tag und einen Block mit den anderen restlichen zwei Zeilen getrennt.
Folgende Parameter können für Blockattribute festgelegt werden:

```
{
        tagName: "div",
        parse: false,
        nestable: true,
        terminal: true,
        breakOnReturn: true,
        group: false,
        text: {
                plaintext: true
        },
        listAttribute: "bulletList",
        exclusive: true
}
```

Abbildung 3.11: Eigenschaften eines `blockAttributes`

- Wie bei `textAttributes` kann bei der Eigenschaft `tagName` ein gültiges HTML-Tag festgelegt werden, in dem der Block gewrappt werden soll.
- Bei `parse` kann mit den Werten `true` oder `false` oder angegeben werden, ob ein kopierter Text dieselbe Formatierung des Blocks beibehält oder nicht.
- Mit `nestable` kann der Text mit dem Blockattribut in ein anderes HTML-Tag verschachtelt werden.
- `terminal` gibt an, ob dieses Blockattribut mit anderen kombiniert werden kann. Ist der Wert `true`, können keine anderen Blockattribute angewendet werden und andere Attribute Buttons mit `blockAttributes` sind deaktiviert.
- Soll die Formatierung nach einmal tätigen der Enter-Taste abgebrochen werden, muss `breakOnReturn` auf `true` gesetzt werden. Ansonsten müsste die Enter-Taste zweimal getätigt werden.
- Damit das HTML-Element in ein anderes HTML-Tag als Teil einer Liste gewrappt wird, kann bei `listAttribute` der Name eines anderen Blockattributes angegeben werden.

```
Trix.config.blockAttributes.myHeading2 = {
    tagName: "h2",
    terminal: true,
    breakOnReturn: true,
    group: false
}
```

Abbildung 3.12: Beispiel eines `blockAttributes`, der den Text als Überschrift im HTML-Heading-Tag `<h2>` wrappt

Parameter eines Attribute Buttons

```
{
   trixAttribute: 'bold',
   trixKey: 'b',
   content: 'Bold',
   title: 'Bold',
   ariaDescription: 'Press ctrl + b for a bold text'
   isHidden: boolean
   dialog: element
}
```

Abbildung 3.13: Eigenschaften eines Attribute Buttons

Hier gibt es zusätzlich zu den Parametern, die in `BaseButton` definiert sind, die Eigenschaften `trixAttribute` und `dialog`.
`trixAttribute` kann entweder ein von Trix bereitgestelltes oder ein selbstdefiniertes `textAttribute` oder `blockAttribute` sein.
Soll der Benutzer nach dem Klick auf einen `<button>` mit einem Pop-up-Dialog interagieren, kann optional zusätzlich in `dialog` noch ein Dialog in Form eines HTML-Elementes mit einer eindeutigen Identifikation mitgegeben werden.

```
<div id="my-dialog">
   <input type="text" placeholder="Enter your first name">
   <input type="text" placeholder="Enter your last name">
</div>
```

Abbildung 3.14: Beispiel eines Dialogs

```
const dialogButton = new TrixExtensions.AttributeButton({
    trixAttribute: 'dialog-group',
    content: 'My Dialog',
    dialog: document.getElementById('my-dialog')
});
```

Abbildung 3.15: Beispiel eines Attribute Buttons mit einem Dialog mit Eingabefeldern

Wenn ein Dialog mitgegeben wird, wird das spezifizierte HTML-Element durch Trix Extension in ein `<div>` gewrappt, welches dann Zugriff auf die `delegate` Klasse `DialogDelegate` hat.
Folgende Funktionen können bei einem Dialog ausgeführt werden:

- `close(dialog)` schließt das zur Verfügung gestellte Dialog.
- `link(href)` erstellt einen Link mit dem selektierten Text als String-Parameter. Falls nichts selektiert wird, wird an der Stelle des Cursors der Link als Text eingefügt.
- `unlink()` entfernt den Link vom selektierten Text, falls möglich.

