

FACHHOCHSCHULE ERFURT

Accessibility WID Dokumentation

Author:
Tom KÄPPLER

Supervisor:
Rolf KRUSE

19. Februar 2023

Inhaltsverzeichnis

1	Einführung	1
2	Zeitplan	2
3	Recherche	3
3.1	Allgemeines	3
3.2	ARIA Authoring Practices Guide (APG)	3
3.2.1	Patterns	3
3.2.2	Landmark Regions	4
3.2.3	Accessible Names and Descriptions	4
3.2.4	Keyboard Interface	4
3.3	Tools zur Analyse	5
3.4	Storyook	5
3.5	Angular	6
3.5.1	Material UI	6
3.5.2	Google Codelab - Angular a11y	6
3.6	React	7
3.6.1	React-Aria	7
3.6.2	HeadlessUI	7
3.6.3	Radix	8
4	Accessibility Tools	9
4.1	Lighthouse	9
4.2	IBM Equal Access Accessibility Checker	9
4.3	WAVE	10
4.4	Axe	10
4.5	JAWS	10
4.6	NVDA	11
5	Turbomeet	12
5.1	Technologien	12
5.1.1	Next.js	12
5.1.2	TypeScript	13
5.1.3	tRPC	13
5.1.4	TailwindCSS	13
5.1.5	Prisma	13
5.1.6	NextAuth.js	13
5.1.7	Radix UI	14
5.1.8	CmdK	14
5.2	Analyse	16
5.2.1	Lighthouse	16
5.3	Accessibility Anpassungen	18
5.3.1	Fehlende Link Beschreibungen	18

5.3.2 Farbkontraste	18
6 Fazit	19
A Abbildungen	20
B Quellcode	23
Literatur	24

Abbildungsverzeichnis

5.1	Dialog Accessibility Dokumentation	14
5.2	Turbomeet Kombobox	15
5.3	Turbomeet Lighthouse Bericht Desktop	16
5.4	Turbomeet Lighthouse Bericht Chromium	17
A.1	GitHub Kombobox	20
A.2	Linear Kombobox	21
A.3	TailwindCSS Kombobox	21
A.4	Turbomeet Lighthouse Bericht Mobile	22

1 Einführung

In den letzten Jahren hat die Bedeutung von Web-Accessibility, also der Zugänglichkeit von Webinhalten für Menschen mit Behinderungen, stark zugenommen. Das Internet bietet eine Vielzahl von Informationen und Dienstleistungen, die für Menschen mit eingeschränkten körperlichen oder geistigen Fähigkeiten von großer Bedeutung sind. Es ist daher von entscheidender Bedeutung, dass Web-Entwickler ihre Websites barrierefrei gestalten, um sicherzustellen, dass alle Benutzer sie problemlos nutzen können.

Zum Glück gibt es heute eine Vielzahl von Tools, die Entwicklern dabei helfen können, barrierefreie Websites zu erstellen. Diese reichen von einfachen Validierungstools, die überprüfen, ob eine Website bestimmte Standards erfüllt, bis hin zu komplexen Werkzeugen, die sich speziell auf die Bedürfnisse von Menschen mit Behinderungen konzentrieren.

In dieser Ausarbeitung wird sich mit einigen der wichtigsten Tools und Techniken beschäftigen, die Entwicklern zur Verfügung stehen, um sicherzustellen, dass ihre Websites barrierefrei sind. Es wird sich unter anderem auch mit der Beispielhaften Umsetzung einer barrierefreien Website beschäftigen und anhand dieser verschiedene Techniken und Libraries vorgestellt.

2 Zeitplan

- Ende November: Recherche, welche Libraries eingesetzt werden und anfangen diese einzubinden
- Mitte Dezember: Recherche zu a11y und Prinzipien davon
- Ende Dezember: Website in der Grundform umgesetzt
- Ende Januar: Prototyp Hauptfunktionen fertigstellen
- Mitte Februar: Dokumentation von a11y Tools und Libraries
- Ende Februar: Website optimieren, dokumentieren und Tests mit den a11y Tools abschließen

3 Recherche

3.1 Allgemeines

Der Begriff *Accessibility* oder abgekürzt *a11y* bezeichnet eine Gestaltung der Umwelt, sodass sie auch von Menschen mit Beeinträchtigungen ohne zusätzliche Hilfen genutzt und wahrgenommen werden kann. Bundesfachstelle Barrierefreiheit, o. D.

Zu diesen Beeinträchtigungen zählen unter anderem:

- Auditiv
- Kognitiv
- Neurologisch
- Physisch
- Sprache

Die Umsetzung von *a11y* auf Webseiten kann außerdem bei temporären Einschränkungen helfen, wie z.B. bei einer Verletzung oder einer Krankheit, einem gebrochenen Arm oder einer verlorenen Brille. Ebenfalls kann es bei situationsabhängigen Limitierungen helfen, wie z.B. helles Tageslicht oder ein lauter Raum in dem ein Audio/Video nicht hörbar ist.

3.2 ARIA Authoring Practices Guide (APG)

Der ARIA Authoring Practices Guide kurz (APG) ist eine Sammlung von Beispielen und Best Practices für die Verwendung von ARIA. W3C Web Accessibility Initiative (WAI), o. D. Es werden verschiedene Themen wie z.B. Farbkontraste, Semantisches HTML, Fokus Kontroller und noch einiges mehr behandelt. Diese werden mit Beispielen und tiefgehenden Erklärungen dargestellt und an praktischen Beispielen erläutert.

3.2.1 Patterns

Der Guide stellt für quasi alle Elemente die in einer normalen Webseite vorkommen, ein Pattern, also eine Beschreibung welche Dinge beachtet werden müssen, bereit. Dies fängt bei einfachen HTML-Elementen wie z.B. Buttons oder Links an und geht bis hin zu komplexeren Elementen wie z.B. Accordions, Carousels oder Dialogen (Modale). Initiative (WAI), o. D.(c)

Für den Dialog wird z.B. ersteinmal grundlegend erklärt wie genau ein Dialog funktioniert und welche Elemente er beinhaltet. Danach wird auf die Tastatur Interaktionen eingegangen, welche implementiert werden sollten und auch welche HTML-Aria-Eigenschaften dafür genutzt werden können, um die best mögliche *a11y* zu gewährleisten. In dem direkt bereitgestellten Beispiel wird dann die Implementierung dieser Eigenschaften gezeigt und bis in kleinste Detail erläutert was jede einzelne davon bewirkt und warum diese genutzt wird.

3.2.2 Landmark Regions

Landmark Regions sind Bereiche einer Webseite, welche für Screenreader und andere Hilfsmittel wie z.B. Sprungmarken wichtig sind. Initiative (WAI), o. D.(b) Diese werden in der Dokumentation mit Beispielen und Erklärungen vorgestellt und erklärt.

Zuerst wird auf die HTML Struktur Elemente eingegangen. Dazu gehören z.B. `<header>`, `<main>`, `<footer>` und `<nav>`. Diese werden mit Beispielen und Erklärungen vorgestellt und erklärt. Danach wird auf die ARIA Eigenschaften eingegangen, welche für die Landmark Regions genutzt werden können. Diese sind z.B. `role="banner"`, `role="main"`, `role="contentinfo"` und `role="navigation"`. Mit diesen können bestimmte Bereiche oder Informationen für Screenreader und andere Hilfsmittel markiert werden.

3.2.3 Accessible Names and Descriptions

Accessible Names and Descriptions sind die Namen und Beschreibungen von Elementen, welche z.B. für Screenreader wichtig sind. Initiative (WAI), o. D.(d) In der Dokumentation wird ausführlichst darauf eingegangen wie genau man diese Namen und Beschreibungen aufbauen und vergeben sollte und welche Möglichkeiten es dafür gibt. Dies wird auch hier wieder durch eine Vielzahl an Beispielen zu den verschiedenen Möglichkeiten unterstützt.

