Hochschule Flensburg

B A C H E L O R – T H E S I S

Thema: Stärken und Grenzen automatisierter Accessibility-Prüftools für die Bewertung von Webseiten

von: Sophia Wild

Matrikel-Nr.: 690613

Studiengang: Medieninformatik

Betreuer/in und Erstbewerter/in: Prof. Dr. Torben Wallbaum

Zweitbewerter/in: Prof. Dr. Sven Bertel

Ausgabedatum: 17.09.25

Abgabedatum: 17.11.25

Inhaltsverzeichnis

Abkürzungsverzeichnis

BFSG ..... *Barrierefreiheitsstärkungsgesetz*
BFSGV ..... *Verordnung zum Barrierefreiheitsstärkungsgesetz*
BITV ..... *Barrierefreie-Informationstechnik-Verordnung*
UW ..... *University of Washington*
W3C ..... *World Wide Web Consortium*
WCAG ..... *Web Content Accessibility Guidelines*

Anmerkung zur Umsetzung

Im Laufe der Thesis wurden zwei Webseiten für die Tests verwendet. Die Domains und ggf. das entsprechende GitHub Repository sind im Folgenden angegeben:

Accessible University (UW)

Domain: https://www.washington.edu/accesscomputing/AU/

GitHub Repository: https://github.com/terrill/AU

Before and After Demonstration (W3C)

Domain: https://www.w3.org/WAI/demos/bad/

1 Einleitung

Mit der Umsetzung der EU-Richtlinie zur Stärkung der Barrierefreiheit, ist im Juni 2025 in Deutschland ein neues Gesetz in Kraft getreten, das „Barrierefreiheitsstärkungsgesetz“ (BFSG). Durch diese Rechtslage sollen private Unternehmen verpflichtet werden, Barrierefreiheit im Web zu gewährleisten. Dies betrifft Unternehmen, die Produkte oder Dienstleistungen anbieten [1].

Die gesetzlichen Vorgaben dienen dem Ziel, dass ein uneingeschränkter Zugang auf Webseiten für alle Menschen eine Selbstverständlichkeit sein soll. Testergebnisse zeigen jedoch, dass auf Webseiten nach wie vor erhebliche Defizite in Bezug auf Barrierefreiheit vorhanden seien. So geht aus einer Studie von Aktion Mensch und Google aus dem Jahr 2025 hervor, dass zwei Drittel der in Deutschland meistbesuchten Onlineshops nicht barrierefrei sind [2].

Eine mögliche Lösung zur Überprüfung der Barrierefreiheit stellt der Einsatz von Prüftools dar. Diese ermöglichen eine Analyse und Bewertung von Webseiten anhand definierter Kriterien, wie der „Web Content Accessibility Guidelines“ (WCAG). Inzwischen stehen zahlreiche solcher Tools zur Verfügung, deren Ergebnisse sich in Umfang, Preis und Genauigkeit unterscheiden. In der wissenschaftlichen Literatur lassen sich bereits mehrere Studien finden, die den Fokus auf den Vergleich einzelner Prüftools legen, um das jeweils effektivste Tool oder die jeweils effektivste Kombination zu ermitteln [3], [4]. Allerdings betont eine Studie von Ara, Sik-Lányi, Kelemen und Guzsvinecz aus dem Jahr 2024, dass die meisten Accessibility-Prüftools lediglich einen bestimmten Anteil von etwa 50 % der WCAG-Kriterien überprüfen [5]. Für die vollständige Analyse einer Webseite könnten deshalb zusätzlich manuelle Tests erforderlich sein.

Aus dieser Grundproblematik ergibt sich die Leitfrage dieser Arbeit: **„Welche Stärken und Grenzen weisen Accessibility-Prüftools bei der Bewertung der Barrierefreiheit von Webseiten auf?“** Das Ziel dieser Arbeit ist es, den Fokus weniger auf die Identifikation des leistungsstärksten Tools zu legen, sondern vielmehr auf die Ermittlung des Umfangs, in dem die bestehenden Tools bereits praxistauglich sind. Daraus ergibt sich folgende Unterfrage: „In welchen Bereichen bleibt eine ergänzende manuelle

Prüfung weiterhin unverzichtbar und welches Verbesserungspotenzial besteht für die Weiterentwicklung dieser Tools?“ Die vorliegende Arbeit richtet sich damit sowohl an Anbieter solcher Prüftools, die ihre Systeme kontinuierlich weiterentwickeln, als auch an Unternehmen, die auf Prüftools zurückgreifen.

Um die Leitfrage dieser Arbeit zu beantworten, wurden im Rahmen der Analyse praktische Tests durchgeführt. Die Tools haben gezielt demonstrative Webseiten überprüft, die entwickelt wurden, um eine Vielzahl an Barrieren aufzuzeigen. Alle Barrieren der Demo-Webseiten wurden zu beginn tabellarisch zusammengeführt. Anschließend konnten die Testergebnisse der Prüftools, axe DevTools von der Firma Deque Systems und WAVE von der Firma WebAIM in den Tabellen eingetragen werden. Durch dieses Vorgehen konnten die tatsächlich existierenden Barrieren mit den automatisierten Testergebnissen verglichen und bewertet werden.

Es wird davon ausgegangen, dass sich automatisierte Prüftools in ihrer Funktionsweise gegenseitig ergänzen, da sie über unterschiedliche Stärken und Spezialisierungen verfügen. Im Rahmen dieser Arbeit wird diese Annahme überprüft, um festzustellen, ob sich Unterschiede in der Erkennungsleistung und Ergebnisdarstellung tatsächlich gegenseitig ausgleichen oder ob ein Tool insgesamt umfangreichere Ergebnisse liefert.

Die Leser werden in Kapitel 2 zunächst in die theoretische Grundlage anhand bestehender fachlicher Literatur eingeführt. Hierzu werden zentrale Begrifflichkeiten wie „Barrierefreiheit“ und „WCAG“ erörtert. In Kapitel 3 wird die Ausgangslage für die technische Umsetzung dargestellt, sodass die Auswahl und die Anwendungsmöglichkeiten der Prüftools und der Demo-Webseiten nachvollziehbar werden. Auf der Grundlage der vorangegangenen Ausführungen, wird die konkrete Durchführung der Tests schriftlich begleitet und es wird aufgezeigt, nach welchen Kriterien die Ergebnisse bewertet werden. Die Ergebnisse werden im Kapitel 4 aufgezeigt und im Kapitel 5 analysiert und gegenübergestellt, die daraus resultierenden Erkenntnisse werden abschließend im Fazit formuliert.

2 Theoretischer Hintergrund

In dem vorliegenden Kapitel werden die Grundlagen des Themas dieser Arbeit definiert. Um die Stärken und Grenzen automatisierter Accessibility-Prüftools aufzeigen zu können, muss ein grundlegendes Verständnis für die Konzepte und Rahmenbedingungen der digitalen Barrierefreiheit geschaffen werden. Hierzu werden zunächst die Leitlinien zur barrierefreien Gestaltung von Webinhalten sowie die rechtlichen Rahmenbedingungen erläutert. Im weiteren Verlauf wird erörtert, was unter dem Begriff „Accessibility-Prüftools“ zu verstehen ist. Dabei werden die beiden Prüftools axe DevTools und WAVE repräsentativ betrachtet.

2.1 Digitale Barrierefreiheit

Mit der Fortschreitung der Digitalisierung ist auch die Vielfalt der Nutzer von Webanwendungen gestiegen. Die Gewährleistung der Barrierefreiheit erlangt dadurch eine entscheidende Priorität. García-Santiago und Olvera-Lobo zeigen in der Studie „How accessibility guidelines are used in Spanish World Heritage websites“ auf, dass die Zugänglichkeit von Webseiten nicht nur individuelle Vorteile hat, sondern auch einen Gewinn für die Gesellschaft bietet [6]. Barrierefreie Webangebote fördern die soziale Inklusion, indem sie Menschen mit Behinderungen den gleichen Zugang zu Informationen und Dienstleistungen ermöglichen. Dies trägt zur Reduzierung von Diskriminierung und sozialer Ausgrenzung bei. Darüber hinaus profitieren alle Nutzer von barrierefreien Websites, da diese in der Regel benutzerfreundlicher, klarer strukturiert und besser navigierbar sind [6].

Der Begriff „Barrierefreiheit“ ist laut des BFSG § 3 Absatz 1, wie folgt definiert: „Produkte und Dienstleistungen sind barrierefrei, wenn sie für Menschen mit Behinderung in der allgemein üblichen Weise, ohne besondere Erschwernis und grundsätzlich ohne fremde Hilfe auffindbar, zugänglich und nutzbar sind“ [7]. Im Hinblick auf die Barrierefreiheit im Internet bedeutet das, Webinhalte so bereitzustellen, dass sie unabhängig von einer Behinderung oder dem Gerät zugänglich sind.

Laut der WCAG werden folgende Behinderungen berücksichtigt: Blindheit und Sehbehinderung, Gehörlosigkeit und Hörverlust, Bewegungseinschränkungen, Sprachbehinderungen, Lichtempfindlichkeit,

Lernbehinderung sowie kognitive Einschränkungen. Dies schließt auch mögliche Kombinationen der aufgelisteten Behinderungen mit ein [8]. Über dieses breite Spektrum hinaus profitieren zusätzlich Menschen mit unterschiedlichen Altersstufen, Fähigkeiten, Bildungsniveaus und Kompetenzen von den barrierefreien Gestaltungsprinzipien der WCAG. Somit trägt die Umsetzung der WCAG nicht nur zur Berücksichtigung von Menschen mit Behinderung bei, sondern stärkt auch die allgemeine Benutzerfreundlichkeit und mithin die Einbeziehung größerer Nutzergruppen.

Außerdem sei darauf hinzuweisen, dass die Einhaltung der WCAG und somit auch die gesetzlichen Richtlinien nicht zwangsläufig zur Beseitigung sämtlicher Barrieren führen und folglich nicht sämtlichen Bedürfnissen von Menschen mit Behinderungen gerecht werden. Allerdings befinden sich Gestaltungsprinzipien wie die WCAG in einer permanenten Weiterentwicklung. Das Ziel ist es, auf den dynamischen Wandel der digitalen Welt reagieren zu können und für neue Technologien barrierefreie Lösungen bereitzustellen, damit Menschen mit Behinderungen ein gleichberechtigter Zugang zu Webinhalten ermöglicht wird.

2.1.1 Web Content Accessibility Guidelines (WCAG)

Im Mai 1999 legte das World Wide Web Consortium (W3C) den Grundstein für einen international anerkannten Standard für Barrierefreiheit im Web, die Web Content Accessibility Guidelines (WCAG) [9]. Dieser Standard dient als Grundlage für EU-Richtlinien und Gesetze zur digitalen Barrierefreiheit, die im Kapitel 2.1.1 „Rechtliche Rahmenbedingungen“ näher betrachtet werden. Nach der Publikation der WCAG 1.0 folgten in Abständen aktualisierte Versionen. Die WCAG 2.2 ist derzeit die aktuellste Version und ist im Dezember 2024 erschienen [10]. In Bezug auf die Weiterentwicklung ist zu betonen, dass die WCAG 2.2 die vorherigen Versionen, WCAG 2.0 und WCAG 2.1, weder ersetzt noch ungültig macht. Vielmehr ergänzt sie diese und baut auf ihren Grundlagen auf [8].

Die WCAG sollen einen möglichst äquivalenten Zugang aller Menschen zu Informationen und Funktionen im Internet gewährleisten. Dazu zählen Webinhalte wie Text, Bilder, Videos, Audioinhalte und interaktive Elemente.

Die WCAG sind in mehrere Ebenen von Leitlinien untergliedert. Alle Ebenen wirken zusammen und sollen eine übersichtliche Darstellung ermöglichen.

Die erste Ebene bilden die allgemeinen **Grundsätze (Principles)**. Diese sind nach vier Hauptgrundsätzen klassifiziert: Wahrnehmbar (perceivable), Bedienbar (operable), Verständlich (understandable) und Robust (robust). Die folgende Tabelle 1 veranschaulicht eine Zusammenfassung dieser Grundsätze und zeigt Beispiele auf.

Tabelle 1 Grundsätze der WCAG

Grundsatz	Definition	Beispiel
Wahrnehmbar	Informationen und Komponenten der Benutzeroberfläche müssen für Benutzer auf eine Weise darstellbar sein, die sie wahrnehmen können.	• Alternativtext bei Bildern und Videos • Untertitel bei Audioinhalten • Ein Farbkontrast von mindestens 4,5:1 bei informativen Webinhalten
Bedienbar	Komponenten der Benutzeroberfläche und die Navigation müssen bedienbar sein.	• Alle Webinhalte sind über eine Tatstatur bedienbar • Webseiten haben eine korrekte Überschriftenstruktur und Fokusreihenfolge
Verständlich	Informationen und die Funktionsweise der Benutzeroberfläche müssen verständlich sein.	• Die Standardsprache der Webseite kann bestimmt werden • Fehlerhafte Eingaben werden textlich beschrieben
Robust	Der Inhalt muss robust genug sein, damit er von einer Vielzahl von Benutzeragenten, einschließlich unterstützender Technologien, interpretiert werden kann.	• Webinhalte benutzen einen passenden Tag und werden korrekt verschachtelt

Die nächste Ebene besteht aus den insgesamt 13 **Richtlinien (Guidelines)** der WCAG. Diese konkretisieren die vier Grundsätze und beschreiben die übergeordneten Ziele der Barrierefreiheit. Unter jeder Richtlinie sind ein oder mehrere **Erfolgskriterien (Success Criteria)** definiert, die anhand von Konformitätsstufen eingestuft sind. Es existieren drei Stufen:

- Konformitätsstufe A = Die niedrigste Konformität, vermittelt einen Standard, der ein geringes Maß an Barrierefreiheit gewährleistet.
- Konformitätsstufe AA = Die mittlere Konformität, vermittelt einen Standard, der den Anforderungen einer ausreichenden Barrierefreiheit grundsätzlich genügt.
- Konformitätsstufe AAA = Die höchste Konformität, vermittelt einen Standard, der den höchsten Grad an Barrierefreiheit sichert.

Insgesamt werden 86 Erfolgskriterien definiert, davon 31 mit der Konformitätsstufe A, 24 mit der Konformitätsstufe AA und 31 mit der Konformitätsstufe AAA. Jedem Erfolgskriterium stehen **Techniken (Sufficient and Advisory Techniques)** zur Verfügung, die entweder ausreichend für die Erfüllung eines Erfolgskriteriums sind, oder eine beratende Funktion erfüllen [11].

Im Folgenden werden die Ebenen der WCAG mittels eines praktischen Beispiels veranschaulicht. Dabei wird der Punkt „Ein Farbkontrast von mindestens 4,5:1“ näher betrachtet. Ein ausreichender Farbkontrast ist eine wesentliche Voraussetzung für die visuelle Wahrnehmbarkeit von Webinhalten und wird daher dem **Grundsatz 1 wahrnehmbar** zugeordnet. Um die Lesbarkeit von Webinhalten zu verbessern und eine klare Unterscheidung zwischen Vorder- und Hintergrund zu ermöglichen, ist ein ausreichender Farbkontrast erforderlich. Aufgrund dieser Merkmale ist die **Richtlinie 1.4 unterscheidbar** für Kontraste zuständig. Die Kriterien zur Verwendung von Farbkontrasten sind in zwei Konformitätsstufen unterteilt. Gemäß dem **Erfolgskriterium 1.4.3 Kontrast (AA)** ist bei der visuellen Darstellung von Text und Bildern von Text ein Kontrastverhältnis von mindestens 4,5:1 erforderlich, um einen minimalen Farbkontrast zu gewährleisten [12]. Ausgenommen sind Texte, die ausreichend groß, nebensächlich oder Teil eines Logos sind. In den entsprechenden **Techniken und Fehlern für 1.4.3** ist genau beschrieben, dass ein Kontrastverhältnis von mindestens 4,5:1 für

Texte gilt, die eine Schriftgröße unter 18 Pt. bzw. unter 14 Pt. aufweisen und fett gedruckt sind. Für größere Texte ist ein Kontrastverhältnis von mindestens 3:1 ausreichend [13]. Das **Erfolgskriterium 1.4.6 Kontrast (AAA)** baut auf dem Erfolgskriterium 1.4.3 Kontrast (AA) auf und definiert höhere Anforderungen an den Farbkontrast [14]. Zur Gewährleistung dieses Erfolgskriteriums wird ein Farbkontrast von mindestens 7:1 für Texte vorausgesetzt. Größere Texte ab 18 Pt. bzw. 14 Pt. und fett gedruckt, benötigen einen Farbkontrast von mindestens 4,5:1 [15]. Die folgende Abbildung 1 veranschaulicht die drei Grenzwerte der Kontrastbereiche und ordnet sie den Konformitätsstufen zu.

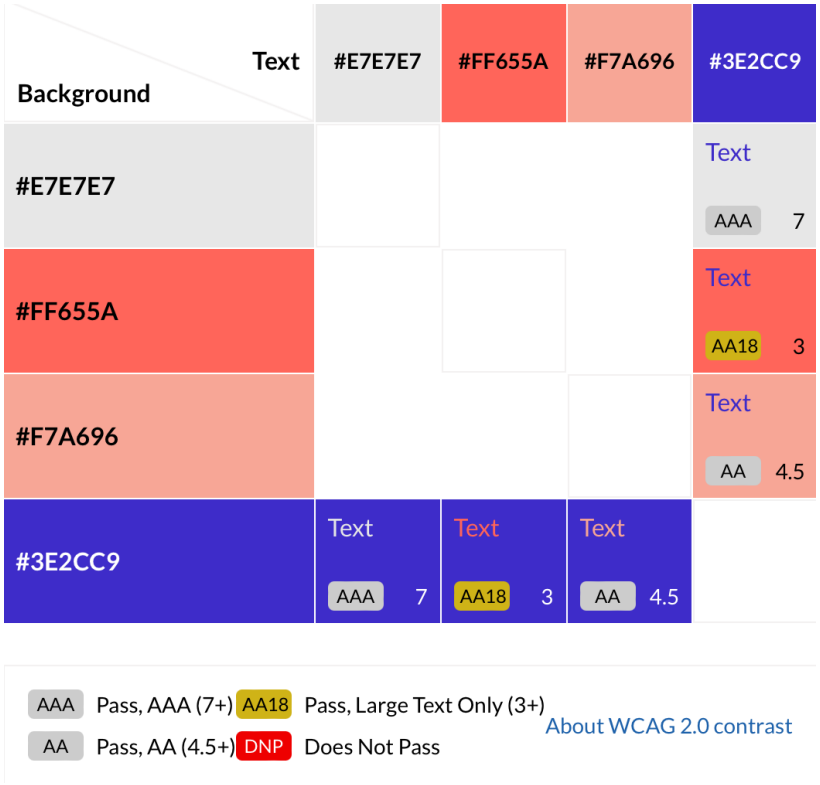


Abbildung 1: Contrast Grid
Quelle: https://contrast-grid.eightshapes.com/?version=1.1.0&background-colors=&foreground-colors=%23E7E7E7%0D%0A%23FF655A%0D%0A%23F7A696%0D%0A%233E2CC9&es-color-form__tile-size=compact&es-color-form__show-contrast=aaa&es-color-form__show-contrast=aa&es-color-form__show-contrast=aa18&es-color-form__show-contrast=dnp

2.1.2 Rechtliche Rahmenbedingungen

„Niemand darf wegen seiner Behinderung benachteiligt werden“. So steht es im Grundgesetz, Art. 3 Abs. 3 [16]. Dieses Diskriminierungsverbot bildet die verfassungsrechtliche Grundlage für sämtliche Regelungen zur Gleichstellung und Teilhabe von Menschen mit Behinderungen. Auf dieser Grundlage wurde das Behindertengleichstellungsgesetz (BGG) geschaffen,

worauf wiederum die Barrierefreie-Informationstechnik-Verordnung (BITV 2.0) basiert und die Anforderungen zum § 12 des BGG umsetzt [17], [18]. Gemäß der seit 2011 geltenden BITV 2.0 sind elektronisch zur Verfügung gestellten Informationen und Dienstleistungen von öffentlicher Stellen des Bundes, barrierefrei zu gestalten [18].

