



Barrierefreiheit in  
Frontend-Frameworks: Eine  
systematische Evaluierung anhand der  
WCAG 2.1 Standards am Beispiel von  
Vuetify

PRAXISPROJEKT

ausgearbeitet von

*Meike Jungilligens*

vorgelegt an der

TECHNISCHEN HOCHSCHULE KÖLN  
CAMPUS GUMMERSBACH

im Studiengang

MEDIENINFORMATIK

Prüfer: Prof. Christian Noss

Gummersbach, Oktober 2025

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Problemstellung . . . . .	1
1.2 Zielsetzung und Forschungsfrage . . . . .	1
1.3 Stand der Forschung . . . . .	2
1.4 Aufbau der Arbeit . . . . .	3
<b>2 Theoretischer Hintergrund</b>	<b>4</b>
2.1 Rechtliche Grundlagen der Barrierefreiheit . . . . .	4
2.2 WCAG 2.1 Richtlinien . . . . .	6
2.3 Frontend-Frameworks und Vuetify . . . . .	8
<b>3 Methodik</b>	<b>10</b>
3.1 Auswahl der Komponenten . . . . .	10
3.2 Erstellung der Bewertungstabellen . . . . .	12
3.3 Testdurchführung . . . . .	12
<b>4 Ergebnisse der Analyse</b>	<b>13</b>
4.1 Container- und Navigationskomponenten . . . . .	13
4.2 Formular-Komponenten . . . . .	14
4.3 Bildliche Komponenten . . . . .	14
4.4 Feedback-Komponenten . . . . .	15
<b>5 Transfer in die Praxis</b>	<b>16</b>
5.1 Stakeholderanalyse und Anforderungen . . . . .	16
5.2 Aufbereitung der Ergebnisse . . . . .	18
<b>6 Diskussion</b>	<b>24</b>
<b>7 Fazit und Ausblick</b>	<b>26</b>
<b>A Anhang</b>	<b>29</b>
A.1 Vuetify-Komponenten nach Atomic Design . . . . .	29
A.2 Bewertungstabellen . . . . .	31
A.2.1 Container- und Navigationskomponenten . . . . .	31
A.2.2 Formular-Komponenten . . . . .	33
A.2.3 Bildliche Komponenten . . . . .	39
A.2.4 Feedback-Komponenten . . . . .	40

# 1 Einleitung

## 1.1 Problemstellung

Allein in Deutschland leben rund 7,9 Millionen Menschen mit einer Beeinträchtigung (REHADAT-Statistik, 2023), wobei diese Zahl nur Personen mit einem Grad der Behinderung von mindestens 50 erfasst. Zählt man Menschen mit leichteren Behinderungen oder Beeinträchtigungen hinzu, liegt diese Zahl deutlich höher. Diese Beeinträchtigungen wirken sich nicht selten auf die Fähigkeit, sich im Web zurechtzufinden, aus. Digitale Inhalte und Anwendungen sollen für alle Menschen zugänglich sein, und dennoch ist Barrierefreiheit im Web häufig nur unzureichend umgesetzt (Martins and Duarte, 2024). Um dem entgegenzuwirken, ist am 28. Juni 2025 das Barrierefreiheitstärkungsgesetz (BFSG) in Kraft getreten, welches einen Schwerpunkt auf die hürdenlose Gestaltung digitaler Produkte legt und somit die Barrierefreiheit von Webseiten gesetzlich verankert. Die Umsetzung der gesetzlichen Vorgaben liegt also auch in der Hand der Webentwickler.

In der Implementierung von User Interfaces (UI) im Web haben Frontend-Frameworks wie React, Vue oder Angular in den letzten Jahren an großer Bedeutung gewonnen und sind aus der modernen Webentwicklung nicht mehr wegzudenken. Ihre komponentenbasierte Architektur bietet die Möglichkeit, Projekte modular und wiederverwendbar strukturieren zu können. Zusätzlich gibt es eine Vielzahl an ergänzenden UI-Bibliotheken, die vorgefertigte Komponenten wie Buttons oder Eingabefelder bereitstellen, sodass sich die Entwickler auf wesentliche Aufgaben wie eigene Funktionen konzentrieren können. Wie gut diese Frameworks die Umsetzung von Barrierefreiheitsvorgaben von Haus aus unterstützen, ist bisher kaum untersucht worden.

## 1.2 Zielsetzung und Forschungsfrage

Ziel dieser Arbeit soll eine Erweiterung des Forschungsstandes zum Thema Barrierefreiheit in Frontend-Frameworks sein und die Erkenntnisse für Vuetify-Entwickler praxisnah aufzubereiten. Dazu wird der Stand der Barrierefreiheit anhand einer Auswahl an Komponenten analysiert. Im Anschluss an die Auswertung der Ergebnisse werden relevante Informationen und Handlungsempfehlungen abgeleitet und übersichtlich, in Form eines Posters, dargestellt.

Die erste Forschungsfrage lautet „Wie barrierefrei sind die Standardkomponenten des Vuetify-Frameworks in ihrer Grundkonfiguration gemäß den

Anforderungen der WCAG 2.1?“. Grundkonfiguration bedeutet hier ohne Modifikationen, die über die in der Vuetify-Dokumentation vorhandene Darstellung hinausgehen. Die Frage wird durch eine komponentenbasierte Analyse beantwortet und die Ergebnisse anschließend durch die Postergestaltung in die Praxis übertragen. Der Postergestaltung liegt die Frage „Welche Barrieren bestehen bei der Nutzung von Vuetify-Komponenten in Bezug auf die WCAG 2.1, und welche Maßnahmen müssen Entwickler ergreifen, um diese zu beseitigen?“ zugrunde.

### 1.3 Stand der Forschung

Die bestehende Forschung zur Barrierefreiheit ist größtenteils technologieneutral und fokussiert sich auf Testverfahren sowie allgemeine Handlungsempfehlungen in der Entwicklung. Auf dieser Grundlage entstanden bereits zahlreiche Leitfäden, die die barrierefreie Entwicklung unterstützen. Während diese Empfehlungen gut dokumentiert sind, bestehen nach wie vor Lücken bei der Erforschung der Barrierefreiheit spezifischer Frameworks. Studien behandeln vor allem Fragen wie „Wie können Vorgaben von Entwicklern umgesetzt werden?“ und weniger „Wie tragen genutzte Entwicklungs-Tools zur Erreichung von Barrierefreiheit bei?“.

React, Vue und Angular zählen zu den am weitesten verbreiteten JavaScript-basierten Frontend-Frameworks (International Journal for Research in Applied Science & Engineering Technology (IJRASET), 2022) und bilden die Grundlage zahlreicher Komponentenbibliotheken. Trotz ihrer weiten Verbreitung und der steigenden Anforderungen an Barrierefreiheit ist die Forschung in diesem Bereich bislang begrenzt. Einzig im React-Kontext sind zwei Bachelorarbeiten vorhanden. Beide betrachten eine Auswahl an Komponenten aus mehreren React-basierten Komponenten-Frameworks oder -Bibliotheken und vergleichen diese miteinander bezüglich ihrer Barrierefreiheit.

Arbeit 1 (Karlsson, 2021) untersuchte Chakra UI, React Bootstrap, Ant Design, Grommet, PrimeReact und React Suite. Untersucht und verglichen wurden bibliotheksübergreifend die Komponente Akkordeon sowie verschiedene Formularelemente (Textfelder, Checkboxen, Radiobuttons, Slider, Select). Der zugrundeliegende Standard war WCAG 2.1 AA und die Tests wurden automatisiert (Axe Lighthouse), manuell und mit dem Screenreader NVDA durchgeführt. In jeder Bibliothek hat mindestens eine Komponente nicht alle Kriterien erfüllen können, wobei es starke Unterschiede in den Testergebnissen der einzelnen Bibliotheken gab: Chakra UI, was am besten abgeschnitten hat, erfüllte nur ein Kriterium nicht, während React Suite ins-

gesamt 19 Tests nicht bestanden hat. Besonders vor dem Hintergrund, das der Großteil der getesteten Bibliotheken vorgab, WCAG-konform zu sein, ließ die Arbeit darauf schließen, dass in einem Großteil der im Frontend verwendeten Bibliotheken und Frameworks (auch über den Test-Umfang hinaus) Barrierefreiheitsprobleme vorhanden sind, die von Entwicklern nicht sofort erkannt werden.

Arbeit 2 (Lenz, 2023) betrachtete die React-Komponentenbibliotheken Material UI, Radix UI und Semantic UI. Untersucht wurden Komponenten wie Dialoge, Akkordeons und Tooltips. Die Untersuchung resultierte in einer community-basierten Plattform zur Peer-Review von Komponenten, wobei die aufgeführten Komponenten einer beispielhaften Erstanwendung der Plattform genutzt wurden. Zu beachten ist hier, dass zunächst nur die Tastaturbedienbarkeit und die Screenreader-Kompatibilität der Komponenten der Frameworks getestet wurde, nicht aber visuelle Kriterien wie Kontraste oder Zielgrößen. In der Untersuchung wies nur Semantic UI starke Schwächen in den getesteten Kriterien auf.

Detaillierte Untersuchungen zu Vue- und Angular-Komponentenbibliotheken kamen bisher nicht vor.

## 1.4 Aufbau der Arbeit

Der Hauptteil der Arbeit gliedert sich in vier Kapitel auf. Das Grundgerüst für das praktische Vorgehen bildet die theoretische Auseinandersetzung mit der Domäne. Dieser Teil dient der Herleitung der Analyse zu Grunde liegenden Bewertungskriterien sowie der ausgewählten untersuchten Technologien. Im Anschluss wird die Methodik der Analyse und dessen Durchführung erläutert. Hier finden sich der Aufbau der Untersuchung sowie die ausgewählten Testobjekte und Testwerkzeuge wieder. In dem darauffolgenden Kapitel werden die Ergebnisse der Analyse inklusive der Bewertung der einzelnen Komponenten beschrieben. Die reine Darlegung der Ergebnisse wird ergänzt durch die Einordnung und Bedeutung dieser. Die wichtigsten Erkenntnisse werden anschließend für den Praxisgebrauch aufbereitet, um ein erstes Transferprodukt aus der Analyse zu generieren. Die hier erstellten Poster werden zunächst mithilfe einer Stakeholder- und einer anschließenden Anforderungsanalyse konzipiert und inhaltlich strukturiert. Die visuelle Gestaltung erfolgt auf Basis dieser Vorarbeit und stellt das Schlusswerk dieses Projektes dar.

## **2 Theoretischer Hintergrund**

Die folgende Auseinandersetzung mit den theoretischen Hintergründen des Forschungsthemas dient dazu, dieses gesetzlich und technologisch einzubetten und bietet einen kontextuellen Rahmen für die formulierten Fragestellungen. Begonnen wird zunächst mit einer Auseinandersetzung mit den rechtlichen Grundlagen digitaler Barrierefreiheit, insbesondere der Barrierefreiheit von Webseiten, was die Aktualität und Relevanz des Themas bezeugen wird. Die gesetzliche Verankerung sorgt für ein erhöhtes Interesse von Privatunternehmen, die digitale Barrierefreiheit zeitig umzusetzen. Aus den Gesetzesgebungen wird übergegangen in die inhaltliche Grundlage dieser und ein zentrales Dokument für dieses Projekt: die Web Content Accessibility Guidelines (WCAG). Da diese die Grundlage für die im späteren Verlauf entwickelten Testkriterien sind, wird hier Kontext zur Entstehung und Bedeutsamkeit ihrer gegeben. Den Abschluss dieses Kapitels stellt die Auseinandersetzung mit der Bedeutung von Vuetify in der modernen Webentwicklung mit JavaScript-Frameworks dar, was als Herleitungsgrundlage für das Untersuchen dieses Frameworks gilt.

