F.-A.-Brockhaus-Gymnasium

Schuljahr: 2023/24

Facharbeit
Grundkurs Informatik

Barrierefreiheit im Web: Möglichkeiten als Nutzer und Entwickler

Eine Arbeit zu den Möglichkeiten der Barrierefreiheit im Internet, welche sich intensiv mit der Umsetzung einer barrierefreien Website beschäftigt.

Verfasser: Louis Taube

Betreuerin: Frau Webs

Abgabetermin: 01.03.2024

Inhaltsverzeichnis

1 Einleitung **1**
2 Begriffserklärung **2**
3 Möglichkeiten als Nutzer **3**
3.1 Screenreader 3
3.2 Vergrößerungssoftware 4
3.3 Tastaturbedienung 4
4 Möglichkeiten als Entwickler **6**
4.1 Rechtliche Grundlagen und Normen 6
4.2 Barrierefreies Design 7
4.3 Seitenstruktur 8
5 Mehraufwand einer barrierefreien Website **10**
5.1 Vorgehensweise und Technologien 10
5.2 Programmierung einer barrierefreien Website 11
5.3 Überprüfung der Barrierefreiheit 13
5.4 Bewertung des Mehraufwandes 14
6 Fazit **15**
7 Anhang **16**
7.1 Anh. 1: Umfrage Entwickler 16
7.2 Anh. 2: Einblick UI 18
7.3 Anh. 3: Eindruck Vokabel-Box 19
7.4 Anh. 4: Analysebericht Lighthouse 20
8 Literaturverzeichnis **29**
9 Selbstständigkeitserklärung **31**

1 Einleitung

Barrierefreiheit – ein Thema für uns alle. Ein Thema, welches in unserer Gesellschaft zurecht immer mehr Zuspruch und Platz bekommt.
Dazu digitalisiert sich unsere Gesellschaft immer weiter. Fast jeder Deutsche besitzt bereits einen Internetzugang. Ein Zugang zum Internet bedeutet auch Zugriff auf zahlreiche Möglichkeiten und Werkzeuge. Nutzern mit Einschränkungen sollten diese Werkzeuge aber nicht verwehrt bleiben. Aus diesem Grund beschäftigt sich diese Facharbeit intensiv mit den Möglichkeiten für Nutzer und Entwickler, um die Barrieren in der Webnutzung zu überwinden.

Das Ziel dieser Facharbeit ist es, auf die Wichtigkeit des Themas aufmerksam zu machen und herauszufinden, mit welchen Methoden die Umsetzung dieses Themas bereits auf verschiedenen Webseiten erfolgt. Des Weiteren habe ich ein persönliches Interesse daran, die Webseiten, an denen ich arbeite, auch weitestgehend barrierefrei gestalten zu können. Ich möchte herausfinden, wie viel Mehraufwand die barrierefreie Gestaltung einer Webseite benötigt und wie man diesen Aufwand reduzieren kann. Dazu werde ich eine eigene Website programmieren, welche über Barrierefreiheit informiert und dabei bestmöglich barrierefrei sein sollte. Zudem werde ich im Theorieteil meiner Facharbeit mein Wissen über die Möglichkeiten der Barrierefreiheit von Webseiten für Nutzer und Entwickler erweitern und so sehr vertiefen, dass ich in der Lage bin, die Barrierefreiheit einer Website zu bewerten.

Um die Inhalte meiner Facharbeit bestmöglich auswählen zu können, habe ich eine anonyme Umfrage[1] mit mehreren Informatikern durchgeführt, welche ihre Einschätzung zur Wichtigkeit von Barrierefreiheit auf Webseiten, sowie ihre eigenen Erfahrungen mitteilen konnten. Diese diente aber nur der Unterthemen-Spezialisierung und floss nicht weiter in diese Arbeit ein.

[1] Anh. 1: Umfrage Entwickler

2 Begriffserklärung

Spricht man von Barrierefreiheit im Internet, handelt es sich häufig nicht nur um die vereinfachte Nutzung der Websites von körperlich eingeschränkten Personen, sondern von jeglicher Art der Einschränkung [1]. Wie zum Beispiel ein hohes Alter oder ein falsches Endgerät, welche hierbei ebenfalls als Einschränkungen gelten. An diesem Punkt unterscheidet sich der Begriff der Barrierefreiheit in der Entwicklung zum Begriff Barrierefreiheit im reellen und allgemeinen Leben.

Wenn man sich die Definition bezüglich Barrierefreiheit in der Webnutzung anschaut, erkennt man, dass es sehr viele Einschränkungen und Aspekte geben kann, die den Umgang damit für Menschen mit Einschränkungen jeglicher Art erschweren können. Keine Website kann zu 100 Prozent barrierefrei sein, das ist eine utopische Vorstellung! Darum sollte es in der Webentwicklung immer das Ziel sein, die Webseite so gut wie möglich barrierefrei zu gestalten.

3 Möglichkeiten als Nutzer

Für Menschen, welche das Internet täglich ohne Einschränkungen nutzen können, ist es zu einer Selbstverständlichkeit geworden, sich frei im Web zu bewegen, ihr einziger Kritikpunkt gilt dann eher den schlechten oder älteren Webseiten, oder dem veralteten User Interface (Benutzeroberfläche). Was aber für diese Nutzer nur ein kleines optisches Problem bedeutet, kann für Anwender mit Einschränkungen ein echtes Hindernis sein. Diese Hindernisse werden leider nicht immer, eher sehr selten, von den Verantwortlichen entfernt. Daher müssen sich Betroffene selbst Hilfe zur Anwendung suchen. Diese Hilfe kann in der Software-, sowie in Hardware-Lösungen liegen. Einige ausgewählte IT-Lösungen möchte ich Ihnen hierzu gern vorstellen.

3.1 Screenreader

Eine der Software Anwendungsmöglichkeiten für Menschen mit Sehschwäche oder vollkommener Blindheit sind Screenreader. Diese lesen dem Nutzer die Inhalte einer Website laut vor. Er soll die visuellen Inhalte von Webseiten in Sprache umwandeln bzw. gibt die Inhalte so wieder, dass Menschen mit Sehbeeinträchtigung diese wahrnehmen können. Hierbei entsteht allerdings das Problem, dass sie nicht die ganze Website umwandeln soll, sondern nur die Inhalte und Informationen, welche für den Nutzer wichtig sein sollen. Dabei sind die Screenreader auf die Hilfe von Entwicklern angewiesen. Des Weiteren muss ein Screenreader auch in der Lage sein, Inhalte zu bedienen, wie z. B. Input Felder auszufüllen oder Checkboxen zu aktivieren bzw. zu deaktivieren.

Die Screenreader sind IT-Lösungen, welche vom Nutzer auf dem digitalen Endgerät installiert werden müssen bzw. abhängig vom Betriebssystem schon vorinstalliert sind und nur noch aktiviert werden müssen. Entwickler können die Screenreader unterstützen, indem Sie auf z.B. eine klare Quelltextstruktur[2] achten. Etablierte Screenreader sind z. B. JAWS (Windows), VoiceOver (MacOS) oder TalkBack (Android) [1]. Dennoch sind Screenreader

[2] weitere Informationen: 4.3 Seitenstruktur

keine perfekte Lösung, da sie von der sauberen und strukturierten Arbeit des Entwicklers abhängig sind.