Allerdings bezieht sich die BITV auf keine privatwirtschaftlichen Akteure. Unternehmen, Banken, Onlineshops sowie App-Anbieter waren bislang nicht gesetzlich verpflichtet, ihre digitalen Angebote barrierefrei zu gestalten. Zur Behebung dieses Mangels, hat die Europäische Union die Richtlinie (EU) 2019/882 verabschiedet. Gemäß dieser Richtlinie werden Mitgliedstaaten in die Pflicht genommen, Barrierefreiheit auch für bestimmte Produkte und Dienstleistungen der Privatwirtschaft zu gewährleisten [19].

In Deutschland wurde diese Richtlinie durch das Barrierefreiheitsstärkungsgesetz (BFSG) umgesetzt, welches im Juli 2021 verabschiedet wurde und am 28. Juni 2025 in Kraft trat [1]. Gemäß den Vorgaben des BFSG sind Hersteller und Dienstleistungsanbieter dazu angehalten, ihre neuen Produkte und digitalen Angebote barrierefrei zu gestalten. Ausgenommen sind Kleinunternehmen, die nicht unter das BFSG fallen. Die Verordnung zum Barrierefreiheitsstärkungsgesetz (BFSGV) ist die Durchführungsverordnung zum BFSG und konkretisiert die technische Umsetzung und organisatorische Details [20].

Für die technische Umsetzung verweisen die BITV 2.0 und das BFSGV auf die europäische Norm EN 301 549, welche verbindliche Anforderungen an die digitale Barrierefreiheit definiert [21]. Die vorliegende Norm gründet sich in zentralen Aspekten auf die WCAG des W3C. Die WCAG stellen somit die international anerkannte Grundlage für barrierefreie Webinhalte dar. Obwohl die WCAG an sich nicht rechtlich bindend sind, erlangen sie durch ihre Integration in die EN 301 549 und das darauf aufbauende BITV 2.0 und BFSG faktisch rechtliche Relevanz. Die WCAG stellt folglich das technische Fundament dar, auf dem das aktuelle deutsche und europäische Barrierefreiheitsrecht im digitalen Kontext fußt.

2.2 Automatisierte Accessibility-Prüftools

Obwohl konkrete technische Vorgaben zur Verfügung stehen, stellen die WCAG-Richtlinien Entwickler:innen vor erheblichen Herausforderungen. Diese ergeben sich insbesondere aus den zeitaufwendigen manuellen Analysen von Webseiten, die zur Sicherstellung der Konformität erforderlich sind. Der Einsatz von Accessibility-Prüftools kann diesen Prozess unterstützen. Accessibility-Prüftools sind Softwareprogramme oder Online-Dienste, die dazu dienen, Webinhalte anhand definierter Richtlinien auf Barrierefreiheit zu überprüfen [22]. Sie können beim Testen von Webseiten Zeit, Mühe und Geld sparen. Das W3C listet auf seiner Website derzeit mehr als 100 solcher Tools auf [23]. Zur Unterstützung bei der Auswahl können die Tools anhand verschiedener Kriterien gefiltert werden, darunter nach Zweck, Kosten, Typ und unterstützten Standards, wie der WCAG 2.2. Die Tools umfassen unterschiedliche Ausprägungen, darunter vollautomatische Lösungen zur umfassenden Erkennung von Barrierefreiheitsproblemen, beispielsweise als npm-Paket, REST-API oder Browser-Extension [22].

Die Auswahl des passenden Tools ist von diversen Faktoren abhängig, darunter der gegenwärtige Entwicklungsstand der Webseite. Bestimmte Tools unterstützen spezifische Rollen in einem Projektteam, wie beispielsweise Content-Autoren, Code-Entwickler, Designer und Produktverantwortliche. Bei der Wahl spielen zudem die zu prüfenden Inhalte sowie die verfügbaren Ressourcen eine entscheidende Rolle. Einige Tools prüfen beispielsweise nur einzelne Seiten, während andere ganze Webseiten oder passwortgeschützte Inhalte analysieren können [22].

Es ist jedoch zu betonen, dass automatisierte Tools nicht allein alle Aspekte der Barrierefreiheit abdecken können. Manuelle Tests sind nach wie vor unerlässlich, um eine vollständige Konformität mit den WCAG zu gewährleisten [22]. Einige Tools, darunter auch das Tool WAVE, bieten jedoch unterstützende Funktionen bei manuellen Prüfungen an [24]. So stellen diese Tools beispielsweise Checklisten bereit oder führen die Anwender:innen durch den Prüfprozess.

Prüftool axe von der Firma Deque Systems

Deque Systems ist ein US-amerikanisches Technologieunternehmen, das sich auf die Schaffung digitaler Zugänglichkeit spezialisiert hat. Das Unternehmen entwickelt Tools, bietet Schulungen und Beratungsleistungen an, um die Gestaltung digitaler Produkte barrierefrei und in Konformität mit den WCAG-Standards zu unterstützen [25].

Deque Systems vermarktet eine Vielzahl von Produkten, wobei bei den Pro- und Enterpriseversionen mit Kosten ab 45 US-Dollar pro Benutzer und Monat zu rechnen sind [26]. Darüber hinaus werden kostenfreie Optionen bereitgestellt. Zu den Programmen zählen das seit 2015 bestehende Open-Source-Framework „axe“ sowie die „axe DevTools Browser-Extension“, welche für die Browser Chrome, Firefox und Edge als Basisversion kostenlos zur Verfügung steht. Für Zusatzfunktionen der Browser Extension werden ebenfalls Gebühren erhoben [25].

Prüftools WAVE von der Firma WebAIM

WAVE (Web Accessibility Evaluation Tool) ist ein Produkt von WebAIM (Web Accessibility In Mind), einer Organisation der Utah State University. Ziel von WebAIM ist es, Entwickler:innen und Institutionen dabei zu unterstützen, Webseiten für Menschen mit Behinderungen zugänglich zu gestalten. WAVE bietet eine Reihe von Evaluationstools und gibt an, zahlreiche potenzielle Verstöße gegen die WCAG erkennen zu können [27].

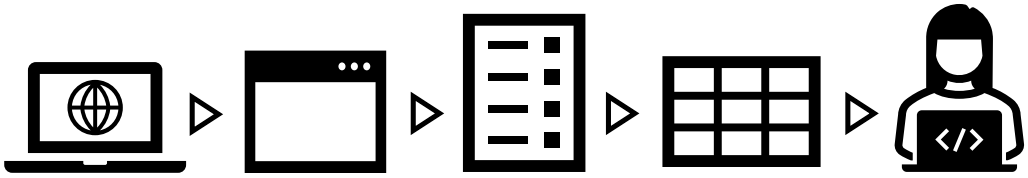
Zu den Tools gehören unter anderem die „WAVE Browser-Extensions“ für Chrome, Firefox und Edge, die eine schnelle und kostenfreie Analyse einzelner Webseiten ermöglichen. Für umfangreichere Testvorhaben, bei denen eine größere Anzahl von Webseiten automatisiert überprüft werden sollen, stehen die kostenpflichtigen Dienste wie „WAVE API and Testing Engine“ und „Accessibility Impact (AIM)“ zur Verfügung [27]. Die Preise richten sich nach dem Nutzungsumfang und der Anzahl der zu prüfenden Seiten. Die API beginnt beispielsweise bei rund 10 US-Dollar, während der AIM-Report ab 500 US-Dollar erhältlich ist [28], [29].

3 Methodik

Die Bewertung der eingesetzten automatisierten Prüftools erfolgte unter Einbeziehung zweier zentraler Aspekte, einerseits die Effektivität und andererseits die Benutzerfreundlichkeit der Tools. Die Effektivität wird anhand der Fähigkeit gemessen, Barrieren zuverlässig zu erkennen. Zur Benutzerfreundlichkeit gehören Aspekte wie Installation, Übersichtlichkeit und Nachvollziehbarkeit der Ergebnisse.

Für die Analyse der Tools wurde ein empirischer Ansatz gewählt. Dadurch konnte sichergestellt werden, dass die Analyse nicht auf Herstellerangaben oder behaupteten Fähigkeiten beruht, die sich möglicherweise als unzutreffend oder unvollständig erweisen könnten. Die Tools wurden daher mittels eigenständig durchgeführter Tests angewendet, wodurch die tatsächlich auftretenden Fehlermeldungen analysiert werden konnten. So war es möglich, eine objektive und nachvollziehbare Bewertung auf Grundlage praktischer Beispiele durchzuführen.

Tabelle 2: Methodisches Vorgehen



Tool Auswahl	Demo-Webseiten Auswahl	Auflistung der Barrieren	Strukturierung nach Prüfkriterien	Automatisierte Prüfung
• Auswahl repräsentativer Prüftools	• Auswahl Demo-Webseiten, die gezielt Barrieren aufzeigen	• Auflistung der existierenden Barrieren der Demo-Webseiten	• Zusammenstellung relevanter Anforderungen • Gruppierung der aufgelisteten Barrieren nach Anforderungen	• Automatisierte Prüfung der Webseiten mit Tool a und b • Dokumentation der Ergebnisse

Das methodische Vorgehen dieser Arbeit wird in der vorangestellten Tabelle 2 dargestellt. Im ersten Schritt erfolgt eine ausführliche Begründung zur Auswahl der Prüftools. Im weiteren Verlauf wird aufgezeigt, welche zwei Demo-Webseiten als Umgebung für die automatisierte Tests genutzt wurden. Es wurde sich aktiv für den Einsatz von Demo-Webseiten entschieden, weil diese explizit für die Veranschaulichung von Barrieren entwickelt wurden. Auf diese Weise wird eine Bandbreite an Barrieren gewährleistet, die die Negativbeispiele der WCAG abdecken. Zusätzlich sind die aufgezeigten Barrieren der Demo-Webseiten dokumentiert, sodass alle Barrieren für die Bewertung der Testergebnisse miteinbezogen werden konnten. Eine plötzliche Aktualisierung der Webseiten während der Testphase konnte ebenfalls mit hoher Wahrscheinlichkeit ausgeschlossen werden, da die Demo-Webseiten nicht regelmäßig aufgrund von ändernden Web-Inhalten aktualisiert werden. Die Anwendung beider Tools als Browser-Extension erfolgte in einem Google-Chrome-Browser mit der Version 141.0.7390.123. Im darauffolgenden Schritt wurden alle auf den Demo-Webseiten dokumentierten Barrieren einzeln aufgelistet und anschließend nach definierten Anforderungen gruppiert. Die daraus resultierenden Tabellen stellen eine strukturierte Ausgangsbasis für die anschließende Auswertung der Testergebnisse zur Effektivität bereit.

3.1 Auswahl und Anwendungsmöglichkeiten der Prüftools

Die Auswahl geeigneter Prüftools ist für die Qualität und Aussagekraft der im Rahmen dieser Arbeit durchgeführten Testverfahren von zentraler Bedeutung. In diesem Kapitel werden zunächst die Auswahlkriterien erläutert. Auf dieser Grundlage fiel die Entscheidung auf die Prüftools WAVE von WebAIM und axe DevTools von Deque Systems. Darauf aufbauend wird in einem zweiten Schritt die Anwendung der Tools beschrieben. Dabei wird erläutert, wie die Browser-Extension installiert, genutzt und Ergebnisse dargestellt werden.

Auswahl der Prüftools

Im Rahmen dieser Arbeit sollten ausschließlich Tools berücksichtigt werden, die eine Webseite basierend auf der WCAG-Verordnung bewerten können. Außerdem war es entscheidend, dass diese Tools aktiv gepflegt werden und kostenlos oder als Open Source zur Verfügung stehen. Ein weiteres Auswahlkriterium bestand darin, dass die Tools als REST-API, NPM-Paket oder Browser-Extension genutzt werden konnten. Die vorliegende Eingrenzung gewährleistet, dass die ausgewählten Tools eine praxisnahe Einsatzmöglichkeit aufweisen und langfristig gewartet werden. Da eine Vielzahl an Prüftools diese Kriterien erfüllen, ermöglichte die statistische Verbreitung in der Fachpraxis eine weitere Eingrenzung. Laut einer Umfrage von WebAim im Jahr 2021, „Survey of Web Accessibility Practitioners #3“, zählen axe und WAVE zu den am häufigsten verwendeten Tools im Bereich automatisierter Barrierefreiheitsprüfung. In dieser Umfrage gaben 64 % der Befragten an, axe oder ein axe-Plugin zu verwenden. 53 % der Befragten gaben an, WAVE als Chrome- oder Firefox-Extensions zu verwenden [30]. Zudem kann anhand der Ergebnisse dieser Umfrage darauf geschlossen werden, dass in der Fachpraxis häufig gleich mehrere Tools zum Bewerten von einer Website zum Einsatz kommen.

Basierend auf den Auswahlkriterien und der statistischen Verbreitung der Tools sollten die Prüftools WAVE, der Firma WebAIM und die axe DevTools, der Firma Deque Systems für die Tests verwendet werden. Beide Tools wurden jeweils separat als auch in Kombination betrachtet und bilden für diese Arbeit einen repräsentativen Ausschnitt praxisrelevanter Tools zur automatisierten Überprüfung der Barrierefreiheit.

Sowohl axe als auch WAVE bieten eine kostenlose Browser-Extension an. Die Kostenstruktur für weitere Funktionen ist bei beiden Tools unterschiedlich [25], [27]. Nach den zuvor genannten Kriterien zur Toolauswahl fiel die Wahl im Fall von WAVE auf die Browser-Extension. Axe verfügt im Kern über ein kostenloses Open-Source-Framework (axe-core) und eine darauf basierende Browser-Extension (axe Devtools). Allerdings lässt sich anmerken, dass sich bei einem direkten Vergleich der über npm installierbaren Bibliothek „axe-core“ und der „axe DevTools“ als Browser-Extension, keine signifikanten Unterschiede, in Bezug auf die Anzahl der identifizierten Probleme einer Webseite ergaben. Die Browser-Extension hat

zusätzliche Probleme in der Kategorie „Moderate (mittel)“ angezeigt. Allerdings wurden diese Probleme nur in der Browser-Version identifiziert, weil die Funktion „Best Practices“ ausgewählt war. Diese Funktion erlaubt es auch über etablierte Regelwerke hinaus, Barrieren zu identifizieren [31]. Der größte Unterschied der beiden Tools liegt in der Nutzung und Installation der jeweiligen Varianten. Während bei der Browser-Extension keine Programmierkenntnisse erforderlich sind, erfordert die npm-Variante eine geeignete Testumgebung wie beispielsweise Visual Studio Code, Node.js und npm. Darüber hinaus ist die Ausführung eines Java Scripts erforderlich und die Ergebnisse von „axe-core“ werden in Form eines umfangreichen JSON-Reports aufgelistet. Wie dieser Report auszuwerten ist, wird in der Dokumentation von „Deque Systems“ angegeben [32]. Die Ergebnisse der Browser-Extension werden dagegen direkt visuell aufbereitet.

Aufgrund der aufgezeigten Gründe wurde für die Tests der beiden Tools die kostenlose Browser-Extension verwendet. Im vorliegenden Fall kam die axe Devtools Browser-Extension in der Version 4.117.0 (basierend auf axe-core: Version 4.10.3) zum Einsatz, die am 17. Oktober 2025 veröffentlicht wurde. Für die Browser-Extension von WAVE wurde, die am 4. September 2024 veröffentlichte Version 3.2.7.1 verwendet. Die präsentierten Testergebnisse spiegeln lediglich die Funktionalität der Browser-Extension mit den angegebenen Versionen wider. Es sei daher darauf hingewiesen, dass im Rahmen dieser Arbeit kein Bezug auf zusätzliche Funktionen, Versionen oder Testverfahren genommen wird. Zudem wird im weiteren Verlauf der Arbeit davon ausgegangen, dass unter den Begrifflichkeiten „axe“ und „WAVE“ die zugehörige Browser-Extension zu verstehen ist, ohne dies explizit zu erwähnen. Sofern dies für das Verständnis des jeweiligen inhaltlichen Kontextes nicht erforderlich ist.

Anwendung der Prüftools

Die Installation der Browser-Extensions kann in beiden Fällen mit wenigen Klicks über die jeweiligen Webseiten des Anbieters durchgeführt werden. Zur Prüfung einer Webseite wird diese im Browser geöffnet. Im Fall von dem Prüftool axe DevTools, werden die automatisch erkannten Probleme in der Entwicklerkonsole des Browsers dargestellt. Dazu kann in der Konsole der Reiter „axe DevTools“ ausgewählt werden. Durch Klicken auf die

Schaltfläche „Scan ALL of my page“ wird die Analyse der Seite gestartet. In der Regel erscheinen die Testergebnisse nach kurzer Zeit in der Konsole. Abbildung 2 zeigt einen Ausschnitt der Konsole mit einer abgeschlossenen Analyse der Demo-Webseite „Before and After Demonstration/home“ der W3C.

Abbildung 2: axe Konsole UI
Quelle: Screenshot der UI beim Scan der Seite https://www.w3.org/WAI/demos/bad/before/home.html

Die Gesamtanzahl der identifizierten Probleme wird im oberen Bereich des Fensters hervorgehoben angezeigt. Gleichzeitig erfolgt eine Klassifizierung der Probleme nach Schweregrad in vier Kategorien, „Critical“, „Serious“, „Moderate“ und „Minor“. Im Rahmen dieser Arbeit wurden bei der Auswertung der Testergebnisse alle Schweregrade gleichwertig berücksichtigt. Die von axe erkannten Probleme werden innerhalb der Benutzeroberfläche in Dropdown-Menüs gegliedert. Unter jedem Eintrag finden sich eine kurze Beschreibung des Problems sowie der zugehörige HTML-Codeausschnitt. Über einen weiterführenden Link wird zudem auf die Deque University-Seite verwiesen. Dort werden die jeweiligen Fehlertypen ausführlich beschrieben und zum Teil den entsprechenden WCAG zugeordnet. Zusätzlich werden Beispielcode und Empfehlungen zur Behebung der jeweiligen Barriere bereitgestellt.

Für die Anwendung des Prüftools WAVE kann eine zu prüfende Webseite im Browser geöffnet und über einen Rechtsklick die Option „WAVE this page“ ausgewählt werden. Anschließend wird die Seite neu geladen und die identifizierten Probleme werden, anders als bei der Anwendung von axe, direkt auf der Webseite dargestellt. Die identifizierten Elemente

werden mithilfe farblich codierter Symbole markiert, die jeweils einer bestimmten Kategorie zugeordnet sind.