Außerdem werden Grundregeln für die Namensgebung vorgestellt. Diese sind z.B.:

- Beachtung von Warnungen und ausführliches Testen
- Sichtbaren Text bevorzugen
- Native Techniken bevorzugen (z.B. `<label>`)
- Browser Fallbacks vermeiden
- Kurze und nützliche Namen und Beschreibungen nutzen

3.2.4 Keyboard Interface

Damit eine Seite komplett und vorallem auch barrierefrei nur mit Hilfe einer Tastatur bedienbar ist, müssen einige Dinge beachtet und explizit implementiert werden. Dies muss passieren, da die meisten Browser, im Gegensatz zu nativen Form-Elementen, keinen direkten Tastatur Support für das Steuern von GUI Komponenten bieten. Initiative (WAI), o. D.(a) Dies muss z.B. für Menüs, Grids, Toolbars und Dialoge von den Entwicklern übernommen werden.

Die wichtigste und fundamentalste Tastaturnavigation ist die Tabulator Navigation. Diese wird per `Tab` und `Shift + Tab` gesteuert und ermöglicht es den Fokus von einem UI Element auf das nächste zu bewegen. Zusätzlich können dann die Pfeiltasten genutzt werden um den Fokus innerhalb einer Komponente zu bewegen. Initiative (WAI), o. D.(a)

Anschließend wird in der Dokumentation noch erweiternd auf die folgenden Themen eingegangen:

- Erkennbarer und vorhersehbarer Tastaturfokus
- Fokus vs. Selektion und die Wahrnehmung eines doppelten Fokuses

- Wann sollte die Auswahl dem Fokus automatisch folgen
- Tastaturnavigation zwischen Komponenten (Die Tab-Sequence)
- Tastaturnavigation innerhalb von Komponenten
- Fokussierbarkeit von deaktivierten Steuerelementen
- Tastenzuweisungskonventionen für allgemeine Funktionen
- Tastaturkurzbefehle und wie diese vergeben werden sollten

3.3 Tools zur Analyse

Die folgenden Tools wurden in der Recherche gefunden und können zur Analyse von Webseiten verwendet werden. Eine ausführliche Beschreibung der einzelnen Tools findet sich in den jeweiligen Kapiteln.

- Lighthouse (SEO, Performance, Accessibility Analyse)
- IBM Equal Access Accessibility Checker
- Deque Axe (Accessibility Analyse)
- Wave (Accessibility Analyse)
- JAWS (Screenreader)
- NVDA (Screenreader)

3.4 Storybook

Storybook ist eine Open-Source-JavaScript-Bibliothek, die es Entwicklern ermöglicht, Komponenten isoliert und unabhängig voneinander zu entwickeln, zu testen und zu dokumentieren. Storybook, o. D. Mit Storybook können Entwickler eine interaktive und benutzerfreundliche Benutzeroberfläche erstellen, um ihre Komponenten in verschiedenen Zuständen zu präsentieren, ohne dabei auf eine vollständige Anwendung angewiesen zu sein.

Storybook ist besonders nützlich für Projekte, die auf Komponentenarchitekturen und Design-Systemen basieren, da es Entwicklern hilft, Komponenten unabhängig voneinander zu testen und sicherzustellen, dass sie ordnungsgemäß funktionieren, bevor sie in die Anwendung integriert werden.

Die Bibliothek bietet außerdem verschiedene Add-Ons und Tools, die es Entwicklern ermöglichen, die Zugänglichkeit ihrer Komponenten zu überprüfen und sicherzustellen, dass sie die Anforderungen der Web Content Accessibility Guidelines (WCAG) erfüllen.

Ein Beispiel für ein Accessibility-Add-On in Storybook ist das a11y Add-On, das eine Integration mit der Accessibility Testing-Engine *axe-core* bietet. Mit diesem Add-On können Entwickler ihre Komponenten auf Barrierefreiheitsprobleme testen, wie z.B. fehlende Alternativtexte für Bilder oder mangelnde Kontrastverhältnisse für Texte und Hintergründe.

Ein weiteres Beispiel ist das *Storybook Addon Docs* Add-On, das es Entwicklern ermöglicht, Dokumentationen für ihre Komponenten zu erstellen, einschließlich der Zugänglichkeitsanforderungen und -hinweise für jede Komponente.

Zusammenfassend bietet Storybook verschiedene Möglichkeiten, um die Zugänglichkeit von Komponenten zu testen und sicherzustellen, dass sie für alle Benutzer, einschließlich derjenigen mit Behinderungen, zugänglich sind.

3.5 Angular

3.5.1 Material UI

Angular Material UI (MUI) ist eine Sammlung von UI Komponenten, die auf Angular basieren. Angular Components Team, o. D. Diese sind teilweise bereits mit a11y ausgestattet, wie z.B. die [Selects](#) oder [Checkboxes](#). Neben den automatisch eingestellten ARIA-Tags gibt die Dokumentation für die meisten Komponenten weitere Tags an, welche vom Entwickler gesetzt werden sollten. Siehe z.B. [Dialog](#).

MUI bietet außerdem ein *Component Dev Kit (CDK)* an. Dieses bietet verschiedene Hilfsfunktionen, welche für die Entwicklung von eigenen a11y optimierten Komponenten verwendet werden können. Angular Components Team, o. D.

Die Vorteile von Material UI sind:

- **Zugänglichkeit:** Material UI legt Wert auf Accessibility-Features und stellt sicher, dass alle Komponenten vollständig zugänglich sind, um eine barrierefreie Nutzung der Webanwendung zu gewährleisten.
- **Leicht anpassbar:** Material UI bietet eine Vielzahl von Möglichkeiten, um Komponenten an das Design der Anwendung anzupassen. Dabei können Themes, Farben und andere Design-Elemente einfach angepasst werden.
- **Große Community:** Material UI hat eine große Community von Entwicklern, die dazu beitragen, die Bibliothek ständig zu verbessern und zu erweitern.

Ein Nachteil ist, dass die Bibliothek möglicherweise nicht so gut anpassbar ist wie andere UI-Bibliotheken, insbesondere wenn es um das Layout geht. Dies kann zu Einschränkungen bei der Gestaltung von Webanwendungen führen. Bestimmte Stylings müssen über teilweise umständliche CSS-Regeln angepasst werden und überschrieben werden.

Insgesamt ist Material UI eine robuste Bibliothek mit vielen nützlichen Funktionen und einem großen Unterstützungssystem. Es ist eine gute Wahl für Entwickler, die eine umfassende UI-Lösung für Angular suchen, besonders wenn Zugänglichkeit und Benutzerfreundlichkeit eine hohe Priorität haben.

3.5.2 Google Codelab - Angular a11y

Google Codelab sind Lernkurse zur eigenständigen Bearbeitung. In diesem Codelab wird ein Beispielprojekt erstellt, welches die Grundlagen von a11y in Angular beinhaltet. Dabei werden Themen wie z.B. Farbkontraste, Semantisches HTML, Fokus Kontroller und verschiedene andere Dinge behandelt.

3.6 React

3.6.1 React-Aria

React-Aria ist eine React-Bibliothek, die von Adobe entwickelt wurde und es Entwicklern ermöglicht, barrierefreie Benutzeroberflächen in React zu erstellen. Die Bibliothek bietet eine Sammlung von React-Komponenten und Hooks, die speziell für die Unterstützung von Accessibility-Funktionen entwickelt wurden.

Die Vorteile von React-Aria sind:

- **Accessibility-Features:** React-Aria bietet Entwicklern eine einfache und konsistente Möglichkeit, Accessibility-Features wie Tastaturzugänglichkeit und Screen-Reader-Kompatibilität in ihre React-Komponenten zu integrieren.
- **Einfachheit:** React-Aria bietet eine einfache API und eine umfassende Dokumentation, die es Entwicklern erleichtert, barrierefreie Komponenten zu erstellen.
- **Flexibilität:** React-Aria bietet eine Vielzahl von Konfigurationsoptionen, um sicherzustellen, dass die Accessibility-Funktionen den Bedürfnissen des Benutzers entsprechen. Entwickler können z.B. die Reihenfolge der Fokusnavigation anpassen oder die Funktionsweise von ARIA-Attributen konfigurieren.
- **Aktive Community:** React-Aria hat eine aktive Entwickler-Community, die regelmäßig Updates und Verbesserungen zur Verfügung stellt und Fragen beantwortet.

Ein Nachteil von React-Aria ist, dass die Bibliothek möglicherweise nicht für alle Anwendungsfälle geeignet ist. Da die Accessibility-Funktionen von React-Aria zusätzliche Markup- und Code-Elemente erfordern, kann dies zu einer erhöhten Komplexität von React-Komponenten führen.