### **2.1 Rechtliche Grundlagen der Barrierefreiheit**

Die Entwicklung des Behindertenrechts, vor allem im digitalen Bereich, durchlief in den letzten drei Jahrzehnten mehrere Etappen auf nationaler und internationaler Ebene. 2002 wurde in Deutschland zunächst das Behindertengleichstellungsgesetz (BGG) verabschiedet, das auf dem im Grundgesetz verankerten Benachteiligungsverbot für Menschen mit Behinderungen basiert. Dieses regelte vor allem die Barrierefreiheit im öffentlichen Bereich und beinhaltete in seiner Urversion, neben Zielvorgaben zu baulichen Anlagen oder Transportmitteln, auch einen Paragraph zu barrierefreier Informationstechnik. mit der Vorgabe, Internetauftritte des Bundes barrierefrei zu gestalten (BGG, 2002, § 11). Die Barrierefreie-Informationstechnik-Verordnung (BITV), ebenfalls 2002 erlassen, diente der konkreten Umsetzung des BGG. Die BITV enthielt konkrete Vorgaben und Kriterien für das Umsetzen von Barrierefreiheit, die der damalig aktuellen WCAG 1.0 entsprachen (BITV, 2002, Anlage 1). 2011 wurde die BITV 2.0 (analog zu einer Aktualisierung des BGG) verabschiedet und bezog sich nun auf die WCAG 2.0 (BITV 2.0, 2011, Anlage 1).

Das BGG und die BITV wurden zunächst aus nationaler Initiative hin erlassen und reflektierten erst in ihren späteren Versionen zukünftig entstehende EU-Richtlinien. Diese Richtlinien müssen bis zu einem festgeleg-

ten Datum von den Mitgliedsstaaten in nationales Recht umgesetzt werden. So wurden das BGG und folglich die BITV nach Verabschiedung der EU-Richtlinie 2016/2102, die die Barrierefreiheit im öffentlichen Bereich regelt und bestehende Gesetze harmonisiert (EU 2016/2102, 2016), an diese angepasst. Seitdem führt die BITV auch keine konkreten Standards mehr auf, sondern verweist auf die „harmonisierten Normen [der EU]“ (BITV 2.0, 2021, § 3, Abs. 2). Diese Normen werden verstanden als der Inhalt der EU-Norm EN 301 549, die momentan eine Referenz auf die WCAG 2.1 beinhaltet („Accessibility requirements for ICT products and services“, 2021, Clause 9.6). Die Norm wurde anlässlich der oben genannten Richtlinie erschaffen und wird mittlerweile auch außerhalb der EU als Referenz für das zu erreichende Maß an Barrierefreiheit genutzt (wie beispielsweise in Kanada (Accessibility Standards Canada, 2024) oder Australien (Australian Human Rights Commission, 2025)).

Die drei Jahre später verabschiedete EU-Richtlinie 2019/882, auch unter der Bezeichnung „European Accessibility Act“ (EAA) bekannt, markierte einen weiteren Meilenstein auf dem Weg zur digitalen Barrierefreiheit. Ein zentrales Ziel in dieser Richtlinie ist die Miteinbeziehung des privaten Sektors in die verpflichtende Barrierefreiheit (EU 2019/882, 2019, Art. 1). Die Barrierefreiheit in der Digitalisierung der öffentlichen Hand war zu dem Zeitpunkt schon seit fast zwei Jahrzehnten Rechtsgegenstand, und erleichterte Menschen mit Behinderungen unter anderem die Nutzung von Identifikations- oder Zahlungsverfahren. Gegenstände des privaten Sektors, wie Online-Shops oder Video-on-Demand-Services, die von eigenständigen Unternehmen betrieben werden, fielen bis jetzt jedoch nicht darunter und sollen mit dieser Richtlinie barrierefreiheitstechnisch reguliert werden. Diese Zielsetzung ermöglicht Menschen mit Behinderungen eine Teilnahme am kulturellen und wirtschaftlichen Angebot des Internets über die Seiten der öffentlichen Träger hinaus. In dieser Richtlinie wird bei Einhaltung der EN 301 549 ebenfalls eine Konformität mit der EU 2019/882 vermutet (EU 2019/882, 2019, Art. 15, Abs. 1), was die EU-Norm als weiterhin geltenden Standard untermauert.

Die nationale Umsetzung der Richtlinie in Deutschland fand in Form des Barrierefreiheitsstärkungsgesetzes (BFSG) statt, welches am 16.07.2021 erlassen und am 28.06.2025 in Kraft getreten ist. Der konkrete Inhalt davon ist das Aufstellen von Barrierefreiheitsvorschriften für ab dem 28.06.2025 neu online gestellte Webseiten. Auch hier, wie in der EU-Richtlinie selbst, findet ein Bezug auf die „europäischen Standards“ als richtunggebende Leitlinien statt, was aktuell über die EN 301 549 einen weiteren Verweis auf die WCAG 2.1 darstellt.

## 2.2 WCAG 2.1 Richtlinien

Die Web Content Accessibility Guidelines (WCAG) stellen den internationalen Standard für Barrierefreiheit dar und sind Grundlage für Gesetze und Richtlinien in verschiedenen Rechtsbereichen. Sie wurden vom World Wide Web Consortium (W3C) erschaffen, dessen Gründer Tim Berners-Lee unter anderem das Hypertext Transfer Protocol (HTTP) oder die Auszeichnungssprache HTML entwickelt hat (Gesellschaft für Informatik, 2025). Das W3C übernimmt die Rolle der Standarisierung von Implementationen im Web, die die universelle Nutzbarkeit und Harmonisierung des Internets sichern (W3C, 2025e). Der Standarisierungsprozess besteht aus mehreren Iterationen von Peer-Diskussionen und Entwürfen der Dokumente, wobei die finalen Versionen von W3C selbst als Empfehlungen bezeichnet werden. Die Empfehlungen werden in einzelnen Arbeitsgruppen von den Mitgliedern erarbeitet und überprüft. 1997 schlossen sich mehrere dieser Arbeitsgruppen innerhalb des W3C zur Web Accessibility Initiative (WAI) zusammen, die sich seitdem mit der Verfassung und Aktualisierung der WCAG beschäftigt (W3C, 2025a). Zu den teilnehmenden Arbeitsgruppen gehört unter anderem auch die Gruppe Accessible Rich Internet Applications (ARIA), deren Werk eine semantische HTML-Erweiterung ist, die relevante Informationen, die sonst nur visuell erkennbar wären, für Screenreader programmatisch bereitstellt (W3C, 2025c). Die Verwendung von ARIA-Rollen in HTML spielt deswegen in der Web-Barrierefreiheit eine zentrale Rolle.

Das WCAG-Dokument ist in die vier Prinzipien der Barrierefreiheit im Web gegliedert: Wahrnehmbarkeit, Bedienbarkeit, Verständlichkeit und Robustheit (W3C, 2025d, Section 0.2: Layers of Guidance). In der englischen Sprache sind diese Prinzipien unter dem Akronym POUR (Perceivability, Operability, Understandability, Robust) bekannt. Dinge wie Untertitel bei Videos oder Alternativtexten bei Bildern sind beispielsweise dem Prinzip „Wahrnehmbarkeit“ zuzuordnen und in dem entsprechenden Kapitel reguliert. Für die Untersuchung bestehender Komponenten auf die Erfüllung der Richtlinien müssen diese also zunächst auf ihre Einschränkungsmöglichkeiten überprüft werden. Ein statisches Bild muss nicht auf seine Bedienbarkeit überprüft werden, allerdings verletzt ein Button, sei sein Kontrast nicht ausreichend oder der Fokus nicht sichtbar, schon mindestens die ersten beiden Prinzipien der Barrierefreiheit.

Jede Richtlinie in den WCAG ist einer von drei Konformitätsstufen (A, AA, AAA) zugeordnet, was die Bewertung der Barrierefreiheit einer Webseite über das reine Erfüllen einer bestimmten Anzahl von Kriterien hinaus diffe-

renziert (W3C, 2025d, Section 5.2.1: Conformance Level). Stufe A beschreibt das absolut mindeste Maß an Barrierefreiheit, das erfüllt sein muss. Darunter fallen beispielsweise das Bereitstellen von Textalternativen für Nicht-Text-Kontent oder die Möglichkeit zur Tasteaturnavigation der Webseite. Stufe AA geht darüber hinaus und umfasst unter anderem das Verwenden von ausreichenden Kontrastwerten bei Text und Nicht-Text oder das Bereitstellen eines klaren Navigationsschemas, was die Website-Bedienung für Menschen mit einer Sehbehinderung erleichtert. Das Höchstmaß an durch WCAG definierter Barrierefreiheit wird mit Stufe AAA erreicht. Die Richtlinien dieser Stufe sind nicht selten Steigerungen einer der vorherigen Stufen, oder setzen das Bereitstellen mehrerer Darstellungs- oder Navigationsalternativen voraus. Eine Webseite oder ein digitales Produkt erreicht eine bestimmte Konformitätsstufe lediglich bei Erfüllung aller zugehörigen Kriterien.

Wie auch das World Wide Web selbst befinden sich die WCAG in einer ständigen Phase der Weiterentwicklung, um dem digitalen Wandel nachkommen zu können. Die WCAG 1.0 erlangte 1999 Empfehlungsstatus, und war damit die derzeitige Referenz für das Umsetzen von Barrierefreiheit im Web (W3C, 1999). Inhaltlich fokussierte sich diese erste Version vor allem auf konkrete technische Umsetzungsstrategien in HTML und CSS, allgemeinere Hinweise zu beispielsweise Kontrasten oder Textalternativen nahmen eine untergeordnete Rolle ein. Schon hier fand sich die Klassifizierung der Kriterien nach den drei Konformitätsstufen wieder. Mit den 2008 veröffentlichten und 2009 ins Deutsche übersetzten WCAG 2.0 wurde die Vorgängerversion als internationaler Standard abgelöst (W3C, 2008). Neu in dieser Version war die erstmalige Unterteilung in die vier oben beschriebenen Prinzipien der Barrierefreiheit, in die die einzelnen Richtlinien kategorisch eingesortiert wurden. Das Dokument weist kapitelübergreifend zwölf Kriterien auf, die sich nicht signifikant von der ersten Version unterscheiden.

Die aktuellen Gesetze und Verordnungen basieren auf den 2018 veröffentlichten WCAG 2.1, da diese zur Zeit der Gesetzesgebung die aktuellste und vom W3C zur Befolgung empfohlene Version der Richtlinien war. Der Unterschied zur Vorgängerversion bestand hauptsächlich in der Miteinbeziehung von mobilen Endgeräten, Benutzern mit Sehschwächen sowie Benutzer mit kognitiven Behinderungen, die in 17 neuen Kriterien abgebildet war (W3C, 2025d, Section 0.5.1: New Features in WCAG 2.1).