3.8.3 Clickable Button

Wie im Abschnitt 3.8.1 bereits erwähnt worden ist, können keine weiteren Action Buttons erstellt werden, da der Texteditor Trix diese nicht erkennen kann. Abhilfe hierfür, um dennoch benutzerdefinierte Funktionen, wie Kopieren, Ausschneiden etc., ausführen zu können, bietet der Clickable Button von Trix Extension.

```
function myCopyFunction() {
   document.execCommand('copy');
}

const button = new TrixExtensions.ClickableButton({
   onMousedown: () => myCopyFunction(),
   trixKey: 'c',
   content: 'Copy',
   title: 'Copy',
   ariaDescription: 'Press ctrl + c for copying the text'
   isHidden: false
}
```

Abbildung 3.16: Beispiel eines Clickable Buttons, der den selektierten Text kopiert

Neben den `BaseButton`-Eigenschaften muss beim Parameter `onMousedown` eine Funktion mitgegeben werden. Der zu der `<trix-toolbar>` hinzugefügte HTML-Button hört nun auf das JavaScript Event `mousedown`. Sobald der `<button>` geklickt worden ist, wird das JavaScript Event ausgelöst und die im Parameter übergebene Funktion ausgeführt.

3.8.4 Dropdown Button

Mit Hilfe des Dropdown Buttons wird ein `<button>` mit einem Dropdown Menü in der `<trix-toolbar>` erstellt.

```
const button = new TrixExtensions.DropdownButton({
   identifier: 'text-styles',
   buttonGroups: [{
       ...
   }],
   content: 'Text Styles',
   title: 'Text Styles',
   ariaDescription: 'Press ctrl + c for copying the text',
   isHidden: false
})
```

Abbildung 3.17: Beispiel eines Dropdown Buttons

Neben den Parametern aus `BaseButton` wird im Parameter `identifier` eine eindeutige Identifikation für den in `<trix-toolbar>` hinzugefügten `<button>` festgelegt.
Im Parameter `buttonGroups`, der ein Array ist, werden Buttons definiert, die jeweils als `<button>` im Dropdown Menü in einem ungeordneten HTML-Listen-Element platziert werden. Wird der Dropdown Button geklickt, wird das Menü eingeblendet.

3.8.5 Gruppierung der Buttons

Die einzelnen `<button>` Elemente in der `<trix-toolbar>` müssen gruppiert werden, ansonsten gibt es einen Error. Hierfür muss zuvor eine Factory erstellt werden, wie im Abschnitt 3.7.3. Mit dieser können die HTML-Buttons in einem `<span>` von anderen Gruppen getrennt werden.
Eine `ButtonGroup` die folgenden Parameter:

```
{
   name: 'My Button Group',
   ariaLabel: 'My Button Group',
   buttons: []
}
```

Abbildung 3.18: Parameter einer Button Group

- `name` kennzeichnet die eindeutige Bezeichnung der Button Group.
- Optional kann ein `ariaLabel` mitgegeben werden. Der `<span>` erhält dann das Attribut `aria-label`, um zusätzliche Barrierefreiheit bereitzustellen.
- Im Array `buttons` können mehrere Buttons definiert werden, die gemeinsam eine Gruppe in der `<trix-toolbar>` bilden.

Es gibt zwei Arten wie eine Button Group erstellt werden kann:

- 1. Die Gruppe wird direkt mit der Factory erstellt und zur `<trix-toolbar>` hinzugefügt.
- 2. Die Gruppe wird separat als Objekt erstellt und erst später der Factory übergeben und zur `<trix-toolbar>` hinzugefügt.

```
// Button Group mit der Factory erstellen
factory
   .createButtonGroup({
      name: 'Formatting Tools',
      ariaLabel: 'Formatting',
      buttons: [
         new TrixExtensions.AttributeButton({
             trixAttribute: 'my-attribute'
         })
      ]
   });

// Button Group nachträglich erstellen
let buttonGroup = new TrixExtensions.ButtonGroup({
    name: 'Formatting Tools',
    ariaLabel: 'Formatting',
    buttons: [
        new TrixExtensions.AttributeButton({
            trixAttribute: 'my-attribute'
        })
    ]
});

factory
    .createButtonGroup(buttonGroup);
```

