Accessibility Basics

Barrierefreiheit von Websites und Webapplikationen

Weltweit haben über 1 Milliarde Menschen eine Körperliche Einschränkung (https://www.audioeye.com/resources/accessibility-essentials/). Aus diesem Grund ist es wichtig, dass die wesentlichen Regeln der Barrierefreiheit eingehalten werden, damit möglichst viele Personen unsere Websites verwenden können. Aus diesem Grund gibt es die WCAG 2.2 Richtlinien welche definieren wie eine Website Barrierefrei funktionieren kann. Die meisten dieser Punkte bedeuten keinen zusätzlichen Programmieraufwand wenn man die Punkte von Anfang an im Kopf hat.

Lieblingszitat aus den Richtlinien "Inhalte sind so zu gestalten, dass keine epileptischen Anfälle ausgelöst werden" Beispiel dazu: Palfinger | Annual Report 2022

Bei Accessibility-Verstößen erhält man normalerweise zunächst eine Verwarnung mit Frist zur Nachbesserung und erst dann (theoretisch) Strafen.

Die Richtlinien lassen sich hierbei in die 4 Bereiche Wahrnehmbar, Bedienbar, Verständlich und Robust unterteilen, die vollständigen Richtlinien können unter folgendem Link nachgelesen werden: Web Content Accessibility Guidelines (WCAG) 2.2 .

Neuerungen in WCAG 2.2 : BIK BITV-Test | Standards und Gesetze - WCAG 2.2. ist da!

Umsetzungen erfolgen grundsätzlich nach Konformitätsstufe AA. Falls ein Kunde auf AAA besteht, muss dies gesondert abgeschätzt werden.

FE = Info vorrangig für Frontend-Developer
BE = Info vorrangig für Backend-Developer
DESIGN = Info vorrangig für Designer
PM = Info vorrangig für Projektmanager

Designphase FE DESIGN PM

- Design muss Kontrastreich sein. Kontrast auch nach der Umsetzung in Devtools prüfen
  Mit den Entwicklertools von Chrome kannst du Text mit geringem Kontrast erkennen und korrigieren | Google for Developers
  Zu schlechte Kontrastverhältnisse direkt in der FE-Umsetzung anpassen
- Keine Slider die nur mit Dragging funktionieren. Es muss Control-Elemente geben (WCAG 2.2)
- Keine Autoplay Slider/Videos. Es muss Control-Elemente geben

tldr FE BE

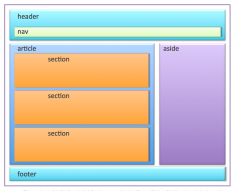
- Am wichtigsten ist es semantisch sinnvolles und valides HTML zu schreiben. Dadurch beugt man den meisten potentiellen Probleme bereits vor.
- Wenn es native HTML Elemente gibt sollten diese auch verwendet werden (z.B. sollte ein Button nicht mit einem div umgesetzt werden.)
- HTML5 Landmarks in logischer Anordnung verwenden (header, footer, main,nav, section,....). Genaue Erklärung: ARIA: landmark role - Accessibility | MDN
- Die Reihenfolge der Elemente im DOM und Grafisch sollte ident zueinander sein, da tabbing auf der Seite ansonsten verwirrend sein kann (http://adrianroselli.com/2015/10/html-source-order-vs-css-display-order.html). Sofern möglich sollte also die Reihenfolge von Elementen im HTML und nicht mit CSS angepasst werden.
- ARIA nur bei dynamischen Elementen verwenden welche mit HTML nicht abbildbar sind.

Mit Details:

HTML5 Sectioning FE

HTML5 Sectioning Elemente geben die Struktur der Website an und können Screenreadern bei der Navigation unterstützen.
Folgende Elemente können verwendet werden:

header, footer, main, nav, section, aside



ARIA: landmark role - Accessibility | MDN

Verwende semantisch korrekte Tags FE BE

Weiterführendes

- All The Tags
- HTML elements reference - HTML: HyperText Markup Language | MDN

Inputs FE BE

Jeder Input sollte ein passendes Label erhalten damit der User weiß was er in dieses Feld schreiben soll. Alle Labels sollten dabei möglichst sinnvolle Texte beinhalten.
Beispiel:

```
<label for="zuname">Zuname</label>
<input name="zuname" id="zuname" maxlength="40">
```

Weiters sollten alle Inputs einen passenden Type erhalten (z.B. type="number"), eine Liste mit allen möglichen Types findet sich hier: https://www.w3schools.com/tags/att_input_type.asp
Sollte laut Design kein Label vorhanden sein, muss es dennoch im HTML vorhanden sein und per CSS ausgeblendet werden.