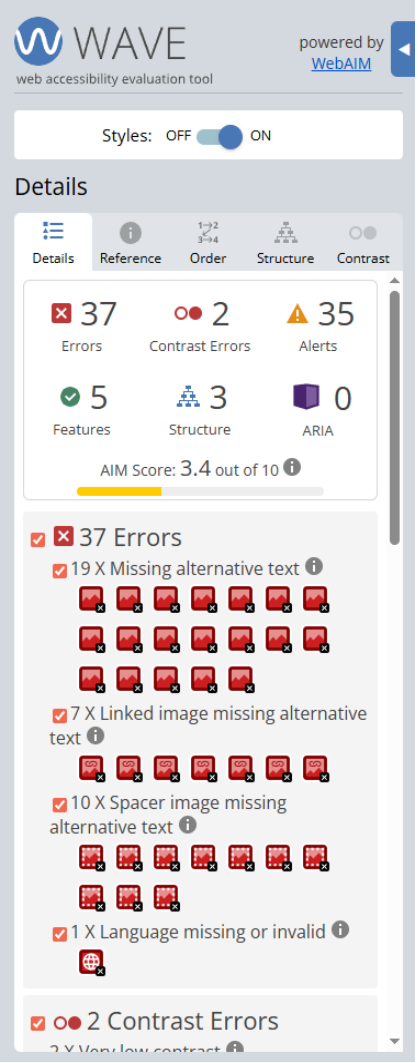


Abbildung 3: WAVE Sidebar UI
Quelle: Screenshot der UI beim Scan der Seite https://www.w3.org/WAI/demos/bad/before/home.html

Über eine linke Seitenleiste können die einzelnen Fehlertypen detailliert eingesehen werden. Abbildung 3 zeigt einen Ausschnitt dieser Seitenleiste mit einer abgeschlossenen Analyse der Demo-Webseite „Before and After Demonstration/home“ der W3C auf. Dort werden alle identifizierten Eigenschaften der Seite aufgelistet. Unter dem Reiter „Reference“ liefert WAVE eine Beschreibung des Fehlertyps, was genau der Algorithmus abfängt und welche Kriterien der WCAG diesem Fehler unterliegen. Durch einen Linksklick kann zudem die betroffene HTML-Codestelle in einem weiteren Fenster geöffnet werden. Für die Auswertung der Testergebnisse wurden im Rahmen dieser Arbeit ausschließlich die von WAVE ausgegebenen Fehlerkategorien „Error“, „Alert“ und „Contrast Errors“ berücksichtigt. Dies ist wichtig zu erwähnen, da WAVE zusätzliche Kategorien hat, die u. a. Features oder strukturelle Elemente kennzeichnen.

3.2 Auswahl der Domo-Webseiten

Für die Analyse wurden zwei Demo-Webseiten ausgewählt. Die „Accessible University Demo Site“ der University of Washington (UW) [33] sowie die „Bevore and After Demonstration“ der Web Accessibility Initiative W3C [34]. Die Auswahl der Websites erfolgte nach bestimmten Kriterien. In erster Linie wurde darauf geachtet, dass die Demo-Webseiten nicht von gewinnorientierten Unternehmen bereitgestellt oder erstellt wurden. Damit kein kommerzielles Interesse Einfluss auf die Gestaltung und Auswahl der

dargestellten Barrieren nimmt. Auf diese Weise kann davon ausgegangen werden, dass die Webseiten primär zu Demonstrations- oder Bildungszwecken erstellt wurden und sich daher besser für eine unabhängige Analyse eignen.

Darüber hinaus sollten die Webseiten kostenlos oder im Idealfall als Open-Source-Ressource öffentlich zugänglich sein. Eine offene Zugänglichkeit ist von entscheidender Bedeutung, um die Transparenz und Reproduzierbarkeit der Untersuchung zu gewährleisten. Die Bereitstellung des Quellcodes ermöglicht es zudem, die Webseite bei Bedarf zu verändern oder um Barrieren zu ergänzen. Ein weiteres entscheidendes Kriterium für die Auswahl der Demo-Webseiten war die Dokumentation der Barrieren. Das konnte entweder auf den Webseiten selbst erfolgen oder in einem zusätzlichen Dokument. Dies ermöglichte eine umfangreiche Erfassung und Zuordnung der Barrieren im Rahmen der Analyse. Der Vorteil bestand vor allem darin, dass dadurch Barrieren nicht übersehen werden.

Accessible University Homepage der University of Washington

Die Website "Accessible University" (AU) demonstriert eine Startseite einer fiktiven Universität und veranschaulicht dabei typische Barrieren, die bei nicht barrierefreiem Webdesign auftreten können. Die Website wurde vom AccesIT entwickelt und wird vom AccesComputing gewartet. Beide Projekte stammen von der University of Washington. Die Seite bietet eine „Before-Version“ mit typischen Barrieren, die Personen mit Behinderungen den Zugriff auf Inhalte und Funktionen erschweren sowie einer „After-Version“ mit einer barrierefreien Umsetzung der gleichen Seite. Zusätzlich existiert eine „Info-Seite“, die detaillierten Erläuterungen der einzelnen Barrieren der „Before-Seite“ aufzeigt. Die gesamte Webseite steht zudem als Open Source zur Verfügung.

Die Entwicklung der Demo-Webseite wurde durch das Nationalinstitute on Disability and Rehabilitation Research des U.S. Department of Education unterstützt und wird fortlaufend mit der Unterstützung der National Science Foundation aktualisiert und gepflegt. Es sei darauf hingewiesen, dass die Inhalte nicht notwendigerweise die Politik der US-Bundesregierung widerspiegeln und keine offizielle Unterstützung implizieren. Im Folgenden

sind alle Barrieren aufgelistet, die auf der „Info-Seite“ angegeben werden. Im Anhang unter "Accessible University Info Page" findet sich zu jeder aufgelisteten Barriere eine ausführliche Beschreibung.

Struktur

1. Fehlende Landmark-Bereiche
2. Keine Überschriften
3. Sprache nicht angegeben

Bilder

4. Bilder ohne Alternativtext
5. Fehlender oder übermäßiger Alternativtext bei dekorativen Bildern

Farbe

6. Unzureichender Farbkontrast
7. Einsatz von Farbe zur Kommunikation von Informationen

Tastatur

8. Unzugängliche Tastaturschnittstelle
9. Keine sichtbare Fokusanzeige
10. Kein Link„Zum Hauptinhalt springen“

Links

11. Redundanter, nicht informativer Linktext
12. Verwendung von Links u. Buttons mit falschem HATML-Element

Eigenschaften

13. Unzugängliches Navigationsmenü
14. Unzugängliches Karussell
15. Unzugänglicher modaler Dialog

Formulare

16. Formular nicht ordnungsgemäß beschriftet
17. Unzugängliches CAPTCHA
18. Unzugängliche Eingabevalidierung

Tabellen

19. Fehlende Markierung für barrierefreie Tabelle
20. Fehlende Abkürzungs-Tag

Video

21. Unzugängliche Audioinhalte
22. Unzugängliche visuelle Inhalte

Before and After Demonstration der Web Accessibility Initiative W3C

Die „Before and After“-Demonstration wurde entwickelt, um die Web-Zugänglichkeit zu veranschaulichen und für Barrierefreiheit im Web zu sensibilisieren. Die Entwicklung der Webseiten erfolgte in Zusammenarbeit mit dem Web Accessibility Initiative (WAI) des W3C und wurde durch die finanzielle Unterstützung des „IST Programmes“ der Europäischen Kommission ermöglicht.

Es werden praktische Beispiele bereitgestellt, die sich an den Kriterien der WCAG orientieren. Dazu existieren, wie auch bei der Website der UW, eine "Before Demonstration" mit zahlreichen Barrieren und eine "After Demonstartion", die eine barrierefreie Version der Seite bietet. So wird einerseits eine Bandbreite typischer Barrieren auf Webseiten angezeigt, andererseits werden auch Lösungswege für diese Barrieren angeboten. Es

besteht die Möglichkeit, Anmerkungen zu aktivieren, um die wichtigsten Barrieren in Bezug auf die Barrierefreiheit hervorzuheben. Diese Anmerkungen sind auf jeder Webseite vorhanden. Es sei darauf hingewiesen, dass nicht alle Arten von Barrieren auf den Demo-Webseiten abgedeckt werden.

Im Folgenden sind alle Barrieren aufgelistet, die in den Anmerkungen der einzelnen Webseiten angegeben werden. Im Anhang unter "Before and After Demonstration" findet sich zu jeder aufgelisteten Barriere eine ausführliche Beschreibung.

Home

1. Bild mit falschem Alternativtext
2. Bild ohne Alternativtext
3. Link nicht optisch unterscheidbar
4. Bild mit unzugänglichem Alternativtext
5. Linktext nicht aussagekräftig
6. Lesereihenfolge nicht sinnvoll
7. Link mit Bild mit leerem Alternativtext
8. Dekoratives Bild ohne leerem Alternativtext
9. Liste nicht als solche gekennzeichnet
10. Bild mit falschem Alternativtext

News

1. Überschrift nicht als solche gekennzeichnet
2. Schriftart schwer lesbar
3. Spalten zu eng beieinander
4. Lesereihenfolge nicht sinnvoll
5. Zweck des Links unklar
6. Link nicht optisch unterscheidbar
7. Hervorgehobener Text nicht als solcher gekennzeichnet
8. Text mit veraltetem Markup
9. Diagramm Darstellung unklar
10. Bild mit falschem Alternativtext

Tickets

1. Bild von Text ohne Textalternative, verwendet als Überschrift
2. Tabellenspaltenüberschriften nicht als solche gekennzeichnet
3. Verwendung von CSS zur Vermittlung semantischer Bedeutung
4. Bild von Text mit falscher Textalternative
5. Datentabelle ohne semantische Struktur
6. Unzureichender Farbkontrast
7. Verwendete Abkürzungen werden nicht erläutert.
8. Keine semantischen Markups und schwer lesbarer Text

Survey

1. Anweisungen zum Ausfüllen des Formulars fehlen
2. Radiobuttons sind nicht innerhalb des Codes gruppiert
3. Radiobutton ist nicht mit seiner Beschriftung verknüpft
4. Selectbox fehlt die Beschriftung
5. Selectbox ist nicht einfach über die Tastatur bedienbar
6. Nicht sinnvolle Lesereihenfolge
7. Lesereihenfolge nicht sinnvoll

Templates

1. Redundanter Seiten-Titel
2. Bild mit falschem Alternativtext
3. Automatische Seitenumleitung bei Änderung einer Selectbox
4. Skript mit einigen Tools nicht kompatibel
5. Keine Mechanismen zur Umgehung wiederholender Inhalte
6. Überschrift nicht als solche gekennzeichnet
7. Bild ohne Alternativtext
8. Seitenkomponenten nicht fokussierbar
9. Linktext nicht aussagekräftig und optisch nicht unterscheidbar
10. Unzureichender Farbkontras

3.3 Definierung der Anforderungen

Für eine übersichtliche Darstellung, der Barrieren von den Demo-Webseiten und der anschließenden Testergebnissen, wurden die aufgelisteten Barrieren nach Anforderungen gruppiert. Dadurch konnte aufgezeigt werden, ob ganze Anforderungsfelder von den Prüftools nicht abgedeckt wurden. Hierzu hätten die Barrieren nach den zugehörigen Kriterien der WCAG gruppiert werden können. Die WCAG sind allerdings in sich komplex und es konnten in den meisten Fällen mehrere Kriterien zu einer Barriere zugeordnet werden, was eine Gruppierung erschwerte. Außerdem bilden die WCAG zwar die Grundlage für internationale Richtlinien und Gesetze, jedoch sind sie mit ihren insgesamt 86 Kriterien auch sehr umfangreich. Dies würde die Darstellung und die anschließende Bewertung sehr detailliert und teilweise redundant gestalten, was eine eher unübersichtliche Auswertung zur Folge hätte.

Daher erfolgte eine Gruppierung der Barrieren nach 18 definierten Anforderungen. Diese Anforderungen sind die Resultate einer Forschungsarbeit von Endres, Steckert und Tuncer aus dem Jahr 2025 [35]. Der Fokus dieser 18 Anforderungen lag auf den in Deutschland geltenden rechtlichen Rahmenbedingungen und orientiert sich an Gesetzen, Normen und Richtlinien. Die Zielsetzung dieser Arbeit war es, einen Leitfaden für Softwareentwickler:innen zu kreieren, der die Anforderungen und Prüfkriterien für barrierefreie digitale Anwendungen für Desktop und mobile Endgeräte, zusammenfasst [35].

Der Arbeit ist eine ausführliche Literaturarbeit vorangestellt, welche die WCAG und die EN 301 459 als Grundlage heranzieht. Nach der Beleuchtung und Analyse aktueller Gesetze, Normen und Richtlinien wurde der Fokus

des Leitfadens auf die WCAG 2.1, die BITV 2.0 und die BFSGV gelegt. Daraus entstand eine tabellarische Neugruppierung der Anforderungen sowie eine Zusammenführung redundanter Anforderungen. Der vorliegende Leitfaden ist praxisorientiert aufgebaut, sodass zu jeder Anforderung eine entsprechende Beschreibung sowie Prüfkriterien aufgeführt werden [35]. Die nachfolgende Aufzählung präsentiert die insgesamt 18 Anforderungen. Eine detaillierte Darstellung des Leitfadens, einschließlich Beschreibungen und Prüfkriterien, ist dem Anhang unter „18 Anforderugen“ zu entnehmen.

Anforderungen:

1. Labels für alle Daten Eingabefelder und – Ausgabefelder, sowie für gruppierte Informationen
2. Textalternativen für nicht-textuellen Inhalt
3. Animationen, Audio- und Videosteuerung (Zeitbasierte Medien)
4. Barrierefreie Struktur durch korrekte Verwendung von HTML-Elementen
5. Farb- und Kontrasteinstellung
6. Zwei-Kanal-Prinzip und Unterscheidbarkeit
7. Schrift, Textvergrößerung bis 200 % und Zeichenabstand
8. Vermeiden von flackernden und blinkenden Inhalten
9. Tastaturbedienbarkeit Interaktiver Elemente
10. Alternative zugängliche Bedienungsformen
11. Interoperabilität mit unterstützenden, assistiven Technologien
12. Navigierbarkeit und konsistente Navigation
13. Bildschirmausrichtung
14. Ausreichend Zeit und Zeitbegrenzungen
15. Fehlervermeidung und korrekte Fehlerbehandlung
16. Aktivierung von Barrierefreiheitsfunktionen, Benutzerdefinierte Einstellungen und Dokumentation
17. Privatsphäre
18. Alternativen zur biometrischen Identifizierung und Steuerung

3.4 Datenaufbereitung und Bewertungsmethode

Die im Rahmen dieser Arbeit dokumentierten Barrieren beider Demo-Webseiten wurden mittels Tabellen zusammengeführt. Dabei wurden redundante Barrieren entfernt. Die Tabellen dienen einer übersichtlichen und vergleichbaren Grundlage, mittels derer aufgezeigt werden kann, in welchem Umfang die Tools die aufgelisteten Barrieren erkennen konnten. Eine Zuordnung zur ursprünglichen Quelle bleibt über eine zusätzliche Spalte nachvollziehbar. Für die Struktur der Ergebnisse wurden die Barrieren jeweils den 18 definierten Anforderungen zugeordnet. In Ausnahmefällen wäre es möglich gewesen, einer Barriere mehrere Anforderungen zuzuordnen. In diesen Fällen wurde stets die Hauptanforderung der jeweiligen Barriere berücksichtigt, sodass jeder Barriere nur einer Anforderung zugeordnet ist. Zur besseren Nachvollziehbarkeit sind alle 18 Anforderungen aufgeführt, einschließlich jener, für die keine Barrieren ermittelt werden konnten. Dadurch wird sichtbar, welche Aspekte barrierefreier Gestaltung die Demo-Webseiten nicht abdeckten.

Die Testergebnisse zur Bewertung der Effektivität sind in den letzten beiden Spalten der Tabellen dargestellt. Symbolische und farbliche Markierungen verdeutlichen, ob das Tool eine Barriere erkennen konnte ✓ oder nicht ✗. Eine positive Bewertung liegt nur vor, wenn ein Tool alle Instanzen einer Barriere vollständig identifizieren konnte. Wurde auch nur ein einzelnes Vorkommen übersehen, galt das Tool in Bezug auf diese Art von Barriere als nicht zuverlässig. Die Häufigkeit des Auftretens einzelner Barrieren wurde daher in der Bewertung nicht berücksichtigt. Die auf diese Weise aufbereiteten Daten bilden die Grundlage für die Darstellung der Testergebnisse im folgenden Kapitel.

4 Testergebnisse

Die Resultate der durchgeführten Tests werden im Folgenden dargestellt. Die Tabellen zeigen, welche Barrieren auf den untersuchten Demo-Webseiten dokumentiert wurden und ob diese von den beiden Prüftools erkannt werden konnten. Eine detaillierte Analyse der Ergebnisse erfolgt im nächsten Kapitel.