Ein weiterer Nachteil ist, dass Entwickler, die nicht vertraut sind mit den Accessibility-Features von React-Aria, eine zusätzliche Lernkurve haben können, um die Bibliothek effektiv einzusetzen.

3.6.2 HeadlessUI

HeadlessUI ist eine React-Bibliothek, die von Tailwind Labs entwickelt wurde und eine Sammlung von vollständig zugänglichen und wiederverwendbaren Headless-Komponenten bietet. HeadlessUI ist darauf ausgelegt, Entwicklern die Erstellung von benutzerdefinierten, zugänglichen Benutzeroberflächen zu erleichtern.

Die Vorteile von HeadlessUI sind:

- **Accessibility-Features:** HeadlessUI-Komponenten sind vollständig zugänglich und unterstützen Tastaturzugänglichkeit, Screenreader-Unterstützung und andere Accessibility-Features, die die Barrierefreiheit von Webanwendungen verbessern.
- **Anpassbarkeit:** HeadlessUI-Komponenten sind *headless*, d.h. sie haben keine vorgegebene visuelle Darstellung und können einfach an das Design der Anwendung angepasst werden.
- **Einfache Integration:** HeadlessUI kann nahtlos in bestehende React-Projekte integriert werden, da es nur minimale Abhängigkeiten hat und einfach zu verwenden ist.

- Gute Dokumentation: HeadlessUI bietet eine ausführliche Dokumentation und viele Beispiele, die Entwicklern helfen, die Bibliothek schnell und effektiv einzusetzen.

Ein Nachteil von HeadlessUI ist, dass die Bibliothek, ähnlich wie React-ARIA, möglicherweise nicht für alle Anwendungsfälle geeignet ist. Da die Komponenten so anpassbar sind, erfordert ihre Verwendung möglicherweise mehr Code als die Verwendung von vorgefertigten UI-Komponenten.

Ein weiterer Nachteil ist, dass die Bibliothek noch relativ neu ist und noch nicht so gut etabliert und weit verbreitet wie einige andere UI Bibliotheken. Dies kann zu einer begrenzten Unterstützung von Drittanbietern und einer kleineren Entwickler-Community führen.

3.6.3 Radix

Radix ist eine React-Bibliothek, die von Modulz entwickelt wurde und eine Sammlung von wiederverwendbaren Komponenten enthält, die auf einer *primitive-first* Designphilosophie basieren. Radix stellt Komponenten zur Verfügung, die den Primitiven von HTML ähneln, um den Zugriff auf wichtige Accessibility-Features sicherzustellen.

Die Vorteile von Radix sind:

- Accessibility-Features: Radix-Komponenten sind vollständig zugänglich und unterstützen Tastaturzugänglichkeit, Screenreader-Unterstützung und andere Accessibility-Features, die die Barrierefreiheit von Webanwendungen verbessern.
- Einfache Integration: Radix kann nahtlos in bestehende React-Projekte integriert werden, da es nur minimale Abhängigkeiten hat und einfach zu verwenden ist. Des Weiteren ist die Bibliothek Modular aufgebaut und installierbar, sodass nur die Komponenten installiert werden, die für die Anwendung überhaupt benötigt werden.
- Anpassbarkeit: Radix-Komponenten sind so konzipiert, dass sie einfach an das Design der Anwendung angepasst werden können. Dabei wird das gleiche Prinzip wie bei HeadlessUI verwendet und die Komponenten sind *headless*.
- Gute Dokumentation: Radix bietet eine ausführliche Dokumentation und viele Beispiele, die Entwicklern helfen, die Bibliothek schnell und effektiv einzusetzen.

Wie auch bei HeadlessUI ist ein möglicher Nachteil von Radix, dass es noch relativ neu ist und möglicherweise nicht so gut etabliert und weit verbreitet wie einige andere UI-Bibliotheken.

Insgesamt bietet Radix jedoch eine Vielzahl von Vorteilen und ist eine nützliche Bibliothek für Entwickler, die benutzerdefinierte, zugängliche Benutzeroberflächen in React erstellen möchten. Die *primitive-first* Designphilosophie kann dabei helfen, die Wartbarkeit und Skalierbarkeit von Code zu verbessern, während die Zugänglichkeit von Webanwendungen gewährleistet bleibt.

4 Accessibility Tools

4.1 Lighthouse

Lighthouse ist ein Open-Source-Tool von Google, das zur Verbesserung der Qualität von Webanwendungen verwendet wird. Es bietet eine umfassende Prüfung der Leistung, Zugänglichkeit, Best Practices, SEO und PWA-Funktionalitäten. Das Tool ist als Chrome-Erweiterung oder über die Kommandozeile verfügbar.

Die wichtigsten Features von Lighthouse sind:

- **Performance-Prüfung:** Lighthouse kann eine detaillierte Analyse der Ladezeit und der allgemeinen Leistung der Webanwendung durchführen. Es zeigt, welche Teile der Anwendung optimiert werden können und bietet Tipps zur Verbesserung der Geschwindigkeit.
- **Accessibility-Prüfung:** Lighthouse kann die Zugänglichkeit der Webanwendung prüfen und zeigt, welche Elemente die Barrierefreiheit beeinträchtigen. Es bietet auch Tipps zur Verbesserung der Zugänglichkeit.
- **Best Practices-Prüfung:** Lighthouse prüft, ob die Webanwendung den gängigen Best Practices entspricht und bietet Empfehlungen zur Verbesserung der Code-Qualität.
- **SEO-Prüfung:** Lighthouse kann auch die Suchmaschinenoptimierung (SEO) der Webanwendung prüfen und gibt Empfehlungen zur Optimierung von Meta Tags, Keywords und anderen wichtigen SEO-Faktoren.

Ein wichtiger Faktor für die Performance Tests ist, dass wenn möglich keine Erweiterungen im Browser installiert sein sollten, da diese die Ergebnisse verfälschen können.

Während meiner Tests ist dies im Besonderen bei der Erweiterung *React Developer Tools* aufgefallen. Diese Erweiterung ist sehr nützlich, da sie es ermöglicht, die React Komponenten einer Seite zu untersuchen. Allerdings kann sie die Ergebnisse der Performance Tests verfälschen, da sie die Seite stark verlangsamt. Daher sollte diese Erweiterung vor dem Start der Performance Tests deaktiviert werden. Am besten ist es einen separaten Browser zu verwenden, in dem keine Erweiterungen installiert sind oder nur die die auch ein normaler Nutzer installiert hat. (z.B. PrivacyBadger, uBlock Origin, etc.)

4.2 IBM Equal Access Accessibility Checker

IBM Equal Access Accessibility Checker ist ein browserbasiertes Tool, das zur Überprüfung der Barrierefreiheit von Webseiten genutzt werden kann. Das Tool ist kostenlos und bietet eine Vielzahl von Funktionen, die die Barrierefreiheit von Webinhalten verbessern können. Einige der wichtigsten Funktionen von IBM Equal Access Accessibility Checker sind:

- **Manuelle Überprüfung:** Es ist möglich, manuelle Überprüfungen für spezifische Elemente durchzuführen, um detailliertere Informationen über Barrierefreiheitsprobleme zu erhalten.
- **Berichterstellung:** Das Tool bietet detaillierte Berichte zur Barrierefreiheit, die auf WCAG 2.1 basieren und dabei helfen können, Barrieren für Menschen mit Behinderungen zu identifizieren und zu beseitigen.
- **Automatisierung:** Das Tool kann auch als Teil eines automatisierten Testprozesses verwendet werden, um die Barrierefreiheit von Webinhalten zu überprüfen. Dies kann beispielsweise in CI/CD-Pipelines integriert werden, um sicherzustellen, dass die Barrierefreiheit von Webinhalten bei jeder Änderung überprüft wird. Dafür können die direkt bereitgestellten NPM Pakete für Karma und Cypress verwendet werden.
- **Kontrastprüfung:** Das Tool kann den Kontrast von Text und Hintergrund automatisch prüfen und anzeigen.

4.3 WAVE

WAVE (Web Accessibility Evaluation Tool) wird als browserbasiertes Tool und API bzw. Test-Engine bereitgestellt. Es ist ein Open-Source-Tool, das von der WebAIM-Organisation entwickelt wird. Das Feature Set ist hierbei sehr ähnlich zu IBM Equal Access Accessibility Checker.