Die zum Zeitpunkt der Anfertigung dieser Arbeit neueste Version sind die WCAG 2.2. Da sich die oben genannte EU-Norm allerdings auf die WCAG 2.1 bezieht und es noch keine Aktualisierung dieser gibt, wurde diese Version

der WCAG als Grundlage der Framework-Evaluation gewählt. Gleichzeitig befindet sich eine neue Hauptversion (WCAG 3.0) schon in der Umsetzung, wird aber vermutlich erst in mehreren Jahren veröffentlicht werden. Die Zwischenversionen (WCAG 2.1, WCAG 2.2) sind eine Übergangslösung (W3C, 2025d, Section 0.6: Later Versions of Accessibility Guidelines).

## 2.3 Frontend-Frameworks und Vuetify

Die rasante Entwicklung moderner Webanwendungen im Laufe der Geschichte der Webentwicklung verlangt nach Tools, die nicht nur schnelle Entwicklungszyklen erlauben, sondern auch Wartbarkeit, Wiederverwendbarkeit und Skalierbarkeit unterstützen. Diesem Anspruch werden vor allem komponentenbasierte Frameworks gerecht. Der komponentenbasierte Ansatz entspringt einem Muster der klassischen Softwarearchitektur: Teile einer Anwendung werden in in sich abgeschlossene Segmente, die „Komponenten“, unterteilt.

Insbesondere die JavaScript-basierten Frameworks React, Angular und Vue.js haben sich in den letzten Jahren als führende Technologien im Bereich der komponentenbasierten Frontend-Entwicklung etabliert (International Journal for Research in Applied Science & Engineering Technology (IJRA-SET), 2022). Das neueste dieser Frameworks ist Vue.js, welches heute zu den am schnellsten wachsenden Frameworks im Frontend-Bereich zählt. Mit über 38.000 GitHub-Stars (Vuetify, 2025c) ist Vuetify eine der am weitesten verbreiteten UI-Frameworks im Ökosystem von Vue.js. Das Projekt wurde 2016 gestartet und orientiert sich am von Google entwickelten Material Design, welches sich durch klare visuelle Prinzipien sowie einheitliche Typografie, Abstände und Farbpaletten auszeichnet. Ziel von Vuetify ist es, Entwickler:innen eine Sammlung hochwertiger, konsistenter und sofort einsetzbarer Komponenten bereitzustellen, die die Gestaltung professioneller Webanwendungen erheblich vereinfachen. Vuetify selbst wirbt damit, dass keine eigenen Design-Skills in der Frontend-Entwicklung mit dem Framework vorhanden sein müssen (Vuetify, 2024b).

Vuetify umfasst insgesamt 76 Komponenten, worunter Bausteine wie Buttons oder Icons, aber auch komplexere Strukturen wie Navigationsleisten, Tabellen oder Timelines fallen. Darüber hinaus lässt sich das Gesamterscheinungsbild durch individuell anpassbare Themes an unterschiedliche Corporate Designs anpassen. Vuetify stellt ein flexibles Grid-System und Utility Features wie vorgefertigte Animationen, Übergänge oder Direktiven bereit, weswegen es als Framework und nicht als reine Komponentenbibliothek bezeichnet wird. Abgerundet wird das Framework durch eine ausführliche Do-

kumentation, die zentrale Konzepte wie Theming und Utilities sowie auch die einzelnen Komponenten inklusive Beispiele und Code Snippets bereitstellt.

Zum Thema Barrierefreiheit befindet sich ein eigener Abschnitt in der Dokumentation. In diesem wird der Fokus auf Mechanismen zum Fokus-Management und die Möglichkeit, über sogenannte Activator Slots Interaktionen flexibel steuern zu können (Vuetify, 2025a), gelegt. Barrierefreiheit wird hier also thematisiert, der Umfang der dokumentierten Maßnahmen ist jedoch im Vergleich zu den Anforderungen der WCAG 2.1 eher begrenzt. Für Entwickler:innen stellt sich daher die Frage, inwiefern Vuetify tatsächlich eine verlässliche Grundlage für barrierefreie Webanwendungen liefert oder ob zusätzliche Anpassungen notwendig sind. Damit bildet Vuetify eine gute Grundlage für dieses Praxisprojekt: Einerseits wirbt es mit Accessibility-Features, andererseits bleibt unklar, wie konsistent und umfassend diese umgesetzt sind. Die systematische Evaluierung im Rahmen dieser Arbeit soll genau diesen Punkt klären.

## 3 Methodik

Die Methodik legt das der Analyse zugrundeliegende Vorgehen fest. Die Auswahl der Komponenten, die auf ihre Barrierefreiheit überprüft werden, wird im Vorab begrenzt, da der Umfang der Arbeit keinen Raum für eine Evaluierung aller 76 Vuetify-Komponenten bietet. Im Anschluss wird ein Bewertungsschema entwickelt, welches auf den WCAG 2.1 basiert und alle drei Konformitätsstufen reflektiert. Die so entstandenen Bewertungstabellen bilden die Basis für ein strukturiertes Testverfahren der einzelnen Komponenten. Danach wird die lokale Testumgebung geplant und aufgebaut. Benutzte Tools und Techniken sind in jenem Unterkapitel beschrieben.

### 3.1 Auswahl der Komponenten

Die Auswahl der Komponenten, die in der Analyse untersucht wurden, fand unter zwei Einschränkungen statt:

	Einschränkung	Begründung
1	Komponenten, die kritischer sind für die Barrierefreiheit, sollen vorrangig betrachtet werden.	Kritische Komponenten bestimmen die grundlegende Barrierefreiheit und Verwendbarkeit der Webseite.
2	Atomare Komponenten werden vorrangig zu komplexen/zusammengesetzten Komponenten betrachtet.	Die Barrierefreiheit von zusammengesetzten Komponenten wird maßgeblich von ihren Bestandteilen beeinflusst.

Tabelle 1: Komponentenauswahl: Einschränkungen und Begründungen

Basierend auf diesen Einschränkungen entstehen Kriterien, nach denen die Komponenten aussortiert werden können. Die erste Einschränkung lässt sich dabei auf zwei Weisen interpretieren: Einerseits werden Komponenten aussortiert, deren Hauptfunktionalität nicht im Frontend liegt, wie beispielsweise das (Nicht-)Verwenden von Server Side Rendering (SSR). Diese Komponenten haben keinen Einfluss auf die Barrierefreiheit der Komponente und der Seite an sich. Andererseits werden Komponenten nicht betrachtet, die eine rein dekorative Funktion haben oder nach WCAG als nicht störend gelten.

Die darauffolgenden Iterationen basieren auf der zweiten Einschränkung. Da die Arbeit eine Grundlage für weitere Forschungen im Bereich Barrierefreiheit in Vue und Vuetify bildet, sollen zunächst die grundlegenden Komponenten betrachtet werden. Eine entsprechende Einstufung der Komponenten nach ihrer Komplexität orientiert sich am Prinzip des Atomic Designs von Brad Frost (Frost, 2016) aufgrund der Verankerung dieser Methode in bestehenden Designkonzepten und der Nähe zur komponentenbasierten Entwicklung. Die Einteilung der Vuetify-Komponenten in die Kategorien des Atomic Designs (atom, molecule, organism, template) war nicht immer eindeutig. Teilweise überschneidet sich die Zuordnung und eine Komponente weist Merkmale zweier benachbarter Kategorien auf, besonders wenn Vuetify selbst verschiedene Erweiterungsmöglichkeiten der Komponente bereitstellt. Die finale Auswahl an zu untersuchenden Komponenten enthält somit auch solche, die gegebenenfalls nicht streng atomar, sondern auch leicht molekular aufgebaut sind. Um eine aussagekräftige Anzahl an Komponenten testen zu können, wurden solche Zwischenfälle teilweise mit in die Testgruppe aufgenommen. Die vollständige Zuordnung findet sich im Anhang (Kapitel 8). Die für die Untersuchung ausgewählten Komponenten, abgeleitet aus allen beschriebenen Einschränkungen, sind die folgenden: Button, Chip, Floating Action Button, Tooltip, Icon, Avatar, Image, Progress Circular, Progress Linear, Badge, Checkbox, Switch, Radio, Text Field, Textarea, Slider und Range Slider.

Diese Einsortierung resultierte in weiteren Überlegungen, weil Vuetify teilweise viele Erweiterungsmöglichkeiten einer einzelnen Komponente bereitstellt: Soll der vollständig mögliche Umfang einer Komponenten hier getestet werden oder nur die einfachste Implementation? Die Entscheidung fiel auf zweiteres, wiederum mit dem Argument, dass mit der Analyse erst einmal ein Überblick über Vuetify stattfindet, und hierbei eher in die Breite gegangen wird (Testen vieler Komponenten) als in die Tiefe (intensives Testen einer handvoll Komponenten). Das Ausmaß des Testens, wie später in 3.3 ausgeführt, beschränkt sich also auf die Implementation, wie Vuetify sie im obersten Teil der jeweiligen Dokumentations-Seite (Abschnitt „Usage“) dargestellt hat. Hier befindet sich eine Demonstration der Komponente und die Möglichkeit der Auswahl weniger Styling-Optionen: den Varianten.

Mit dem Vorhaben dieser Arbeit und der Limitierung der betrachteten Komponenten geht auch eine Einschränkung der relevanten WCAG-Kriterien einher. So sind zum Beispiel redaktionelle Kriterien aufgrund der technischen Ausrichtung der Analyse hier irrelevant, und Kriterien, die sich auf den Zusammenhang mehrerer Komponenten beziehen, sind nicht überprüfbar. Diese

erste Einschränkung der Kriterienliste führt zu einem effizienteren Prozess in der Erstellung der Bewertungstabellen.

### **3.2 Erstellung der Bewertungstabellen**

Der Aufbau und die Dokumentation der Analyse erfolgt pro Komponente tabellarisch. Die Spalten stellen hierbei die einzelnen Komponenten des Tests, wie Kriterium, Testwerkzeug und Ergebnis dar. Diese Darstellung reflektiert den chronologischen Ablauf der Analyse: Als erstes wird die entsprechende WCAG-Kriterie aufgeführt, da die Testkriterien exakt diesen entsprechen sollen, um ein konformes Ergebnis liefern zu können. Darauf folgt eine textuelle Beschreibung oder Zusammenfassung der Anforderungen besagter Kriterie, was die Definierung des zu erreichenden Ergebnisses abschließt.

Die darauffolgende Spalte repräsentiert die Testphase und legt fest, wie das Erfüllen der Kriterie überprüft wird. Um die Barrierefreiheit der ausgewählten Komponenten vollumfänglich evaluieren zu können, werden verschiedene Testverfahren angewendet: das automatische Testen, das manuelle Testen und das Testen mit Hilfswerkzeugen. Diese Dreiteilung wird in verschiedenen Ressourcen zu Barrierefreiheits-Tests genutzt, unter anderem auf der Webseite web.dev von Google-Entwicklern (web.dev, 2024, Artikel 17-19), die auch das Browser-Testwerkzeug Chrome Lighthouse, welches auch in der Testdurchführung (Kapitel 3.3) verwendet wird, kreiert haben. Die letzten beiden Spalten bieten Platz für das Ergebnis und eventuelle Differenzierungen über die Notizen. Das Ergebnis wird farblich analog zu einer Ampel dokumentiert: grün für bestanden, gelb für teilweise bestanden oder rot für nicht bestanden. Ein Kriterium teilweise zu bestehen bedeutet hier für eine Komponente, dass beispielsweise nur bestimmte Varianten das Kriterium bestehen konnten. Entsprechende Fälle werden in der Spalte „Weitere Notizen“ aufgeführt. Die Ampel-Darstellung hat den Vorteil, eine leichte Übersicht über die Konformität einer einzelnen Komponente zu erhalten, indem das Verhältnis zwischen grün, gelb und rot beim Anblick der Komponententabelle schnell ersichtlich wird.