Abbildung 3.19: Beispiel zum Erstellen einer Button Group

Kapitel 4

Resümee

4.1 Halil Bahar

Mit voller Überzeugung und Motivation habe ich mein Praktikum in der Firma Fabasoft begonnen. Seit dem ersten Tag an, durfte ich viele tolle und hilfsbereite Kolleginnen und Kollegen kennenlernen, die uns bis zum Ende des Praktikums begleitet und unterstützt haben.
Am Anfangs des Praktikums lag der Fokus nicht am Programmieren, sondern auf der Vorbereitung für die nächsten Wochen. Wir durften an einem Workshop über Web Accessibility teilnehmen, was mir große Freude bereitet hat.

Anschließend hat das Programmieren begonnen. In den Anfangszeiten hatten wir öfters Probleme, was die Architektur anbelangt. Wir haben lange und detailliert darüber diskutiert, welche weiteren Schritte notwendig sind. Durch diese Probleme konnte ich viele Ideen und Lösungswegen mit meinem Team besprechen und austauschen. Dieser Lernprozess hat mir besonders gut gefallen.
Besonders stolz war ich darauf, dass wir unsere Software rechtzeitig fertiggestellt und die letzten Wochen damit verbracht haben, unsere Software in die Fabasoft Cloud zu integrieren.

4.2 Sonja Cao

Zuversichtlich, dass wir schnell fertig werden, haben wir uns an die Arbeitsplätze gesetzt. Allerdings hat das Einarbeiten und das Aufsetzen der passenden Entwicklungsumgebungen sowie die Sammlung wichtiger Vorkenntnisse über Web Accessibility etwa eine Woche in Anspruch genommen. Nachdem dieser Schritt geschafft wurde, hat auch die tatsächliche Arbeit begonnen.
Mit der Unterstützung des Teams der Firma Fabasoft haben wir den besten Lösungsweg ausfindig machen können, um unser Ziel zu erreichen. Mithilfe der zahlreichen Ratschläge und Feedbacks ist es möglich gewesen Probleme schnell zu beheben, falls vorhanden, und die gewünschten Anforderungen einzupflegen.

Insbesondere das Ziel, einen WYSIWYG Texteditor barrierefrei zu machen, macht die Diplomarbeit zu etwas Besonderem. Dadurch können viele Menschen, unabhängig von ihrer physischen oder psychischen Einschränkungen, den Texteditor problemlos nutzen. Der Gedanke daran, dieses Ziel erreichen zu wollen, hat mich dazu motiviert, so exakt wie möglich zu arbeiten und immer mein bestes zu geben.
Ebenso hat mir das Auseinandersetzen mit diesem umfangreichen Thema noch viel deutlicher gemacht, dass sich viele Menschen mit Behinderungen noch benachteiligt fühlen. Es gibt noch immer zahlreiche Personen, die nicht ihr volles Potenzial entfalten können. Aus diesem Grund werde ich von nun an noch deutlicher darauf achten, dass eine barrierefreie Nutzung jeglicher Webseiten, Applikationen und sonstige Dinge wichtig ist.

Literaturverzeichnis

[1] Web Accessibility Certificate Austria. WACA Zertifikate. (abgerufen am: 18.08.2020). Österreichs erstes Qualitätssiegel um Barrierefreiheit im Web nach den internationalen W3C-Richtlinien nach außen erkennbar zu machen. URL: https://waca.at/zertifikate.

[2] MAKHMALI, Javan and STEPHENSON, Sam. (2013). Trix. (abgerufen am: 09.08.2020). Trix is a rich text editor for everyday writing developed by Javan Makhmali and Sam Stephenson. URL: https://github.com/basecamp/trix.