4.4 Axe

Deque Axe wird, wie auch die Vorgänger, als Browser-Erweiterung und als Bibliothek zur Integration in Test-Engines bereitgestellt. Außerdem bietet es eine Developer Tools Variante mit der Entwickler bereits während der Entwicklung auf Barrierefreiheitsprobleme hingewiesen werden. Die Erweiterung ist eingeschränkt kostenlos, um aber den vollen Funktionsumfang zu erhalten ist eine Pro-Version (45€) notwendig. Dazu zählen unter anderem auch die CLI sowie CI-CD Integration.

4.5 JAWS

JAWS ist ein Screenreader, der von der Firma Freedom Scientific entwickelt wird. Er ist für Windows verfügbar und kann kostenlos "getestet" werden. (40min. Session danach Neustart erforderlich) Er ist der am weitesten verbreitete Screenreader und wird von vielen Behörden und Unternehmen verwendet. Er ist in der Lage, die meisten Webseiten zu lesen und bietet eine Vielzahl von Funktionen, die die Barrierefreiheit von Webinhalten verbessern können. Der größte Nachteil von JAWS ist, dass die Lizenzen sehr teuer sind. Die Preise für eine dauerhafte einzelne Lizenz beginnen bei 1.000 US-Dollar bzw. mit einem Jahresabonnement bei 90 US-Dollar.

Neben Webseiten kann JAWS auch mit Microsoft Office, PDFs, E-Mails und anderen Anwendungen verwendet werden. Sodass auch andere Programme barrierefrei genutzt werden können. Für die Ausgabe steht sowohl eine Sprachausgabe als auch eine Brailleausgabe zur Verfügung. Texte können hierbei sogar in Kurzfassung ausgegeben werden.

4.6 NVDA

NVDA (NonVisual Desktop Access) ist ein Screenreader, der von der Firma NV Access entwickelt wird. Er ist Opensource auf [Github](#) verfügbar und kann kostenlos heruntergeladen werden. Dies auch einer der Gründe, warum er so beliebt ist. Denn die meisten Screenreader sind sehr teuer. Er ist in der Lage, die meisten Webseiten zu lesen und bietet eine Vielzahl von Funktionen. Durch die Open-Source Natur ist es möglich, die Software zu verbessern und neue Funktionen hinzuzufügen. Dies wird auch sehr aktiv von der Community vorangetrieben und es gibt eine Vielzahl von Erweiterungen, die die Funktionalität von NVDA erweitern.

5 Turbomeet

Turbomeet ist der Prototyp an dem verschiedene Libraries und Tools getestet werden. Es ist eine Webseite auf der sich Nutzer anmelden können und Meetings mit verschiedenen Zeitslots erstellen können. Für diese Zeitslots kann dann von anderen Nutzern abgestimmt werden. Wenn ein Meeting abgeschlossen wird bzw. die Deadline erreicht wird, werden die Zeitslots mit den meisten Teilnahmen ausgewählt.

5.1 Technologien

Der ausgewählte Technologie Stack ist der T3-Stack:

- Next.js
- TypeScript
- tRPC
- TailwindCSS
- Prisma
- NextAuth.js

Neben diesen Technologien werden noch weitere Libraries verwendet, die in den folgenden Abschnitten genauer beschrieben werden.

5.1.1 Next.js

Next.js ist ein serverseitiges JavaScript-Framework, das auf React aufbaut. Es ist dafür ausgelegt, schnelle und skalierbare Anwendungen zu entwickeln. Zu den Funktionen von Next.js gehören:

- Serverseitiges Rendern (SSR): Next.js kann die HTML-Ausgabe auf dem Server generieren, was die Ladezeit der Seite verbessert und die Suchmaschinenoptimierung (SEO) erleichtert.
- Automatisches Code-Splitting: Next.js teilt automatisch den Code in kleine, optimierte Teile auf, um die Ladezeit der Seite zu reduzieren.
- Hot-Module-Replacement: Next.js aktualisiert automatisch den Code, während Sie arbeiten, um einen schnellen Entwicklungsprozess zu ermöglichen.
- Static Site Generation (SSG): Next.js kann auch statische Seiten generieren, um schnelle und sichere Websites zu erstellen, die auf einem CDN (Content Delivery Network) gehostet werden können.

5.1.2 TypeScript

TypeScript ist eine freie und quelloffene Programmiersprache, die auf JavaScript basiert. Sie ist eine statisch typisierte Sprache, die es Entwicklern ermöglicht, die Typen von Variablen und Funktionen zu definieren. Dadurch können Fehler frühzeitig erkannt und behoben werden. Außerdem können Entwickler mit TypeScript auch die neuesten Funktionen von JavaScript verwenden, ohne auf die Kompatibilität mit älteren Browsern achten zu müssen.

5.1.3 tRPC

tRPC ist ein Open-Source-Framework, das es Entwicklern ermöglicht, API Endpunkte einfach zu definieren und zu erstellen. Mit dem Motto *“Move fast and break nothing. End-to-end typesafe APIs made easy.”* (<https://github.com/trpc/trpc>) werden die Vorteile von tRPC sehr gut beschrieben. Durch die Nutzung von TypeScript im Backend sowie im Frontend werden Breaking Changes vermieden und die Entwicklung wird beschleunigt. Wenn bestimmte Felder z.B. in der API geändert/entfernt werden, und auf diese im Frontend zugegriffen wird, werden diese direkt mit spezifischen Fehlern annotiert.

Um eine E2E Typensicherheit bei normalen REST und GraphQL APIs zu erhalten muss eine Vielzahl von Tools und Konfiguration vorgenommen werden. Dies ist meist sehr aufwändig und komplex.

5.1.4 TailwindCSS

TailwindCSS ist ein Utility-First CSS-Framework. Das bedeutet, dass es keine vorgefertigten Komponenten wie z.B. Buttons oder Formulare gibt. Stattdessen werden die Komponenten aus einzelnen Utility-Klassen zusammengesetzt. Diese Utility-Klassen sind meist sehr spezifisch und beschreiben nur eine Eigenschaft. So kann z.B. ein Button mit der Klasse `.bg-blue-500` eine blaue Hintergrundfarbe erhalten. Die Klasse `.bg-blue-500` beschreibt nur die Hintergrundfarbe und nicht die Größe oder den Abstand zum nächsten Element. Diese Utility-Klassen können dann beliebig kombiniert werden, um die Komponente zu gestalten.

5.1.5 Prisma

Prisma ist ein Open-Source-ORM (Object-Relational Mapping) für Node.js. Es ermöglicht es Entwicklern, Datenbanken zu modellieren und mit der Datenbank zu interagieren. Prisma bietet eine einfache und intuitive API, die es Entwicklern ermöglicht, Datenbankabfragen zu schreiben, ohne sich um die Syntax der Datenbank kümmern zu müssen. Prisma unterstützt eine Vielzahl von Datenbanken, darunter MySQL, PostgreSQL, SQLite und MongoDB.

Prisma generiert automatisch eine TypeScript Typen Definition für die Datenbank. Mithilfe dieser Typen Definition können Entwickler sicherstellen, dass die Datenbank und die API Endpunkte immer synchron sind.

5.1.6 NextAuth.js

NextAuth.js ist ein Open-Source-Framework, das es Entwicklern ermöglicht, Authentifizierung und Autorisierung in Next.js Anwendungen zu implementieren. Es bietet eine einfache und intuitive API, die es Entwicklern ermöglicht, verschiedene

Authentifizierungsmethoden zu implementieren, wie z.B. OAuth, E-Mail-Authentifizierung, Passwort-Authentifizierung, Magic-Links, etc.