3.2 Vergrößerungssoftware

Ein weitere verbreitete Möglichkeit und Software-Lösung für Nutzer mit Sehschwäche ist Vergrößerungssoftware. Diese ist in vielen Browsern und Betriebssystemen schon vorinstalliert. Der Sinn solcher Software bzw. die Funktion liegt darin, kleine Texte oder Details auf dem Bildschirm zu erkennen. Die Nutzer mit Seheinschränkungen können die Inhalte der Seiten auf ihre passende gewünschte Größe skalieren.

Die Funktionen dieser IT-Lösungen variieren von Modus und Art der Vergrößerungssoftware. Beispielsweise gibt es Anwendungen, welche die gesamte Website vergrößern, oder auch nur einen bestimmten Bereich, wie mit einer Lupe skalieren. Ergänzend dazu ändern viele Vergrößerungssoftware auch noch die Farben, um somit ein besseres Kontrastverhältnis zu erzeugen [2]. Der Nachteil dieser Software-Lösungen liegt darin, dass die Software oft nur einen Teil des Bildschirms vergrößert, wobei teilweise der Kontext des Inhalts verloren gehen kann oder die Nutzer ein Pop-up Window nicht mitbekommen, da es außerhalb ihres Sichtbereiches aufgeht [2].

Vergrößerungssoftware ist eine weitere Möglichkeit für einen effektiveren und erleichterten Umgang auf Websites für Nutzer mit Einschränkungen. Sie sind die perfekte Ergänzung für Nutzer, welche nur eine Sehschwäche haben und nicht vollkommen erblindet sind.

3.3 Tastaturbedienung

Einer der wichtigsten Fragen, welche für Nutzer mit Einschränkungen beantwortet werden muss, ist die Navigation. Wie navigiert man sich auf der Website, wenn man nichts sieht? Oder wie navigiert man sich, wenn man nur eine Hand hat? Die Antwort für diese Nutzer kann die Tastaturnavigation sein.

Die Tastaturnavigation bezieht sich auf die Möglichkeit, Webinhalte nur durch die Tastatur zu erreichen. Dies ermöglicht auch Menschen mit

motorischen Beeinträchtigungen, oder Nutzern, welche Schwierigkeiten mit dem präzisen Umgang mit einer Maus haben, eine zielführende Navigation im Internet. Diese präzise Navigation ist allerdings wieder auf die Sauberkeit des Quellcodes der Seite angewiesen. Die Navigation erfordert, dass sämtliche Funktionen und Interaktionen, welche mit der Maus erfolgen, auch über die Tastatur steuerbar sind. Dies bedeutet im Speziellen, dass Nutzer sich mit Tastenkombinationen durch die Inhalte der Websites, wie z. B. Links, Schaltflächen oder Input-Feldern, navigieren können.

Des Weiteren sind die Tastenkombinationen oft gleich, um den Nutzern das Lernen zu erleichtern. Beispielsweise kann man sich mit der Tabulatortaste von Element zu Element navigieren und mit der Enter- bzw. der Leertaste kann man bestätigen. Dies ist aber nur möglich, wenn Entwickler ihre Elemente alle einzeln implementieren und somit keine Doppelauswahl von einem Element möglich ist. [3]

4 Möglichkeiten als Entwickler

Bei all den Möglichkeiten, welche Nutzer haben, um sich barrierefreier im Internet zu bewegen, ist hervorzuheben, dass die Beseitigung dieser Barrieren eigentlich die Aufgabe der Entwickler der jeweiligen Webseiten ist. Des Weiteren funktioniert ein Großteil der IT-Lösungen für die Nutzer nur als Ergänzung und auch nur dann, wenn sich die Entwickler der Webseiten an bestimmte Normen und Regeln halten, welche vorgegeben werden. Ausgewählte Möglichkeiten für die Barrierefreiheit aus der Entwicklersicht stelle ich Ihnen im folgenden Abschnitt vor.

4.1 Rechtliche Grundlagen und Normen

Barrierefreiheit – ein Thema, welches in vielen Bereichen vorgeschrieben ist, so auch im Web, denn Barrierefreiheit ist mehr als eine moralische Verpflichtung. Entwickler werden zu Recht verpflichtet, sich an bestimmte Vorgaben sowie Regeln zu halten und diese während ihrer Arbeit umzusetzen. Besonders betroffen ist die Arbeit an Behördenseiten. Dadurch sind Webentwickler verpflichtet, einen Zugang für alle Nutzer bereitzustellen. Sie tragen damit zu einem gerechten Internet bei.

Für die Einführung von Regelungen der Barrierefreiheit, berufen sich viele Länder auf die internationalen Web Content Accessibility Guidelines (WCAG). Diese Richtlinien sind vom World Wide Web Consortium (W3C) verfasste und veröffentlichte Richtlinien, sowie Empfehlungen, nach welchen sich Entwickler und Länder bei der Entwicklung bzw. bei der Gesetzgebung orientieren können. Ein Gesetz, welches die Barrierefreiheit in Deutschland regeln soll, ist die „Barrierefreie-Informationstechnik-Verordnung - BITV 2.0”. [4]

Dieses Gesetz setzt die EU-Richtlinie 2016/2102 in Deutschland durch. Dieses BITV 2.0 wird oft kritisiert, da es nur an staatlichen Behörden bzw. Organisationen zur Pflicht wird. Des Weiteren verweist die BITV auf die Europäische Norm „ETSI EN 301 549 - V3.2.1 - Accessibility requirements for ICT products and services”, welche sich an der oben erwähnten WCAG orientiert. [4]-[6]

Somit ist auch die BITV 2.0 nicht die Regelung, die den Wunsch nach Checklisten für die Umsetzung von Barrierefreiheit vieler Entwickler erfüllt, obwohl Einheitlichkeit so ein wichtiges Thema für die Entwicklung von barrierefreien Inhalten ist.

4.2 Barrierefreies Design

Wenn wir über barrierefreies Design reden, sprechen wir nicht davon, dass wir für Menschen mit Einschränkungen ein neues Design und eine neue UI erstellen müssen. Barrierefreies Design ist Design für alle [7]. Die Vorteile liegen nicht nur bei Nutzern mit Einschränkungen, sondern bei allen Nutzern. Beispielsweise erleichtert ein starkes Kontrastverhältnis auch Nutzern, ohne Einschränkungen, die Inhalte einer Website unter Blendung durch ungünstige Sonneneinstrahlung zu erkennen. Oder ein sich verändertes Layout (Responsive Design) ist nicht ausschließlich für die Vergrößerungen auf einem Bildschirm gut, sondern kann sich auch den verschiedenen Größen der Bildschirme anpassen.

Bei der Verwendung von Farben gibt es viele Aspekte bei der Entwicklung zu beachten. Beispielsweise sollten Informationen nicht nur durch Farben übermittelt werden, damit auch farbenblinde Nutzer diese Anwendung nutzen und bedienen können. Weiterhin sollte auf ein starkes Kontrastverhältnis geachtet werden, um auch Nutzern mit Sehschwäche eine gute Arbeit mit der Website zu ermöglichen. Bestimmte Kontrastverhältniswerte werden z. B. in den WCAG empfohlen. Eine weitere Möglichkeit ein hohes Kontrastverhältnis anzubieten, wäre es, einen Button auf der Website einzufügen, welcher die Hintergrundfarbe und die Textfarbe tauschen.