[3] CKSource. CKEditor 4. Smart WYSIWYG HTML editor. (abgerufen am: 06.09.2020). Fully customizable WYSIWYG HTML editor with rich text features. Enterprise-grade with 70 languages and the approval of millions. URL: https://ckeditor.com/ckeditor-4/.

[4] CKSource. CKEditor 5. Rich text editor. (abgerufen am: 06.09.2020). Easy to customize rich text editor with a powerful framework, a modular architecture, and modern features like collaborative editing. URL: https://ckeditor.com/ckeditor-5/.

[5] World Health Organization. (2011, S. 3 ff.). Understanding disability. in: World Report on Disability. (Abgerufen am: 10.04.2021). The Report focuses on measures to improve accessibility and equality of opportunity; promoting participation and inclusion; and increasing respect for the autonomy and dignity of persons with disabilities. URL: https://www.who.int/disabilities/world_report/2011/report.pdf.

[6] HAWKING, Stephen W. (2011, S. ix). Foreword. in: World Report on Disability. URL: https://www.who.int/disabilities/world_report/2011/report.pdf.

[7] RIS. Bundes-Behindertengleichstellungsgesetz. (abgerufen am: 30.08.2020). Das Bundes-Behindertengleichstellungsgesetz (BGStG) bildet eines der gesetzlichen Grundlagen für das Behindertenrecht, dessen Ziel es ist, die Diskriminierung von Menschen mit Behinderungen zu beseitigen oder zu verhindern. URL: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20004228.

[8] RIS. Behinderteneinstellungsgesetz. (abgerufen am: 30.08.2020). Das Behinderteneinstellungsgesetz (BEinstG) bildet eines der gesetzlichen Grundlagen für das Behindertenrecht und regelt die Diskrimierung in der Arbeitswelt. URL: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10008253.

[9] RIS. Bundesbehindertengesetz. (abgerufen am: 30.08.2020). Das Bundesbehindertengesetz (BBG) bildet eines der gesetzlichen Grundlagen des Behindertenrechts und regelt die Beratung, Betreuung und besondere Hilfe für Menschen mit Behinderungen. URL: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10008713.

[10] RIS. Web-Zugänglichkeits-Gesetz. (abgerufen am: 30.08.2020). Das Web-Zugänglichkeits-Gesetz (WZG) hat das Ziel Websites und mobile Anwendungen für alle Nutzerinnen und Nutzer, insbesondere für Menschen mit Behinderungen, barrierefrei zu gestalten. URL: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20010727.

[11] DIRK, Jan. (2019, 30. April). Was ist eine Versionsverwaltung?. in: IT-Talents. (abgerufen am: 15.11.2020). Die Versionsverwaltung (kurz VCS, engl. Version Control System) protokolliert Änderungen, die an einer Datei oder einer Reihe von Dateien über die Zeit hinweg getätigt wurden. URL: https://www.it-talents.de/blog/it-talents/was-ist-eine-versionsverwaltung.

[12] Atlassian. What is Git. (abgerufen am: 15.11.2020). Git is a distributed version control system. Every dev has a working copy of the code and full change history on their local machine, by Linus Torvalds URL: https://www.atlassian.com/git/tutorials/what-is-git.

[13] Atlassian. Git Merge. (abgerufen am: 09.04.2021). Merging is Git's way of putting a forked history back together again. URL: https://www.atlassian.com/git/tutorials/using-branches/git-merge.

[14] GitLab. About GitLab. (abgerufen am: 30.11.2020). GitLab is an open source end-to-end software development platform with built-in version control, issue tracking, code review, CI/CD, and more. URL: https://about.gitlab.com/what-is-gitlab/.

[15] MDN Web Docs. HTML: HyperText Markup Language. (abgerufen am: 10.04.2021). HTML is the most basic building block of the Web and defines the meaning and structure of web content. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

[16] MDN Web Docs. JavaScript. (abgerufen am: 11.02.2021). JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[17] HARBAND, Jordan and SMITH, Kevin. (2020). ECMAScript® 2020 Language Specification. (abgerufen am: 10.04.2021). ECMAScript 2020 language specification, 11th edition - ECMAScript is a programming language based on several technologies like JavaScript and JScript. URL: https://262.ecma-international.org/11.0/.