1 Labels für alle Daten Eingabe- und Ausgabefelder, sowie für gruppierte Informationen

Tabelle 3: Labels für alle Daten Eingabe- und Ausgabefelder, sowie für gruppierte Informationen

Barriere	Quelle	axe	WAVE
Radiobuttons sind nicht innerhalb des Codes gruppiert	W3C	✗	✓
Radiobutton ist nicht mit seiner Beschriftung verknüpft	W3C	✓	✓
Selectbox fehlt die Beschriftung	W3C	✓	✓
Text-input fehlt die Beschriftung	UW	✓	✓
Checkbox fehlt die Beschriftung	UW	✓	✓

2 Textalternativen für nicht-textuellen Inhalt

Tabelle 4: Textalternativen für nicht-textuellen Inhalt

Barriere	Quelle	axe	WAVE
Bild ohne Alternativtext	UW/W3C	✓	✓
Dekoratives Bild ohne leerem Alternativtext	UW/W3C	✗	✓
Bild mit falschem Alternativtext	W3C	✗	✗
Bild von Text mit falscher Alternativtext	W3C	✗	✗
Bild mit unzugänglichem Alternativtext (title="image")	W3C	✗	✗
Link mit Bild mit leerem Alternativtext (alt="")	W3C	✓	✓

Unklare Darstellung des Diagramms mit leerem Alternativtext (alt="")	W3C	✗	✗

3 Animationen, Audio- und Videosteuerung (Zeitbasierte Medien)

Tabelle 5: Animationen, Audio- und Videosteuerung (Zeitbasierte Medien)

Barriere	Quelle	axe	WAVE
Unzugängliche Audioinhalte	UW	✗	✓
Unzugängliche visuelle Inhalte	UW	✗	✓

4 Barrierefreie Struktur durch korrekte Verwendung von HTML-Elementen

Tabelle 6: Barrierefreie Struktur durch korrekte Verwendung von HTML-Elementen

Barriere	Quelle	axe	WAVE
Bild von Text wird als Überschrift verwendet wird, ohne Alternativtext	W3C	✓	✓
Es sind keine Überschriften vorhanden	UW	✓	✓
Überschrift nicht als solche gekennzeichnet	UW/W3C	✗	✓
Hervorgehobener Text nicht als solcher gekennzeichnet	W3C	✗	✗
Lesereihenfolge nicht sinnvoll	W3C	✗	✗
Text mit veraltetem Markup	W3C	✗	✗
Verwendung von CSS zur Vermittlung semantischer Bedeutung	W3C	✗	✗
Abkürzungen sind nicht gekennzeichnet	UW/W3C	✗	✗
Liste nicht als solche gekennzeichnet	W3C	✗	✗
Datentabelle ohne semantische Struktur	W3C	✗	✓
Fehlende Landmark-Bereiche	UW	✓	✓
Verwendung von Links u. Buttons mit falschem HATML-Element	UW	✗	✗

5 Farb- und Kontrasteinstellung

Tabelle 7: Farb- und Kontrasteinstellung

Barriere	Quelle	axe	WAVE
Unzureichender Farbkontrast (4,5:1)	UW/W3C	✓	✓

6 Zwei-Kanal-Prinzip und Unterscheidbarkeit

Tabelle 8: Zwei-Kanal-Prinzip und Unterscheidbarkeit

Barriere	Quelle	axe	WAVE
Link nicht optisch unterscheidbar	W3C/UW	✗	✗
Verwendung von Farbe zur Vermittlung von Informationen (Formular)	UW	✗	✗

7 Schrift, Textvergrößerung bis 200 % und Zeichenabstand

Tabelle 9: Schrift, Textvergrößerung bis 200 % und Zeichenabstand

Barriere	Quelle	axe	WAVE
Schriftart schwer lesbar	W3C	✗	✗
Spalten zu eng beieinander	W3C	✗	✗
Schwer lesbarer Text und kein semantisches Markup	W3C	✗	✓

8 Vermeiden von flackernden und blinkenden Inhalten

9 Tastaturbedienbarkeit Interaktiver Elemente

Tabelle 10: Tastaturbedienbarkeit Interaktiver Elemente

Barriere	Quelle	axe	WAVE
Selectbox ist nicht über die Tastatur bedienbar	W3C	✗	✗
Seitenkomponenten nicht fokussierbar	W3C	✗	✓
Navigationsmenü unterstützt nur die Tabulatortaste	UW	✗	✗
Keine sichtbare Fokusanzeige	UW	✗	✗

Unzugängliches Karussell	UW	✗	✗
Unzugänglicher modaler Dialog	UW	✗	✗

10 Alternative zugängliche Bedienungsformen

Tabelle 11: Alternative zugängliche Bedienungsformen

Barriere	Quelle	axe	WAVE
Unzugängliches CAPTCHA	UW	✗	✗

11 Interoperabilität mit unterstützenden, assistiven Technologien

Tabelle 12: Interoperabilität mit unterstützenden, assistiven Technologien

Barriere	Quelle	axe	WAVE
Skript mit einigen Tools nicht kompatibel	W3C	✗	✗

12 Navigierbarkeit und konsistente Navigation

Tabelle 13: Navigierbarkeit und konsistente Navigation

Barriere	Quelle	axe	WAVE
Linktext nicht aussagekräftig	UW/W3C	✗	✗
Zweck des Links unklar	W3C	✗	✗
Redundanter Seiten-Titel	W3C	✗	✗
Automatische Seitenumleitung bei Änderung einer Selectbox	W3C	✗	✓
Keine Mechanismen zur Umgehung wiederholender Inhalte	W3C	✗	✗
Kein Link „Zum Hauptinhalt springen“	W3C	✗	✗
Redundanter Linktext	UW	✗	✗
Unzugängliches Navigationsmenü	UW	✗	✗

13 Bildschirmausrichtung

14 Ausreichend Zeit und Zeitbegrenzungen

15 Fehlervermeidung und korrekte Fehlerbehandlung

Tabelle 14: Fehlervermeidung und korrekte Fehlerbehandlung

Barriere	Quelle	axe	WAVE
Die zum Ausfüllen des Formulars erforderlichen Anweisungen fehlen	W3C	✗	✗
Unzugängliche Eingabevalidierung	UW	✗	✗

16 Aktivierung von Barrierefreiheitsfunktionen, Benutzerdefinierte Einstellungen und Dokumentation

Tabelle 15: Aktivierung von Barrierefreiheitsfunktionen, Benutzerdefinierte Einstellungen und Dokumentation

Barriere	Quelle	axe	WAVE
Sprache nicht angegeben	UW/W3C	✓	✓

17 Privatsphäre

18 Alternativen zur biometrischen Identifizierung und Steuerung

5 Auswertung der Testergebnisse

Im Folgenden werden die Testergebnisse, die auf Grundlage der zuvor erstellten Tabellen ermittelt wurden, ausgewertet. Wie bereits in Kapitel 3.4 aufgezeigt, umfasst die Auswertung sämtliche der 18 definierten Anforderungen an barrierefreie Webseiten. Jedoch konnte den Anforderungen 8 (Vermeiden von flackernden und blinkenden Inhalten), 13 (Bildschirmausrichtung), 14 (Ausreichend Zeit und Zeitbegrenzungen), 17 (Privatsphäre) und 18 (Alternativen zur biometrischen Identifizierung und Steuerung) keine Barrieren zugeordnet werden.

Die Analyse der Testergebnisse ergab, dass beide Prüftools gemeinsam in der Lage waren, 20 der 51 verschiedenen Arten von Barrieren erfolgreich zu identifizieren und zu kennzeichnen. Dies entspricht rund 39 % aller in den Tabellen aufgeführten Barrieren. Dabei identifizierte axe etwa 22 % und WAVE etwa 39 % der getesteten Arten von Barrieren korrekt. Die Tabellen zeigen jedoch lediglich auf, ob ein Tool alle Instanzen einer Art von Barrieren vollständig erkennen konnte. In diesem Kapitel werden daher besonders solche Fälle näher betrachtet, in denen nur eine teilweise Erkennung möglich war. Dadurch soll nachvollziehbar aufgezeigt werden, warum bestimmte Barrieren von den Tools erkannt wurden und in welchen Bereichen ihre Erkennungslücken liegen. Die Auswertung erfolgt zunächst in einer übergreifenden Betrachtung, in der die Funktionsweisen beider Tools direkt miteinander verglichen werden können. Anschließend werden die Ergebnisse und Erkennungsstrategien der einzelnen Tools erläutert.

Wie aus Tabelle 3 zu entnehmen ist, konnten beide Tools nahezu alle Barrieren der **Anforderung 1**: „Labels für alle Daten-Eingabefelder und -Ausgabefelder sowie für gruppierte Informationen“ erfolgreich identifizieren. Das Tool axe ordnete sämtliche Typen von Eingabefeldern dem Fehlertyp „Form elements must have labels“ zu. Eine Ausnahme bilden dabei Select-Elemente, die gesondert unter „Select element must have an accessible name“ aufgeführt wurden. Lediglich das Erkennen von nicht **gruppierten Radiobuttons innerhalb des Codes** war für axe nicht möglich. WAVE konnte fehlende Gruppierungen hingegen erkennen und wies den betreffenden

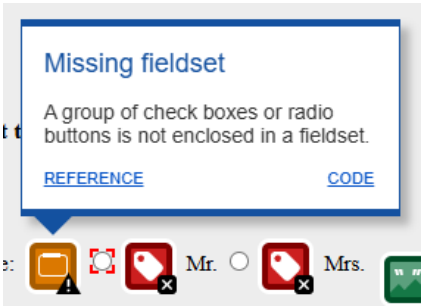


Abbildung 4: WAVE Overlay UI Quelle: https://www.w3.org/WAI/demos/bad/before/survey.html

Elementen den Fehlertyp „Missing fieldset“ zu. Dieser überprüft sowohl Checkboxen als auch Radiobuttons auf eine korrekte Gruppierung innerhalb eines Formulars. Alle weiteren Barrieren dieser Anforderung listete WAVE unter dem Fehlertyp „Missing form label“ auf (siehe Abbildung 4).

Der Fehlertyp „**Bilder ohne Alternativtext**“ konnte von beiden Tools gleichermaßen erfolgreich identifiziert werden (siehe Tabelle 4). Das Tool axe gruppierte dabei alle identifizierten Bilder ohne Alternativtext dem Fehlertyp „Images must have alternate text“ zu. WAVE differenzierte diesen Fehlertyp zusätzlich und unterschied zwischen den Kategorien „Missing alternative text“, „Linkes Image missing alternative text“ und „Spacer image missing alternative“ (siehe Abbildung 5).

Beide Prüftools konnten außerdem zuverlässig erkennen, ob ein **Bild innerhalb eines Links einen leeren Alternativtext** besaß (siehe Tabelle 4). Leere Alternativtexte sind gemäß der WCAG grundsätzlich nicht falsch. Für dekorative Bilder gilt die Verwendung eines leeren Alternativtextes (alt=““) als korrekte Praxis. Die Richtlinien formulieren hierzu: „If non-text content is pure decoration, is used only for visual formatting, or is not presented to users, then it is implemented in a way that it can be ignored by assistive technology.“ [36]. Bei informativen Bildern hingegen, ist die Angabe eines aussagekräftigen Alternativtexts zwingend erforderlich [36]. Da ein verlinktes Bild von den Prüftools als informatives Element korrekt eingestuft wird, bewerteten beide diese Fälle als Fehler. Allerdings waren die geprüften Tools nicht in der Lage, eigenständig zwischen dekorativen und informativen Bildern zu unterscheiden. Warum das Tool WAVE in einzelnen Fällen auch **dekorative Bilder mit Alternativtext** beanstandete, wird im entsprechenden Abschnitt „WAVE-Analyseansätze“ näher erläutert.

Auf beiden Demo-Webseiten konnten Barrieren in Verbindung mit Tabellen festgestellt werden (siehe Tabelle 6). Im Folgenden wird die Barriere „**Datentabelle ohne semantische Struktur**“ näher beleuchtet. Beide Prüftools waren gleichermaßen in der Lage, leere Tabellenüberschriften (<th><th>) zu erkennen und haben dies als Fehler markiert. Jedoch erkannte lediglich

WAVE, ob innerhalb einer Tabelle eine Überschrift (<th>-Tag) vorhanden war. Tabellen ohne eine Tabellenüberschrift wurden von WAVE mit der Warnung „Layout table“ versehen (siehe Abbildung 5). Es sei anzumerken, dass die untersuchten Tabellen weder einen Titel (caption-Tag) noch korrekt gesetzte scope-Attribute oder header-Attribute aufwiesen. Diese Mängel wurden von beiden Tools nicht erkannt.

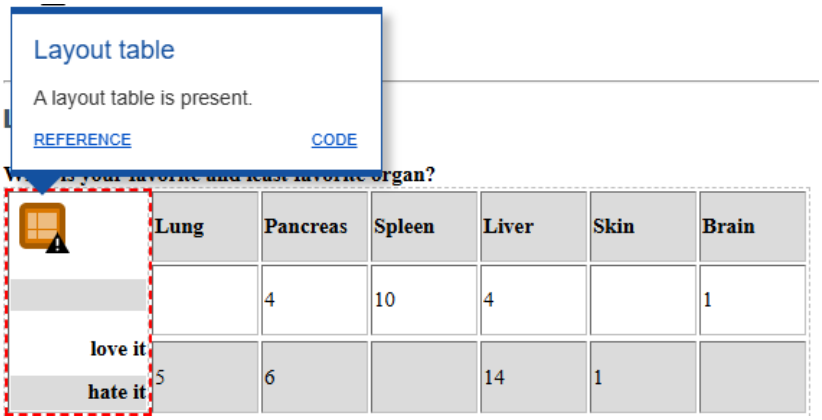


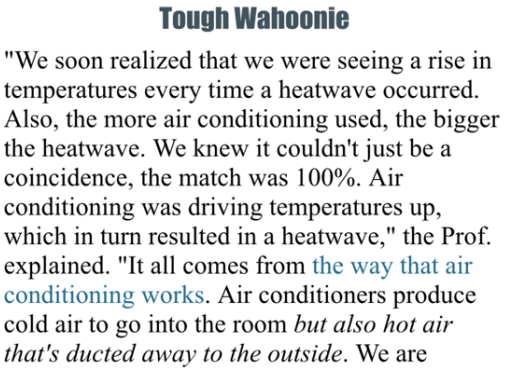
Abbildung 5: WAVE Overlay UI
Quelle: https://www.w3.org/WAI/demos/bad/before/survey.html

Darüber hinaus zeigten sich bei beiden Tools deutliche Erkennungslücken bei den **Anforderungen zur Tastaturbedienbarkeit** (siehe Tabelle 10) und **Navigierbarkeit** (siehe Tabelle 13). Es wurden nur vereinzelte Fehler erkannt, wie beispielsweise fehlende Fokusbereiche oder eine automatische Seitenumleitung bei Änderung einer Selectbox.

5.1 Axe-Analyseansätze

Das Tool axe zeigte bei der Erkennung von **optisch nicht ausreichend vom Fließtext unterscheidbaren Links** ein unterschiedliches Verhalten auf (siehe Tabelle 8). In einigen Fällen ordnete axe Link-Elementen den Fehlertyp „Links must be distinguished from surrounding text in a way that does not rely on color“ zu. Grundlage dieser Bewertung ist ein erforderlicher Mindestkontrast von 3:1 zwischen Link- und Fließtextfarbe [37]. Auf der Demo-Webseite von W3C konnte axe keine Barriere dieser Art feststellen, da der vorhandene Farbkontrast von 3,6:1 den Mindestwert überstieg. Dort war die Fließtextfarbe Schwarz (#000000) und die Linkfarbe ein helles Blau (#226c8e) (siehe Abbildung 6). Auf der Demo-Webseite der UW hingegen wurde der Kontrast von 2,8:1 korrekt als unzureichend erkannt. Hier war der Fließtext ebenfalls schwarz (#000000) und die

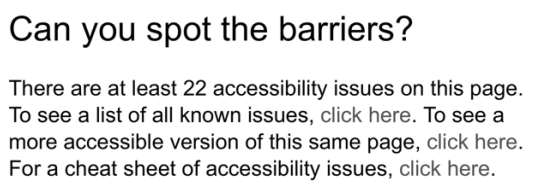
Linkfarbe grau (#555555) (siehe Abbildung 7). Damit konnte axe in diesem Fall die Anforderungen 6 des **Zwei-Kanal-Prinzips und Unterscheidbarkeit** in einem gewissen Rahmen erfüllen.



Tough Wahoonie

"We soon realized that we were seeing a rise in temperatures every time a heatwave occurred. Also, the more air conditioning used, the bigger the heatwave. We knew it couldn't just be a coincidence, the match was 100%. Air conditioning was driving temperatures up, which in turn resulted in a heatwave," the Prof. explained. "It all comes from the way that air conditioning works. Air conditioners produce cold air to go into the room *but also hot air that's ducted away to the outside*. We are

Abbildung 6: Textblock der W3C-Demo-Webseite Quelle: https://www.w3.org/WAI/demos/bad/before/news.html



Can you spot the barriers?

There are at least 22 accessibility issues on this page. To see a list of all known issues, click here. To see a more accessible version of this same page, click here. For a cheat sheet of accessibility issues, click here.

Abbildung 7: Textblock der UW-Demo-Webseite Quelle: https://www.washington.edu/accesscomputing/AU/before.html

5.2 WAVE-Analyseansätze

Die Ergebnisse von Tabelle 4 zeigen, dass WAVE den Fehlertyp „**Bild mit falschem Alternativtext**“ nicht immer korrekt identifizieren konnte. Dennoch ist festzuhalten, dass das Tool in mehreren Fällen fehlerhafte oder unzureichende Alternativtexte erkannt hatte. So kennzeichnete WAVE auf der UW-Webseite unter anderem einzelne Bilder mit der Warnung „Suspicious alternative text“. Auf den Seiten der Demo-Webseite von W3C hingegen konnte WAVE kaum unzureichende Alternativtexte identifizieren, obwohl dort eine Vielzahl unterschiedlich ausgeprägter Fehler auftraten. Bei der Erkennung von **Nicht aussagekräftige Linktexte** zeigte WAVE ein ähnliches Verhalten auf (siehe Tabelle 13). Zum Teil markierte WAVE Link-Elemente mit der Warnung „Suspicious linktext“. Über diese Funktion konnten beispielsweise Linktexte wie „Click here“ identifiziert werden, andere wie „Read More“ jedoch nicht.

Dieses Verhalten ist darauf zurückzuführen, dass WAVE die inhaltliche Bedeutung eines Alternativtextes nicht interpretieren kann. Stattdessen überprüft WAVE Alternativ- und Linktexte anhand typischer Muster potenziell für unzureichender Formulierungen. So wurden beispielsweise lange

Alternativtexte ab einer Zeichenanzahl von mehr als 100 Zeichen mit der Warnung „Long alternative text“ gekennzeichnet. Alternativtexte wurden unter anderem auf Begrifflichkeiten wie „graphic of“, „bullet“ oder „image of“ geprüft oder auf reine Dateinamen. So konnte das Prüftool WAVE Alternativtexte identifizieren, die eventuell keinen echten Informationswert bieten. Bei Linktexten ging WAVE ähnlich vor und überprüft ebenfalls auf Wortmuster wie die typischen Füllphrasen „click here“, „more“, „more ...“ oder „read more“ (siehe Abbildung 8). Da diese Listen jedoch begrenzt sind, blieb der Linktext „Read More...“ auf der Demo-Webseite von W3C unentdeckt.

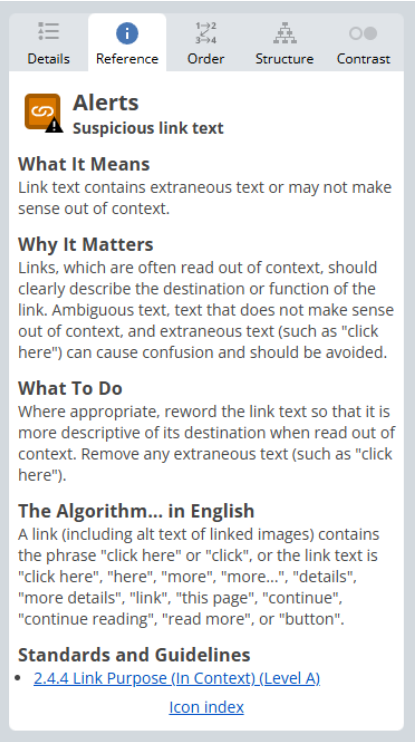


Abbildung 8: WAVE Reference „Suspicious link text“
Quelle: Screenshot der UI beim Scan der Seite https://www.w3.org/WAI/demos/bad/before/home.html

Auf diese Weise konnte WAVE auch **dekorative Bilder ohne leeres alt-Attribut** erkennen (siehe Tabelle 4). Da die betroffenen Bilder mit nichtssagenden Alternativtexten wie „bullet“ (W3C-Seite) oder „horizontal line graphic“ (UW-Seite) versehen waren. WAVE stufte diese ebenfalls als „Suspicious alternative text“ ein und erfasste sie somit korrekt als potenzielle Fehler. Darüber hinaus erkennt WAVE grundsätzlich **Bilder mit leeren Alternativtexten** und kennzeichnet diese. Befindet sich ein leeres alt-Attribut innerhalb eines Links, wird dies, wie zuvor erläutert, als Error eingestuft, da ein verlinktes Bild stets als informativ gilt. In allen anderen Fällen markiert WAVE leere Alternativtexte dagegen als Feature, da diese für dekorative Bilder gemäß WCAG korrekt umgesetzt sind.