### **3.3 Testdurchführung**

Die ausgewählten Komponenten werden in einem minimal aufgebauten Vuetify-Projekt implementiert. Der Projektaufbau erfolgt mit Vuetifys Scaffolding-Werkzeug create-vuetify und befolgt die offizielle Installations-Dokumentation (Vuetify, 2025b). Der minimale Aufbau soll sicherstellen, dass keine eigenen Konfigurationen die Testergebnisse verfälschen können. Auch das Theme

wurde nicht angepasst. Das Testen selbst findet im Browser statt. Dazu werden die ausgewählten Komponenten gemäß der Dokumentation implementiert, je nach Verfügbarkeit auch in ihren verschiedenen Varianten. Nach dem lokalen Starten des Projekts werden die Tests im Browser Google Chrome durchgeführt.

Einige Testkriterien, wie das Überprüfen der Kontrastwerte (u.a. WCAG-Kriterium 1.4.3) oder die Zielgröße eines Elements (WCAG-Kriterium 2.5.5) können automatisiert mit dafür entwickelten Browser-Dev-Tools getestet werden. Lighthouse zählt zu diesen Tools, und wurde basierend auf der Verbindung zu Google Chrome für dieses Projekt ausgewählt. Des Weiteren werden eine Reihe an Kriterien mithilfe des Screenreaders NVDA getestet. Das manuelle Testen wird als Ergänzung sowie als Überprüfungswerkzeug genutzt. So wird vermieden, dass Testergebnisse zum Beispiel durch fehlerhaftes Funktionieren von Lighthouse oder NVDA verfälscht werden können. Die Ergebnisse jedes Tests werden pro Komponente fortlaufend in die vorab erstellten Tabellen eingetragen.

## 4 Ergebnisse der Analyse

Die Ergebnisse der durchgeführten Analyse inklusive der ausgefüllten Bewertungstabellen werden hier aufgeführt. Der Übersichtlichkeit halber werden ähnliche untersuchte Komponenten in Kategorien zusammengefasst, die die folgenden Unterkapitel bilden. Die Zusammenfassung orientierte sich an der offiziellen Vuetify-Dokumentation, welche alle 76 Komponenten in Gruppen, jeweils nach Funktion oder Darstellungsweise sortiert, unterteilte (Vuetify, 2024a). Die ausgefüllten Ergebnistabellen befinden sich in Kapitel A.2.

### 4.1 Container- und Navigationskomponenten

Die Komponenten Button, Chip und Floating Action Button (FAB) wiesen aufgrund ihres gleichen Stylings die gleichen Defizite in den Tests zu Kontrast-Kriterien auf. Dabei waren allerdings nur bestimmte Varianten der Komponenten betroffen: Die Text-Kontrast-Kriterie in Konformitätsstufe AAA (1.4.6) bestanden die Variante „plain“ der drei Komponenten nicht, die Nicht-Text-Kontrast-Kriterie in AA (1.4.11) bestanden die Varianten „default“ und „tonal“ nicht. Die „plain“-Komponente besitzt einen Box Shadow, der sie vom Hintergrund abtrennt. Es ist nicht eindeutig durch die WCAG definiert, ob ein Schatten als sichtbare Trennung gelten kann, weswegen die Variante hier durchgefallen ist, da sie ansonsten keine klare Abhebung

vom Hintergrund aufweist. Laut WCAG ist Kriterie 1.4.11 nur erfüllt, wenn das Element entweder einen ausreichenden Kontrast zum Hintergrund oder aber eine deutlich erkennbare Grenze (beispielsweise in Form eines Rahmens) aufweist (W3C, 2025b). Ein Schatten, der verschwommen ist und eventuell an manchen Seiten des Elements weniger auftritt als an anderen, kann diese Anforderung nicht verlässlich erfüllen. Zusätzlich dazu hat der FAB-Inhalt (hier getestet mit einem Icon, ohne Text) die Kriterie 4.1.2 nicht bestanden, da der Name des Inhalts nicht erkannt werden kann.

Die Tooltip-Komponente wies bei einer Testkriterie Defizite auf. Kriterie 1.4.13 (AA) beschreibt erwünschtes Verhalten von Inhalten, die wie ein Tooltip zum Beispiel bei Überfahren einer verbundenen Komponente erscheinen. Die Tooltip-Komponente von Vuetify erfüllt diese nicht, da sie nicht verwertbar und nicht überfahrbar ist. Diese Funktionalität kann über Props der Komponente hinzugefügt werden, darauf muss aber besonderes Augenmerk von den Entwicklern selbst gelegt werden. Da diese Props manuell gesetzt werden müssen, und somit die Grundausstattung der Komponente nicht barrierefrei ist, wird Kriterie 1.4.13 als nicht bestanden bewertet.

## 4.2 Formular-Komponenten

Die Formular-Komponenten hatten ebenfalls, wie in der Tabelle ersichtlich wird, in mehreren Varianten Defizite in den Kontrast-Kriterien aufzuweisen. Der Nicht-Text-Kontrast, teilweise vom Fokus, teilweise von einzelnen Farbflächen, reichte bei keiner Komponente aus. In den beiden Textfeld-Komponenten gab es außerdem Probleme beim Text-Kontrast in den Varianten, wobei auf Konformitätsstufe AA drei von sechs und auf Konformitätsstufe AAA zwei von sechs das Kriterium bestanden. Ein anderes Problemfeld war das Kriterium Name, Rolle, Wert (4.1.2), das falsche (Checkbox) oder keine (Slider, Range Slider) Namen vorlas.

## 4.3 Bildliche Komponenten

Diese Gruppe, die aus rein darstellenden Komponenten besteht, erfüllte teilweise nicht die erforderliche Kriterie zur Bereitstellung einer Textalternativen. Typischerweise ist diese im Code in Form eines „alt“-Props zu setzen, was bei der „Icon“ und der „Avatar“-Komponente nicht möglich war. Die „Image“-Komponente verfügte zwar über ein solches Attribut, dieses ist in der Standardkonfiguration aber nicht gesetzt.

## **4.4 Feedback-Komponenten**

Die „Badge“-Komponente ist die einzige, die alle an sie gestellten Testkriterien erfüllt hat. Die beiden Fortschrittsanzeigen wiesen dahingegen Fehler bei der programmatischen Erkennung ihres Namens auf (Kriterie 4.1.2). Die Animation der Fortschrittsanzeigen kann zwar nicht gestoppt werden, was Kriterie 2.2.2 verletzt, wird aber hier als unentbehrlich gewertet, da die Pausierung der Animation eines Ladebalkens o.ä. zu Verwirrungen führen könnte. Damit ist das Kriterium als bestanden zu werten.

## 5 Transfer in die Praxis

Die visuelle Darstellung der Ergebnisse in Form eines Posters stellt das Transferprodukt der Analyseergebnisse dar und bietet eine erste praktische Anwendung für diese. Der Nutzen dieses Produkts liegt in der Bereitstellung von Hilfe und Hinweisen für Stakeholder im Umfeld der praktischen Vuetify-Entwicklung, weswegen die genauen Anforderungen an Inhalt und Gestaltung des Posters an diesen ausgerichtet werden. Es erfolgt also zunächst eine Aufstellung und Analyse der relevanten Stakeholder, woraus im Anschluss Anforderungen abgeleitet werden. Ein solches Vorgehen resultiert in einer Liste an Rahmenbedingungen für die darauffolgende Aufbereitung der Ergebnisse. Diese gliedert sich dabei in die Auswahl des Inhalts sowie in die Gestaltung des Posters. Die Inhaltsauswahl bezieht sich dabei auf das Filtern der für den Anwendungskontext relevantesten Ergebnisse, die auf dem Poster abgebildet werden. Was hierbei als „am relevantesten“ gilt, ergibt sich aus der vorangehenden Stakeholderanalyse.

### 5.1 Stakeholderanalyse und Anforderungen

Grundlage für die Definition der Stakeholder sind die in der Arbeit schon benannten drei Hauptrollen bei der Umsetzung von Barrierefreiheit auf Webseiten: Redaktion, Design und Technik. Wie auch im Rest der Arbeit wird dabei ein technischer Fokus gesetzt und die redaktionelle Komponente, also der reine Inhalt der Webseite, vernachlässigt. Da allerdings aus der Analyse Fehlerquellen hervorgehen, in denen auch Designer eine Verantwortung tragen, wird diese Rolle neben den Entwicklern als primärer Stakeholder aufgenommen. Zusätzlich dazu ist die Aufgabe des Testens von Webseiten, ob während der laufenden Entwicklung oder am fertigen Produkt, hier von Relevanz, da oft erst durch das Testen bestimmte Barrierefreiheitsfehler erkannt werden können. Die Tester wurden hier allgemein als QA betitelt, und ihre Aufgaben auf das Testen bezüglich Barrierefreiheit reduziert, da das das Teilaufgabengebiet ist, was für diese Arbeit relevant ist.

Als weitere sekundäre Stakeholder wurde außerdem die Teamleitung/der Product Owner mit aufgeführt, die unter anderem aus Gründen der Gesetzesgebung und Erweiterung der potenziellen Nutzergruppe ein grundlegendes Interesse an der Umsetzung einer barrierefreien Webseite hat. Die inhaltlichen Anforderungen werden allerdings nur von den primären Stakeholdern, also den Rollen, die an der praktischen Entwicklung und Wartung der Webseite beteiligt sind, abgeleitet. Hier ist besonderes Augenmerk darauf zu legen, dass hier lediglich von „Rollen“ im Sinne von groben Aufgabengebieten gespro-

chen wird. Bei der konkreten Teamaufteilung können Mitglieder (vor allem in kleineren Teams) auch mehrere Verantwortungen übernehmen, gleichzeitig kann aber auch eine noch detailliertere Rollenaufteilung vorliegen. Die vorliegende Arbeit abstrahiert diese Arbeitsaufteilung auf die aufgeführten Rollen.