[18] MALCHER, Ferdinand, HOPPE, Johannes und KOPPENHAGEN, Danny. (2020, S. 27-49). Angular - Grundlagen, fortgeschrittene Themen und Best Practices. Teil II: TypeScript. TypeScript ist eine Obermenge von JavaScript, greift die aktuellen ECMAScript-Standards auf und integriert zusätzliche Features

[19] TypeScript. Intro to the TSConfig Reference. (abgerufen am: 11.04.2021). A TSConfig file in a directory indicates that the directory is the root of a TypeScript or JavaScript project. URL: https://www.typescriptlang.org/tsconfig.

[20] OpenJS Foundation. About Node.js®. (abgerufen am: 10.04.2021). Node.js is a JavaScript runtime environment built on Chrome's V8 JavaScript engine and executes JavaScript code outside a web browser. URL: https://nodejs.org/en/about/.

[21] caustik's blog. (2012, 19. August). Node.js w/1M concurrent connections!. (abgerufen am: 13.02.2021). Using a swarm of 500 Amazon EC2 test clients, each establishing 2000 active connections to a single 15GB cloud server. URL: https://blog.caustik.com/2012/08/19/node-js-w1m-concurrent-connections/.

[22] Deno. Deno Manual. (abgerufen am: 13.02.2020). Deno is a simple, modern and secure runtime for JavaScript and TypeScript that uses V8 and is built in Rust. URL: https://deno.land/manual.

[23] About npm. in: npm Docs. (abgerufen am: 12.02.2020). Node Package Manager (npm) is the default package manager for the JavaScript runtime Node.js and consists of a CLI tool and an online repository to host JavaScript packages. URL: https://docs.npmjs.com/about-npm.

[24] registry. The JavaScript Package Registry. in: npm Docs. (abgerufen am: 09.04.2021). To resolve packages by name and version, npm talks to a registry website that implements the CommonJS Package Registry specification for reading package info. URL: https://docs.npmjs.com/cli/v7/using-npm/registry.

[25] Verdaccio. (abgerufen am: 09.04.2021). Verdaccio is a lightweight private proxy registry build in Node.js. URL: https://github.com/verdaccio/verdaccio.

[26] JFrog Ltd. JFrog Artifactory. (abgerufen am: 09.04.2021). Artifactory provides compelling features including, HA, P2 repository, build server integration, NuGet support, repo replication. URL: https://jfrog.com/artifactory/features/.

[27] Facebook. Jest. (abgerufen am: 11.04.2021). Jest is a delightful JavaScript Testing Framework with a focus on simplicity. URL: https://jestjs.io/.

[28] webpack. (abgerufen am: 11.02.2021). Webpack is a module bundler. It mainly bundles JavaScript files for usage in a browser and is capable of transforming, bundling, or packaging any resource or asset. URL: https://webpack.js.org/.

[29] W3C. (2018, 5. Juni). Web Content Accessibility Guidelines (WCAG) 2.1. (abgerufen am: 18.08.2020). Web Content Accessibility Guidelines (WCAG) 2.1 covers a wide range of recommendations for making Web content more accessible. URL: https://www.w3.org/TR/WCAG21/.

[30] MDN Web Docs. Web Components. (abgerufen am: 03.10.2020). Webkomponenten sind eine Gruppe von Web-Technologien, die es ermöglichen, benutzerdefinierte, wiederverwendbare HTML Elemente zu erstellen, deren Funktionalität gekapselt ist und damit vollständig getrennt von anderem Code. URL: https://developer.mozilla.org/de/docs/Web/Web_Components.

[31] Source Making. Design Patterns. (abgerufen am: 30.10.2020). In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. URL: https://sourcemaking.com/design_patterns.

[32] Source Making. Factory Method Design Pattern. (abgerufen am: 30.10.2020). Factory Method Pattern defines an interface for creating an object, but lets subclasses decide which class to instantiate. URL: https://sourcemaking.com/design_patterns/factory_method.