Bei der Überprüfung der Demo-Webseite von W3C durch WAVE konnte die Barriere „**Keine semantischen Markups und schwer lesbarer Text**“ korrekt identifiziert werden (siehe Tabelle 6). WAVE kennzeichnete den betreffenden Textblock mit der Warnung „Justified text“. Damit ist gemeint, dass der Text im Blocksatz formatiert war, was die Lesbarkeit tatsächlich beeinträchtigen kann. Auch Text-Elemente mit einer kleinen Schriftgröße von

10 Pixeln oder kleiner konnten erkannt werden und wurden mit der Warnung „Very small text“ markiert.

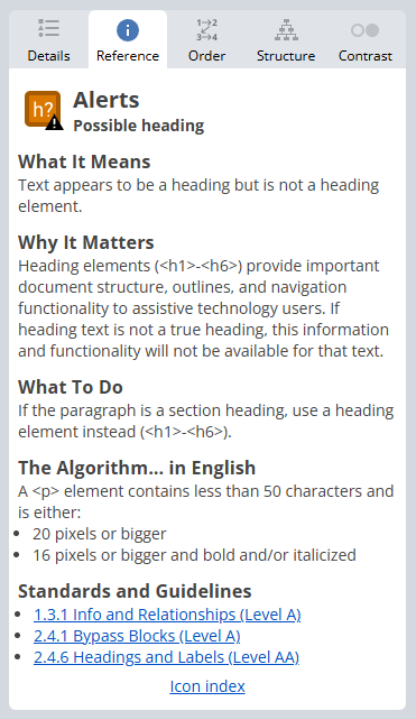


Abbildung 9: WAVE Reference „Possible heading“
Quelle: Screenshot der UI beim Scan der Seite https://www.w3.org/WAI/demos/bad/before/home.html

WAVE stellte die identifizierten Barrieren in einer eigenen Oberfläche dar. Unter dem Reiter „Reference“ wurden für jedes erkannte Problem detaillierte Informationen bereitgestellt, darunter eine textliche Beschreibung („What It Means“), die Bedeutung für die Barrierefreiheit („Why It Matters“), sowie konkrete Handlungsempfehlungen („What To Do“). Zusätzlich wird der angewendete Prüfalgorithmus kurz und bündig in Form von Stichpunkten beschrieben („The Algorithm ... in English“) und den zugehörigen WCAG-Kriterien zugeordnet. Abbildung 9 zeigt beispielhaft die WAVE-Referenzansicht zur Warnung „Possible heading“.

6 Diskussion

Die Auswertung der Testergebnisse zeigt auf, welche Barrieren die getesteten Demo-Webseiten nicht abdeckten. Fünf Anforderungen, darunter flackernde Inhalte und Zeitbegrenzungen, traten auf keiner der getesteten Seiten auf. Dadurch war es nicht möglich, die Leistungsfähigkeit der Prüftools in diesen Bereichen zu bewerten. Diese Einschränkung reduziert die Aussagekraft der Ergebnisse, da die ermittelte Effektivität der Tools nur für die tatsächlich vorkommenden Barrieren gilt. Trotzdem lassen sich auf Basis der 13 vorhandenen Tabellen aussagekräftige Erkenntnisse über die Funktionsweise und Zuverlässigkeit beider Tools ableiten. Außerdem wurden Barrieren entdeckt, die von den Demo-Webseiten nicht als solche angegeben waren. Diese wurden in der tabellarischen Auswertung nicht berücksichtigt. Die Diskussion der Ergebnisse erfolgt daher unter Berücksichtigung dieser Einschränkungen und fokussiert sich auf die identifizierten Stärken und Schwächen von axe und WAVE innerhalb der aufgeführten Testergebnisse.

Des Weiteren sei anzumerken, dass die Testergebnisse nicht uneingeschränkt für andere Testumgebungen übertragbar sind. Ein Tool, welches im Rahmen dieser Tests, alle Barrieren eines bestimmten Fehlertyps auf den Demo-Webseiten identifizierte, ist nicht zwangsläufig in der Lage, auch in anderen Umgebungen Barrieren dieser Art zu erkennen. Dennoch vermitteln die Ergebnisse dieser Arbeit ein ausreichendes Bild über die Prüftools von denen Stärken und Grenzen abzuleiten sind.

Die in der Einleitung aufgestellte Hypothese, dass sich beide Tools in ihrer Funktionsweise gegenseitig ergänzen würden, konnte im Rahmen der Analyse nicht bestätigt werden. Vielmehr zeigte sich, dass WAVE insgesamt mehr Arten von Barrieren identifizierte und alle aufgelisteten Fehler erkannte, die auch axe gefunden hatte. Dabei erkannte WAVE 20 der 51 verschiedenen Barrieren korrekt, während axe nur 11 der 51 Fehlerarten identifizierte. Nur in Einzelfällen schnitt axe besser ab. So identifizierte das Prüftool den Fehlertyp „Link nicht optisch unterscheidbar“, wenn auch nicht ausreichend, und erkannte somit zumindest teilweise eine Barriere, die für WAVE gänzlich unentdeckt blieb.

Jedoch zeigt sich hier auch eine Schwäche. Im vorliegenden Fall wurde eine relevante Anforderung an barrierefreie Webseiten korrekt formuliert: „Links must be distinguished from surrounding text in a way that does not rely on color“. Diese Regel legt nahe, dass Links nicht nur durch Farbe, sondern auch durch weitere visuelle Merkmale unterscheidbar sein sollen (Anforderung 6: Zwei-Kanal-Prinzip und Unterscheidbarkeit). In der Praxis prüft axe jedoch lediglich den Farbkontrast zwischen Link- und Fließtext. Dadurch weicht die technische Umsetzung vom Regelinhalt ab, was die Aussagekraft einzelner Prüfungsbeschreibungen relativiert.

Eine weitere Erkenntnis ist, dass keines der beiden Prüftools inhaltliche Barrieren erfassen konnte. Dies wird insbesondere bei der Analyse von Alternativ- und Linktexten deutlich. Weder axe noch WAVE waren in der Lage, einen beschreibenden Text anhand seiner Sinnhaftigkeit im Kontext zu bewerten. Für eine umfassende inhaltliche Bewertung wären KI-gestützte Prüftools erforderlich, die Bilder oder Link-Elemente analysieren und in einen Kontext setzen können. Ob ein Bild dekorativ oder inhaltlich relevant ist, kann nur schwer über einen Algorithmus entschieden werden, weshalb diese Validierung von einer manuellen Prüfung abhängig bleibt. In sämtlichen Fällen wäre es eine anwenderfreundliche Unterstützung, leere Alternativtexte mit einem Hinweis auf manuelle Überprüfung zu kennzeichnen.

Die Analyse der Testergebnisse zeigt bereits auf, dass WAVE deutlich mehr Barrieren identifizieren konnte als axe. Dies ist darauf zurückzuführen, dass axe Webseiten gemäß den WCAG auf das Vorhandensein bestimmter Tags, Attribute und Rolle des zugrunde liegenden HTML-Codes überprüft. Eine inhaltliche oder semantische Bewertung der Elemente fand nicht statt. Im Gegensatz dazu konnte das Tool WAVE Probleme erkennen, die sich nicht ausschließlich durch Code-Attribute erfassen ließen. Unter semantischen Barrieren ist zu verstehen, dass Inhalte, Strukturen oder Funktionen nicht mit den dafür vorgesehenen HTML-Elementen ausgezeichnet sind, wodurch deren Bedeutung für assistive Technologien nicht korrekt vermittelt werden kann. Der Code funktioniert technisch, vermittelt jedoch falsche oder unvollständige Informationen über die Struktur. Axe überprüfte beispielsweise nur das Vorhandensein von Überschrift-Elementen (<h1>-Tags). WAVE konnte darüber hinaus Absätze identifizieren

(<p>-Tags), die einer Überschrift entsprechen könnten. Dabei überprüfte WAVE den Inhalt eines Absatzes auf Länge und Formatierung.

Ebenso markierte das Tool WAVE Textabschnitte mit potenziellen Lesbarkeitsproblemen, wenn eine kleine Schriftgröße oder Blocksatz verwendet wurde. Im getesteten Rahmen erkannte WAVE einen schwer lesbaren Text daher korrekt. Allerdings wurden nicht korrekt semantisch gekennzeichnete hervorgehobene Inhalte oder Abkürzungen in Textabschnitten und Tabellen nicht erkannt. Diese können ebenfalls die Lesbarkeit beeinträchtigen.

In dieser teilweise korrekten Überprüfung liegt das Problem. WAVE überzeugte zwar durch eine hohe Transparenz bei der Darstellung seiner Prüflogik. So wird für jeden identifizierten Fehlertyp nachvollziehbar beschrieben, auf welche spezifischen Algorithmen, Regeln oder Formatierungsmerkmale sich die Bewertung bezieht. Diese Offenlegung erleichtert es Anwender:innen, die Ursache eines Hinweises nachzuvollziehen. Eine teilweise oder unvollständige Erkennung von Barrieren kann jedoch auch unter diesen Voraussetzungen problematisch sein, da den Anwender:innen durch die identifizierten Fehlertypen eine trügerische Sicherheit vermittelt wird. Zudem sind Anwender:innen darauf angewiesen sich bei der Anwendung von Prüftools intensiv mit den Kriterien der WCAG auseinanderzusetzen, denn welche der von der WCAG definierten Barrieren die Prüftools nicht abdecken, wird in der Browser-Extension nicht aufgezeigt. Nur so kann die Gefahr vermieden werden, dass tatsächliche bestehende und von den WCAG als solche definierten Barrieren übersehen und nicht behoben werden. Dies ist notwendig um die Anforderungen hinsichtlich der Barrierefreiheit von Webseiten im Internet nun auch in Deutschland geltenden, rechtlichen Bestimmungen zu genügen.

7 Fazit und Ausblick

Fazit

Das Ziel der vorliegenden Arbeit war es, durch die Analyse von zwei automatisierten Accessibility-Prüftools den Umfang ihrer Praxistauglichkeit sowie bestehende Grenzen in der Bewertung von Webseiten genau zu beleuchten. Die Forschungsergebnisse zeigen, dass axe und WAVE gemeinsam rund 39 % der auf den Demo-Webseiten dokumentierten Arten von Barrieren identifizierten. Ein Ergänzungsverhältnis zwischen beiden Prüftools konnte dabei jedoch nicht festgestellt werden. So identifizierte WAVE abgesehen von den elf gemeinsamen Arten von Barrieren, noch neun weitere, die für axe unentdeckt blieben. Während axe vorrangig den HTML-Code auf vorhandene HTML-Elemente und -Attribute überprüfte, konnte WAVE zusätzlich semantische und visuelle Aspekte erfassen. Dennoch stießen beide Prüftools hinsichtlich der inhaltlichen Bewertung an ihre Grenzen. Damit erfassen automatisierte Prüftools zwar Strukturelle oder technische Fehler, die sich aus dem Code oder dem Design ergeben. Jedoch keine inhaltlichen oder kontextuellen Barrieren, die für eine umfassende Barrierefreiheit entscheidend sind.

Darüber hinaus blieben mehrere dokumentierte Barrieren vollständig unerkannt. Besonders betroffen waren die Anforderungsbereiche „Zwei-Kanal-Prinzip und Unterscheidbarkeit", „Alternative zugängliche Bedienungsformen", „Interoperabilität mit unterstützenden, assistiven Technologien" sowie „Fehlervermeidung und korrekte Fehlerbehandlung". In keinem dieser Bereiche konnten WAVE oder axe die vorhandenen Barrieren identifizieren. Dies deutet darauf hin, dass automatisierte Prüftools insbesondere dort an ihre Grenzen stoßen, wo Barrieren interaktiv, kontextabhängig oder nicht eindeutig algorithmisch prüfbar sind.

Durch diese empirische Forschung wurde somit gezeigt, dass automatisierte Accessibility-Prüftools innerhalb ihres Anwendungsbereichs eine wertvolle, wenn auch begrenzte Unterstützung bieten. Sie ermöglichen eine schnelle Erstprüfung, schaffen Bewusstsein für Barrierefreiheit und sind dennoch nicht in der Lage, ein großes Spektrum an Barrieren abzudecken.

Für die Praxis bedeutet dies, dass Tool-Anbieter ihre Prüftools künftig stärker um semantische und kontextuelle Dimensionen erweitern sollten, um eine effektivere Bewertung zu ermöglichen. Zukünftige Weiterentwicklungen, etwa durch KI-gestützte Verfahren, könnten hierbei unterstützend sein, indem sie inhaltliche Zusammenhänge besser erkennen und interpretieren. Gleichzeitig ist eine transparente Darstellung der Prüfumfänge und ihrer Grenzen notwendig, um Fehlinterpretationen durch Anwender:innen zu vermeiden. Bis dahin bleibt eine Auseinandersetzung mit den verwendeten Algorithmen und eine kritische Einordnung automatischer Prüfergebnisse unerlässlich.

Ausblick

Die im Rahmen dieser Arbeit erstellten Tabellen bilden eine Grundlage für aufbauende Tests und detailliertere Analysen. Mithilfe der verwendeten Demo-Webseiten und der daraus resultierenden Tabellen ist es möglich, weitere Prüftools unter denselben Voraussetzungen zu testen und die Ergebnisse mit den bestehenden zu vergleichen. In diesem Zusammenhang wäre ebenso das Testen manueller oder KI-unterstützter Prüftools möglich. Zusätzlich besteht die Möglichkeit die Tabellen um weitere Barrieren zu erweitern. So könnte durch das Einbeziehen weiterer Webseiten die Bandbreite der Arten von Barrieren vergrößert und ergänzt werden. Zusätzlich bieten die Demo-Webseiten aufgrund des Open-Source-Codes die Option, den HTML-Code zu erweitern und gezielt bestimmte Barrieren zu testen. Insbesondere die Anforderungen, die in der Arbeit nicht abgedeckt wurden, könnten auf diese Weise nachträglich umgesetzt und getestet werden. Beispielsweise wäre es möglich die Demo-Webseite der UW um ein blinkendes Element zu erweitern. Insgesamt bietet die Arbeit damit zahlreiche Ansätze, wodurch die Leistungsfähigkeit automatisierte Prüftools näher analysiert werden kann.

8 Verzeichnisse

8.1 Literatur

[1] "Barrierefreiheits-stärkungs-gesetz (BFSG)," *Portal Barrierefreiheit der Dienstekonsolidierung des Bundes*, 02. November 2023. Zugriff am: 8. Oktober 2025. [Online.] Verfügbar: https://www.barrierefreiheit-dienstekonsolidierung.bund.de/Webs/PB/DE/gesetze-und-richtlinien/barrierefreiheits-staerkungsgesetz/barrierefreiheitsstaerkungsgesetz-node.html

[2] A. Mensch, "Dritter Testbericht: So barrierefrei sind Online-Shops in Deutschland 2025," [Online]. Verfügbar unter: https://delivery-aktion-mensch.stylelabs.cloud/api/public/content/aktion-mensch-dritter-testbericht-onlineshops-barrierefreiheit.pdf?v=8d8cb330

[3] T. Nguyen, "Evaluating Automated Accessibility Checker Tools," [Online]. Verfügbar unter: https://cedar.wwu.edu/cgi/viewcontent.cgi?article=1792&context=wwu_honors

[4] J. R. Pool, "Accessibility Metatesting," in *20th International Web for All Conference*, Austin TX USA, 2023, S. 1–4, doi: 10.1145/3587281.3587282.

[5] J. Ara, C. Sik-Lanyi, A. Kelemen und T. Guzsvinecz, "An inclusive framework for automated web content accessibility evaluation,"

[6] L. García-Santiago und M.-D. Olvera-Lobo, "How accessibility guidelines are used in Spanish World Heritage websites: an exploratory study," *LHT*, Jg. 39, Nr. 1, S. 144–165, 2021, doi: 10.1108/LHT-05-2019-0113.

[7] BFSG. "§ 3 BFSG Barrierefreiheit, Verordnungsermächtigung - Barrierefreiheitsstärkungsgesetz (BFSG)." Zugriff am: 26. September 2025. [Online.] Verfügbar: https://bfsg-gesetz.de/3-bfsg/

[8] "Web Content Accessibility Guidelines (WCAG) 2.2." Zugriff am: 6. November 2025. [Online.] Verfügbar: https://www.w3.org/TR/WCAG22/#abstract

[9] "Web Content Accessibility Guidelines 1.0." Zugriff am: 27. September 2025. [Online.] Verfügbar: https://www.w3.org/TR/WAI-WEBCONTENT/

[10] "WCAG 2.2: Neue Version im Dezember 2024 veröffentlicht," *Portal Barrierefreiheit der Dienstekonsolidierung des Bundes*, 02. November 2023. Zugriff am: 27. September 2025. [Online.] Verfügbar: https://www.barrierefreiheit-

dienstekonsolidierung.bund.de/Webs/PB/DE/gesetze-und-richtlinien/wcag/wcag_2_2/wcag-2-2-node.html

[11] "Web Content Accessibility Guidelines (WCAG) 2.2." Zugriff am: 8. Oktober 2025. [Online.] Verfügbar: https://www.w3.org/TR/WCAG22/#wcag-2-layers-of-guidance

[12] "Web Content Accessibility Guidelines (WCAG) 2.2." Zugriff am: 8. Oktober 2025. [Online.] Verfügbar: https://www.w3.org/TR/WCAG22/#contrast-minimum

[13] "Understanding Success Criterion 1.4.3: Contrast (Minimum) | WAI | W3C." Zugriff am: 13. November 2025. [Online.] Verfügbar: https://www.w3.org/WAI/WCAG22/Understanding/contrast-minimum.html

[14] "Web Content Accessibility Guidelines (WCAG) 2.2." Zugriff am: 8. Oktober 2025. [Online.] Verfügbar: https://www.w3.org/TR/WCAG22/#contrast-enhanced

[15] "Understanding Success Criterion 1.4.6: Contrast (Enhanced) | WAI | W3C." Zugriff am: 13. November 2025. [Online.] Verfügbar: https://www.w3.org/WAI/WCAG22/Understanding/contrast-enhanced.html

[16] "Art 3 GG - Einzelnorm." Zugriff am: 8. Oktober 2025. [Online.] Verfügbar: https://www.gesetze-im-internet.de/gg/art_3.html

[17] "BGG - Gesetz zur Gleichstellung von Menschen mit Behinderungen." Zugriff am: 8. Oktober 2025. [Online.] Verfügbar: https://www.gesetze-im-internet.de/bgg/BJNR146800002.html

[18] "BITV 2.0 - Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz." Zugriff am: 8. Oktober 2025. [Online.] Verfügbar: https://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html

[19] "Gesetz zur Umsetzung der Richtlinie (EU) 2019/882 des Europäischen Parlaments und des Rates über die Barrierefreiheitsanforderungen für Produkte und Dienstleistungen und zur nderung anderer Gesetze," [Online]. Verfügbar unter: https://www.bmas.de/SharedDocs/Downloads/DE/Gesetze/barrierefreiheitsstaerkungsgesetz.pdf?__blob=publicationFile&v=3

[20] "BFSGV - Verordnung über die Barrierefreiheitsanforderungen für Produkte und Dienstleistungen nach dem Barrierefreiheitsstärkungsgesetz 1." Zugriff

am: 26. Oktober 2025. [Online.] Verfügbar: https://www.gesetze-im-internet.de/bfsgv/BJNR092800022.html

[21] "Harmonisierte Europäische Norm (EN) 301 549," *Portal Barrierefreiheit der Dienstekonsolidierung des Bundes*, 02. November 2023. Zugriff am: 27. September 2025. [Online.] Verfügbar: https://www.barrierefreiheit-dienstekonsolidierung.bund.de/Webs/PB/DE/gesetze-und-richtlinien/en301549/en301549-node.html

[22] W. W. A. Initiative. "Selecting Web Accessibility Evaluation Tools." Zugriff am: 9. Oktober 2025. [Online.] Verfügbar: https://www.w3.org/WAI/test-evaluate/tools/selecting/

[23] W. W. A. Initiative. "Web Accessibility Evaluation Tools List." Zugriff am: 27. September 2025. [Online.] Verfügbar: https://www.w3.org/WAI/test-evaluate/tools/list/