Eine große Hilfe für Nutzer mit Einschränkungen ist das Vergrößern von Inhalten mit externen Anwendungen oder internen Buttons der Website. Die Aufgabe der Entwickler liegt darin, dass die Inhalte so designt werden, dass sie mindestens bis zu 200 % skaliert werden können, ohne die Funktion der Website zu verlieren. Dies gilt für die komplette Website. Um diese Skalierung zu ermöglichen, wird oft ein Responsive Design benötigt [8]. Ein Responsive

Design ist ein flexibles Design, es passt sich der Bildschirmbreite und Höhe an, ohne dass der Entwickler mehrere Designs anfertigen muss [9, Abs. 4].

Bei der Entwicklung von Websites ist Regelmäßigkeit wichtig. Entwickler sollten beispielsweise für Überschriften einer Ebene immer das gleiche Layout und Design verwenden. Des Weiteren sollte Text am besten rechts- oder linksbündig positioniert sein und eine Zeile sollte nicht mehr als 80 Zeichen beinhalten [8].

4.3 Seitenstruktur

Wenn man von Möglichkeiten für Entwickler ihre Seiten barrierefrei zu machen spricht, ist eine saubere organisierte Seitenstruktur, nicht nur im visuellen, sondern auch im Quelltext, für einen Entwickler am einfachsten und schnellsten umzusetzen.

Eine große Hilfe für Screenreader für Nutzer ist eine klare und lineare Quelltext-Struktur. Gemeint ist hierbei die Anordnung der HTML-Elemente im Quellcode. Diese sollten linear sein und durch das Stylen der Elemente, z. B. mit CSS oder einem CSS-Framework wie TailwindCSS[3], sollte keine große Veränderung der Position und der Elemente im Vergleich zur visuellen Website vorhanden sein. Beispielsweise sollte eine Navigationsleiste (Navbar), welche an den oberen Bildschirmrand geheftet ist, auch im Quellcode ganz oben stehen. Dadurch ermöglicht der Entwickler den Nutzern, mit deren Screenreadern ähnliche Websites mit dem gleichen Layout zu besuchen, da der Screenreader sich am Quellcode orientiert.
Entwickler sollten ebenso beachten, falls sie nicht englische Inhalte entwickeln, die verwendete Sprache im HTML-Element zu kennzeichnen (`<html lang="de">`), dies sollte genauso bei Zitaten in Blockquote-Elemente erfolgen [8].

HTML in seiner Urform, ohne Frameworks, bietet auch einige Möglichkeiten, um Screenreader zu unterstützen. Eine davon ist, dass der Entwickler die richtigen HTML-Elemente nutzt. Dies klingt am Anfang sehr einfach, kostet aber doch ein wenig Disziplin, da Entwickler eben nicht nur ein

[3] www.tailwindcss.com

Span-Element (`<span>Lorem Ipsum<span/>`) nutzen und das Layout mit CSS verändern kann, sondern dass für eine Überschrift die richtige Ebene genutzt werden muss. Beispielsweise Ebene 1 (`<h1>Lorem Ipsum<h1/>`). Dies bezieht sich auch auf alle verschiedenen HTML-Elemente, solange der Entwickler die Möglichkeit hat, etwas über ein Element zu spezialisieren, sollte er dies auch tun. Der Aufwand lohnt sich, denn bei den richtigen Elementen erkennt auch der Screenreader, ob es sich um einen normalen Text handelt oder um eine wichtige Überschrift.
Einzelne CSS-Frameworks bieten genauso Möglichkeiten an, um die Screenreader zu unterstützen. Beispielsweise bietet TailwindCSS bestimmte Klassen an, um Dinge visuell von der Website auszublenden, aber dem Screenreader diese Elemente trotzdem anzuzeigen [10].

Weitere Möglichkeiten, welche ein wenig zeitintensiver sind, wären Alternativinhalte. Das fängt bei einfachen Beschreibungen von wichtigen Bildern mithilfe des `alt=""` Attributs bei der Implementierung von Bildern an (`<img src="" alt=""/>`), kann aber eben genauso etwas komplizierter werden, wenn eine Beschreibung von Diagrammen implementiert wird. Was man dabei auf keinen Fall vergessen darf, ist, dass es nicht darum geht, alles zu beschreiben, sondern nur die wichtigsten Inhalte wiederzugeben. Ein Bild, welches nur für die Gestaltung im Hintergrund liegt und keinen informativen Mehrwert hat, sollte auch nicht beschrieben werden.

5 Mehraufwand einer barrierefreien Website

Ein großes Argument von Web-Entwicklern gegen die Barrierefreiheit im Web ist die Zeit. Einige Entwickler sind der Meinung, dass die Zeit, die eine barrierefreie Website mehr benötigt, zu viel ist und nicht genügend entlohnt wird. Des Weiteren wird oft kritisiert, dass keine verbindlichen Richtlinien für Entwickler in der freien Wirtschaft existieren, was dazu führt, dass Auftraggeber oft für Barrierefreiheit nicht zahlen wollen. Aber wie groß ist der zeitliche Mehraufwand wirklich? - Dies möchte ich in diesem Abschnitt meiner Arbeit untersuchen. Dafür habe ich eine Website[4] programmiert, welche die Vorgaben meiner Facharbeit für Barrierefreiheit versucht umzusetzen und gleichzeitig noch über Barrierefreiheit informiert.

5.1 Vorgehensweise und Technologien

Um den Zeitaufwand meiner barrierefreien Website bestmöglich bewerten zu können, habe ich die gleichen Frameworks und Programmiersprachen verwendet, wie in meinen sonstigen Projekten. Bei Frameworks handelt es sich um Grundgerüste für das Programmieren, welche bestimmte Prozesse beim Programmieren optimieren und effizienter laden [11]. Als Programmiersprache verwende ich Typescript bzw. JavaScript, wobei es sich bei Typescript um eine Weiterentwicklung von JavaScript handelt. Als Web-Framework kommt SvelteKit[5] zum Einsatz mit dem CSS-Framework TailwindCSS als Ergänzung. In Verbindung mit TailwindCSS basiert mein Layout und Design der barrierefreien Website auf einer eigenen UI-Bibliothek[6], die in Zusammenarbeit mit anderen Entwicklern während einer aktuellen Neuaufsetzung unseres gemeinsamen Projektes „Vokabel-Box”[7] entstanden ist, aber von mir mithilfe von TailwindCSS umgesetzt wurde.

Für die Entwicklung und Planung der Website entwarf ich ein Figma Design[8], wobei es sich um einen Vorentwurf der Webseite handelt, welcher das

[4] **Aufrufbar unter:** facharbeit.louist2469.de
[5] https://kit.svelte.dev/
[6] siehe 7.2 Anh. 2: Einblick UI
[7] siehe 7.3 Anh. 3: Eindruck Vokabel-Box
[8] **Aufrufbar unter:** facharbeit.louist2469.de/figma-design

Layout wiedergibt. Dieser Vorentwurf trug dazu bei, ein einheitliches Layout zu entwickeln und sich auch erste Gedanken zur Umsetzung der Barrierefreiheit zu machen.