[33] GeeksforGeeks. (2018, 7. Juni) Delegation vs Inheritance in Java. (abgerufen am: 11.04.2021). Delegation is simply passing a duty off to someone/something else. URL: https://www.geeksforgeeks.org/delegation-vs-inheritance-java/.

[34] TU Wien. (2013, 4. Oktober) Delegation Pattern. (abgerufen am: 11.04.2021). Delegation is a technique where an object expresses certain behavior to the outside but in reality delegates responsibility for implementing that behaviour to an associated object. URL: https://javarevisited.blogspot.com/2013/06/why-favor-composition-over-inheritance-java-oops-design.html.

[35] W3C. (2019, 14. August). WAI-ARIA Authoring Practices 1.1. (abgerufen am: 10.04.2021). WAI-ARIA Authoring Practices is a guide for understanding how

to use WAI-ARIA 1.1 to create an accessible Rich Internet Application. URL: https://www.w3.org/TR/wai-aria-practices/.

[36] W3C. (2006, Dezember). WAI-ARIA. (Stand: 09.09.2020). (abgerufen am: 10.09.2020). The Accessible Rich Internet Applications Suite, defines a way to make Web content and Web applications more accessible to people with disabilities. URL: https://www.w3.org/WAI/standards-guidelines/aria/.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Arbeitsteilung

A.1 Halil Bahar

A.1.1 Praktische Arbeit

- Implementierung der Grundstruktur und weiterer Strukturen
- Qualitätssicherung
- Testen der Module

A.1.2 Theoretische Ausarbeitung

Einleitung

- Ist-Zustand
- Barrierefreiheit im Web
- Aufgabenstellung
- Zielsetzung
- Sollzustand
- Projektablauf und Produkt

Ausgewählte Technologien

- JavaScript
- TypeScript
- Node.js und npm
- Jest
- Webpack

Ausgewählte Aspekte

- Web Components
- JavaScript Events und EventListener
- MutationObserver
- Delegation
- Barrierefreie Toolbar
- Button Elemente

A.2 Sonja Cao

A.2.1 Praktische Arbeit

- Implementierung weiterer Strukturen
- Testen der Module

A.2.2 Theoretische Ausarbeitung

Einleitung

- Ausgangssituation
- Ist-Zustand
- Motivation
- Barrierefreiheit im Web
- Sollzustand
- Projektablauf und Produkt

Ausgewählte Technologien

- Git und GitLab
- Hypertext Markup Language
- TypeScript

Ausgewählte Aspekte

- WCAG 2.1
- Web Components
- Factory Method Pattern
- JavaScript Events und EventListener
- MutationObserver
- Barrierefreie Toolbar
- Button Elemente

Anhang B

Protokolle

B.1 Meetings in der Firma Fabasoft

B.1.1 Protokoll vom 08. Juli 2020

Anwesende

- **Thomas Bühringer**
- **Halil Bahar**
- **Sonja Cao**

Ort
Meetingraum

Inhalt

- Besprechung der Arbeit (Ist-Zustand, Problemstellung, Aufgabenstellung, Zielsetzung)
- Terminvereinbarung für einen Crashkurs in "Web Accessibility"

B.1.2 Protokoll vom 09. Juli 2020

Anwesende

- **Mario Batusic**
- **Halil Bahar**
- **Sonja Cao**

Ort
Büro

Inhalt

- Crashkurs in "Web Accessibility"
- Besprechung der Anforderungen an einen barrierefreien Texteditor

B.1.3 Protokoll vom 15. Juli 2020

Anwesende

- **Mario Batusic**
- **Halil Bahar**
- **Sonja Cao**

Ort
Büro

Inhalt
Mario testet die erste Implementierung zur Nutzung der Toolbar über die Tastatur und gibt sein Feedback dazu.