[24] "WAVE Help." Zugriff am: 9. Oktober 2025. [Online.] Verfügbar: https://wave.webaim.org/help?utm_source=chatgpt.com

[25] Deque. "Company Overview | Deque Systems." Zugriff am: 27. Oktober 2025. [Online.] Verfügbar: https://www.deque.com/company/

[26] Deque. "axe DevTools Pricing & Plans | Deque." Zugriff am: 27. September 2025. [Online.] Verfügbar: https://www.deque.com/axe/devtools/pricing/

[27] "WAVE Web Accessibility Evaluation Tools." Zugriff am: 29. Oktober 2025. [Online.] Verfügbar: https://wave.webaim.org/

[28] "WAVE Stand-alone API and Testing Engine." Zugriff am: 29. Oktober 2025. [Online.] Verfügbar: https://wave.webaim.org/standalone

[29] "Accessibility IMpact (AIM)." Zugriff am: 29. Oktober 2025. [Online.] Verfügbar: https://wave.webaim.org/aim/

[30] "WebAIM: Survey of Web Accessibility Practitioners #3 Results." Zugriff am: 8. Oktober 2025. [Online.] Verfügbar: https://webaim.org/projects/practitionersurvey3/#testingtools

[31] "Using Best Practice or Experimental Rules | Deque Docs." Zugriff am: 5. November 2025. [Online.] Verfügbar: https://docs.deque.com/devtools-for-web/4/en/best-practice-and-experimental#enable-best-practice-rules

[32] "Using the Accessibility Results with Java | Deque Docs." Zugriff am: 27. September 2025. [Online.] Verfügbar: https://docs.deque.com/devtools-for-web/4/en/java-use-results

[33] "Accessible University Demo Site." Zugriff am: 29. Oktober 2025. [Online.] Verfügbar: https://www.washington.edu/accesscomputing/AU/

[34] "Before and After Demonstration: Overview." Zugriff am: 29. Oktober 2025. [Online.] Verfügbar: https://www.w3.org/WAI/demos/bad/

[35] M. Endres, A. Steckert und Z. Tuncer, "Barrierefreie Interaktion zwischen Mensch und Technologie – Konzept eines Leitfadens für Softwareentwickler:innen basierend auf Gesetzen, Normen und Richtlinien," *Z. Arb. Wiss.*, Jg. 79, Nr. 2, S. 159–180, 2025. doi: 10.1007/s41449-025-00465-6. [Online]. Verfügbar unter: https://link.springer.com/content/pdf/10.1007/s41449-025-00465-6.pdf

[36] "Web Content Accessibility Guidelines (WCAG) 2.2." Zugriff am: 27. September 2025. [Online.] Verfügbar: https://www.w3.org/TR/WCAG22/#non-text-content

[37] "Links must be distinguished from surrounding text in a way that does not rely on color | Axe Rules | Deque University | Deque Systems." Zugriff am: 7. November 2025. [Online.] Verfügbar: https://dequeuniversity.com/rules/axe/4.0/link-in-text-block

8.2 Abbildungen

8.3 Tabellen

Anhang

1. Accessible University Info Page

Diese Auflistung fasst alle Barrieren der Accesible Unerversity (AU) Home page zusammen.

Quelle: https://www.washington.edu/accesscomputing/AU/info.html Zugriff am 16.11.25

This page summarizes the accessibility issues demonstrated on the Accessible University (AU) home page Before accessibility. With each issue, a solution is proposed. To see these accessibility solutions implemented, view the AU Home Page After accessibility.

Structure

Everyone benefits when web pages are organized into distinct sections with headings, *subheadings*, lists, and other structural elements. Structure helps users to understand how the content is organized and makes it easier for them to parse and read the content. If a person is using a screen reader to access content audibly rather than visually, they too benefit from structure, just as visual users do. For screen reader users to benefit, structure needs to be built into the page in the underlying code.

Missing Landmark Regions

Most web pages have a common overall structure. The AU home page is no exception. It has a banner, footer, navigation menu, main content, and a sidebar. These page regions are clearly identifiable to sighted users, but they are not properly coded in a way that makes them similarly recognizable by screen reader users.

In 2014, the World Wide Web Consortium (W3C) published a new specification called Accessible Rich Internet Applications (ARIA), which introduced eight so-called *landmark roles* that could be added to HTML elements to identify specific regions of the page. For example, the regions mentioned above would be coded as **role="banner"**, **role="contentinfo"** (for the footer), **role="navigation"**, **role="main"**, and **role="complementary** (for the sidebar). With properly-coded landmark regions, screen reader users can quickly locate and jump to the section of the page that meets their needs.

Not long after ARIA was published, HTML added new semantic elements that serve the same purpose. Now, landmark regions on the AU home page can be coded with **<header>**, **<footer>**, **<nav>**, **<main>**, and **<aside>** respectively.

Both of these methods for identifying page regions — ARIA landmark roles and HTML semantic elements — are well-supported by screen readers, easy to implement, and have a huge positive impact on accessibility. For the *Accessible Version* of the AU home page, we used HTML semantic elements.

No headings

Visually, the AU home page includes several headings. However, these headings are not coded as headings using HTML heading tags (e.g., **<h1>**, **<h2>**). Instead, they are just plain text that is bold and large. If HTML heading tags are used, screen reader users have access to this structural information. This helps them to understand how the document is organized, and helps with navigation: Screen reader users can jump between headings in a document using a single keystroke, such as the "h" key (for "heading").

Headings should form an outline of the page content, without skipping levels. On the AU home page, the logo, which contains the text "Accessible University," is arguably the main heading of the page, so on the *Accessible Version* it's coded as an **<h1>**heading. All other headings on the page are coded as **<h2>** headings.

Language not specified

The AU home page includes content in both English and Spanish. Screen readers support both of these languages (and many others), and can switch on the fly between languages if they know to do so. If they don't know to do so, an English-speaking screen reader will read the Spanish section using English pronunciation rules, which produces poor if not indecipherable Spanish. The technique for addressing this issue is to provide a **lang** attribute on the **<html>** element, with the value being the official language code for the primary language used in the document. For example, an English document would include this html tag:

```
<html lang="en">
```

If the web page includes content that differs from the primary language, the change in language should be identified in the code. For example, to markup a paragraph as being in Spanish, the paragraph tag would look like this:

```
<p lang="es">
```

Images

Images are inherently inaccessible to people who are unable to see them. The technique for making them accessible to screen reader users depends on whether they are *informative* or *decorative*.

No alternate text on *informative* images

On the AU home page, the logo and the four photos that are featured in the carousel are all *informative*. Their purpose is to communicate information to the user. When images contain text, such as the AU logo, the primary message that's being communicated is the text itself. When images do not contain text, the message is often more subtle. In the AU carousel, all four images are working together to communicate that Accessible University has a pretty awesome campus.

To make these messages accessible, images require an **alt** attribute that succintly communicates the image's message. This is commonly called "alternate text" or simply "alt text".

Most web authoring tools provide some means of entering alt text for images, often in an *Image Properties* dialog. Some web authoring tools pre-populate the alt text field with the image filename, which is not good alt text, and should always be replaced. On the AU home page, the logo is an example of an image which has the file name as its default alt text. For screen reader users, this makes no sense and should be replaced with **alt="Accessible University"**.

Missing or excessive alternate text on *decorative* images

Some of the images on the AU home page are *decorative*; they are not intended to communicate information to the user. Specifically, the page includes a horizontal line that serves as a separator between sections on the page. These images are tagged with **alt="horizontal line graphic"** which is unnecessary information, and makes for a noisy experience for screen reader users, distracting them from the page content they actually care about. One solution is to provide an empty **alt**attribute (**alt=""**) for all decorative images. This is standard markup that instructs screen readers to ignore the image. Another solution is to move all decorative images to the background using Cascading Style Sheets (CSS). The latter is the solution we've used for addressing this issue on the *Accessible Version* of the AU home page.

Color

Insufficient color contrast

On the AU home page, the navigation menu is an example of content that is difficult to see due to insufficient contrast between foreground and background colors. The W3C Web Content Accessibility Guidelines (WCAG) 2.x includes specific contrast ratios that must be met in order to meet the guidelines at various levels. There are a variety of free tools available that check for contrast. See the Tools and Resources page on the UW Accessible Technology website for an annotated list of accessibility tools, including several color checkers.

Furthermore, contrast is frequently overlooked when colors change on focus, such as when a keyboard user tabs to them or a mouse user hovers their pointer on the item. This problem can be observed in AU's navigation menu. The menu already has poor color contrast, but when a user tabs to an item in the submenu, it's even worse! See the *Keyboard* section below for additional information about keyboard accessibility, and when testing with a keyboard, always be on the lookout for contrast problems that might not reveal themselves if you only test with a mouse.

Color used to communicate information

The AU home page includes two instances of color being used as the sole means of communicating information.

First, in the application form, required fields are marked with blue text. People who are color blind or have other visual disabilities may be unable to perceive the differences in these colors. Even if a more obvious color was chosen, some users would still be unable to identify which fields are required. The solution is to communicate information using other means in addition to color. On the *Accessible Version* of the AU home page, we have addressed this issue by presenting labels for required fields in bold text, marking them with a star, and using the HTML **required** attribute, so browsers can do their part to communicate to users that a field is required.

Second, the links within the main content use color as the sole means of distinguishing links from other text. Look closely! They're difficult for most people to spot. Even if the color choice was more dramatic, some users would still be unable to distinguish link text from non-link text. This is why browsers have always underlined links by default, and still do. It's a good, accessible practice and with modern CSS that are a wide variety of techniques for adding nicely styled, aesthetically pleasing, on-brand underlines to link text.

Keyboard

Any page content or features that are accessible with a mouse should also be accessible with keyboard alone. Some users whose disabilities impact upper body mobility or dexterity are physically unable to use a mouse and may be using the keyboard instead. Also, if a web page is not accessible by keyboard, this is often a good indication of other underlying accessibility problems.

Keyboard accessibility is easy to test, as it requires no special tools or skills. Simply try navigating through a web page using the Tab key (or Shift+Tab to go back). Use Enter or spacebar to follow links or activate buttons. Use other keys, such as the arrow keys and Escape key if the interface seems to call for use of those keys.

Web pages tend to fail keyboard accessibility tests for one of two reasons. First, the page may be coded improperly, making it truly impossible for keyboard users to access some of the content or features. Second, the page may actually be technically accessible to keyboard users but there is no visible indication of focus, so it's impossible for sighted users to tell where they are on the page.

Inaccessible keyboard interface

On the AU home page, the biggest problem for keyboard users is the modal dialog that pops up when they follow the third "click here" link in the "Can you spot the barriers?" section. Without a mouse, it's impossible to close the dialog. For additional information about this feature, see the *Inaccessible modal dialog* section below.

Another problem is the navigation menu. The menu is technically accessible to keyboard users since it's possible to access all menu items, including those contained in dropdown submenus, with keyboard alone. However, this menu only supports the tab key, and moving through it with keyboard is extremely cumbersome. A properly coded navigation menu would additionally support other keys, such as the arrow keys and escape key, so keyboard users can navigate the menu more efficiently. For additional information about this feature, see the *Inaccessible navigation menu* section below.

No visible indication of focus

As a keyboard user tabs through a web page, they should always be able to tell where they are on the page. Some web browsers provide a built-in visible focus indicator such as a solid blue outline that identifies the element that currently has focus. Other browsers provide a thin dotted line that can be difficult to see. A best practice is to design your own visible focus indicator that works reliably across all browsers. For more on this issue, including sample CSS code, see the Visible Focus page on the UW Accessible Technology website.

No "skip to main content" link

The AU home page has content at the top of the page (a navigation menu for the demo site and AU's website naviation menu) that, even if coded properly for accessibility, can be burdensome for keyboard users who just want to tab to a link or button elsewhere on the page. This isn't a problem for screen reader users if the page is coded properly (e.g., using landmark regions and HTML headings as described above in the *Structure* section); they can easily jump to particular page content given functionality that's provided by their assistive technologies. However, keyboard users who are not using assistive technologies are much more limited.

The solution is to provide a "Skip to main content" link in the top left corner of the page, which links directly to the main content. This should be the first element that receives focus when a user starts tabbing from the top of the page. For the *Accessible Version* of the AU home page, we have designed this link so it's visibly hidden by default so as not to confuse mouse users who don't understand its purpose. It's hidden by positioning it off-screen using CSS, which keeps it visible to screen reader users. Even though they have other ways of navigating on the page, they may still find this link to be useful. When a keyboard user tabs to the link, its CSS properties change so that it moves onto the visible page.

Links

Redundant, uninformative link text

The AU home page includes three links that say "click here". Screen reader users may encounter these links out of context. For example, they might be tabbing through focusable elements on the page, or they might be navigating through a list of links (many screen readers include this feature). The solution is to always ensure that link text is unique and meaningful on its own, even if read out of context.

Links vs buttons

On web pages, links and buttons are different elements, which serve different purposes. Links take the user to a new location, such as a new web page or new section of the current page; whereas buttons trigger some action, such as showing content on the page that was previously hidden, playing a video, or submitting a form. This distinction matters because it affects user expectations. If a screen reader clicks a link expecting to be taken to a new location and that doesn't happen, the result will likely be a very confused user.

On the AU home page, the third "click here" links in the "Can you spot the barriers?" section triggers a modal dialog. This link should be a button.

Features

As web pages become more interactive, the interface can become more challenging to make accessible and usable for people with disabilities. Designers and developers need to consider how to make the feature accessible to keyboard users and how to code it properly so that screen assistive technology users can understand the interface and use it effectively. A key resource when designing, coding, or choosing an interactive web feature is the W3C's ARIA Authoring Practice Guide (APG). The APG includes a variety of design patterns for common web features, including all of the features listed below. Whenever possible, web features should be designed and developed in accordance with the APG design patterns, as doing so would ensure a reliable, consistent interface throughout the web.

Inaccessible navigation menu

The website navigation menu on the AU home page includes sub-menus that appear when users hover over the menus with a mouse. They also are technically accessible to keyboard users since it's possible to access all menu items, including those contained in dropdown submenus, with keyboard alone. However, this menu only supports the tab key, and moving through it with keyboard is extremely cumbersome. The AU navigation menu is simple and short compared to most menus in the real world, but it still requires 20 keystrokes to get from one end to the other.

A properly coded navigation menu would additionally support other keys, such as the arrow keys and escape key, so keyboard users can navigate the menu more efficiently. Also, ARIA is needed in order to communicate to screen reader users that submenus are available and whether they're expanded or collapsed. The APG includes a design pattern called Disclosure (Show/Hide), which works well for navigation menus.

Inaccessible carousel

Carousels or slideshows are common features on university home pages. Unless designed and coded with accessibility in mind, these features present a variety of accessibility issues. For example:

- Keyboard users are unable to access all components.
- Screen reader users are unable to operate the controls or access and understand all content.
- People who are distracted by movement or who need more time to read the content are unable to pause the animation.

The APG includes a design pattern for a Carousel (Slide Show or Image Rotator). Following this design pattern will result in a much more accessible interface. However, it is important to test the carousel (and other complex interactive features) with assistive technologies, ideally recruiting actual users of these technologies to do the testing. Does the interface make sense? What is the experience like? How can it be improved? These are all important questions that must be explored in order to ensure features are not just accessible, but usable. Part of this line of inquiry should also be weighing the pros and cons of using a particular feature. What are the benefits of presenting content in a carousel? Do your analytics confirm that users are interacting with it as expected? Adding features just because they're trendy and cool may not be sound justification, especially if the features are challenging to implement and make your web page more difficult to use.

Inaccessible modal dialog

In the "Can You Spot the Barriers?" section of the AU home page, the third "click here" link opens the content in a modal dialog, which appears in the foreground while background content is masked behind a dark transparency. Despite its visible appearance, this is not truly a *modal* dialog. If a keyboard user presses the Tab key, their focus may remain in the background, making it difficult (maybe even impossible) to tab to the dialog so that they can dismiss it. Screen reader users may not even be aware that a dialog has appeared, leaving them confused because they clicked on a link but nothing seemed to happen. (See also "Links vs buttons" above).

The APG includes a design pattern for Dialog (Modal) that spells out the keyboard model and ARIA markup that are needed in order to make a modal dialog accessible. Also, HTML now includes a **<dialog>** element that greatly simplifies the

coding required to create a modal dialog. We have used the latter as a solution for making the dialog accessible on the *Accessible Version* of the AU home page.

Forms

Form not properly labeled

On the AU home page, the application form mostly includes simple text fields and a set of checkboxes. Each of these form fields is accompanied by a label and sighted users know which labels accompany which form fields by their position. However, this is not coded properly for accessibility. Labels must be coded using the HTML **<label>** element and explicitly associated with the fields they represent. Without proper markup, some screen readers may attempt to guess the labels for each form field, while others will not. In this case, screen readers that attempt to guess the labels (i.e., JAWS) do so incorrectly within the group of checkboxes.

Another problem with the group of checkboxes is that the overall prompt "Desired major(s)" is not explicitly associated with the checkboxes. Screen reader users navigating in "forms mode", pressing the tab key to jump between fields, will skip past this prompt and not know what the group of checkboxes is all about. If a group of checkboxes or radio buttons was coded this way in an online quiz, users would be presented with the possible answers, but not with the question. Coding this properly involves wrapping the entire set of fields, including the overall prompt, in a **<fieldset>** element and coding the overall prompt as a **<legend>**.

Additional information, including sample code, is available on the Forms on websites page on the UW Accessible Technology website.

Inaccessible CAPTCHA

Online forms often include images of distorted characters called CAPTCHAs ("Completely Automated Public Turing test to tell Computers and Humans Apart"). CAPTCHAs are designed to prevent spammers and other unwanted users from filling out and submitting the form automatically. The usual method for making images accessible (alternate text) would not be feasible for CAPTCHAs since this same technique would make the images accessible to bots. The Forms on websites page on the UW Accessible Technology website includes a section on "Avoiding CAPTCHA" with considerations about how to block bots without burdening users.

For the *Accessible Version* of the AU home page, the method we've used to address this problem, for demonstration purposes only, is a text-only only CAPTCHA, which asks simple logic questions that are accessible to humans but more challenging for bots. As bots get smarter, this solution becomes less reliable. If spam is a significant problem for a particular form, techniques for addressing that problem must be considered very carefully in order to avoid blocking legitimate users.

Inaccessible input validation

On the AU home page, when the user submits the appplication form, it's validated using JavaScript before the data is submitted to the server. If the user's submission fails validation, an error message is displayed at the top of the form.

This feature has many problems:

- Screen reader users are not informed that this message has appeared. They submitted the form but nothing seems to have happened.
- The error message appears in a location that is nowhere near the submit button, where the user is currently focused. Users who are zoomed in significantly may not see the error message when it appears.
- The error message is unclear. It states that there are errors, but does not identify what those errors were. The burden of finding the errors is left to the user, which adversely impacts usability for everyone, but can be especially challenging for screen reader users.

Client-side form validation is an important part of making online forms accessible. However, care must be taken to ensure it's done in an accessible way. For more information, see the W3C Web Accessibility Initiative's Forms Tutorial, which includes a detailed section on Validating Input.

Tables

Missing accessible table markup

Data tables can be challenging for screen reader users to understand, particularly if they have many columns, or a complex layout with nested rows or columns, as is the case with the AU Enrollment Trends table. Imagine reading a table from left to right and top to bottom, with no visual access to the column headers. When you're halfway through the table, will you remember which column you're in? This is not unlike what screen reader users experience as they try to read a data table unless the table includes semantic markup that explicitly defines the relationships between the table's parts. For example, table headers should be marked up with the **<th>** element ("th" stands for "table header"), and headers should include the **scope** attribute, which identifies whether it's a row header (**<th scope="row">**) or a column header (**<th scope="col">**).