Buttons FE BE

Buttons mit Textinhalten bereiten in der Regel keine Probleme mit Barrierefreiheit. Allerdings beinhalten viele Buttons keinen Text sondern lediglich Icons welche für Screenreader keine sinnvollen Informationen beinhalten. Ein Beispiel dazu findet sich in nahezu jedem header mit Burger-Menü oder einem User-Icon. Deshalb muss mittels ARIA Labels bestimmt werden worum es sich bei diesem Element handelt. Damit das Icon nicht vorgelesen wird erhält dieses noch das Attribut aria-hidden="true"

Beispiel:

```
<button type="button" aria-haspopup="true" aria-label="Navigation"><i class="icon icon-menu" aria-hidden="true"><
```

Icons FE

Verwende aria-label & title um Icons mit einer Textalternative auszuzeichnen.

Ausnahmen

Bei Icons, die ausschließlich zur visuellen Unterstützung verwendet werden und deren Bedeutung bereits durch ihren Kontext klar ist (z. B.: durch andere Texte), sollte auf diese Attribute verzichtet werden. Sie sollten nur mit aria-hidden=”true” ausgezeichnet werden, um sie für Screenreader komplett zu verstecken.

```
// Bad
<span class="icon icon-warning"></span>

// Good
<span class="icon icon-warning" aria-label="Warnung" title="Warnung"></span>

<h2>
  <span class="icon icon-warning" aria-hidden="true"></span>
  Warnung
</h2>
```

Weiterführendes

- WCAG 1.1 Text Alternatives

Bilder FE BE

Text innerhalb von Bildern sollte mittels HTML angegeben werden. Sollte der Text direkt im Bild sein und auch als solches gerendert werden wird der Text nicht von Screenreadern erkannt und dadurch auch nicht vorgelesen. Eine Ausnahme besteht für Logos da diese in der Regel Text enthalten

Alle rein visuellen Inhalte wie zum Beispiel Bilder sollten Text alternativen erhalten.

```
<img src="smiley.png" alt="Smiley face" height="42" width="42">
```

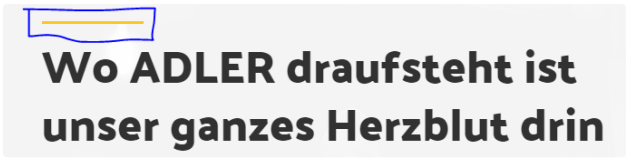
Der Text in diesem Bild ist Teil des pngs. Aus diesem Grund kann der Text nicht von Screenreadern gelesen werden und ist nicht Barrierefrei.

Dekorationselemente FE

Elemente welche nur als Dekoration verwendet werden sollten von Screenreadern ignoriert werden können da diese keinen Inhalten Mehrwert liefern.

Beispiel:

```
<div aria-hidden="true" style="be-fancy"></div>
```



Zeitliche Anpassung FE DESIGN PM

Alle Elemente welche sich nach einer bestimmten Zeit automatisch verändern, müssen vom User zeitlich angepasst werden können. Dazu zählen zum Beispiel Slider die automatisch weiter Sliden, oder Toasts die nach einer bestimmten Zeit verschwinden. Die einfachste Lösung um diese Regel einzuhalten wäre es, dass sich diese Elemente entweder gar nicht von selbst bewegen oder sehr einfach pausiert werden können.