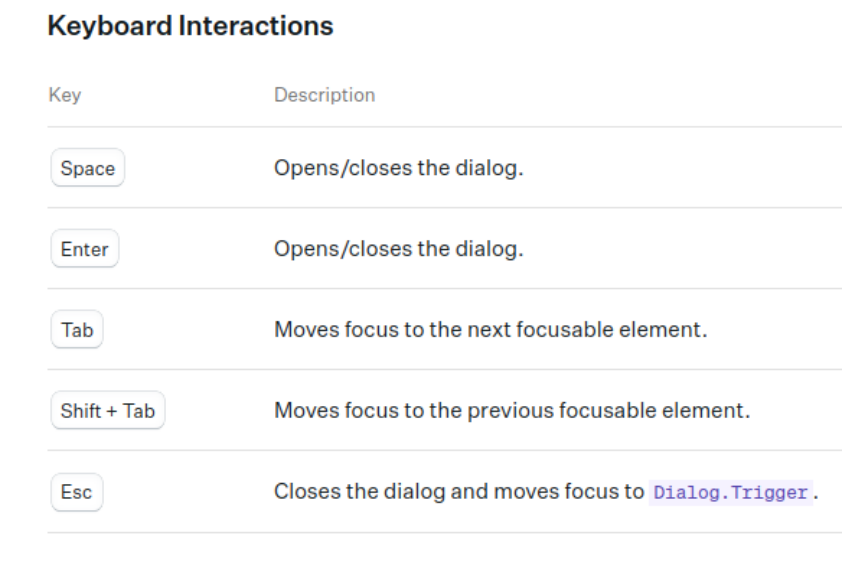
5.1.7 Radix UI

Wie bereits in dem Recherche Kapitel beschrieben ist Radix UI ein *primitive-first* UI-Framework. Es werden z.B. Komponenten wie Buttons, Inputs und Dialoge bereitgestellt. Diese sind immer bereits mit der Funktionalität und Accessibility Features ausgestattet. Außerdem sind die Komponenten un-styled, d.h. sie haben keine vorgefertigte CSS-Klasse. Stattdessen können die Komponenten mit den Utility-Klassen von TailwindCSS gestyled werden.

Alle Komponenten von Radix UI folgen dem WAI-ARIA Standard und implementieren die Accessibility Patterns der APG wie in im Kapitel 3.2.1 beschrieben. Dadurch werden den Entwicklern bereits viele Accessibility Features zur Verfügung gestellt und es muss nicht jedes Mal alles neu/manuell implementiert werden.

Bestimmte Felder wie z.B. die label eines Inputs können mit einem aria-label versehen werden. Dadurch wird der Input für Screenreader und andere Assistive Technologien lesbar. Diese Werte kann Radix UI nicht automatisch generieren, da es nicht weiß, was in den Feldern steht. Deshalb müssen diese Werte manuell angegeben werden. Aber auch hier hilft Radix UI, indem es in der Dokumentation explizit darauf hinweist, dass diese Werte angegeben werden müssen.

In der Dokumentation zu den einzelnen Komponenten werden außerdem die Accessibility Features beschrieben. So wird z.B. bei einem Dialog beschrieben, dass der Dialog mit der Tastatur bedienbar ist und wie dies funktioniert. Außerdem wird beschrieben, dass der Dialog mit einem Screenreader gelesen werden kann. In Abbildung 5.1 ist ein Screenshot der Dokumentation zu einem Dialog zu sehen.



Keyboard Interactions	
Key	Description
Space	Opens/closes the dialog.
Enter	Opens/closes the dialog.
Tab	Moves focus to the next focusable element.
Shift + Tab	Moves focus to the previous focusable element.
Esc	Closes the dialog and moves focus to <code>Dialog.Trigger</code> .

ABBILDUNG 5.1: Radix UI Dialog Accessibility Dokumentation

5.1.8 CmdK

CmdK ist eine Open-Source-Library, die eine Komponenten basierte Accessibility optimierte *Combobox* bereitstellt. Eine *Combobox* ist eine Kombination aus einem *Input* und einer *Dropdown* Liste. Diese wird inzwischen häufig auf Webseiten/Tools

wie GitHub, Linear und TailwindCSS (siehe Abbildungen A.1, A.2, A.3) genutzt um zum einen eine Suchfunktion zu implementieren oder um eine Liste von Aktionen / Optionen anzuzeigen. Die Tastenkombinationen für solche Funktionen sind von System zu System unterschiedlich. Zum Beispiel stellt macOS unter *Cmd + Leertaste* eine Globale Such/Kommandofunktion zur Verfügung auf Webseiten wird meist eher *Cmd + K* (macOs) oder *Ctrl + K* (Windows) verwendet. Eine weitere Variante ist das einfache drücken von *"/*". Dies ist aber z.B. auf dem deutschen QWERTZ-Tastatur Layout nicht direkt möglich.

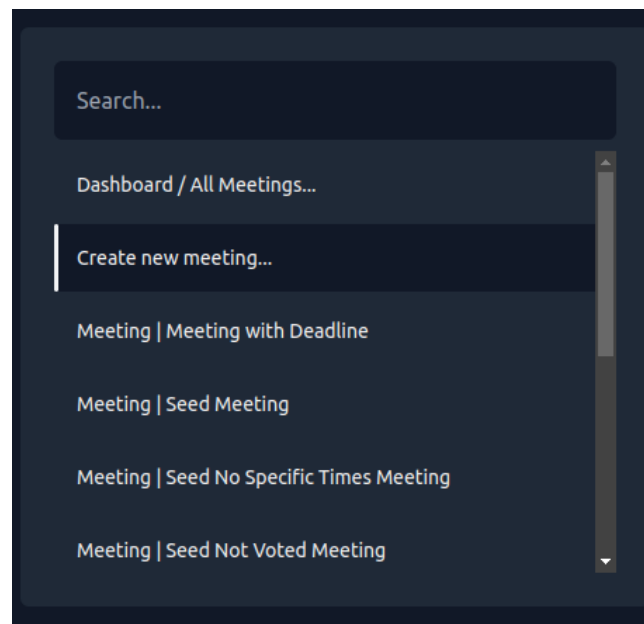


ABBILDUNG 5.2: Turbomeet Kombobox für Seitennavigation/Suche

Die Implementierung in Turbomeet (siehe Abbildung 5.2) lässt quasi alle oben genannten Kombinationen zu um Systemunabhängig für verschiedene Nutzer einfach bedienbar zu sein.

Zur Auswahl stehen unter anderem die Hauptseite wie das Dashboard, Nutzerprofil oder die Datenschutzerklärung, dazu werden außerdem Direkt-Verlinkungen zu den eigenen Meetings mit angezeigt. Des Weiteren werden Funktionen wie das erstellen von neuen Meetings oder das wechseln des Themes direkt ansprechbar gemacht.

Diese Funktionalität ist auch für die Accessibility ein sehr interessantes Feature. So kann z.B. ein Screenreader Nutzer die Kombobox öffnen und die verfügbaren Optionen durchlesen. Wenn ein Nutzer eine Seite bereits kennt, kann er durch die Suche direkt zu dieser springen ohne erst durch verschlungene Navigationen und Submenüs zu navigieren. Hierbei sind aber auch wieder die Entwickler gefragt um die Suchbegriffe so zu wählen, dass sie für den Nutzer intuitiv sind. Dies kann z.B. auch durch eine duplizierte Navigation unter verschiedenen Begriffen erreicht werden. Dies wäre in einer normalen Navigation nicht möglich, da die Navigation dann zu unübersichtlich wird.

Ein großer Vorteil von CmdK ist, dass es für die Dialog/Popover Implementierung Radix-UI als Basis nutzt. Dies ist für Turbomeet von Vorteil, da Radix sowieso

genutzt und geladen wird und somit keine zusätzlichen Abhängigkeiten hinzugefügt werden müssen und darauf vertraut werden kann, dass die Accessibility Features von Radix auch in CmdK implementiert sind.

5.2 Analyse

5.2.1 Lighthouse

Da Lighthouse eines der am einfachsten und schnellsten Tools für eine Analyse auch während der Entwicklung ist habe ich mich vorrangig darauf gestützt. Die Ergebnisse waren hier auch immer recht gut und ich konnte so schnell die kleinen Dinge verbessern, die noch nicht optimal waren. Die Ergebnisse von einem Zwischenstand sind in der folgenden Abbildung 5.3 zu sehen. Die kompletten Berichte sind als PDF verfügbar, aber nicht im Anhang eingebunden, da diese über 100 Seiten umfassen. Daher sind sie über den folgenden Link abrufbar: [GitHub.com/TKSpectro/web-interaction-documentation](https://github.com/TKSpectro/web-interaction-documentation).