The AU Enrollment Trends table includes additional challenges since it has nested column headers, plus row headers, that apply to each data cell. The W3C Tables Tutorial includes recommended markup for a variety of distinct table configurations.

For the *Accessible Version* of the AU home page, we made the AU Enrollment Trends table accessible by properly tagging each column and row header as explained above and by assigning a unique id to each header. Then, a **headers** attribute was added to each data cell, the value of which is a space-separated list of ids of all headers that apply to that cell (e.g., **<td headers="col1a col2a row1">**). When a table includes all of this markup, screen reader users can easily ascertain their current position with a table, and their screen reader can announce all of the headers that apply to the current cell.

Missing abbreviation tags

Abbreviations can be difficult for all users to understand. In the AU Enrollment Trends table, the column headers are abbreviations for different majors. These abbreviations may be new to some users, and for all users there may be ambiguities (does "Eco" stand for "Ecological Sciences" or "Economics"?) The **<abbr>** element should be used to identify abbreviations with a **title** attribute that spells out the full word or name. This is displayed as a tooltip to mouse users as they hover over the abbreviation, and is announced verbally to screen reader users if their screen reader is configured to support this functionality. In the "Eco" example, the **<abbr>** element would be used like this:

```
<abbr title="Economics">Eco</abbr>
```

Video

Inaccessible audio content

The AU video includes narration that is inaccessible to users who are unable to hear the audio. This includes people who are deaf or hard of hearing, as well as people who are watching the video in a quiet setting where they're unable to turn up the sound, and people who are watching the video in a noisy setting. The solution is to add captions to the video. For details, see the Captionspage on the UW Accessible Technology website.

Inaccessible visual content

The AU video features rich imagery that is designed to communicate information about the unviersity to viewers. The video's narration captures some of this, but leaves out a lot of important content, and a person listening to the audio alone does not come away with an equivalent impression to that of a person who's listening to the audio accompanied with the visuals. The solution is to supplement the video with *audio description*, a narration track that supplements the video by describing the visual content for people who are unable to see it.

The *Accessible Version* of the AU home page uses Able Player, a free, open-source, accessible HTML media player created and maintained at the University of Washington. Among other features, it enables users to switch between the described and non-described versions of the video using the "Descriptions" toggle button on the player control bar.

2. Before and After Demonstration

Diese Auflistung zeigt die Dokumentierten Barrieren der Before and After Demonstration der W3C auf.
Quellen:

Home	https://www.w3.org/WAI/demos/bad/before/annotated/home.html	Zugriff am 16.11.25
News	https://www.w3.org/WAI/demos/bad/before/annotated/news.html	Zugriff am 16.11.25
Tickets	https://www.w3.org/WAI/demos/bad/before/annotated/tickets.html	Zugriff am 16.11.25
Survey	https://www.w3.org/WAI/demos/bad/before/annotated/survey.html	Zugriff am 16.11.25
Template	https://www.w3.org/WAI/demos/bad/before/annotated/template.html	Zugriff am 16.11.25

Home

Note 01: Image with incorrect text alternative

The text alternative for this image of text is overly verbose and does not serve the equivalent purpose of the image. The text alternative is as follows:

```
<img src="top_logo.gif" alt="Red dot with a white letter 'C' that symbolizes a moon crescent as well as the sun. This logo is followed by a black banner that says 'CITYLIGHTS' which is the name of this online portal. Finally, the slogan of the portal, 'your access to the city', follows in a turquoise green handwriting style and with a slight slant across the top banner." ... >
```

Note: This error is derived from the Template design and occurs throughout the entire website.

Note 02: Image without any text alternative

This image does not have any text alternative, so that it is unclear to some reasers if the image is important or not.

```
<img src="teaser_right1.jpg" ... >
```

Note: Every img element should have an alt attribute. The attribute should be empty if the image should be ignored by assistive technology.

Note 03: Link not visually distinct

Thelink "heat wave linked to temperature" is not sufficiently distinct because it resembles a heading, and will not be recognized as a link by many readers.

Note 04: Image with inadequate text alternative

The text alternative for this image is provided in the title attribute alone, which is not usable for many readers. Moreover, the text alternative "image" does not describe the image.

```
<div style="background: url(BrainInJar.jpg)" title="image" ... >
```

Note: The techniques for retrofitting this image differ depending if the author regards this image as purely decorative or as informative.

Note 05: Link text is not descriptive

The link text "Read more..." is not descriptive to convey the purpose of the link.

Note 06: Reading sequence not meaningful

These three columns of text appear to be visually distinct but according to the markup code they appear as one piece of running text. They would read as follows:

```
"After three years of effort city scientists now agree that the primary cause of the 2003 heatwave was hot air from our Mayor: These kinds of crimes need more creative, effective punishments. For example, we could require compulsory Brain donations: huge drop off in brain donations down due to the 'success' of 'Slow Traffic, Safe Streets' policy"
```

Note 07: Link with image has empty text alternative

This image has an empty text alternative but it is the only content in the link, so that the purpose of the link is unclear to some users.

```
<a href="news.html" ... ><img src="morearrow.gif" alt="" ... ></a>
```

Note 08: Decorative image without empty text alternative

This image has the text alternative "bullet", which does not provide any useful information. The image should have an empty text alternative to indicate that it is a decorative image that should be ignored by assistive technology.

```
<img src="list_bullets.gif" alt="bullet" ... >
```

Note: The image can also be dispayed using CSS rather than the img element.

Note 09: List not marked up as such

The items "Killer bees" and "Onions" are formatted to visually resemble a list but this structural information is not represented in the HTML code.

Note 10: Image with incorrect text alternative

This image has an incorrect text alternative, "1234 56789"; possibly because the text alternative is outdated or a placeholder.

```
<img src="telefon_white_bg.gif" alt="1234 56789" ... >
```

News

Note 01: Heading not marked up as such

The heading "Heat wave linked to temperatures" has been styled to look like a heading but has not been marked up as a heading within the HTML code.

```
<font size="5" color="#FFFFFF"><P class=headline>Heat wave linked to temperatures</P></font>
```

Note 02: Font difficult to read

The text on this page uses serif fonts such as 'Times New Roman' rather than sans-serif fonts such as 'Lucida', which are considered easier to read on-screen.

```
td { color: #000000; text-decoration: none; font: 15px/17px "Times New Roman", Times, serif; }
```

Note 03: Columns too close together

The two main columns of text need more space between them.

```
<div style="padding: 10px 0px 10px 0px; margin: 0px; ... ">
```

Note 04: Reading sequence not meaningful

The section "Your Shout" appears visually below and as part of the "Man Gets Nine Months in Violin Case" article but it only appears after the "Tough Wahoonie" section within the HTML code. Some readers will read this section out of the intended order.

Note 05: Link purpose unclear

This image is part of a link so its text alternative should describe the link purpose rather than describe the image.

```
<a href="..." ...><img src="smzappa2.jpg" alt="picture of Clara F." ... / ></a>
```

Note: This seems to be a stray link that was placed in error or that was there for an earlier image.

Note 06: Link not visually distinct

The link "the way that air conditioning works" is not distinguished from the surrounding text because it has a similar color and there are no other cues (such as underline) to indicate it is a link.

Note 07: Emphasized text not marked up as such

The text "but also hot air that's ducted away to the outside" is styled visually to resemble emphasized text but it is not marked up using the em element.

```
<span style="font-style: italic;">but also hot air that's ducted away to the outside</span>
```

Note 08: Text with deprecated markup

The text "Return to Sender" is formatted using deprecated markup code, such as font element and align attribute, that may not be supported by browsers and assistive technology.

```
<p ... align="center"><font size="3">Return To Sender</font></p>
```

Note 09: Diagram representation unclear

This diagram does not have any text alternative and is of very poor quality so that it is unclear what it represents.

```
<img src="chart1.jpg" alt="" ... / >
```

Note 10: Image with incorrect text alternative

The text alternative for this image provides more information than what is visually conveyed to most readers. This information should be provided to all readers rather than in the text alternative alone.

```
<img src="blanca.jpg" alt="Mrs B. Wahoonie investigates shipping hot air to Australia" width="240" height="180" border="0">
```

Tickets

Note 01: Image of text without text alternative used as a heading

This is an image of text that visually looks like a heading but is not marked up as a heading. It also does not have a text alternative.

```
<p style="padding-top:5px;"><img src="../img/headline_ticket_offers.gif" border="0"></p>
```

Note: It is more preferable to present text that is styled visually with CSS rather than images of text.

Note 02: Table column headings not marked up

Table headings are indicated by the colored background but have not been marked up semantically using the th element. Also, the table title has not been associated with the table.

```
<td><b><font color="41545D">Frank Zappa</font ></b></td>
```

Note 03: Using CSS to convey semantic meaning

The "special deals" text has been made bold visually using CSS rather than semantically using the strong element.

```
<big style="font-weight:bold;">special deals</big>
```

Note 04: Image of text with incorrect text alternative

The text alternative for this image does not provide the phone number "(1) 269 M-U-S-I-C" that is displayed graphically.

```
<img src="../img/telefon_lightgrey_bg.gif" alt="music line phone number" border="0" valign="absmiddle">
```

Note 05: Data table without semantic structure

Table is not identified using the caption element, and the header cells are not marked up or associated with the data cells.

```
<table ...>
<tr>
<td><b>ADULT</b></td>
<td><b>FS</b></td>
<td><b>RS</b></td>
<td><b>DC</b></td>
<td><b>ST</b></td>
</tr>
<tr>
<td><b>Thelonius Mank - reincarnate</b></td>
<td><b>$20.90</b></td>
<td><b>$20.90</b></td>
<td><b>$20.90</b></td>
<td><b>$20.90</b></td>
</tr>
```

Note 06: Insufficient color contrast

Contrast ratio between foreground color (#41545D) and background color (#A9B8BF) is too low for some people to differentiate.

```
{color:#41545D; background-color:#A9B8BF;"}
```

Note 07: Acronyms used to indicate seat options are not explained

The acronyms "FS", "RS", "DC", and "ST" used to indicate seat options are not expanded or otherwise explained.

```
<td><b>FS</b></td>
```

Note 08: No semantic markup and text is hard to read

Heading is not marked up as such, list is not marked up as such, and text is hard to read due to full justification, use of all-capitals, use of underlining, and tight line-spacing (leading).

```
<p id="terms" class="annot_link_parent" align="justify"><b><div style="text-align: center;"><font size="4">Citylights Terms and Conditions</font></div><br><br>[1] You agree through the act of calling ... etc
```

Survey

Note 01: Instructions necessary for form completion are missing

Instructions necessary for form completion, especially regarding any required fields, are missing.

Note 02: Radio buttons not grouped within the code

Radio buttons not grouped using the fieldset element, and are not described using the legend element.

Note 03: Radio button is not associated with its label

Radio button is not associated with its label "none" using the label element.

```
<input class="lign" type="radio" name="res" value="1">
```

Note 04: Select box is missing caption

Select box is missing caption using the title attribute or label element.

```
<select name="cc"> ... </select>
```

Note 05: Select box not easily usable by keyboard

The items in this long list are not grouped to facilitate effective use by keyboard.

Note: The items in this list are sorted by country rather than by city, making it more difficult for users to complete the requested task.

Note 06: Reading sequence not meaningful

The sequence of labels and input fields does not match the visual order and is not correct when the form is navigated by keyboard. The sequence is as follows:

```
Name: [radio button] Mr. [radio button] Mrs. [text input] [text input] [text input] eMail Address Retype eMail
```

Note: The first [text input] is for "eMail Address", the second for "Name", and the third for "Retype eMail" but this is not clear because the form fields are not associated with their labels.

Note 07: Reading sequence not meaningful

Data in this table is organized using document structures such as the p element rather than table structures such as the th and td elements, so that the order of data cells does not make sense to some readers.

```
<td rowspan="4" bgcolor="#ffffff" style="border-right: 1px dashed silver;">
<p style="background:#DBDBDB;padding-top:3px;padding-bottom:2px;"><br></p>
<p style="margin-bottom:0px;padding-top:3px; padding-bottom:5px" a-
lign="right"><strong>love it</strong></p>
<p style="margin-top:5px;background:#DBDBDB;padding-top:3px; padding-bot-
tom:4px" align="right"><strong>hate it</strong></p>
</td>
```

Templates

Note 01: Page title and layout causes confusion

This and all other pages are identified as "Welcome to Citylights!" using the title element so that it is unclear for some readers on which page they are on. The page layout also uses deeply nested table elements (up to 5 levels alone for the template) that can cause confusion and disorientation to some readers.

```
<title>Welcome to Citylights!</title>
```

Note: Errors on the template are propagated throughout the entire website.

Note 02: Image with incorrect text alternative

The text alternative for this image of text is overly verbose and does not serve the equivalent purpose of the image. The text alternative is as follows:

```
<img src="top_logo.gif" alt="Red dot with a white letter 'C' that symbolizes a moon crescent as well as the sun. This logo is followed by a black banner that says 'CITYLIGHTS' which is the name of this online portal. Finally, the slogan of the portal, 'your access to the city', follows in a turquoise green handwriting style and with a slight slant across the top banner." ... >
```

Note: Errors on the template are propagated throughout the entire website.

Note 03: Automatic page redirect on change

Selecting an item from the drop-down menu automatically redirects the user to another page. This is confusing and disorienting to some users, especially keyboard users who are trying to move from one item to the next rather than actively selecting an item.

<select onchange="...">

Note: Errors on the template are propagated throughout the entire website.

Note 04: Script not compatible with some tools

The script uses document.write() technique which should be avoided. Content inserted that way into a page is not included in the Document Object Model (DOM) and may not be properly communicated to the accessibility APIs used by some assistive technology.

document.write(today);

Note: Errors on the template are propagated throughout the entire website

Note 05: No mechanisms to bypass repeated content

The page does not offer mechanisms to bypass the header and navigation blocks which are repeated on each page. Such skip-links make website use more effective for some readers and keyboard users.

Note: Errors on the template are propagated throughout the entire website.

Note 06: Headings not marked up as such

The template suggests styling for headings without use of the corresponding markup within the HTML code.

```
<p class="headline">Template</p>
```

Note: Errors on the template are propagated throughout the entire website.

Note 07: Images without any text alternative

The items in the left navigation menues are images that do not have any text alternative, so that it is unclear to some readers what the links correspond to.

```
<a href="..." ... ><img src="..." ... >
```

Note: Errors on the template are propagated throughout the entire website.

Note 08: Page components not focusable

The is automatically removed from the menu items using scripting, so that some users cannot navigate to these items.

<a href="..." onfocus="blur();" ... >

Note: Errors on the template are propagated throughout the entire website.

Note 09: Link text not descriptive and not visually distinct

The template suggests use of the link text "Read more...", which is not descriptive to convey the purpose of the link. It also uses color that is similar to the surrounding text making it hard for some users to identify the link.

Note: Errors on the template are propagated throughout the entire website.

Note 10: Insufficient color contrast

Contrast ratio between foreground color (#41545D) and background color (#A9B8BF) is too low for some people to differentiate.

```
{color:#41545D; background-color:#A9B8BF;"}
```

Note: Errors on the template are propagated throughout the entire website.

18 Anforderungen

Die Auflistung zeigt die Resultate des Konzeptes eines Leitfadens für Softwareentwickler:innen basierend auf Gesetzen, Normen und Richtlinien von Endres, Steckert und Tuncer aus dem Jahr 2025 [35].

Anforderung 01: Labels für alle Daten Eingabefelder und -Ausgabefelder, sowie für gruppierte Informationen

Grundsätzlich gilt, jede Seite soll einen Titel besitzen, der das Thema oder die Absicht einer Internetseite beschreibt. Dies schließt ebenfalls den Seitentitel im Browser-Tab mit ein (Tab-Browsing), um den Standort für Screenreader-Nutzer zu identifizieren und die Verwaltung der Bookmarks zu erleichtern.

Interaktionen und Transaktionen mithilfe von Formularen stellen einen besonderen Komplexitätsgrad dar, welche ohne die nachfolgenden Punkte für blinde Menschen zu einer sicheren Barriere führen wird und Formulare für Screenreader-Nutzer unbrauchbar macht.

- Zur **Navigierbarkeit** sind für alle Ein- und Ausgabefelder (bspw. Input-Felder, Radio-Buttons, Radio-Button-Groups, etc.) aussagekräftige Überschriften und Beschriftungen zu spezifizieren.
- Im sogenannten <label>-Element wird eine gut sichtbare und aussagekräftige Beschriftung vorausgesetzt. Weiterhin ist es essenzielle, dass die Beschriftung mit dem zugehörigen Feld (bspw. mit dem for-Attrlbut) verknüpft ist. Für das automatische Ausfüllen von Nutzerdaten in Formularen ist weiterhin die korrekte Verwendung der autocomplete-Attrlbute (bspw. autocomplete="family-name" für Familienname) notwendig.
- Eine sichtbare Beschriftung von Bedienelementen im zugänglichen Namen (bspw. Links, Beschriftungen von Textfeldern, Buttons oder Checkboxen) sorgen für gute **Eingabemodalitäten**.

Anforderung 02: Textalternativen für nicht-textuellen Inhalt

Speziell für seheingeschränkte und blinde Menschen ist es nicht möglich Bilder und Grafiken (Icons, Piktogramme, etc.) zu sehen. In Verbindung mit dem Screenreader erhalten sie mithilfe des Alternativtextes, welcher im HTML-Code nicht sichtbar ist, Zugang zum Bild.

- Jedes Bild sollte einen **individuellen Alternativtext** haben. Bei Verlinkungen wird das Linkziel im alt-Text hinterlegt, während bei Grafiken ohne Link ein beschreibender Text im alt-Text spezifiziert wird. Bei rein dekorativen Grafiken bleibt der alt-Text leer (alt="").
- Bei der Verwendung von Icons, wie bspw. einem Such-Symbol ist das Element mit einem Tooltip zu versehen und ein beschreibender Term "Suche" als alt-Text festzulegen. Gleiches gilt für das Logo des Seitenanbieters. Piktogramme müssen leicht verständlich und nachvollziehbar sein.
- **Nicht-Text-Inhalt** bedeutet ebenfalls, dass Benutzer:innen Inhalte in die benötigten Formen ändern können, wie bspw. Großschrift, Braille, Symbole oder einfachere Sprache.
- In Verbindung mit Videos sind Untertitel (Close Captions) zur Verfügung zu stellen. Weitere **Videofähigkeiten** sind Alternativen für Bedienelemente, Grafiken und Objekte. Auch Alternativen für CAPTCHAs sind hier von Bedeutung.

Anforderung 03: Animationen, Audio- und Videosteuerung (Zeitbasierte Medien)

Um auch Menschen mit Seh- oder Hörbehinderung, sowie weiteren Zielgruppen mit eingeschränkten Wahrnehmungssinnen, Zugang zu visuellen und auditiven Inhalten von Medien zu ermöglichen, ist es erforderlich, Alternativen wie Transkription, Untertitel oder Audiodeskription anzubieten. Für höreingeschränkte Menschen ist zu den Inhalten der Audio-Dateien eine Transkription bereitzustellen, um sämtliche auditive Informationen (Sprache, Dialog, Geräusche) in Textform abzubilden.