Reihenfolge von Inhalten / Focus Reihenfolge FE

Die Reihenfolge der dargestellten Elemente sollte visuell ident zu jener im HTML sein. Vor allem wenn die Reihenfolge wichtig ist um die Inhalte zu verstehen. Aus diesem Grund sollte die Reihenfolge der Elemente nicht mittels flex oder grid verändert werden. Auch Stylings wie float: right sollten nur dann verwendet werden wenn sich das Element bereits an der letzten Stelle befindet und die Reihenfolge nicht beeinflusst wird.
Es sollte auch getestet werden ob alle Inhalte mit möglichen Nutzer Interaktionen mittels tabbing erreicht werden können. Sollte dies nicht möglich sein können fehlende Elemente mit tabindex="0" versehen werden um auch erreicht werden zu können. Wichtig ist es aber keine höheren Werte als 0 zu vergeben da diese die Reihenfolge des Tabbings verändern.

Texte FE BE

Um Texte verständlich zu machen werden verschiedene Tags verwendet.

Es ist nicht Barrierefrei wenn für alle Textelemente p Tags verwendet werden und für den jeweiligen Use Case gestyled werden. Für Personen die die Website normal verwenden macht dies zwar keinen Unterschied, allerdings ändert sich die Bedeutung des Textes für Screenreader. (https://accessibility.psu.edu/headingshtml/).

h1: page title
h2: major heading
h3: major sub heading
.....
p: paragraph

Visuell kann ein p Tag wie eine h1 gestyled werden, semantisch aber nicht!

Alle Überschriften und Labels sollten auch hier wieder sinnvolle Texte beinhalten.

Farben FE

Farben dürfen nicht der einzige Indikator sein, dass mit einem Abschnitt eine Interaktion möglich ist. Sie müssen immer in einem bestimmten Kontext verwendet werden. Grund dafür ist, dass screen reader nicht auf reines styling reagieren und die jeweiligen Passagen dann nur textuell erkennen können. Als reines Dekorationselement dürfen Farben verwendet werden.

Don't:

```
<div style="color:red;">Ich löse ein Event aus</div>
```

Audio FE DESIGN PM

Sollte auf einer Website ein Ton länger als 3 Sekunden abgespielt werden, müssen Kontrollelemente erreichbar sein. Die Datei muss dabei pausiert oder gestoppt und in der Lautstärke geregelt werden können. Die Lautstärke muss direkt auf der Website gesteuert werden können. Die allgemeine Systemlautstärke ist nicht ausreichend um WCAG konform zu sein da auch der Screenreader selbst von dieser Lautstärke abhängig ist.

Animationen und Videos FE DESIGN PM

Alle Animationen und Videos welche automatisch starten und länger als 5 Sekunden dauern, müssen von Nutzern kontrolliert werden können. Unter kontrollieren fallen pausieren, stoppen oder hiden.

Animationen sind unter anderem auch scrolling. Automatisches scrollen zu einer bestimmten Stelle der Seite darf also nur weniger als 5 Sekunden dauern.

Es müssen also alle Videos pausierbar sein. Kritisch sind hier zum Beipiel hero Elemente welche automatisch abgespielt werden wenn keine Kontrollelemente eingeblendet werden.

Selbst wenn Videos entsprechende Controls beinhalten, sollte immer getestet werden ob diese auch erreichbar sind.

Eine Ausnahme ist es wenn eine Animation ein zwingender Bestandteil einer Aktivität ist. Dazu gehört zum Beispiel ein loading spinner im Product Grid.

In jedem Fall darf eine Animation pro Sekunde nicht mehr als 3 flashes enthalten da ansonsten die Gefahr für einen Epileptischen Anfall besteht.

Tastaturnavigation FE

Es muss möglich sein mit der Tastatur durch die gesamte Website zu navigieren ohne eine Maus zu verwenden. Das bedeutet, dass es auch keine “Fallen” gibt mit denen man mit der Tastatur nicht mehr hinaus kommt. Ein häufiger Fehler ist z.B., dass man mittels Tastatur eine Lightbox öffnen kann welche man dann nicht mehr verlassen kann, weil es keinen entsprechenden close button gibt.

Weiters erhält ein aktives Element einen focus state, dieser hat je nach Browser verschiedene default stylings. Stylings sollten nicht entfernt werden da ansonsten nicht erkennbar ist wo man sich gerade befindet. Eine Lösung wäre: während der Umsetzung zusätzlich zum jeweiligen :hover State den :focus State zu stylen.