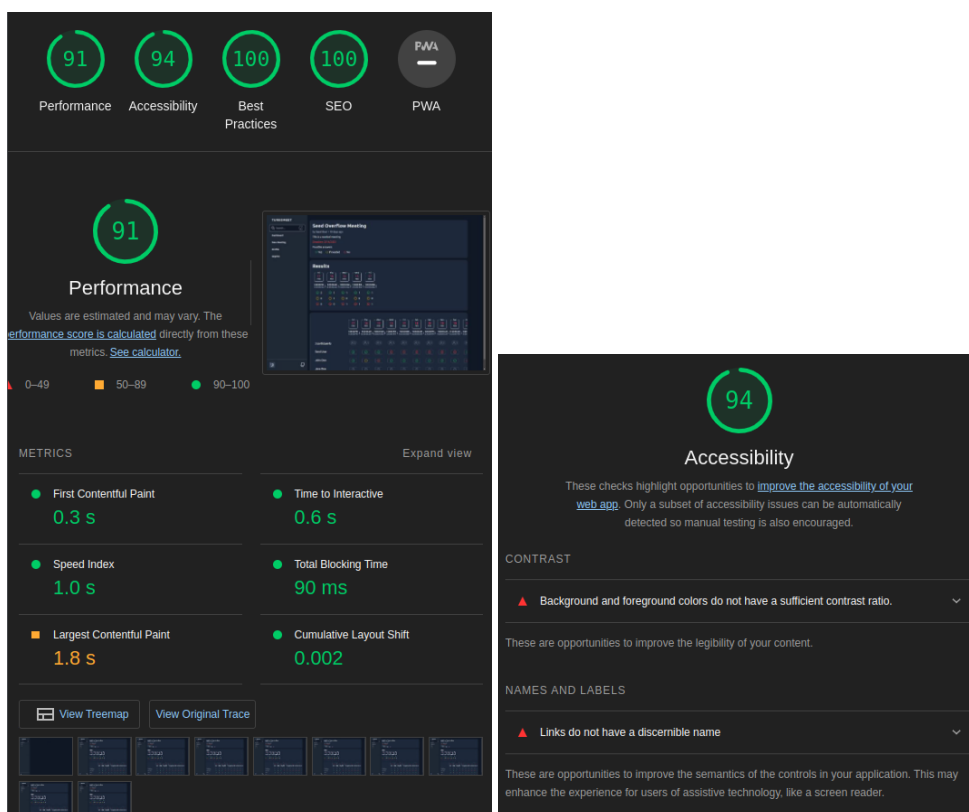


ABBILDUNG 5.3: Lighthouse Bericht für Desktop

Im Bereich Accessibility wurde darauf hingewiesen, dass die Hinter- /Vordergrund (Text) Farben zu wenig Kontrast haben und dass bei Links die Beschreibungen fehlen. Als ich dann aber die Analyse noch einmal für das Mobile-Preset durchgeführt habe (siehe Abbildung A.4) sank die Performance von 91% auf gerade einmal 52%, die TTI (Time to Interactive) von 0.6s auf 6.7s und die LCP (Largest Contentful Paint) von 1.8s auf 9.2s.

Glücklicherweise gab Lighthouse aber auch direkt den richtigen Tipp, welcher bereits im Kapitel 4.1 beschrieben wurde. Durch das Testen in meinem normalen

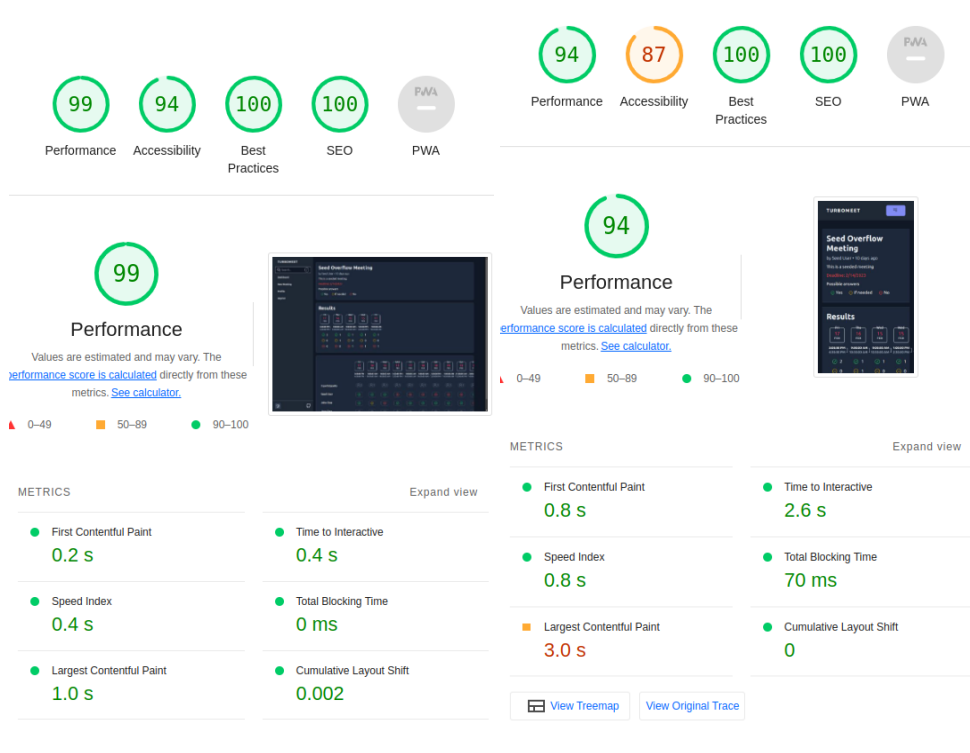


ABBILDUNG 5.4: Lighthouse Bericht Chromium (links: Desktop, rechts: Mobile)

Browser waren verschiedene Erweiterungen aktiv, welche sich in den Render Prozess von React einhängen um so während der Entwicklung States und ähnliche Dinge auszulesen. Aus diesem Grund schlägt Lighthouse hier vor, die Erweiterungen zu deaktivieren oder auf einen Inkognito Modus umzusteigen. Mit einem erneuten Test in einem cleanen Chromium Browser wurde nun wieder eine sehr gute Performance erreicht, jedoch sank die Accessibility trotzdem noch auf 87%. Dies liegt daran, dass Lighthouse in der Mobilen Variante alle Links als Buttons interpretiert und somit die Beschreibungen für die Button (Links) verlangt. Anscheinend weißt Lighthouse diesem dann auch mehr Gewicht zu als in der Desktop Variante. Dies ist aber auch kein Problem, da die Beschreibungen für die Links sowieso noch fehlen und somit schnell nachgeholt werden können.

Daraus lässt sich mitnehmen, immer beide Varianten zu testen, da die Ergebnisse sich unterscheiden können und auch verschiedene Optimierungen für Desktop und Mobile getroffen werden müssen.

Die Punkte SEO und Best Practices wurden hierbei in beiden Varianten mit 100% bewertet und dies ohne das extra Aufwände dafür getätigt werden mussten. Dies liegt daran, dass Turbomeet auf einen Technologie Stack aufbaut, welcher zum einen Next.js und das SSR Feature nutzt, wodurch die Seiten bereits vor dem Laden der Seite, auf dem Server, gerendert werden und somit auch schon für Suchmaschinen verfügbar sind (Crawlable für Google Bots). Zum anderen wird TypeScript genutzt, durch welches die neusten Features von JavaScript auch in älteren Browser polyfilled und deprecated APIs vermieden werden und somit eine sehr gute Performance erreicht wird.

5.3 Accessibility Anpassungen

Durch die Analyse mit Hilfe von Lighthouse wurden wie im letzten Kapitel beschrieben kleinere Accessibility Probleme festgestellt. Um diese zu beheben wurden die folgende Anpassungen durchgeführt.

5.3.1 Fehlende Link Beschreibungen

Dieser Fehler ist sehr einfach zu beheben, da hierzu einfach nur das *aria-label* Tag auf den entsprechenden HTML-Elementen gesetzt werden muss. Wie aber bereits in den Recherche Kapitel 3.2.3 beschrieben, muss darauf geachtet werden, ein möglichst kurzes aber erklärendes Label zu verwenden.