Sobald ich den groben Entwurf meiner Website fertig hatte, konnte ich mich nun der Umsetzung widmen. Dies tat ich mithilfe von TailwindCSS in meinem SvelteKit-Projekt[9]. Für eine bessere Einschätzung der Barrierefreiheit lag mein Fokus nach der Umsetzung darin, die Barrierefreiheit zu überprüfen und wenn nötig zu verbessern. Um eine klare Dokumentation meiner Arbeit durchzuführen und gleichzeitig eine Sicherung meiner täglichen Arbeit zu haben, erstellte ich ein GitHub-Projekt[10] mit meiner Website. Bei Github handelt es sich um ein internationale Plattform mit einer großen Community, welche sich auf die Sicherung und die Verwaltung von Programmierprojekten spezialisiert hat [12]. Dieses GitHub-Projekt wurde dann anschließend über die Web-Deployment-Plattform Vercel[11] veröffentlicht und mit der zugeordneten Domain bzw. Subdomain verknüpft [13].

5.2 Programmierung einer barrierefreien Website

Für eine bestmögliche Umsetzung des visuellen Figma-Entwurfs, welcher auch barrierefrei ist, konzentrierte ich mich erstmals nach der Aufsetzung des Projektes um die Umsetzung der HTML-Struktur, in Ergänzung mit TailwindCSS.

Die Schwierigkeit lag für mich darin, die richtigen HTML-Elemente zu benutzen. Dies hing damit zusammen, dass ich in meinen vorigen Projekten die meisten Elemente nur durch CSS nachgestylt habe und die Funktionen durch JavaScript implementiert hatte, wobei das richtige Element nebensächlich war.

Eine weitere Schwierigkeit bei der anfänglichen Umsetzung lag darin, dass ich im Vergleich zum Figma-Design keine abweichende Design für die Bildschirmgrößen entwickeln wollte, mein Ziel lag darin, das Design für alle Bildschirmgrößen so umzusetzen, dass so wenig Code wie möglich benötigt wird und er bestmöglich Effizient ist. Der Vorteil von diesem Aspekt liegt darin,

[9] **Mehr Informationen:** 5.2 Programmierung einer barrierefreien Website
[10] **Aufrufbar unter:** https://github.com/LouisT2469/Barrierefreiheit-im-Web
[11] https://vercel.com/

dass TailwindCSS durch ihre unterschiedlichen Klassen, auch Breakpoints hat, welche man vor die anderen Designklassen setzen kann, um damit ab bestimmten Größen die Inhalte zu verändern [14]. In Verbindung mit Flex-Boxen konnte ich die verschiedenen Bildschirmgrößen gut umsetzen. Trotz alledem war dies nicht überall möglich, beispielsweise bei der Navigationsleiste am oberen Bildschirmrand.

Diese anfänglichen Schwierigkeiten ließen sich aber lösen und hielten mich nicht davon ab, schon von Anfang an einzelne Aspekte der Barrierefreiheit umzusetzen. Ein Aspekt davon war die Verwendung von beschreibenden Wortgruppen, welche ich in bestimmte Elemente mit Funktionen als Attribut zuwies, beispielsweise Elemente, die einen auf andere Seiten weiterleiten. Für diese beschreibenden Wortgruppen nutze ich das Attribut `aria-label="Lorem Ipsum"`. Des Weiteren achtete ich von Beginn an auf die richtige Verwendung der Funktion entsprechenden Elemente, sowie auf eine klare gerenderte Quelltextstruktur.

Für eine bessere Barrierefreiheit fing ich nach dem das Grundgerüst der Website stand an, an weiteren Features der Barrierefreiheit zu arbeiten. Ich begann mit der Arbeit an einem Button, welcher die Schriftgröße vergrößern sollte. Für die Umsetzung der verschiedenen Schriftgrößen nutze ich ein spezielles Feature von Svelte, das Binding-Feature. Dieses ermöglichte mir, dass ich innerhalb von HTML-Tags Variablen verwenden konnte, um die Inhalte dynamisch zu machen. Für effizienteres Arbeiten und um nicht die gleiche Code-Logik immer wieder verwenden zu müssen, erstelle ich einen Text-Komponente, welcher die benötigte Logik für die Größenveränderung von 100% zu 200% beinhaltet. Diese Komponente kommt in meinem Projekt immer dann zum Einsatz, wenn ich irgendwo Text einfügen möchte. Auf das Feature eines Buttons, der die Farben tauschen würde, um ein besseres Kontrastverhältnis zu erzeugen, habe ich verzichtet, da meine Farben schon einen hohen Kontrast haben. Aber auch dies wäre mit einer Verbindung aus Svelte und TailwindCSS möglich gewesen. Ich hätte If-Verzweigungen in den HTML-Elementen genutzt, um verschiedene Farben bzw. Themes zu

implementieren. Diese Themes würden dann durch einen Button getauscht werden.

Nach dem erfolgreichen Testen der Website schloss ich diesen Abschnitt der Seite und hatte Version 1.0 meines Projektes.

5.3 Überprüfung der Barrierefreiheit

Ein nerviger, aber dennoch elementarer Schritt zur Fertigstellung von barrierefreien Webseiten, ist die Überprüfung der Barrierefreiheit. Während dieser Phase untersucht man die Website auf die Zugänglichkeit für Nutzer und passt, falls nötig, einzelne Parameter oder Klassen an. Des Weiteren geht es um die Überprüfung der Erfüllung aller Standards bzw. Regeln, wenn welche vorhanden sind.

Ich habe für die Überprüfung meiner Website verschiedene Werkzeuge und Methoden genutzt. Angefangen bei der erneuten Sichtung meines Quelltextes und der Überprüfung der linearen Quelltextstruktur nach dem Rendern der Seite. Des Weiteren habe ich mir verschiedene Screenreader installiert und diese auf meiner Website laufen lassen und wenn nötig, den Quelltext verändert.
Darüber hinaus habe ich mit den Barriere-Tools von Microsoft Edge das Kontrastverhältnis überprüft, sowie die Richtigkeit der HTML-Elementreihenfolge. Zu guter Letzt habe ich mein Projekt auf verschiedenen Geräten und Browsern geöffnet und, falls nötig, das Layout noch angepasst.
Als finale Prüfung der Barrierefreiheit der Website, lies ich eine Weiterentwicklung der Google Erweiterung „Lighthouse” meine Website analysieren. Diese Erweiterung prüft verschiedene Aspekte der Website, wie z. B. die Barrierefreiheit und wertet diese in einem Dokument aus[12]. [15]

Durch diese Kombination aus Testmethoden, konnte ich sicherstellen, dass meine Website meinen Kriterien der Barrierefreiheit entspricht und im besten Falle für alle Nutzer zugänglich ist.

[12] siehe 7.4 Anh. 4: Analysebericht Lighthouse

5.4 Bewertung des Mehraufwandes

Die Bewertung des Mehraufwands für die Entwicklung einer barrierefreien Website ist eine wichtige Überlegung. Insbesondere für Entwickler, aber auch für deren Auftraggeber. In diesem Abschnitt möchte ich den Mehraufwand meiner Website analysieren und bewerten. Zunächst möchte ich aber festhalten, dass die Zeit, die für die Implementierung von Barrierefreiheit von verschiedenen Faktoren abhängig ist.
Darunter sind z. B. die Komplexität der Website, die verwendeten Techniken und natürlich die Erfahrung des Entwicklers mit Barrierefreiheit. Des Weiteren ist die Bewertung von Aufwand subjektiv. Ich werde also in diesem Abschnitt von meinen Erfahrungen und Eindrücken berichten.