Bezeichnung	Bezug zum System	Objektbereich	Erwartung/Erfordernis
Vuetify-Entwickler (primär)	Hauptnutzer, Poster soll bei barrierefreier Entwicklung helfen, Arbeitsinstrument	Inhalt (häufige Fehlerquellen bei Komponenten), Struktur (schnelle Erfassung der Kerninformationen)	Gute Erfassbarkeit der Informationen, konkrete Hinweise/Nennung der betroffenen Komponenten, Handlungsempfehlungen, Hilfe bei Codeverbesserung
QA-Tester (primär)	Poster als Hinweis auf Prüfkriterien bei Barrierefreiheits-Test / Aufzeigen von häufigen "Problembereichen"	Inhalt (ausformulierte, überprüfbare Kriterien), Terminologie (Bezug zu WCAG-Kriterien erkennbar), Struktur (schnelle Erfassung der Kerninformationen)	Klare, testbare Anforderungen, Zuordnung zu gängigen Testverfahren, einfache Übertragbarkeit in Testdokumentation, Fokus auf Reproduzierbarkeit
Designer (primär)	Poster als Richtlinie, gibt Orientierung bei Gestaltung barrierefreier UI	Inhalt (visuelle Aspekte wie Farbkontraste, Fokusindikatoren, Lesbarkeit), Struktur (schnelle Erfassung der Kerninformationen)	Praktische Design-Guidelines, visuelle Beispiele guter/schlechter Umsetzungen, schnelle Erkennbarkeit von Gestaltungsfehlern, Bezug zu Styleguides
Projektleiter (sekundär)	Poster als Zeichen, dass Barrierefreiheit berücksichtigt wird -> gesteigerte Qualität des Produkts	Form (Seriosität/Qualität der Informationen)	Übersichtliche Darstellung, Nachweis von Professionalität, Unterstützung für Kommunikation mit Kunden/Stakeholdern, Signal für Qualitätsstandards 

Abbildung 1: Analyse der Stakeholder für das Poster

Analog zur Anforderungsanalyse in der Softwareentwicklung werden die Anforderungen hier in funktionale (Inhalt) und nicht-funktionale (Darstellung) aufgeteilt.

Bezeichnung	Funktionale Anforderungen	Nicht-funktionale Anforderungen
Vuetify-Entwickler (primär)	<ul style="list-style-type: none"> <li>Poster muss zentrale Barrierefreiheits-Fehler in Vuetify aufzeigen inkl. Benennung betroffener Komponenten           <ul style="list-style-type: none"> <li>Poster muss konkrete Handlungsempfehlungen beinhalten</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Poster muss übersichtlich gestaltet sein:           <ul style="list-style-type: none"> <li>Reduktion auf das Wichtigste</li> <li>Zentrale Informationen müssen herausstechen (z.B. Name der betroffenen Komponente) -&gt; Erfassungshilfe</li> </ul> </li> </ul>
QA-Tester (primär)	<ul style="list-style-type: none"> <li>Poster sollte prüfbare Kriterien enthalten</li> <li>Poster muss fachlichen Bezug auf WCAG-Kriterien nehmen           <ul style="list-style-type: none"> <li>Poster sollte Testschritte enthalten</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Poster muss eine nachvollziehbare Darstellung der einzelnen Testschritte aufweisen           <ul style="list-style-type: none"> <li>Zentrale Informationen müssen herausstechen (z.B. Name der betroffenen Komponente) -&gt; Erfassungshilfe</li> </ul> </li> </ul>
Designer (primär)	<ul style="list-style-type: none"> <li>Poster sollte Gestaltungsrichtlinien für barrierefreie UI-Elemente enthalten</li> <li>Poster sollte Beispiele für „gute“ und „schlechte“ UI-Gestaltung enthalten</li> </ul>	<ul style="list-style-type: none"> <li>Poster sollte visuell ansprechend sein (klare Hierarchie, Farben, Icons)           <ul style="list-style-type: none"> <li>Zentrale Informationen müssen herausstechen (z.B. Name der betroffenen Komponente) -&gt; Erfassungshilfe</li> </ul> </li> </ul>
Projektleiter (sekundär)	[leer, da keine direkte Nutzung]	<ul style="list-style-type: none"> <li>Poster sollte auf einen Blick die Berücksichtigung von Barrierefreiheit signalisieren</li> <li>Poster sollte seriös und professionell gestaltet werden </li> </ul>

Abbildung 2: Anforderungen der Stakeholder an das Poster

Bei der Konzeptionierung des Werks aus Sicht der einzelnen primären Stakeholder kristallisierte sich bei der Anforderungsanalyse ein Interessenkonflikt heraus. Objekt dieses Konflikts war die Fokussetzung bei der inhaltlichen Ausrichtung des Posters. Beide Teams haben in ihren jeweiligen Aufgabenbereichen unterschiedliche potenzielle Hürden und Fragen bezüglich Barrierefreiheit. Für Vuetify-Entwickler erweist sich die Darstellung von Umsetzungshilfen von Barrierefreiheit im Entwicklungsprozess als geeignet, also Handlungshinweise, die mit konkreten Code-Beispielen einhergehen. Das QA-Team benötigt stattdessen praxisorientierte Hinweise zum Testen von Zugänglichkeit, wie Screenreader-, Tastatur- und Kontrastprüfungen, um typische Hürden aufzudecken. Als Designer liegt der Fokus eher bei der barrierefreien, also vor allem kontrastreichen, Entwicklung des der Webseite zugrundeliegenden Designs oder Design-Systems.

Wie bereits beschrieben ist auch zu beachten, dass Teams nicht immer streng nach dem in der Stakeholderanalyse vorgelegten Schema (Designer, Entwickler, Tester) aufgeteilt sind. Mitglieder in kleineren Teams übernehmen möglicherweise mehrere Verantwortungen und haben Interesse an mehreren Aspekten des Sicherstellens von Barrierefreiheit, was wiederum ein Argument für eine Gesamtdarstellung ist. Das erschwert die Aufteilung nach Stakeholdern in der Anforderungsanalyse, kann aber durch den Ansatz, ein Poster pro Fehlergruppe zu erstellen, gelöst werden. Somit kann jedes Poster stakeholderübergreifend gestaltet werden. Außerdem ist die Postergruppe im Falle einer fortführenden Analyse, die gegebenenfalls noch weitere häufige Fehlerquellen aufdeckt, mit diesen leicht erweiterbar.

## 5.2 Aufbereitung der Ergebnisse

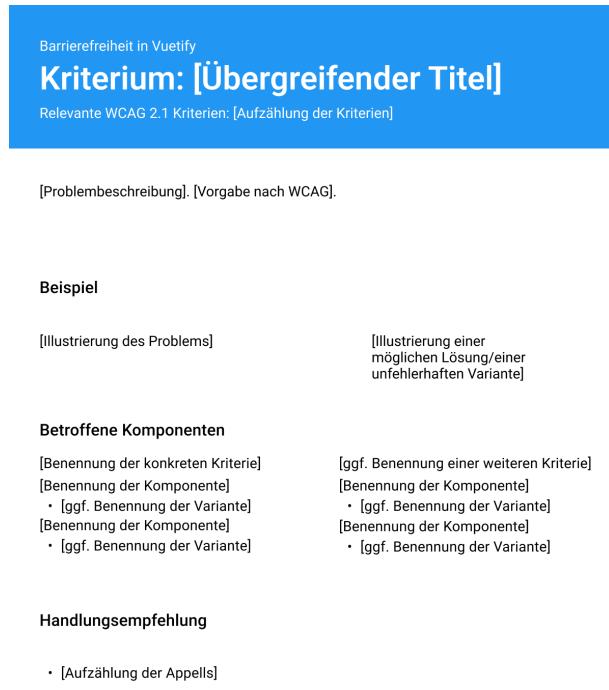
Um die aus der Usability Experience bekannte Design Gap (Wood, 1998) zwischen Anforderungsanalyse und Gestaltung zu schließen, wird tabellarisch festgehalten, welche Anforderungen wo und wie im Inhalt sowie in der Gestaltung berücksichtigt werden. Dieses Vorgehen verknüpft die Anforderungen mit konkreten Design-Entscheidungen und vereinfacht die Postergestaltung.

Anforderungen	Abgeleiteter/s Inhalt/Design
<ul style="list-style-type: none"> <li>- Bezug auf WCAG-Kriterien (Tester)</li> <li>- Zentrale Infos müssen herausstechen (Entwickler, Tester, Designer)</li> <li>- Signalwirkung: Barrierefreiheit wird berücksichtigt (PL)</li> </ul>	Kopfbereich (Titel, Meta-Titel, WCAG-Referenzen)
<ul style="list-style-type: none"> <li>- Problem + Kriterium klar formuliert (Entwickler)</li> <li>- Prüfbarkeit durch Kriterien (Tester)</li> <li>- Nachvollziehbare Darstellung (Tester)</li> </ul>	Kurze Problembeschreibung & Erwartung/Kriterium
<ul style="list-style-type: none"> <li>- Beispiele für gute/schlechte Gestaltung (Designer)           <ul style="list-style-type: none"> <li>- Unterstützung beim schnellen Erfassen (Entwickler)</li> </ul> </li> <li>- Anschaulichkeit &amp; visuelle Attraktivität (Designer, PL)</li> </ul>	Visuelle Darstellung (gutes/schlechtes Beispiel)
<ul style="list-style-type: none"> <li>- Zentrale Barrierefreiheits-Fehler inkl. Komponenten (Entwickler)</li> <li>- Zentrale Infos müssen herausstechen (Entwickler, Tester, Designer)</li> <li>- Fokussierung auf relevante Testbereiche (Tester)</li> </ul>	Aufzählung der betroffenen Vuetify-Komponenten
<ul style="list-style-type: none"> <li>- Konkrete Handlungsempfehlungen (Entwickler)           <ul style="list-style-type: none"> <li>- Testschritte enthalten (Tester)</li> </ul> </li> <li>- Gestaltungsrichtlinien für barrierefreie UI (Designer)</li> <li>- Signal: lösungsorientiert (PL)</li> </ul>	Appell / Handlungsempfehlung
<ul style="list-style-type: none"> <li>- Übersichtlichkeit, Reduktion auf das Wesentliche (Entwickler)</li> <li>- Visuell ansprechend, klare Hierarchie, Farben, Icons (Designer)</li> <li>- Seriös &amp; professionell (PL)</li> </ul>	Gesamtes Poster-Layout (Gestaltung, Branding, Struktur)  miro

Abbildung 3: Mapping zwischen Anforderungen und Inhalt/Design

Diese einzelnen Inhaltsblöcke wurden zunächst in Figma in mehreren Layout-Entwürfen verschieden angeordnet, bevor das finale Layout detaillierter ausgearbeitet wird. Um den sich wiederholenden Gestaltungsprozess der Poster einfacher zu gestalten, wurde so aus den vorher festgelegten Inhaltsblöcken zunächst eine Vorlage entwickelt, in die im Anschluss die tatsächli-

chen Inhalte eingefügt werden können. Diese Vorlage bietet außerdem ein Grundgerüst für eine mögliche Erweiterung der Posterreihe.



Technology  
Arts Sciences  
TH Köln

Abbildung 4: Verallgemeinertes Poster zur Konkretisierung der Inhaltsanordnung

Die visuelle Gestaltung orientiert sich, um die Assoziation zu Vuetify deutlich zu machen, am Material Design. So sind Schriftarten, Abstände und Größen nach den von Google vorgeschriebenen Richtlinien ihres Design Systems angelegt worden. Farblich wird das Blau des Vuetify-Logos als Akzentfarbe übernommen. Die finalen Versionen der Poster sind, zusätzlich zu der Abbildung hier, in einem separaten GitHub-Repository<sup>1</sup> zu sehen.