Die Anpassung für z.B. die GitHub Verlinkung sieht wie folgt aus:

```
1 <Link
2   href="https://github.com/TKSpectro/turbomeet"
3   aria-label="Github Projekt turbomeet"
4 >
5   <FiGithub className="float-right h-7 w-7" />
6 </Link>
```

5.3.2 Farbkontraste

Der Punkt Farbkontraste wurde an vielen verschiedenen Elementen angemerkt. Dies kann in dem ausführlichen Bericht nachvollzogen werden. Das Problem hierbei war, dass diese Kontraste auch z.B. zwischen dem Seitenhintergrund und den Card Elementen entstand, welche nur die vordefinierten Farben von TailwindCSS (*bg-gray-900* und *bg-gray-800*) nutzten. Die Standardwerte der Grautöne sind durch einen relativ starken blauen Unterton eher kritisch für die Accessibility. Genau dieses Problem hatte ich bereits in einem vorherigen Projekt, welches ebenfalls TailwindCSS genutzt hat, erkannt und behoben. ([GitHub.com/TKSpectro/budgetify](https://github.com/TKSpectro/budgetify)). Die Lösung ist mit Hilfe der Tailwind Konfigurationsdatei möglich. In dieser können die definierten Farben durch Accessibility freundlichere Grautöne überschreiben werden. (siehe Listing B.1)

6 Fazit

Das Ziel meiner Arbeit war es verschiedene Technologien und Libraries anhand eines praktischen Beispiels zu benutzen und im Bezug auf Accessibilityunterstützung während der Entwicklung zu untersuchen. Hierbei sind im Besonderen die Libraries RadixUI und CmdK sehr positiv aufgefallen.

RadixUI nimmt dem Entwickler sehr viel manuelle Arbeit und Recherche ab, da es sinnvolle Defaults mit sich bringt und durch die WAI-ARIA Konformität auch immer auf den aktuellsten Standards agiert. Für die Teile die RadixUI nicht automatisch generieren/übernehmen kann, gibt es in der Dokumentation sehr gute Beispiele und Hinweise, die es dem Entwickler ermöglichen, die Accessibility Features selbst zu implementieren. Dies ist insbesondere für die Nutzung von ARIA-Attributen sehr hilfreich, da diese sehr umfangreich sind und somit auch sehr viel Vorwissen bzw. extra Recherchen erfordern.

CmdK bietet eine Combobox Menü Implementierung, welche auf RadixUIs Dialog basiert und somit Accessibility Features wie z.B. Tastatursteuerung und Screenreader Unterstützung direkt mitbringt. Meiner Meinung nach sind diese Art von Menüs unglaublich sinnvoll und sollten in Zukunft, vorallem auch auf nicht Technologisch orientierten Webseiten und Tools, vermehrt eingesetzt werden. Sie bieten eine unkomplizierte und vorallem effiziente Möglichkeit, Aktionen und Features zu suchen und zu nutzen. Hierbei ist auch der Nutzen für Screenreader Nutzer sehr groß, da diese nun nicht mehr durch eine Menüstruktur navigieren müssen, um an die gewünschte Aktion zu gelangen.

Analyse Tools wie Lighthouse sind während der Entwicklung sehr gut einsetzbar, da Sie per Knopfdruck direkt Auswertungen zu Accessibility Punkten liefern. Außerdem geben sie bei Problemen direkt Hinweise, wie diese behoben werden können. Dies ist insbesondere für Entwickler, die wenig Erfahrung mit Accessibility Features haben, sehr hilfreich. Damit diese Tests automatisiert werden können und somit auch in einem CI/CD Prozess integriert werden können, gibt es Tools wie Axe oder die Lighthouse CLI, welche die Auswertungen in einem JSON Format liefern. Diese können dann in einem CI/CD Prozess verwendet werden, um die Accessibility Features zu testen und zu überprüfen.

Auch wenn Bibliotheken wie RadixUI Entwicklern viele a11y Features direkt mitbringen, ist es dennoch wichtig, sich mit Accessibility Features und deren Implementierung auseinanderzusetzen. Dafür kann ich den, im Kapitel 3.2 angesprochenen, *ARIA Authoring Practices Guid* sehr empfehlen. Denn dieser gibt einen sehr guten Überblick über die verschiedenen ARIA-Features und deren Implementierung. Außerdem gibt er sehr gute Beispiele und Hinweise, wie diese Features in der Praxis angewendet werden können. Auch die Kapitel zu Benennung und Beschreibung von Elementen sind sehr hilfreich, da dies ein sehr wichtiger Aspekt der Accessibility ist.