In meinem Fall war der Mehraufwand zu früheren Projekten leicht spürbar. Wobei ich auch viel Zeit in die Informationsbeschaffung investieren musste. Der Mehraufwand war für mich aber genauso groß, wie wenn ich mich mit anderen neuen Themen beschäftige. Allerdings musste ich feststellen, dass zum Thema Barrierefreiheit nur sehr wenig aktuelles Material verfügbar ist. Dies bestätigte auch meine kleine Umfrage unter Entwicklern bzw. Informatikern. Existierende Checklisten sind oft von keiner offiziellen Stelle und wenn, dann sehr schwammig gehalten. Dinge, die sich aber leichter umsetzen ließen, waren z. B. das Responsive Design oder Alternativtexte, zu dem Gelingen dieser Punkte trug TailwindCSS bei.

Zusammenfassend muss ich aber sagen, dass dieser zusätzliche Zeitaufwand mich nicht davon abhält, der festen Überzeugung zu sein, dass sich das Investieren in Barrierefreiheit lohnt. Barrierefreiheit im Web bringt nicht nur Vorteile für Nutzer mit Einschränkungen, sondern hat auch positive Auswirkungen auf die Reichweite und Vergrößerung der Zielgruppe von Webseiten.

6 Fazit

Für unsere digitale Gesellschaft ist eine Auseinandersetzung mit Barrierefreiheit im Internet von großer Bedeutung. In dieser Facharbeit habe ich versucht, eine Vielfalt an Möglichkeiten für Nutzer, genauso wie für Entwickler darzulegen. Es ist mir wichtig, nochmal zu betonen, dass dies lediglich ausgewählte Möglichkeiten waren, für die Behebung der größten Barrieren. Zukünftig werden noch viele weitere Techniken entwickelt werden, insbesondere mit Künstlicher Intelligenz ist noch sehr viel für Entwickler, genauso für Nutzer, möglich.

Wie ich schon in Kapitel 3. „Möglichkeiten für Nutzer" dargelegt habe, stehen für Nutzer zahlreiche Möglichkeiten, wie Screenreader zur Verfügung. Ihre Existenz entbindet aber nicht die Webentwickler, sich mit dem Thema zu beschäftigen.

Wir als Gesellschaft müssen daran arbeiten, dass wir auch im Web besondere Richtlinien bzw. einfache Checklisten für Entwickler bereitstellen, damit diese mit möglichst wenig Aufwand ihre Seite barrierefrei gestalten können. Die Gesellschaft muss in einen gerechten Zugang ins Internet für alle Nutzer investieren!

Des Weiteren ist Barrierefreiheit im Web ein Thema, was sich kontinuierlich verändert, Entwickler und Auftraggebende sollten sich immer wieder bemühen, aktuelle Barrieren von ihren Website zu entfernen, um der utopischen Vorstellung der unerreichbaren 100 Prozent der Barrierefreiheit immer ein Stück näherzukommen.

7 Anhang

7.1 Anh. 1: Umfrage Entwickler

Um möglichst praxisnah auf die in meiner Facharbeit zu behandelnden Möglichkeiten zur Herstellung von barrierefreien Websites einzugehen, habe ich eine Umfrage unter Entwicklern durchgeführt. Dies ermöglichte mir einen realistischen Einblick in die aktuelle Situation zu erhalten.

Fragen:

1. Zu welcher Altersgruppe gehören Sie?*
   - ☐ unter 25 Jahre
   - ☐ 25 - 34 Jahre
   - ☐ 35 - 44 Jahre
   - ☐ 45 - 54 Jahre
   - ☐ 55 -64 Jahre
   - ☐ 65 Jahre oder älter

2. In welchem Bereich der Informatik/ IT arbeiten Sie?*

   _____

3. Inwiefern spielt das Thema Barrierefreiheit in Ihrer Entwicklungsarbeit eine Rolle?*

   0 1 2 3 4 5 6 7 8 9 10

4. Hatten Sie schon mal Kontakt mit Barrierefreiheit im Web während Ihrer Arbeit? (wenn ja, wobei?)

   _____

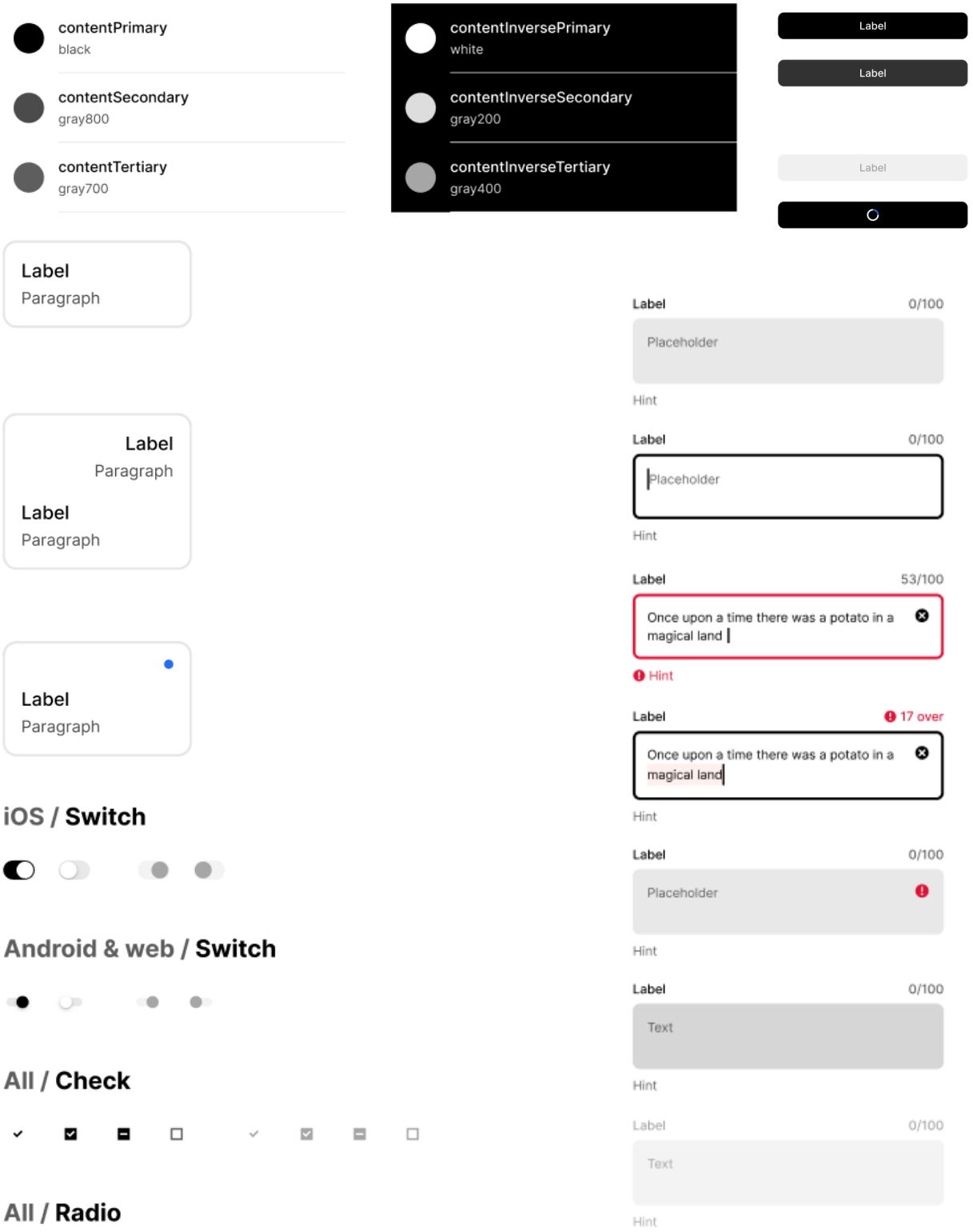
5. Wie bedeutsam sind diese Tools und Optionen zur Barrierefreiheit im Web aus Ihrer Sicht?*
   - Responsive Design