B.1.4 Protokoll vom 16. Juli 2020

Anwesende

- **Mario Batusic**
- **Halil Bahar**
- **Sonja Cao**

Ort
Büro

Inhalt

- Funktionalität der Pfeiltasten vertauscht (z. B. Die Pfeiltaste nach oben sollte dieselbe Funktion haben wie die Pfeiltaste nach rechts.)
- Besprechung der fehlenden HTML-Attribute für die Barrierefreiheit

B.1.5 Protokoll vom 17. Juli 2020

Anwesende

- **Mario Batusic**
- **Halil Bahar**
- **Sonja Cao**

Ort
Büro

Inhalt

- Verwendung der `Tab`-Taste sollte nicht ermöglicht werden
- HTML-Attribut `alt` für `<img>`-Tags einpflegen

B.1.6 Protokoll vom 17. Juli 2020

Anwesende

- **Thomas Bühringer**
- **Halil Bahar**
- **Sonja Cao**

Ort
Meetingraum

Inhalt
Besprechung der geplanten weiteren Vorgehensweise:

1. vorhandene Tools des Texteditors barrierefrei machen
2. Überlegung, welche Buttons implementiert werden sollen und wie
3. Pull Request machen, nachdem alle Funktionalitäten implementiert worden sind

B.1.7 Protokoll vom 21. Juli 2020

Anwesende

- **Mario Batusic**
- **Halil Bahar**
- **Sonja Cao**

Ort
Büro

Inhalt
Tools des CKEditors sollten in Trix ebenfalls ermöglich werden

B.1.8 Protokoll vom 23. Juli 2020

Anwesende

- **Thomas Bühringer**
- **Mateusz Szostak**
- **Halil Bahar**
- **Sonja Cao**

Ort
Büro

Inhalt

- Planung zur Erstellung der eigenen Erweiterung, da keine Rückmeldung von den Entwicklern von Trix gekommen ist
- Besprechung der Schwierigkeiten des Event Handling von Trix
- nächstes Ziel ist die Erweiterung des DOMs mit weiteren HTML-Tags
- Besprechung welche Tools in die Toolbar eingepflegt werden sollen
- Dialogfenster sollten erstellt werden können

B.1.9 Protokoll vom 29. Juli 2020

Anwesende

- **Mario Batusic**
- **Halil Bahar**
- **Sonja Cao**

Ort
Büro

Inhalt
Besprechung welche Funktionalitäten nun möglich sind und welche fehlen

B.1.10 Protokoll vom 29. Juli 2020

Anwesende

- **Thomas Bühringer**
- **Mateusz Szostak**
- **Halil Bahar**
- **Sonja Cao**

Ort
Büro

Inhalt

- Besprechung der bisherigen Implementierungen
- Besprechung der weiteren Vorgehensweise:
    1. Darstellung von Tabellen ermöglichen
    2. Erstellung von Dialogfenstern mit Buttons, um zurück zur Toolbar oder zum Editor zu gelangen
    3. Beispiel einer Toolbar bereitstellen

B.1.11 Meetings im Zeitraum von 03. August bis 28. August 2020

Anwesende

- **Thomas Bühringer**
- **Mateusz Szostak**
- **Halil Bahar**
- **Sonja Cao**

Ort
Meetingraum

Inhalt

- Besprechung der Anforderungen an die Funktionalitäten zur Implementierung des Texteditors in der Fabasoft Cloud
- Bugfixing
- Korrekturen zur richtigen Darstellung der Textformatierungen
- Darstellung aller HTML-Tags ermöglichen

B.2 Meetings mit Prof. Thomas Stütz

B.2.1 Protokoll vom 24. Juli 2020

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung des Projektumfangs und der verwendeten Technologien

B.2.2 Protokoll vom 03. August 2020

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung über die Erweiterung von Trix:

- Struktur des Projekts
- Verwendete Programmiersprachen sind JavaScript und TypeScript
- Bereitstellung der Barrierefreiheit

Auftrag bis zum nächsten Meeting
Erstellung einer Grafik für einen Überblick über die Schnittstellen und Beschreibung der notwendigen JavaScript Events.