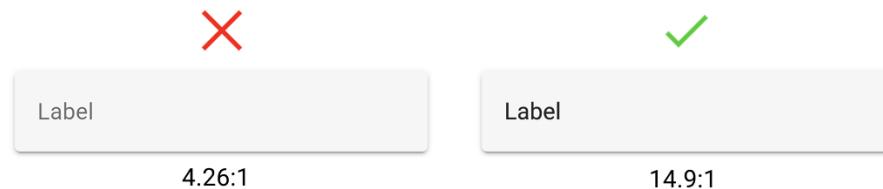
<sup>1</sup><https://github.com/mjung2605/pp2025-barrierefreiheit-vuetify>

Barrierefreiheit in Vuetify

## Kriterium: Farbkontrast

Relevante WCAG 2.1 Kriterien: 1.4.3, 1.4.11

Die Standardfarben einiger Vuetify-Komponenten erzielen kein ausreichendes Kontrastverhältnis. Das vorgegebene Kontrastverhältnis (Konformitätsstufe AA) ist **4,5:1 bei Text und 3:1 bei großem Text und Nicht-Text-Komponenten** wie Bildern, Icons oder Farbflächen.



### Betroffene Komponenten (Auswahl)

Text-Kontrast ungenügend

- Button/Chip
  - Variante: plain
- Text Field/Text Area
  - Variante: solo, solo-filled, solo-inverted

Nicht-Text-Kontrast ungenügend

- Button, Chip, FAB
  - Variante: default, tonal
- Text Field, Text Area
  - Variante: alle
- Slider, Range Slider

### Handlungsempfehlung

- **Kontrast-Check** verwenden, um Kontrastwerte zu überprüfen
- Farben/**Theme mit ausreichenden Kontrasten** wählen bzw. erstellen
- Besonders auf Nicht-Text-Elemente und **Fokus** achten



Technology  
Arts Sciences  
TH Köln

Abbildung 5: Poster zum Problemfeld Kontrast

Barrierefreiheit in Vuetify

## Kriterium: Fokus-Sichtbarkeit

Relevante WCAG 2.1 Kriterien: 2.4.7, 1.4.11

Der Tastaturfokus wird bei einigen Vuetify-Elementen gar nicht oder mit zu wenig Kontrast dargestellt. Bei bedienbaren Elementen muss der Fokus visuell gekennzeichnet sein. Der Fokus muss ein Kontrastverhältnis von 3:1 zum Hintergrund aufweisen.



### Betroffene Komponenten (Auswahl)

Fokus nicht sichtbar

- Text Field/Text Area
- Variante: solo

Fokus-Kontrast ungenügend

- Checkbox
- Switch
- Radio Button
- Slider/Range Slider

### Handlungsempfehlung

- Mit Tab-Taste die Fokussichtbarkeit überprüfen
- **Kontrast-Check** verwenden, um Kontrastwerte zu überprüfen
- Fokus mit **eigenem Styling (CSS)** ausreichend sichtbar machen



Technology  
Arts Sciences  
TH Köln

Abbildung 6: Poster zum Problemfeld Fokussichtbarkeit

Barrierefreiheit in Vuetify

## Kriterium: Name, Rolle und Wert

Relevante WCAG 2.1 Kriterien: 4.1.2

Bei Nutzung eines Screenreaders werden Informationen zu Name, Rolle oder Wert einiger Vuetify-Komponenten nicht oder unvollständig benannt. Die vollständige Bereitstellung dieser durch Software lesbaren Informationen ermöglicht Nutzer:innen von Screenreadern die Nutzung einer Webseite.



### Betroffene Komponenten (Auswahl)

- Progress Circular/Progress Linear
- Switch
- Slider/Range Slider
- FAB

### Handlungsempfehlung

- Mit Screenreader oder in den Dev Tools die Assoziation der Elemente überprüfen
- Fehlende Attribute ergänzen:  
Name mit **aria-label="..."**, Rolle mit **role="..."**, Wert mit **aria-valuenow="..."**



Technology  
Arts Sciences  
TH Köln

Abbildung 7: Poster zum Problemfeld Name, Rolle, Wert

## 6 Diskussion

Die Diskussion der in dieser Arbeit erzielten Ergebnisse unterteilt sich in zwei Sektionen: Die Analyse(-ergebnisse) und die Postererstellung. Die Analyse und ihre Ergebnisse werden von zwei verschiedenen Seiten beleuchtet.

Zunächst ist die Aussagekraft gewisser getesteter Kriterien zu hinterfragen, vor allem bei einer Übertragbarkeit der Ergebnisse in die Praxis (in zum Beispiel einem tatsächlichen Projektumfeld): Bei allen Komponenten wurden ein oder mehrere Kontrast-bezogene Tests durchgeführt. Die Grundlage hierfür war das Vuetify-eigene Theme sowie bei den bildlichen Komponenten die in Vuetifys Beispiel gebenen Bilder. In der praktischen Anwendung im Unternehmenskontext existieren allerdings häufig eigene Themes und eigenes Bildmaterial. Es ist sogar davon auszugehen, dass dies eines der ersten Dinge ist, die an den Komponenten verändert werden, auch wenn alles andere daraufhin gleich bleibt, und sei es nur wegen einer Farbanpassung an das Firmenlogo. Die entsprechenden Ergebnisse, vor allem bei Komponenten wie „Image“ oder „Icon“ haben vor diesem Hintergrund eine stark verminderte Aussagekraft. Trotz dessen wurde ihre Bewertung in die Analyse mit aufgenommen, um pro Komponente eine vollständige Bewertungstabelle zu haben. So sind die Kriterien bei einer Anwendung auf eine Komponente, auf die z.B. schon ein bestimmtes Corporate Design angewendet wurde, durchaus wichtig und aussagekräftig.

Trotz dieser Annahmen kann es interessant sein, die Vuetify-Themes und die bereitgestellten Logos einer Analyse zu unterziehen, die sich auf barrierefreie Farbpaletten konzentriert. Hier sollte jedoch deutlich Bezug auf den Ursprung im Material Design genommen werden, da Vuetify Themes und Logos daraus übernimmt. Dennoch sollten Farbkontraste, wie beispielsweise die Graustufen, die in vielen Formulkarkomponenten für Probleme sorgten, schon in den Themes des Frameworks barrierefreier gemacht werden.

Darüber hinaus wurde die vorliegende Analyse selbst mit einigen Einschränkungen angegangen. Diese halfen, die ausgewählten Komponenten gezielt zu untersuchen, führt aber in manchen Fällen auch zu einer eingeschränkten tatsächlichen Bewertbarkeit, da eventuell nicht das volle Potenzial einer Komponente ausgeschöpft werden konnte. Ein gutes Beispiel hierfür ist die „Tooltip“-Komponente: Das Kriterium 1.4.13 wurde in der getesteten Version der Komponente nicht bestanden, das Problem wäre aber durch eine zusätzlich setzbare Eigenschaft der Komponente gelöst worden. Wäre also eine tiefere Analyse jeder Komponente angemessener gewesen, die eben auch

solche Lösungsmöglichkeiten untersucht hätte? Vielleicht, allerdings wirbt Vuetify selbst damit, dass bei Verwendung des Frameworks keine weiteren eigenen Design-Skills benötigt werden. Das legt dem Nutzer unter anderem nahe, dass die Komponenten in ihrer Basis-Funktionalität schon alles was sie benötigen mitbringen, auch bezüglich Barrierefreiheit. Die Analyse hat gezeigt, dass das nicht der Fall ist.

Die Erkenntnisse aus der Analyse decken sich mit denen der in Kapitel 1.3 betrachteten Forschungen. Auch wenn sich die Komponentenauswahl selbst nur teilweise überschneidet (die Studien legten teils einen Fokus auf komplexere Komponenten), sind ähnliche Ergebnisse aufgetreten: Kontrastwerte, Vorlesen von Namen oder Werten durch Screenreader sind in dieser Analyse und in der Arbeit von A. Karlsson wiederzufinden. Die Analyse von M. Lenz wies ebenfalls wie hier Screenreader-Probleme auf. Zusätzlich dazu kam es zu Problemen mit Fokus-Sichtbarkeit und -bewegung, wobei ersteres auch in Vuetify auffiel. Die große Schnittmenge der herausgearbeiteten Problemfelder, die bibliotheks- und selbst frameworkübergreifend auftreten, unterstreicht die weite Ausbreitung dieser Fehlerquellen. Außerdem unterstützt dies die Annahme, dass die in atomaren Komponenten auftretenden Fehler sich auch auf komplexere Komponenten ausprägen. Die Hypothese für eine Untersuchung weiterer Vuetify-Komponenten ist also, dass ebensolche Probleme mit sehr hoher Wahrscheinlichkeit ebenfalls auftreten werden und somit das gesamte Framework von Barrierefreiheits-Problemen betroffen ist.

Die Erstellung der Poster diente dazu, die in der Analyse gewonnenen Ergebnisse in eine praxisnahe Form zu übertragen. Ziel war es, zentrale Probleme der Barrierefreiheit in Vuetify-Komponenten sichtbar zu machen und für diese konkrete Lösungsmöglichkeiten aufzuzeigen. Damit adressieren die Poster weniger eine wissenschaftliche Vertiefung, sondern vielmehr eine anwendungsorientierte Vermittlung, die Entwickler:innen oder Designer:innen in einem Projektkontext direkt nutzen können. Ein Vorteil dieser Form der Aufbereitung liegt in der schnellen Erfassbarkeit: Während die Analyse sehr detailliert und auf einzelne Kriterien bezogen ist, reduzieren die Poster die Ergebnisse auf das Relevanteste. Dadurch eignen sie sich insbesondere für Teamsituationen, in denen komplexe Probleme schnell kommuniziert werden müssen.

Allerdings ist die Aussagekraft der Poster ebenfalls eingeschränkt, da sie nur einen Ausschnitt der Gesamtergebnisse darstellen. Weniger prominente, aber dennoch relevante Barrieren oder Probleme auf AAA-Level bleiben somit unberücksichtigt. Zudem orientieren sich die Poster an den analysierten

Vuetify-Komponenten in ihrer Standardkonfiguration. In realen Projekten, in denen individuelle Themes, Corporate Designs oder eigene Anpassungen zum Einsatz kommen, können zusätzliche Barrieren auftreten, die in den Postern nicht thematisiert werden. Darüber hinaus basieren die Poster bewusst auf einer generischen Stakeholder-Perspektive. Eine gezieltere Ausrichtung, etwa an den spezifischen Bedürfnissen von Entwickler:innen, Designer:innen oder Entscheidungsträger:innen, könnte ihre Praxistauglichkeit erhöhen. In der vorliegenden Form sind sie als Überblick gedacht, der einen Einstieg erleichtert, aber keine vollständig differenzierte Handlungshilfe für einzelne Rollen bietet.

Insgesamt tragen die Poster dennoch wesentlich dazu bei, die Ergebnisse der Analyse zugänglicher und praxisnäher zu gestalten. Sie schlagen eine Brücke zwischen der detaillierten Bewertung der Komponenten und den konkreten Maßnahmen, die im Projektalltag relevant sind. Damit leisten sie einen Beitrag, Barrierefreiheit nicht nur theoretisch zu bewerten, sondern auch praktisch zu verankern.