- Starke Farbkontraste
- unterschiedliche Schriftgrößen
- Screenreaders

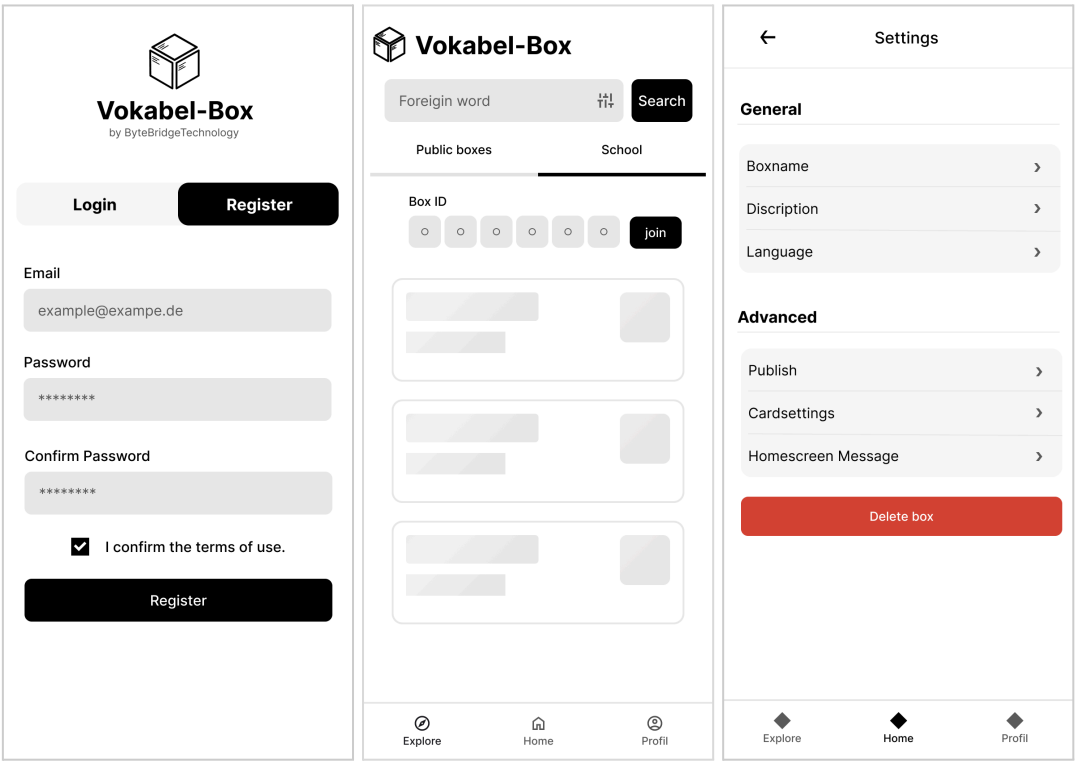
6. Wie wichtig ist Barrierefreiheit im Vergleich zu Funktionalität und Sicherheit auf Websites?
   0 1 2 3 4 5 6 7 8 9 10

7. Welche Herausforderungen sehen Sie bei der Umsetzung von Barrierefreiheit auf Websites, und wie überwinden Sie diese?*
   _____

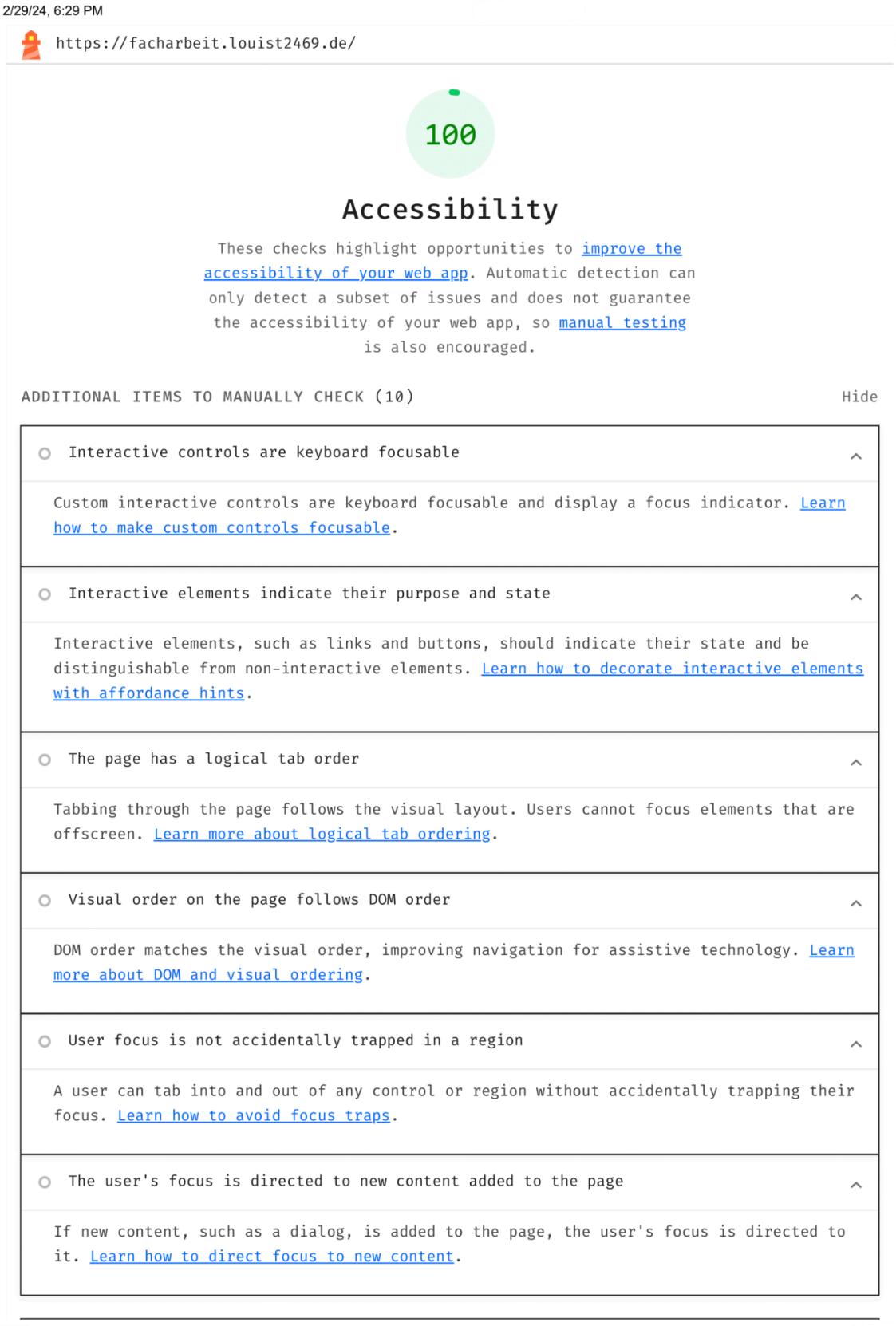
7.2 Anh. 2: Einblick UI



7.3 Anh. 3: Eindruck Vokabel-Box



7.4 Anh. 4: Analysebericht Lighthouse



2/29/24, 6:29 PM

https://facharbeit.louist2469.de/

100

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Hide

- Interactive controls are keyboard focusable

  Custom interactive controls are keyboard focusable and display a focus indicator. Learn how to make custom controls focusable.