- Für **zeitbasierte Medien** sind alternative Audiodateien und stumme Videos vorzuhalten. Aufgezeichnete Videos sind mit Untertiteln auszustatten und für Audiodeskriptionen, bzw. Volltext-Alternativen zu sorgen. Für Live-Videos sind ebenfalls Untertitel und Audiodeskriptionen erforderlich.
- Die Multimediaplayer sind mit **Videofähigkeiten** auszustatten, welche eine Wiedergabe von (synchronen) Untertiteln und Audiodeskriptionen erlauben. Für die Steuerung dieser sind Bedienelemente vorausgesetzt. Ist ein Video nicht bereits von Grund auf mittels <video>-Element und den dazugehörigen HTML-Attributen eingebunden, ist sicherzustellen, dass der Video-Play barrierefrei bedienbar ist.
- Weiterhin wird eine volle Kontrolle über Animation, Video- und Audiostream Im Multimediaplayer vorausgesetzt. Zur Steuerung sind Funktionen wie das Anhalten, Stoppen, Ausblenden oder Anpassen der Lautstärke erforderlich. Auch die Geschwindigkeit soll individuell regelbar sein. Zusätzlich werden erweiterte Audiofunktionen, zum Verringern von störenden Audiosignalen, von Geräten in der Umgebung, und auditive Klarheit benötigt.

Anforderung 04: Barrierefreie Struktur durch korrekte Verwendung von HTML-Elementen

Strukturen und Zusammenhänge sind nicht nur visuell darzustellen, sondern auch textlich und technisch zu vermitteln.

- Im Sinne der **Anpassbarkeit** wird eine Inhaltliche Gliederung vorausgesetzt. Dabei sind die vorgesehenen HTML-Strukturelemente für Überschriften, Listen und Zitate zu verwenden. Datentabellen sind richtig aufzubauen und die Tabellenzellen entsprechend zuzuordnen.
- Internetseiten sollten durch Überschriften strukturiert und visuell voneinander differenziert werden. Um auch Nutzenden mit Hilfsmitteln die Information einer Überschrift zu vermitteln, ist diese entsprechend im HTML-Code als Überschrift zu kennzeichnen. Dabei ist sicherzustellen, dass jeder Text, der visuell einer Überschrift entspricht, auch als solche im HTML ausgezeichnet ist. Die Hierarchie der HTML-Tags von <h1 > bis <h6> ist der Logik des Inhalts anzupassen. Jede Seite ist höchstens mit einem H 1-Heading auszustatten, welches das Thema der Seite beschreibt. Jede Überschrift beschreibt den folgenden Inhalt darunter.
- Pro Seite ist möglichst eine individuelle **Meta Description** zu hinterlegen, um den Inhalt der Seite verständlich zusammenzufassen. Der individuelle Seitentitel (<title> im <head>), pro Seite, ist aussagekräftig zu gestalten und Duplikate auszuschließen.
- Jedes Element ist mit vollständigem Start- und Endtag (bspw. <table></table>) auszustatten. Die korrekte Verschachtelung der Elemente erfolgt gemäß der **Spezifikation**, während keine doppelten Attribute verwendet werden und eindeutige IDs vergeben werden, sofern die Spezifikation dies nicht anders erlaubt. Der Dokumententitel erscheint im Reiter der Titelleiste und Ist somit essenziell für Suchmaschinen, Lesezeichen und der erste Aspekt, den ein Screenreader vorliest.
- Die Verwendung von „sprechenden" und leserlichen URLs, speziell in der Verbindung mit URL-Parametern, trägt dem Verständnis bei Nutzenden von Screenreadern bei.

Anforderung 05: Farb- und Kontrasteinstellung

Für seheingeschränkte Menschen sind Farbe und Kontrast entscheidend, um eine Internetseite bedienen zu können. Sollten sich Vordergrund- und Hintergrundfarbe ähneln, kann es schwer fallen Texte und Grafiken richtig zu lesen. Eine flexible Einstellung der Helligkeit und des Kontrastes ist erforderlich. Gleichzeitig sind Schriften, Bilder und Icons möglichst kontrastreich zu gestalten.

- Der **Kontrast von Text, Graphen, Objekte**, et cetera sollte bei über 4,5:1 liegen im Verhältnis zum Hintergrund liegen. Großer Text und Bilder von großem Text sollen ein Kontrastverhältnis von mindestens 3:1 vorweisen. Spezielle High-Contrast-Themes sollten im Kontrastverhältnis von 7,0: 1 liegen.
- Vordefinierte Kontrastverhältnisse können mithilfe verschiedener Themes (Ein Thema aus der Computertechnik ist eine Zusammenstellung von grafischen Darstellungsdetails und umfasst Einstellungen zu Formen, Farben und Dekorationen von Bedienelementen.), eine **flexible Einstellung** zur Verbesserung der visuellen Klarheit, vorgesehen werden. Typischerweise gibt es hierfür ein normales Standard-Thema, ein Dark-Thema, ein High-Contrast-Black-Theme (HCB) sowie ein High-Contrast-White-Theme (HCW). Das normale Standard-Thema soll dabei den Mindestanforderungen von 4,5:1 entsprechen.

Anforderung 06: Zwei-Kanal-Prinzip und Unterscheidbarkeit

Zur Sicherstellung der Unterscheidbarkeit ist ein Zwei-Kanal-Prinzip erforderlich, wobei die **Benutzung von Farbe** nicht als einziges visuelles Mittel eingesetzt wird,

um Informationen zu vermitteln, eine Handlung zu kennzeichnen oder ein Element unterscheidbar zu machen. Weiterhin ist sicherzustellen, dass das Verständnis und die Bedienung von Inhalten nicht nur auf sensorische Eigenschaften (Form, Farbe, Größe) stützen

- Zur Berücksichtigung von **Farbschwächen** (bspw. Rot-Grün-Blindheit) sind keine gleichen Formen bei der Darstellung zu verwenden, sofern sie sich nur farblich unterscheiden. Gleiches gilt bei der Verwendung von Formen, wenn generell keine Farben wahrgenommen werden können. Zusätzlich sind die Charakteristiken in Textform zu beschreiben. Es muss mindestens eine Bedienungsform geben, die keine Farbunterscheidung erfordert.
- **Unterscheidbar** sind Elemente, wenn sie ohne Farben nutzbar sind, der Ton abschaltbar ist, die Kontraste von Texten ausreichend hoch sind und alle eingeblendeten Inhalte bedienbar sind.

Anforderung 07: Schrift, Textvergrößerung bis 200 % und Zeichenabstand

Die Inhalte auf den Internetseiten lassen sich mit der Zoom-Funktion des Browsers vergrößern, um Texte und weitere Elemente den individuellen Sehbedürfnissen anzupassen. Gleichzeitig dürfen jedoch keine Inhalte abgeschnitten oder überlappt werden, während die Inhalte entsprechend umgebrochen werden. Die Größenänderung von Texten muss bis zu 200 Prozent ohne unterstützende Technologie und ohne den Verlust von Inhalten möglich sein. Speziell für Menschen mit Seheinschränkung ist diese Funktion von großer Bedeutung. Bei der barrierefreien Gestaltung gelten der Schriftart, Schriftgröße, Strichstärke, Schriftweite sowie der Zeichen- und Zeilenabstand eine besondere Beachtung. Zur Leserlichkeit von Schriften und Texten gilt zusätzlich die DIN 1450:2013-04.

- Das horizontale Scrollen innerhalb einer Internetseite soll basierend auf der Vergrößerung vermieden werden. Auch Buttons und Formularelemente müssen zu jeder Zeit sichtbar und nutzbar sein.
- Bei der Vergrößerung der Schrift ist konstant eine statische Breite des UI-Elements zu gewährleisten. Sollten Texte abgeschnitten oder gekürzt werden, sind entsprechende Tooltips einzusetzen, dass keine Informationen verloren gehen. Die Länge des UI-Elements ist dynamisch zu wählen, sodass dieses den gesamten vergrößerten Text beinhalten kann.
- Neben der Vergrößerung spielt auch der Zeichenabstand von Text eine Rolle im Sinne der **Unterscheidbarkeit**. Hierfür sind Einstellungen vorzusehen, um:
    - die Zeilenhöhe auf mindestens das 1,5-fache,
    - den Abstand nach Absätzen auf mindestens das 2-fache,
    - den Buchstabenabstand auf mindestens das 0,12-fache
    - und den Wortabstand auf mindestens das 0,16-fache
- Der Schriftgröße zu erhöhen.

Anforderung 08: Vermeiden von flackernden und blinkenden Inhalten

Inhalte auf der Webseite hören spätestens nach dreimaligem Blinken Im Zeitraum von einer Sekunde auf, um fotosensitive Anfälle oder physische Reaktionen zu verhindern. Dies gilt als Grundsatz für alle visuellen Bedienungsformen.

Anforderung 09: Tastaturbedienbarkeit Interaktiver Elemente

Um Menschen, die nicht mit einer Computermaus arbeiten können, sowie blinden Menschen und Menschen mit motorischen Einschränkungen, eine Steuerung der Internetseite zu ermöglichen, gilt es sicherzustellen, dass sie diese mit der Tastatur bedienen können und keine Maus erforderlich ist. In Bezug auf den Umgang mit assistiven Technologien, ist die Tastaturbedienbarkeit eine Voraussetzung.

- Zur **Tastaturbedienbarkeit** können Tastaturkurzbefehle verwendet oder abgeschaltet werden. Zur Erfüllung dieser Anforderung müssen alle Links, Buttons und Formularelemente mit der Tastatur, oder einem alternativen Eingabegerät, erreichbar und bedienbar sein. Der aktuelle Fokus muss zu jeder Zeit sichtbar sein. Bei der Steuerung mithilfe der Tabulator-Taste ist sicherzustellen, dass die Elemente entsprechend der Lesereihenfolge (in westl. Kulturen von links nach rechts und von oben nach unten) fokussiert werden. Sämtliche Inhalte müssen über die Tabulator-Taste und ggf. ergänzt durch die Pfeiltasten erreichbar und bedienbar sein. Dies schließt alle Texte, Bedienelemente und Multimedia-Inhalte, wie bspw.: Dialoge, Datumsauswahl, Formulare, Slider, Karussells, Menüs, Tooltips und Popups, mit ein.
- **Navigierbar** durch eine schlüssige Reihenfolge bei der Tastaturbedienung. Die aktuelle Position des Fokus ist unter anderem im Sinne der Navigierbarkeit erforderlich. Alle interaktiven Elemente sollen dabei fokussierbar und ansteuerbar sein. Bei der Verwendung der Tastatur ist es entscheidend zu sehen, wo sich der Tastaturfokus befindet, um die angebotenen Funktionen der Internetseite zu nutzen. Eingabefelder mit aktuellem Fokus sind ebenfalls hervorzuheben, hierfür kann sowohl ein Indikator, eine veränderte Hintergrundfarbe oder eine blinkende Umrandung verwendet werden.

Anforderung 10: Alternative zugängliche Bedienungsformen

Werden auf einer Webseite Technologien oder Komponenten verwendet, welche bestimmte Benutzergruppen ausschließen, ist eine alternative Darstellung erforderlich. Bspw. statt visuellen Graphen oder Charts kann die Darstellung tabellarisch in Textform alternativ dargestellt werden.

- Werden **hörbare Signale** eingesetzt, um Informationen zu vermitteln, über eine Handlung zu informieren oder zu einer Reaktion aufzufordern, ist eine Alternative erforderlich.
- Sofern eine **stimmliche Eingabe** erforderlich ist, muss mindestens eine Bedienungsform angeboten werden, welche keine stimmliche Eingabe (auch keine oralen Laute wie Sprechen, Pfeifen oder Schnalzen) erfordert. Bei **manueller Bedienung** sind Bedienungsformen notwendig, welche keine feinmotorische Steuerung und Bedienung, Handmuskelkraft oder gleichzeitige Bedienung von mehr als einem Bedienelement erfordern. Eine Bedienung bei **eingeschränkter Kraft und Reichweite** muss ermöglicht werden. Außerdem soll eine Bedienungsform bereitgestellt werden, welche die Nutzung bei **kognitiven Einschränkungen** erleichtert und vereinfacht.
- Weitere **Eingabemodalitäten** fordern Alternativen für komplexe Zeiger-Gesten (Wischgesten von Slider-Bereichen, Löschen von Inhalten, Mehrpunkt-Gesten) und alternative Bewegungsaktivierungen, die bspw. auf Gerätesensoren reagieren.

Anforderung 11: Interoperabilität mit unterstützenden, assistiven Technologien

Internetangebote müssen sicherstellen, dass sie mit assistiven Technologien verwendet werden können. Speziell blinde Menschen greifen auf Vorlesesoftware (Screenreader) zurück, um damit den Computer oder das Smartphone mit einer grafischen Benutzeroberfläche zu bedienen. Alle Inhalte der Internetseite werden dabei ausgesprochen. Dies schließt Schaltflächen, Menüs, Texte, Abbildung und weitere Elemente mit ein. Die Steuerung der Software erfolgt über die Tastatur oder Gesten an Eingabegeräten.
Auch Menschen mit Legasthenie oder sehbehinderte Menschen profitieren von einer Vorlesefunktion, wenn sie das Ausgabegerät grundsätzlich sehend bedienen können, jedoch Schwierigkeiten beim Lesen haben.

Anforderung 12: Navigierbarkeit und konsistente Navigation

Zur Navigation auf Internetseiten wird ein konsistenter Einstiegspunkt, sowie eine gleichbleibende Navigation mit festen Navigationszielen erwartet. Die Navigation soll die Nutzenden dabei unterstützen zu navigieren, Inhalte zu finden und zu bestimmen, wo sie sich befinden. Relevant ist vor allem die aktuelle Position des Fokus, in Verbindung mit der Tastaturbedienbarkeit. Auch mobile Anwendungen sind mit einem Navigationsmenü ausgestattet.

- Zur Förderung der **Vorhersehbarkeit**, sollen Navigationsmenüs immer an der gleichen Stelle und in der gleichen Reihenfolge organisiert sein. Dabei ist vor allem eine konsistente Bezeichnung notwendig.
- Im Sinne der **Navigierbarkeit** gilt es alternative Zugangswege zu den angebotenen Inhalten zur Verfügung zu stellen. So soll es mindestens zwei unterschiedliche Wege geben. Weiterhin soll es den Nutzenden ermöglicht werden Gruppen, Bereiche oder sich wiederholende Elemente zu überspringen.

Anforderung 13: Bildschirmausrichtung

Eine Anpassung der Displayausrichtung muss möglich sein, ohne Inhalte oder Funktionalität zu verlieren. Als typische Ausrichtungen zählen das Querformat (waagerecht) und der Porträtmodus (hochkant). Die Bedienung von Inhalten ist damit nicht nur auf eine einzige Bildschirmausrichtung beschränkt, sofern eine bestimmte Ausrichtung nicht unentbehrlich ist.

Anforderung 14: Ausreichend Zeit und Zeitbegrenzungen

Sofern mit Zeitlimits auf einer Webseite gearbeitet wird, sind die Benutzer:innen über dieses zu informieren. Gleichzeitig soll es den Nutzenden möglich sein, die Zeitlimits zu verzögern, zu verlängern oder abzuschalten. Dabei ist die verbleibende Zeit jederzeit transparent dazustellen.

- Es gilt **ausreichend Zeit** und eine flexible Zeitmenge für die Interaktion zur Verfügung zu stellen, während die Konsistenz der Funktionalität gewahrt wird.
- Ablenkungen durch **bewegliche und blinkende Inhalte** sind zu vermeiden. Bewegte Inhalte sind auf 5 Sekunden zu begrenzen oder abschaltbar sein. Inhalte, die sich automatisch aktualisieren sollen zum Lesen pausiert werden können.

Anforderung 15: Fehlervermeidung und korrekte Fehlerbehandlung

Im Umgang mit Fehlermeldungen gibt es einige Rahmenbedingungen, welche es zu erfüllen gibt. Dazu zählt, dass diese so dargestellt werden, dass sie visuell und maschinenlesbar sind sowie verständlich und aussagekräftig formuliert sind.

- Wird ein Fehler auf der Internetseite angezeigt, so ist auch die Position des Fehlers, die Fehlerbeschreibung und ein Korrekturvorschlag vorzusehen. Zur Behebung der Fehler sind entsprechende Hilfestellungen zu integrieren.
- Zur Prävention von Fehlern sind entsprechende Dialoge zu realisieren, welche eine Bestätigung der Aktion voraussetzen. Ein solcher Dialog dient der Zusammenfassung der Eingaben (bspw. bei einer Überweisung oder einer Bestellung) und bietet die Möglichkeit, im Sinne der Eingabeunterstützung, Fehler frühzeitig zu erkennen, Dateneingaben rückgängig zu machen oder diese zu korrigieren.

Anforderung 16: Aktivierung von Barrierefreiheitsfunktionen, Benutzerdefinierte Einstellungen und Dokumentation

Sofern ein Internetauftritt Funktionen für Barrierefreiheit beinhaltet, um spezielle Bedürfnisse verschiedener Nutzergruppen zu erfüllen, ist sicherzustellen, dass die Aktivierung der Barrierefreiheitsfunktionen barrierefrei erfolgen kann. Zu diesen Funktionen zählen eine Vorlesefunktion, die Kontrasterhöhung und Anpassung der Schriftgröße, sowie Einstellungen zum Deaktivieren des automatischen Abspielens von Videos, Audio oder Animationen.

- Weiterhin gilt die Erhaltung der Barrierefreiheitsinformationen, bei einer Konvertierung der Webseite in andere Formate als **allgemeine Anforderung**.
- Ergänzend sind die **Benutzerpräferenzen**, wie bspw. geänderte Vorder- und Hintergrundfarben, Schriftarten, Schriftgrößen, zu berücksichtigen.
- Wird eine **Dokumentation** bereitgestellt, sind in dieser ebenfalls die Eigenschaften und Funktionen zur Barrierefreiheit zu benennen.
- Zum Verständnis und der **Lesbarkeit**, sind die Texte in der Hauptsprache anzugeben und anderssprachige Wörter und Abschnitte auszuzeichnen. Hierfür gilt eine Sprache für die gesamte Internetseite oder Anwendung. übersetzbare Texte werden entsprechend der benutzerdefinierten Einstellungen aus der Datenbank gelesen

Anforderung 17: Privatsphäre

Bei der Nutzung der angebotenen Barrierefreiheitsfunktionen, muss die Privatsphäre der Nutzenden geschützt wird.

- Werden Funktionen zur Barrierefreiheit von Webseiten angeboten, ist eine Bedienungsform erforderlich, mit der die Privatsphäre der Nutzenden gewahrt wird.

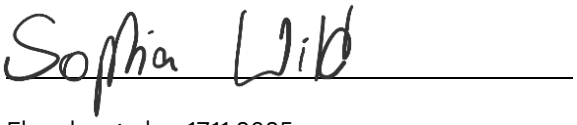
Anforderung 18: Alternativen zur biometrischen Identifizierung und Steuerung

Werden biometrische Merkmale (z.B. Fingerabdruck. Gesichtserkennung. Spracherkennung) zur Identifikation der Nutzenden eingesetzt. ist dieses nicht nur auf ein Merkmal zu beschränken.

- Als **alternative nicht-biometrische Methoden** stehen die Eingabe von Benutzername und Passwort zur Verfügung.
- Auch eine **zweite biometrische Methode** kann eingesetzt werden. wenn zusätzliche Kombinationen aus: Fingerabdruck, Irismuster. Gesichtserkennung und Sprachmuster-Erkennung angeboten werden.

Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen benutzt habe.



Flensburg, den 17.11.2025