## 7 Fazit und Ausblick

Die Arbeit hat gezeigt, dass die Standardkomponenten von Vuetify nicht durchgehend barrierefrei nutzbar sind. Zwar erfüllen einzelne Elemente die grundlegenden Anforderungen der WCAG, doch treten wiederholt Defizite auf, etwa bei Farbkontrasten, Fokus-Indikatoren oder der Screenreader-Nutzung. Damit widerspricht die Analyse dem Anspruch von Vuetify, ohne zusätzliche Designanpassungen einsatzbereit zu sein. Die Poster haben die zentralen Ergebnisse in eine praxisnahe Form übertragen. Sie veranschaulichen häufige Fehlerquellen, betroffene Komponenten sowie Lösungsansätze und erleichtern damit den Einstieg in die Thematik. Trotzdem können sie nur einen Überblick bieten und ersetzen keine umfassende projektspezifische Prüfung.

Zusammenfassend verdeutlicht die Arbeit, dass Barrierefreiheit auch in modernen UI-Frameworks nicht selbstverständlich ist. Sie erfordert bewusstes Prüfen, gezielte Anpassungen und eine konsequente Orientierung anhand der WCAG, die sich ebenfalls weiterentwickeln.

Die Arbeit unterlag einigen Limitierungen, die die künftige Forschung aufgreifen kann. So beschränkte sich die Analyse auf 17 grundlegende Vuetify-Komponenten in ihrer Standardkonfiguration. Eine Ausweitung auf komplexe Komponenten oder auf das Framework in seiner Gesamtheit könnte

weitere Erkenntnisse liefern. Dabei wäre insbesondere die Frage relevant, ob Barrieren durch die Komponente selbst oder durch ihre Teilkomponenten entstehen.

Außerdem fokussieren sich die Poster ausschließlich auf die Konformitätsstufe AA, da diese aktuell durch das BFSG für den privaten Sektor maßgeblich ist. Künftige Arbeiten könnten darüber hinaus die Anforderungen der Stufe AAA berücksichtigen, die zwar derzeit nur im öffentlichen Sektor verpflichtend sind, deren Relevanz aber durch die Weiterentwicklung von WCAG und den Gesetzgebungen zunehmen dürfte. Praktisch bleibt außerdem festzuhalten, dass einzelne Probleme sich zwar durch manuelle Anpassungen beheben lassen, langfristig jedoch die Framework-Entwickler selbst barrierefreie Defaults bereitstellen sollten. Nur so kann der Grundgedanke komponentenbasierter Bibliotheken (sprich: eine sofortige und mit Bezug auf die Barrierefreiheit auch rechtssichere Einsatzfähigkeit) konsequent eingelöst werden.



# A Anhang

## A.1 Vuetify-Komponenten nach Atomic Design

Containment	<ul style="list-style-type: none"><li>• Button - Atom</li><li>• Card - Molecule</li><li>• List - Molecule</li><li>• Chip - Atom</li><li>• Divider - Atom</li><li>• Expansion Panel - Molecule</li><li>• Menu - Organism</li><li>• Dialog - Molecule/Organism</li><li>• Bottom Sheet - Organism</li><li>• Overlay - keine eigenständige Komponente</li><li>• Toolbar - Organism</li><li>• Tooltip - Atom</li><li>• Sheet - Atom</li></ul>
Navigation	<ul style="list-style-type: none"><li>• App Bars - Organism</li><li>• Floating Action Button - Atom</li><li>• Navigation Drawers - Molecule/Organism</li><li>• Pagination - Molecule</li><li>• Bottom Navigation - Organism</li><li>• Breadcrumbs - Molecule</li><li>• Footer Component</li><li>• Speed Dials - Molecule</li><li>• System Bar - Organism</li><li>• Tabs - Organism</li></ul>
Form Inputs & Controls	<ul style="list-style-type: none"><li>• Autocomplete - Molecule</li><li>• Combobox - Molecule</li><li>• Text Field - Molecule</li><li>• Checkbox - Atom ?</li><li>• Switch - Atom ?</li><li>• Radio - Atom/Molecule (Radio Group)</li><li>• File Input - Molecule</li><li>• Form - Organism</li><li>• Inputs - Molecule/allgemeine Komponente</li><li>• Number Input - Molecule</li><li>• OTP Input - Molecule</li><li>• Select - Molecule</li><li>• Slider - Atom/Molecule</li><li>• Range Slider - Atom/Molecule</li><li>• Textarea - Atom/Molecule</li></ul>
Data & Display	<ul style="list-style-type: none"><li>• Confirm Edit - Molecule</li><li>• Data Iterator - Molecule/Organism</li><li>• Data Table - Molecule/Organism</li><li>• Infinite Scroll - Molecule/Organism</li><li>• Sparkline - Atom/Molecule</li><li>• Table - Molecule</li><li>• Virtual Scroll - Molecule/Organism</li></ul>

Abbildung 8: Zuordnung Vuetify-Komponenten zu den Stufen des Atomic Designs nach Brad Frost (1)

miro

Grids/Layouts	<ul style="list-style-type: none"> <li>• Grid - keine Komponente</li> </ul>
Selection	<ul style="list-style-type: none"> <li>• Carousel - Molecule/Organism, implementierungsabhängig</li> <li>• Button Toggle - Molecule</li> <li>• Item Group - Organism/sehr allgemein</li> <li>• Slide Group - Molecule/allgemein bzw. individuell</li> <li>• Button Group - Molecule</li> <li>• Chip Group - Molecule</li> <li>• Window - Molecule</li> <li>• Stepper - Molecule</li> </ul>
Feedback	<ul style="list-style-type: none"> <li>• Alert - Molecule</li> <li>• Badge - Atom (abhängig von Icon/Img)</li> <li>• Banner - Molecule</li> <li>• Empty State (404) - Organism ? (implementierungsabhängig)</li> <li>• Skeleton Loader - alles/keine Komponente</li> <li>• Snackbar - Molecule</li> <li>• Rating - Molecule?</li> <li>• Timeline - Organism/Molecule ?</li> <li>• Hover - keine Komponente</li> <li>• Progress Circular - Atom</li> <li>• Progress Linear - Atom</li> </ul>
Images & Icons	<ul style="list-style-type: none"> <li>• Aspect Ratios - Helper</li> <li>• Avatars - Atom</li> <li>• Icon - Atom</li> <li>• Image - Atom</li> <li>• Parallax - Organism</li> </ul>
Pickers	<ul style="list-style-type: none"> <li>• Color Picker - Molecule</li> <li>• Data Picker - Molecule</li> </ul>
Providers	<ul style="list-style-type: none"> <li>• Defaults Provider - keine Komponente</li> <li>• Locale Provider - keine Komponente</li> <li>• Theme Provider - keine Komponente</li> </ul>
Misc	<ul style="list-style-type: none"> <li>• Lazy - keine Komponente</li> <li>• No SSR - keine Komponente</li> </ul>

miro

Abbildung 9: Zuordnung Vuetify-Komponenten zu den Stufen des Atomic Designs nach Brad Frost (2)

## A.2 Bewertungstabellen

### A.2.1 Container- und Navigationskomponenten

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		Default: passed Outline: passed Tonal: passed Text: passed Plain: failed (6.19:1)
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		Default: failed (box shadow) Outline: passed Tonal: failed (1.32:1) Text: passed Plain: passed
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tasturfälle	Der auf das Element bewegte Tastaturookus kann auch wieder wegbelegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	tool		miro

Abbildung 10: Bewertung der Komponente Button

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		entspricht Button
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tasturfälle	Der auf das Element bewegte Tastaturookus kann auch wieder wegbelegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		Der FAB hat keinen Textinhalt, der als Name von Software erkannt werden kann. Daher muss dieser manuell im Code als ARIA-Label oder Titel gesetzt werden.

Abbildung 11: Bewertung der Komponente Floating Action Button

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		entspricht Button
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		entspricht Button
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		entspricht Button
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		nur relevant bei Elementen mit "closable"-Property
2.1.2 (A) Keine Tastaturläufe	Der auf das Element bewegte Tastaturfokus kann auch wieder wegbewegt werden	man		nur relevant bei Elementen mit "closable"-Property
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		nur relevant bei Elementen mit "closable"-Property
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		nur relevant bei Elementen mit "closable"-Property
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	tool		miro

Abbildung 12: Bewertung der Komponente Chip

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		
1.4.13 (AA) Inhalt bei Überfahren mit dem Zeiger oder Tastaturfokus	Bei Überfahren o.ä. sichtbar werdender Inhalt muss verwerfbar, überfahrbar und beständig sein.	man		default nicht überfahrbar, bei programmatischer Kontrolle des Anzeigens (im Wiki: "Always show"): durch Nutzer nicht schließbar und überdeckt ggf. andere Komponenten
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		miro

Abbildung 13: Bewertung der Komponente Tooltip

## A.2.2 Formular-Komponenten

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		Element bestanden, Fokus nicht (1,32:1)
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tastaturlafalle	Der auf das Element bewegte Tastaturfokus kann auch wieder wegbewegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		
2.5.3 (A) Beschriftung (Label) im Namen	Von Software erkannter Name entspricht dem/enthält den angezeigten Namen	tool		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		miro

Abbildung 14: Bewertung der Komponente Checkbox

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		Element bestanden, Fokus nicht (1,32:1)
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tastaturfalle	Der auf das Element bewegte Tastaturfokus kann auch wieder wegbewegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		
2.5.3 (A) Beschriftung (Label) im Namen	Von Software erkannter Name entspricht dem/enthält den angezeigten Namen	tool		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name, Rolle und Wert des Elements müssen durch Software erkennbar sein.	auto/tool		ARIA attributes are not used as specified for the element's role ABER wird vorgelesen. (ist falsch intern als Checkbox implementiert, bei der tatsächlichen Anwendung aber eher unkritisch) 

Abbildung 15: Bewertung der Komponente Switch

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		Element bestanden, Fokus nicht (1,32:1)
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tastatursäfte	Der auf das Element bewegte Tastaturfokus kann auch wieder wegbewegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		
2.5.3 (A) Beschriftung (Label) im Namen	Von Software erkannter Name entspricht dem/enthält den angezeigten Namen	tool		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name, Rolle und Wert des Elements müssen durch Software erkennbar sein.	auto/tool		miro

Abbildung 16: Bewertung der Komponente Radio

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		Default: passed Outlined: passed Underlined: passed Solo: failed (4.29:1) Solo-filled: failed (4.2:1) Solo-inverted: failed (4.29:1)
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		Default: failed (5.59:1) Outlined: passed Underlined: passed Solo: failed (4.29:1) Solo-filled: failed (4.2:1) Solo-inverted: failed (4.29:1)
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		Default: failed (2.88:1) Outlined: failed (2.67:1) Underlined: failed (2.67:1) Solo: failed (box shadow) Solo-filled: failed (1.08:1) Solo-inverted: failed (box shadow)
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tastaturfälle	Der auf das Element bewegte Tastaturfokus kann auch wieder wegbewegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		Fokus nicht sichtbar bei Varianten "solo" und "clearable solo" (bis auf Verschiebung des Labels und Anzeigen des blinkenden Cursors, das aber auch bei Textinhalt = kein eindeutiger Fokus-Bezug)
2.5.3 (A) Beschriftung (Label) im Namen	Von Software erkannter Name entspricht dem/enthält den angezeigten Namen	tool		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		miro