B.2.3 Protokoll vom 11. August 2020

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung der ersten schriftliche Ausarbeitung der Diplomarbeit:

- Bezeichnung der Kapitel
- Verbesserungsmöglichkeiten zur besseren Verständlichkeit der Arbeit

Auftrag bis zum nächsten Meeting
Verbesserung der aktuellen und Erstellung neuer Diagramme und Grafiken (Überblick über die Schnittstellen, DOM als Baumstruktur, Aussehen des Texteditors Trix) zum besseren Verständnis der Arbeit und Beginn der schriftlichen Ausarbeitung des Kapitel *Einleitung*.

B.2.4 Protokoll vom 31. August 2020

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung der schriftlichen Ausarbeitung der Diplomarbeit:

- Korrigieren einzelner Rechtschreib- und Grammatikfehler
- Überlegungen über funktionale und nichtfunktionale Anforderungen

Auftrag bis zum nächsten Meeting
Beschreibung, warum der Texteditor in Fabasoft Produkten ersetzt werden muss, Verfassen kürzerer Sätze, Trennung zwischen Projektablauf und Produkt und Änderung der funktionalen und nichtfunktionalen Anforderungen.

B.2.5 Protokoll vom 07. September 2020

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung der schriftlichen Ausarbeitung der Diplomarbeit:

- Systemarchitektur mit und ohne der Erweiterung

Auftrag bis zum nächsten Meeting
Skizzieren der Systemarchitektur mit und ohne Erweiterung, Beschreibung der Zusammenhänge und Unterschiede und Verbessern der Beschreibung beim Kapitel "Funktionale Anforderungen".

B.2.6 Protokoll vom 21. September 2020

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung der schriftlichen Ausarbeitung der Diplomarbeit:

- Kleine Grammatik- und Rechtschreibfehler
- Struktur der Kapitel
- Platzierung und Beschreibung der Systemarchitektur und des Texteditors

Auftrag bis zum nächsten Meeting
Ausbessern der Fehler, Zusätzliches Kapitel für erfüllte WCAG 2.1 Kriterien, Kapitel für Web Components

B.2.7 Protokoll vom 09. November 2020

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung der schriftlichen Ausarbeitung der Diplomarbeit:

- Beschreibung der Begriffe Design Patterns und Factory Method Pattern
- Überarbeitung des Kapitels 3.5 JavaScript Events & EventListener

Auftrag bis zum nächsten Meeting
Ergänzung des Factory Method Patterns mit einem UML-Klassendiagramm

B.2.8 Protokoll vom 30. November 2020

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung der schriftlichen Ausarbeitung der Diplomarbeit:

- UML-Abbildung des Factory Method Patterns: Methode `createProduct()` muss `static` sein
- Kapitel Git: Merging beschrieben

- Definieren der Arbeitsaufteilung
- Eventuell Beispiele für WCAG: Webseiten, die die Richtlinien erfüllen und Webseiten, die diese nicht erfüllen

Auftrag bis zum nächsten Meeting
Fertigstellen aller Kapitel so weit wie möglich bis auf Kapitel Technologien

B.2.9 Protokoll vom 14. Februar 2021

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung der schriftlichen Ausarbeitung der Diplomarbeit:

- Fragen bezüglich Kapitel "Verwendete Technologien"geklärt

Auftrag bis zum nächsten Meeting
Beschreiben von JavaScript und TypeScript

B.2.10 Protokoll vom 9. April 2021

Anwesende

- **Prof. Thomas Stütz**
- **Halil Bahar**
- **Sonja Cao**

Ort
Sprachkonferenz über Discord

Inhalt
Besprechung der schriftlichen Ausarbeitung der Diplomarbeit:

- Besprechen des Kapitels "Verwendete Technologien"

Auftrag bis zum nächsten Meeting
Beschreiben von npm, TypeScript und Webpack, Kapitel “Ausgewählte Aspekte“ergänzen, Zitieren von Webseiten und Büchern