- Interactive elements indicate their purpose and state

  Interactive elements, such as links and buttons, should indicate their state and be distinguishable from non-interactive elements. Learn how to decorate interactive elements with affordance hints.

- The page has a logical tab order

  Tabbing through the page follows the visual layout. Users cannot focus elements that are offscreen. Learn more about logical tab ordering.

- Visual order on the page follows DOM order

  DOM order matches the visual order, improving navigation for assistive technology. Learn more about DOM and visual ordering.

- User focus is not accidentally trapped in a region

  A user can tab into and out of any control or region without accidentally trapping their focus. Learn how to avoid focus traps.

- The user's focus is directed to new content added to the page

  If new content, such as a dialog, is added to the page, the user's focus is directed to it. Learn how to direct focus to new content.

- HTML5 landmark elements are used to improve navigation

  Landmark elements (<main>, <nav>, etc.) are used to improve the keyboard navigation of the page for assistive technology. Learn more about landmark elements.

- Offscreen content is hidden from assistive technology

  Offscreen content is hidden with display: none or aria-hidden=true. Learn how to properly hide offscreen content.

- Custom controls have associated labels

  Custom interactive controls have associated labels, provided by aria-label or aria-labelledby. Learn more about custom controls and labels.

- Custom controls have ARIA roles

  Custom interactive controls have appropriate ARIA roles. Learn how to add roles to custom controls.

These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

PASSED AUDITS (14) Hide

- [aria-*] attributes match their roles

  Each ARIA role supports a specific subset of aria-* attributes. Mismatching these invalidates the aria-* attributes. Learn how to match ARIA attributes to their roles.

- [aria-hidden="true"] is not present on the document <body>

  Assistive technologies, like screen readers, work inconsistently when aria-hidden="true" is set on the document <body>. Learn how aria-hidden affects the document body.

- [aria-*] attributes have valid values

  Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid values. Learn more about valid values for ARIA attributes.

- [aria-*] attributes are valid and not misspelled

  Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid names. Learn more about valid ARIA attributes.

Buttons have an accessible name

When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. Learn how to make buttons more accessible.

Image elements have [alt] attributes

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more about the alt attribute.

[user-scalable="no"] is not used in the <meta name="viewport"> element and the [maximum-scale] attribute is not less than 5.

Disabling zooming is problematic for users with low vision who rely on screen magnification to properly see the contents of a web page. Learn more about the viewport meta tag.

Background and foreground colors have a sufficient contrast ratio

Low-contrast text is difficult or impossible for many users to read. Learn how to provide sufficient color contrast.

Document has a <title> element

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more about document titles.

<html> element has a [lang] attribute

If a page doesn't specify a lang attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly. Learn more about the lang attribute.

<html> element has a valid value for its [lang] attribute

Specifying a valid BCP 47 language helps screen readers announce text properly. Learn how to use the lang attribute.

Links have a discernible name

Link text (and alternate text for images, when used as links) that is discernible, unique, and focusable improves the navigation experience for screen reader users. Learn how to make links accessible.

Heading elements appear in a sequentially-descending order

Properly ordered headings that do not skip levels convey the semantic structure of the page, making it easier to navigate and understand when using assistive technologies. Learn more about heading order.

Image elements do not have [alt] attributes that are redundant text.

Informative elements should aim for short, descriptive alternative text. Alternative text that is exactly the same as the text adjacent to the link or image is potentially confusing for screen reader users, because the text will be read twice. Learn more about the alt attribute.

NOT APPLICABLE (45) Hide

[accesskey] values are unique

Access keys let users quickly focus a part of the page. For proper navigation, each access key must be unique. Learn more about access keys.

Values assigned to role="" are valid ARIA roles.

ARIA roles enable assistive technologies to know the role of each element on the web page. If the role values are misspelled, not existing ARIA role values, or abstract roles, then the purpose of the element will not be communicated to users of assistive technologies. Learn more about ARIA roles.

button, link, and menuitem elements have accessible names

When an element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to make command elements more accessible.

Elements with role="dialog" or role="alertdialog" have accessible names.

ARIA dialog elements without accessible names may prevent screen readers users from discerning the purpose of these elements. Learn how to make ARIA dialog elements more accessible.

[aria-hidden="true"] elements do not contain focusable descendents

Focusable descendents within an [aria-hidden="true"] element prevent those interactive elements from being available to users of assistive technologies like screen readers. Learn how aria-hidden affects focusable elements.

- ARIA input fields have accessible names

  When an input field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about input field labels.

- ARIA `meter` elements have accessible names

  When a meter element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to name meter elements.

- ARIA `progressbar` elements have accessible names

  When a progressbar element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to label progressbar elements.

- `[role]`s have all required `[aria-*]` attributes

  Some ARIA roles have required attributes that describe the state of the element to screen readers. Learn more about roles and required attributes.

- Elements with an ARIA `[role]` that require children to contain a specific `[role]` have all required children.

  Some ARIA parent roles must contain specific child roles to perform their intended accessibility functions. Learn more about roles and required children elements.

- `[role]`s are contained by their required parent element

  Some ARIA child roles must be contained by specific parent roles to properly perform their intended accessibility functions. Learn more about ARIA roles and required parent element.

- `[role]` values are valid

  ARIA roles must have valid values in order to perform their intended accessibility functions. Learn more about valid ARIA roles.

- Elements with the `role=text` attribute do not have focusable descendents.

  Adding role=text around a text node split by markup enables VoiceOver to treat it as one phrase, but the element's focusable descendents will not be announced. Learn more about the role=text attribute.

- ARIA toggle fields have accessible names

  When a toggle field doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about toggle fields.

- ARIA `tooltip` elements have accessible names

  When a tooltip element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn how to name tooltip elements.

- ARIA `treeitem` elements have accessible names

  When a `treeitem` element doesn't have an accessible name, screen readers announce it with a generic name, making it unusable for users who rely on screen readers. Learn more about labeling treeitem elements.

- The page contains a heading, skip link, or landmark region

  Adding ways to bypass repetitive content lets keyboard users navigate the page more efficiently. Learn more about bypass blocks.

- `<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>`, `<template>` or `<div>` elements.

  When definition lists are not properly marked up, screen readers may produce confusing or inaccurate output. Learn how to structure definition lists correctly.

- Definition list items are wrapped in `<dl>` elements

  Definition list items (<dt> and <dd>) must be wrapped in a parent <dl> element to ensure that screen readers can properly announce them. Learn how to structure definition lists correctly.

- `[id]` attributes on active, focusable elements are unique

  All focusable elements must have a unique id to ensure that they're visible to assistive technologies. Learn how to fix duplicate ids.

- ARIA IDs are unique

  The value of an ARIA ID must be unique to prevent other instances from being overlooked by assistive technologies. Learn how to fix duplicate ARIA IDs.