Abbildung 17: Bewertung der Komponente Text Field

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		Default: passed Outlined: passed Underlined: passed Solo: failed (4.29:1) Solo-filled: failed (4.2:1) Solo-inverted: failed (4.29:1)
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		Default: failed (5.59:1) Outlined: passed Underlined: passed Solo: failed (4.29:1) Solo-filled: failed (4.2:1) Solo-inverted: failed (4.29:1)
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		Default: failed (2.88:1) Outlined: failed (2.67:1) Underlined: failed (2.67:1) Solo: failed (box shadow) Solo-filled: failed (1.08:1) Solo-inverted: failed (box shadow)
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tastaturfälle	Der auf das Element bewegte Tastaturfokus kann auch wieder wegbewegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		Fokus nicht sichtbar bei Varianten "solo" und "clearable solo" (bis auf Verschiebung des Labels und Anzeigen des blinkenden Cursors, das aber auch bei Textinhalt = kein eindeutiger Fokus-Bezug)
2.5.3 (A) Beschriftung (Label) im Namen	Von Software erkannter Name entspricht dem/enthält den angezeigten Namen	tool		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		miro

Abbildung 18: Bewertung der Komponente Textarea

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		<ul style="list-style-type: none"> <li>• Thumb: passed</li> <li>• Linie: failed (2:1)</li> <li>• Fokus: failed (1.22:1)</li> </ul>
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tastaturläufe	Der auf das Element bewegte Tastaturfokus kann auch wieder wegbewegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		
2.5.1 (A) Zeigergesten	Funktionalität, die auf Multipunkt- oder Pfad-basierte Gesten angewiesen ist, muss auch mit einer einfachen Geste verfügbar sein	man		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		ARIA input fields do not have accessible names ABER: Wert wird vorgelesen Rolle, Wert vorhanden (A11y Tree)

Abbildung 19: Bewertung der Komponente Slider

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		<ul style="list-style-type: none"> <li>• Thumb: passed</li> <li>• Linie: failed (2:1)</li> <li>• Fokus: failed (1.22:1)</li> </ul>
2.1.1 (A), 2.1.3 (AAA) Tastatur	Element ist durch eine Tastatur bedienbar	man		
2.1.2 (A) Keine Tastaturläufe	Der auf das Element bewegte Tastaturfokus kann auch wieder wegbewegt werden	man		
2.4.7 (AA) Fokus sichtbar	Bei Bedienung mit Tastatur ist das Element, welches den Fokus erhält, besonders gekennzeichnet	man		
2.5.1 (A) Zeigergesten	Funktionalität, die auf Multipunkt- oder Pfad-basierte Gesten angewiesen ist, muss auch mit einer einfachen Geste verfügbar sein	man		
2.5.5 (AAA) Zielgröße	Das klickbare Element muss mindestens 44x44 CSS-Pixel groß sein.	auto		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		ARIA input fields do not have accessible names ABER: Wert wird vorgelesen Rolle, Wert vorhanden (A11y Tree)

Abbildung 20: Bewertung der Komponente Range Slider

### A.2.3 Bildliche Komponenten

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.1.1 (AA) Nicht-Text-Inhalt	Nicht-Text-Inhalte, z.B. Bilder, haben eine aussagekräftige Textalternative	auto/tool		nicht vorhanden - wegen individueller Bilderauswahl? Property generell kann gesetzt werden, ist aber in dem Docs-Code nicht
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		natürlich auch abhängig von Bild

Abbildung 21: Bewertung der Komponente Image

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.1.1 (AA) Nicht-Text-Inhalt	Nicht-Text-Inhalte, z.B. Bilder, haben eine aussagekräftige Textalternative	auto/tool		nochmal mit NVDA überprüfen
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		natürlich auch abhängig von Bild

Abbildung 22: Bewertung der Komponente Avatar

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.1.1 (AA) Nicht-Text-Inhalt	Nicht-Text-Inhalte, z.B. Bilder, haben eine aussagekräftige Textalternative	auto/tool		nochmal mit NVDA überprüfen
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		natürlich auch abhängig von Bild

Abbildung 23: Bewertung der Komponente Icon

## A.2.4 Feedback-Komponenten

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.3 (AA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 4,5:1	auto/man		relevant/prüfbar nur bei sichtbarer Nummer
1.4.6 (AAA) Kontrast	Der Kontrast zwischen Text und Hintergrund hat ein Mindestverhältnis von 7:1	auto/man		relevant/prüfbar nur bei sichtbarer Nummer
1.4.4 (AA) Textgröße ändern	Text kann ohne Inhalts- oder Funktionsverlust um bis zu 200% vergrößert werden	man		
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		miro

Abbildung 24: Bewertung der Komponente Badge

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		
2.2.2 (A) Pausieren, Beenden, Ausblenden	Pausieren, Beenden, Ausblenden ist möglich für Animationen, die automatisch beginnen und länger als 5 Sekunden dauern sowie sich automatisch aktualisierende Inhalte	man		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		"ARIA progressbar elements do not have accessible names" - werden auch nicht vorgelesen miro

Abbildung 25: Bewertung der Komponente Progress Circular

WCAG-Richtlinie	Kriterium	Testmethode	Bewertung	Anmerkung
1.4.11 (AA) Nicht-Text-Kontrast	Der Kontrast zwischen Element und Hintergrund hat ein Mindestverhältnis von 3:1	auto/man		
2.2.2 (A) Pausieren, Beenden, Ausblenden	Pausieren, Beenden, Ausblenden ist möglich für Animationen, die automatisch beginnen und länger als 5 Sekunden dauern sowie sich automatisch aktualisierende Inhalte	man		
4.1.2 (A) Name, Rolle, Wert	Name und Rolle des Elements müssen durch Software erkennbar sein.	auto/tool		"ARIA progressbar elements do not have accessible names" - werden auch nicht vorgelesen miro

Abbildung 26: Bewertung der Komponente Progress Linear

## Literaturverzeichnis

- Accessibility requirements for ict products and services: En 301 549 v3.2.1 (2021-03).* (2021). [https://www.etsi.org/deliver/etsi\\_en/301500\\_301599/301549/03.02.01\\_60/en\\_301549v030201p.pdf](https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf)
- Accessibility Standards Canada. (2024). *Accessibility standards canada adopts the globally recognized accessibility standard for ict products and services* [Abgerufen am 19.09.2025]. <https://www.canada.ca/en/accessibility-standards-canada/news/2024/05/accessibility-standards-canada-adopts-the-globally-recognized-accessibility-standard-for-ict-products-and-services.html>
- Australian Human Rights Commission. (2025). *Chapter 3 — standards and guidelines on digital accessibility* [Abgerufen am 19.09.2025]. <https://humanrights.gov.au/our-work/disability-rights/chapter-3-standards-and-guidelines-digital-accessibility>
- Behindertengleichstellungsgesetz (BGG), 2002, <https://www.gesetze-im-internet.de/bgg/BJNR146800002.html>
- Frost, B. (2016). *Atomic design methodology* [Abgerufen am 20.09.2025]. <https://atomicdesign.bradfrost.com/chapter-2>
- Gesellschaft für Informatik. (2025). *Tim berners-lee: Begründer des world wide web* [Abgerufen am 20.09.2025]. <https://gi.de/persoenlichkeiten/tim-berners-lee>
- International Journal for Research in Applied Science & Engineering Technology (IJRASET). (2022). Comparative analysis on front-end frameworks for web applications. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 10(7), 298–307. <https://doi.org/10.22214/IJRASET.2022.45260>
- Karlsson, A. (2021). *Evaluating the state of accessibility in react ui component libraries* [Bachelorarbeit]. Linné-Universität. <https://www.diva-portal.org/smash/get/diva2:1568285/FULLTEXT01.pdf>
- Lenz, M. (2023). *Plattform für crowdsourced accessibility-testing von komponenten aus web ui-libraries* [Bachelorarbeit]. Ostschweizer Fachhochschule.
- Martins, B., & Duarte, C. (2024). Large-scale study of web accessibility metrics. *Universal Access in the Information Society*, 23, 411–434. <https://doi.org/10.1007/s10209-022-00956-x>
- REHADAT-Statistik. (2023). *Statistik der schwerbehinderten menschen* [Abgerufen am 02.10.2025]. REHADAT-Statistik. [https://www.rehadat-statistik.de/statistiken/behinderung/schwerbehindertenstatistik/#headline-fa\\_bc42f664-1-1](https://www.rehadat-statistik.de/statistiken/behinderung/schwerbehindertenstatistik/#headline-fa_bc42f664-1-1)

- Richtlinie (EU) 2016/2102 über den barrierefreien Zugang zu Websites und mobilen Anwendungen öffentlicher Stellen, 2016, <https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX%3A32016L2102>
- Richtlinie (EU) 2019/882 über Barrierefreiheitsanforderungen für Produkte und Dienstleistungen, 2019, <https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX%3A32019L0882>
- Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie Informationstechnik-Verordnung - BITV), 2002, <https://www.buzer.de/gesetz/6217/index.htm>
- Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung - BITV 2.0), 2011, <https://www.buzer.de/BITV.htm>
- Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung - BITV 2.0), 2021, <https://www.buzer.de/BITV.htm>
- Vuetify. (2024a). *All vuetify components* [Abgerufen am 01.10.2025]. <https://vuetifyjs.com/en/components/all/>
- Vuetify. (2024b). *Vuetify - a vue component framework* [Abgerufen am 02.10.2025]. <https://vuetifyjs.com/en/>
- Vuetify. (2025a). *Accessibility (a11y)* [Abgerufen am 02.10.2025]. <https://vuetifyjs.com/en/features/accessibility/#activator-slots>
- Vuetify. (2025b). *Get started with vuetify 3* [Abgerufen am 01.10.2025]. <https://vuetifyjs.com/en/getting-started/installation/#installation>
- Vuetify. (2025c). *Vue component framework* [Abgerufen am 06.10.2025]. <https://github.com/vuetifyjs/vuetify>
- W3C. (1999). *Web content accessibility guidelines 1.0* [Abgerufen am 20.09.2025]. <https://www.w3.org/TR/WAI-WEBCONTENT>
- W3C. (2008). *Web content accessibility guidelines (wcag) 2.0* [Abgerufen am 20.09.2025]. <https://www.w3.org/TR/WCAG20/>
- W3C. (2025a). *About wai* [Abgerufen am 20.09.2025]. <https://www.w3.org/WAI/about/>
- W3C. (2025b). *Understanding success criterion 1.4.11: Non-text contrast* [Abgerufen am 24.09.2025]. <https://w3c.github.io/wcag21/understanding/non-text-contrast.html>
- W3C. (2025c). *Wai-aria overview* [Abgerufen am 20.09.2025]. <https://www.w3.org/WAI/standards-guidelines/aria/>
- W3C. (2025d). *Web content accessibility guidelines (wcag) 2.1* [Abgerufen am 20.09.2025]. <https://www.w3.org/TR/WCAG21/>
- W3C. (2025e). *Web standards* [Abgerufen am 20.09.2025]. <https://www.w3.org/standards/>

web.dev. (2024). *Learn accessibility* [Abgerufen am 01.10.2025]. <https://web.dev/learn/accessibility>

Wood, L. E. (1998). *User interface design: Bridging the gap from user requirements to design*. CRC Press.