A Abbildungen

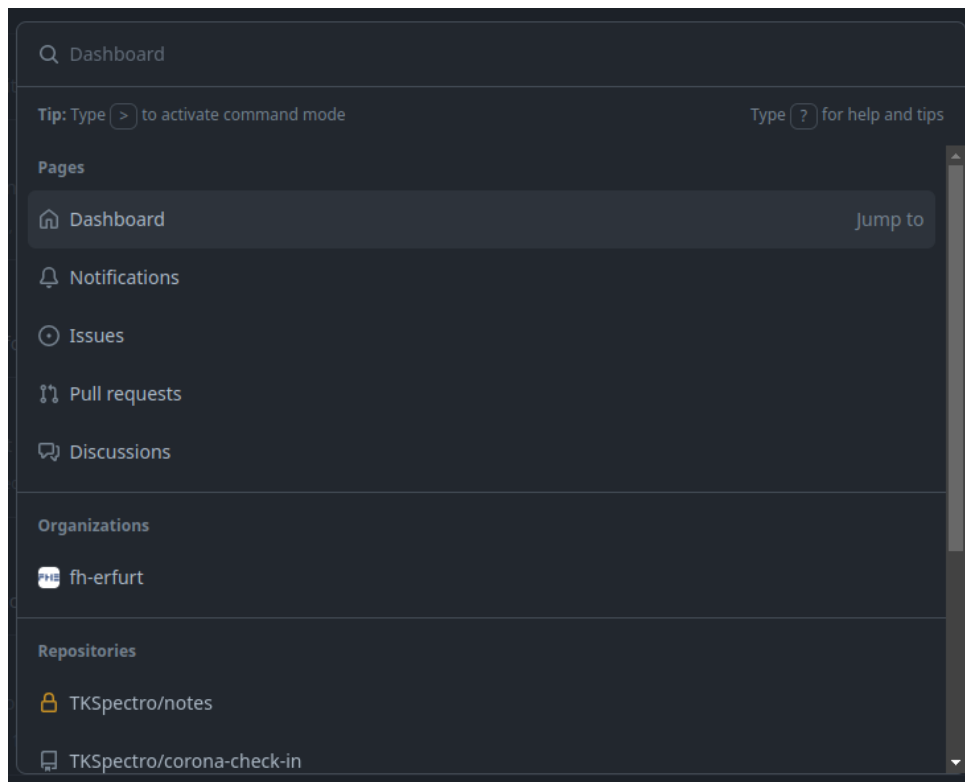


ABBILDUNG A.1: GitHub Kombobox

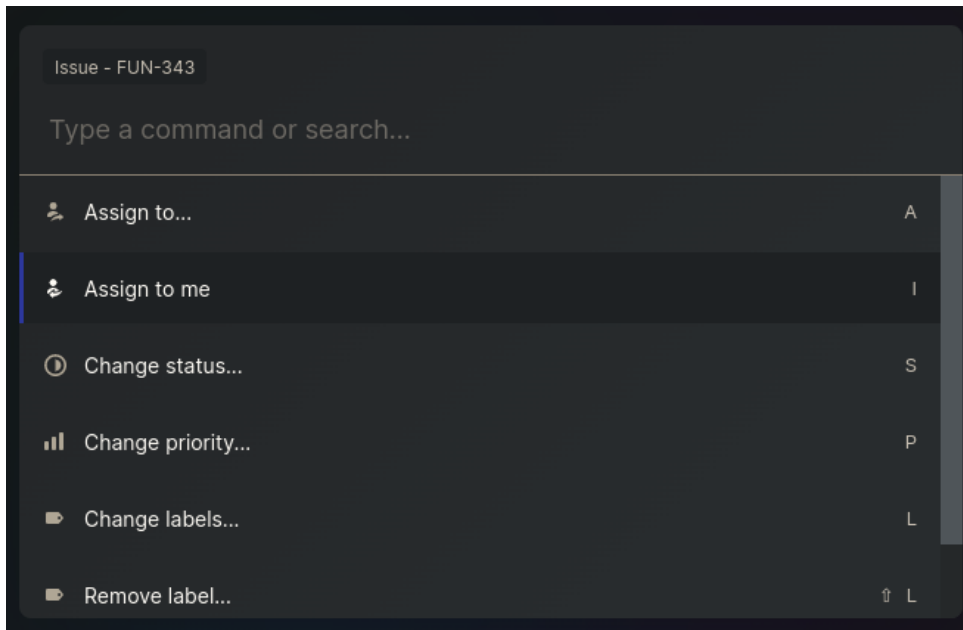


ABBILDUNG A.2: Linear Kombobox

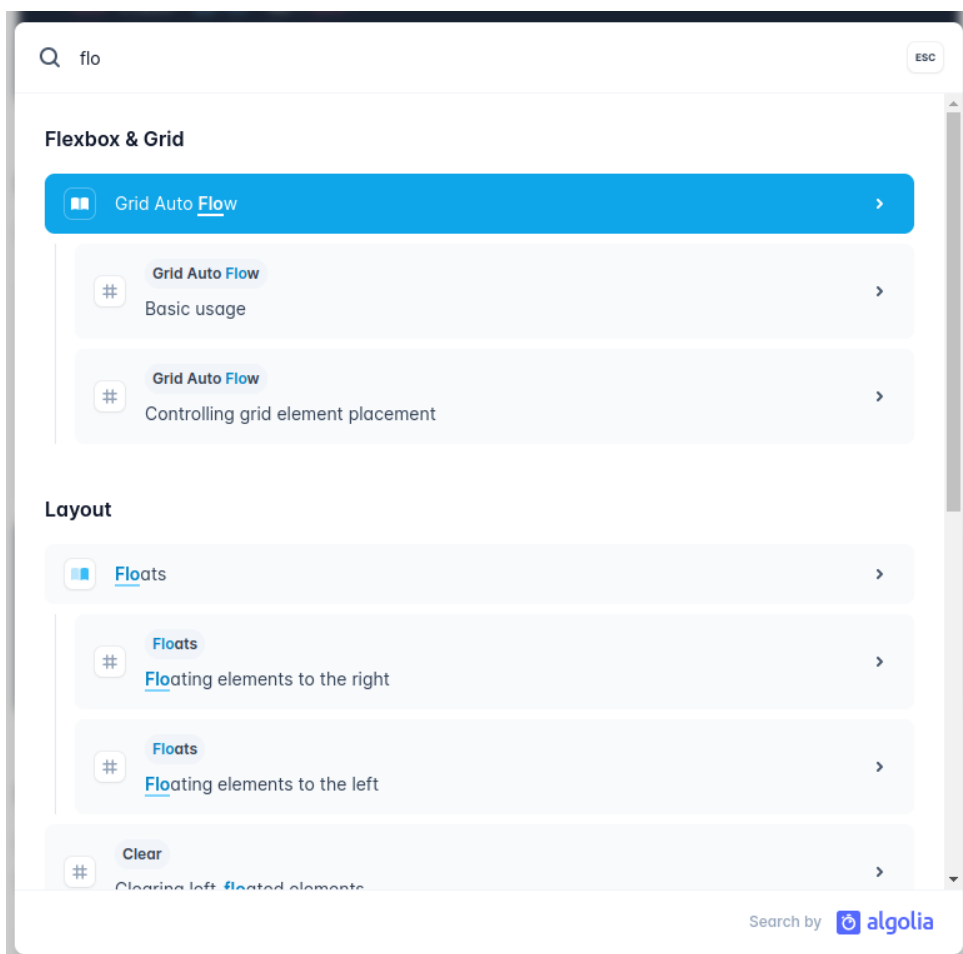


ABBILDUNG A.3: TailwindCSS Kombobox

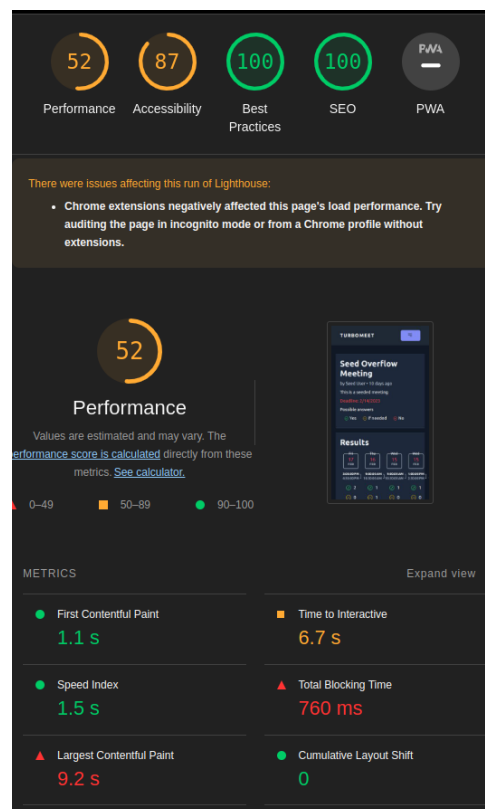


ABBILDUNG A.4: Lighthouse Bericht für Mobile Zwischenstand

B Quellcode

```

1  /** @type {import('tailwindcss').Config} */
2
3  // eslint-disable-next-line @typescript-eslint/no-var-requires
4  const colors = require('tailwindcss/colors');
5
6  module.exports = {
7    content: ['./src/**/*.{js,ts,jsx,tsx}'],
8    darkMode: 'class',
9    theme: {
10     extend: {
11       colors: {
12         primary: colors.emerald[500],
13         success: colors.green[500],
14         warning: colors.yellow[500],
15         danger: '#fe524f',
16         info: colors.emerald[500],
17         gray: {
18           50: '#fafafa',
19           100: '#f4f4f5',
20           200: '#e4e4e7',
21           300: '#d4d4d8',
22           400: '#a1a1aa',
23           500: '#71717a',
24           600: '#52525b',
25           700: '#3f3f46',
26           800: '#27272a',
27           900: '#18181b',
28         },
29       },
30       ringColor: {
31         DEFAULT: colors.indigo[500],
32       },
33       fontFamily: {
34         lato: ['Lato', 'sans-serif'],
35       },
36     },
37   },
38   plugins: [],
39 };

```

LISTING B.1: TailwindCSS Konfiguration

Literatur

- Angular Components Team (o. D.). *Angular Material*. en-US. URL: <https://material.angular.io/> (besucht am 22. 11. 2022).
- Bundesfachstelle Barrierefreiheit (o. D.). *Wie ist Barrierefreiheit definiert?* de. URL: https://www.bundesfachstelle-barrierefreiheit.de/DE/Ueber-Uns/Definition-Barrierefreiheit/definition-barrierefreiheit_node.html (besucht am 21. 11. 2022).
- <https://trpc.io/> (o. D.). *trpc/trpc: Move Fast and Break Nothing. End-to-end typesafe APIs made easy*. URL: <https://github.com/trpc/trpc> (besucht am 14. 02. 2023).
- Initiative (WAI), W3C Web Accessibility (o. D.[a]). *Developing a Keyboard Interface*. en. URL: <https://www.w3.org/WAI/ARIA/apg/practices/keyboard-interface/> (besucht am 13. 12. 2022).
- (o. D.[b]). *Landmark Regions*. en. URL: <https://www.w3.org/WAI/ARIA/apg/practices/landmark-regions/> (besucht am 13. 12. 2022).
- (o. D.[c]). *Patterns*. en. URL: <https://www.w3.org/WAI/ARIA/apg/patterns/> (besucht am 13. 12. 2022).
- (o. D.[d]). *Providing Accessible Names and Descriptions*. en. URL: <https://www.w3.org/WAI/ARIA/apg/practices/names-and-descriptions/> (besucht am 13. 12. 2022).
- Storybook (o. D.). *Storybook: Frontend workshop for UI development*. en. URL: <https://storybook.js.org> (besucht am 22. 11. 2022).
- W3C Web Accessibility Initiative (WAI) (o. D.). *ARIA Authoring Practices Guide*. en. URL: <https://www.w3.org/WAI/ARIA/apg/> (besucht am 13. 12. 2022).