- No form fields have multiple labels

  Form fields with multiple labels can be confusingly announced by assistive technologies like screen readers which use either the first, the last, or all of the labels. Learn how to use form labels.

- <frame> or <iframe> elements have a title

  Screen reader users rely on frame titles to describe the contents of frames. Learn more about frame titles.

- <html> element has an [xml:lang] attribute with the same base language as the [lang] attribute.

  If the webpage does not specify a consistent language, then the screen reader might not announce the page's text correctly. Learn more about the lang attribute.

- Input buttons have discernible text.

  Adding discernable and accessible text to input buttons may help screen reader users understand the purpose of the input button. Learn more about input buttons.

- <input type="image"> elements have [alt] text

  When an image is being used as an <input> button, providing alternative text can help screen reader users understand the purpose of the button. Learn about input image alt text.

- Form elements have associated labels

  Labels ensure that form controls are announced properly by assistive technologies, like screen readers. Learn more about form element labels.

- Links are distinguishable without relying on color.

  Low-contrast text is difficult or impossible for many users to read. Link text that is discernible improves the experience for users with low vision. Learn how to make links distinguishable.

- Lists contain only <li> elements and script supporting elements (<script> and <template>).

  Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more about proper list structure.

- List items (<li>) are contained within <ul>, <ol> or <menu> parent elements

Screen readers require list items (<li>) to be contained within a parent <ul>, <ol> or <menu> to be announced properly. Learn more about proper list structure.

- The document does not use <meta http-equiv="refresh">

  Users do not expect a page to refresh automatically, and doing so will move focus back to the top of the page. This may create a frustrating or confusing experience. Learn more about the refresh meta tag.

- <object> elements have alternate text

  Screen readers cannot translate non-text content. Adding alternate text to <object> elements helps screen readers convey meaning to users. Learn more about alt text for object elements.

- Select elements have associated label elements.

  Form elements without effective labels can create frustrating experiences for screen reader users. Learn more about the select element.

- Skip links are focusable.

  Including a skip link can help users skip to the main content to save time. Learn more about skip links.

- No element has a [tabindex] value greater than 0

  A value greater than 0 implies an explicit navigation ordering. Although technically valid, this often creates frustrating experiences for users who rely on assistive technologies. Learn more about the tabindex attribute.

- Tables have different content in the summary attribute and <caption>.

  The summary attribute should describe the table structure, while <caption> should have the onscreen title. Accurate table mark-up helps users of screen readers. Learn more about summary and caption.

- Cells in a <table> element that use the [headers] attribute refer to table cells within the same table.

  Screen readers have features to make navigating tables easier. Ensuring <td> cells using the [headers] attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more about the headers attribute.

- `<th>` elements and elements with `[role="columnheader"/"rowheader"]` have data cells they describe.

  Screen readers have features to make navigating tables easier. Ensuring table headers always refer to some set of cells may improve the experience for screen reader users. Learn more about table headers.

- `[lang]` attributes have a valid value

  Specifying a valid BCP 47 language on elements helps ensure that text is pronounced correctly by a screen reader. Learn how to use the lang attribute.

- `<video>` elements contain a `<track>` element with `[kind="captions"]`

  When a video provides a caption it is easier for deaf and hearing impaired users to access its information. Learn more about video captions.

- All heading elements contain content.

  A heading with no content or inaccessible text prevent screen reader users from accessing information on the page's structure. Learn more about headings.

- Identical links have the same purpose.

  Links with the same destination should have the same description, to help users understand the link's purpose and decide whether to follow it. Learn more about identical links.

- Touch targets have sufficient size and spacing.

  Touch targets with sufficient size and spacing help users who may have difficulty targeting small controls to activate the targets. Learn more about touch targets.

- Tables use `<caption>` instead of cells with the `[colspan]` attribute to indicate a caption.

  Screen readers have features to make navigating tables easier. Ensuring that tables use the actual caption element instead of cells with the [colspan] attribute may improve the experience for screen reader users. Learn more about captions.

- `<td>` elements in a large `<table>` have one or more table headers.

  Screen readers have features to make navigating tables easier. Ensuring that <td> elements in a large table (3 or more cells in width and height) have an associated table header may improve the experience for screen reader users. Learn more about table headers.

8 Literaturverzeichnis

[1] Mozilla, “What is accessibility? - Learn web development | MDN, https://developer.mozilla.org/en-US/docs/Learn/Accessibility/What_is_accessibility#people_with_visual_impairments, 27.02.2024

[2] ILIAS, ILU: 4. Übersicht verschiedener assistiver Technologien, https://ilu.th-koeln.de/goto.php?target=cat_75998&client_id=thkilu, 27.02.2024

[3] Microsoft, “Barrierefreiheit der Tastaturnavigation”, https://learn.microsoft.com/de-de/windows/apps/design/accessibility/keyboard-accessibility, 29.02.2024

[4] Bundesamt für Justiz, “BITV 2.0 - Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz”, https://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html, 27.02.2024

[5] Bundesfachstelle Barrierefreiheit, “Bundesfachstelle Barrierefreiheit - Die neue BITV 2.0”, https://www.bundesfachstelle-barrierefreiheit.de/DE/Fachwissen/Informationstechnik/EU-Webseitenrichtlinie/BGG-und-BITV-2-0/Die-neue-BITV-2-0/die-neue-bitv-2-0_node.html#doc7b61b41e-e7fa-4086-9857-1ab9e2ac0b6ebodyText1, 27.02.2024

[6] ETSI, “ETSI EN 301 549 - V3.2.1 - Accessibility requirements for ICT products and services.”, https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf, 27.02.2024

[7] Kerstin Probiesch, 2013, "Barrierefreiheit im Web". RKW Kompetenzzentrum, https://www.rkw-kompetenzzentrum.de/publikationen/faktenblatt/barrierefreiheit-im-web/was-bedeutet-barrierefreiheit-im-web/#content, 27.02.2024

[8] pepper — Agentur für Webdesign, Print und Marketing, “Checkliste für barrierefreies Webdesign.”, https://barrierefreies.design/barrierefreiheit-interaktiv-testen/checkliste-fur-barrierefreies-webdesign#checkliste, 27.02.2024

[9] Tobias Kollmann und Alexander Michaelis, 2015, Responsive Webdesign. München, WiSt - Wirtschaftswissenschaftliches Studium. https://web.archive.org/web/20220223150023id_/https://www.beck-elibrary.de/10.15358/0340-1650-2015-7-50.pdf, 27.02.2024

[10] TailwindCSS, “Screen Readers”, https://tailwindcss.com/docs/screen-readers, 27.02.2024

[11] selfhtml, “Framework – SELFHTML-Wiki”, https://wiki.selfhtml.org/wiki/Framework, 27.02.2024

[12] DataScientest, “GitHub: Was steckt dahinter und wie lerne ich den Umgang damit?”, https://datascientest.com/de/github, 27.02.2024

[13] twigbit, “Vercel - Einfach erklärt”, https://www.twigbit.com/glossar/vercel, 27.02.2024

[14] TailwindCSS, “Responsive Design”, https://tailwindcss.com/docs/responsive-design, 27.02.2024

[15] Google,"Überblick", https://developers.google.com/web/tools/lighthouse/, 29.02.2024

9 Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel verfasst habe.

Louis Taube

Leipzig, 29.02.2024