Links FE

Jeder Link sollte textuell sinnvoll sein sodass User wissen wohin der Link führt. Dies geschieht entweder durch den Text im Link selbst oder durch den Kontext in dem sich dieser befindet. Auch hier muss man wie bei Buttons darauf achten ob Links welche nur ein Icon beinhalten

sinnvoll sein können oder nicht. Um eine Textuelle Beschreibung hinzuzufügen welche nicht für alle Nutzer sichtbar ist kann die Klasse sr-only (Bootstrap 3+4) bzw. visually-hidden (Bootstrap 5) vergeben werden. Dadurch können Screenreader zusätzlichen Kontext für einen Link erkennen.

```
<a href="#" aria-label="Read more about aria labels">[Read more...]</a>
```

Zusätzlicher Kontext kann auch, wie in diesem Beispiel, mit einem aria-label gegeben werden.

Sprache FE BE

Die Sprache im header sollte in jedem Fall gleich sein wie jene im Text. Grund dafür ist es, dass Screenreader in den jeweiligen Sprachen vorlesen können und dabei die Betonung der einzelnen Wörter anpassen.
Sollten nur einzelne Textpassagen in einer anderen Sprache geschrieben sein, können auch diese mit lang="" definiert werden. z.B.

```
<a lang="es" href="qa-html-language-declarations.es">Español</a>
```

Focus FE

Elemente welche einen Focus-State bekommen ändern ihren Inhalt nicht. Durch Focus allein dürfen sich also z.B. keine Texte ändern solange der User den Text nicht aktiv ändert.

Navigation auf der Website FE BE DESIGN PM

Für Benutzer mit geringer Aufmerksamkeitsspanne sollten Hilfen zur Orientierung angeboten werden, damit der User weiß wo er sich befindet. Dazu können zum Beispiel Breadcrumbs dienen.

Navigationselemente welche auf mehreren Seite vorkommen, werden immer in der gleichen Reihenfolge dargestellt. Menüs sollten also auf jeder Seite gleich sein.

Table of Contents FE BE

Inhaltsübersicht mit toc package generieren
https://npm-packages.elements.at/-/web/detail/@elements/toc

Components FE DESIGN

Sollte eine bestimmte Komponente mit einer bestimmten Funktionalität auf mehreren Seiten vorkommen, sollte diese auch immer gleich aussehen damit der User weiß welche Funktion diese erfüllt. Dadurch ist die Website für alle User schneller erlernbar und im Design konsistenter.

Fehler FE BE

Sollte ein User ein Formular falsch ausfüllen, soll der Fehler markiert und möglichst genau beschrieben werden. Dabei kann das falsch ausgefüllte Feld z.B. rot markiert werden und ein Text daneben platziert werden. Je genauer der beschreibende Text ist, desto hilfreicher ist es für den Nutzer. Wenn z.B. eine Postleitzahl angegeben werden soll und der Nutzer ein Wort hinein schreibt, kann dieser darauf hingewiesen werden, dass nur Zahlen angegeben werden dürfen. Wobei in diesem Fall ohnehin ein Input vom Type Number verwendet werden sollte.

```
<input type="number">
```

Um Fehler möglichst zu vermeiden muss für jedes Input ein passendes Label angegeben werden.

Valides HTML FE BE

HTML sollte immer valide sein um sicher zu gehen, dass alle Parser die Inhalte auch richtig interpretieren. Um das HTML zu testen können online Tools wie das folgende verwendet werden: https://validator.w3.org/#validate_by_input

Überprüft wird unter anderem ob jede ID nur 1 mal exisiert, ob jedes Elements richtig beginnt und endet, ob jeder Tag alle benötigten Inhalte hat,....

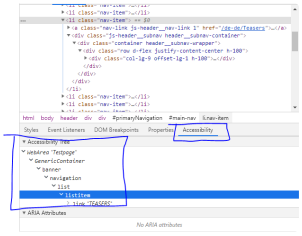
Accessibility Tree FE BE

Screenreader lesen nicht das normale DOM vor, sondern den sogenannten Accessibility Tree welcher dem DOM zwar sehr ähnlich ist aber dennoch Unterschiede aufweist. Mittels den Chrome Dev Tools kann man sich den Accessibility Tree zu jedem Zeitpunkt ansehen.

Durch die Darstellung von diesem Tree werden auch die zuvor genannten HTML5 sectioning Elemente und andere semantische Elemente klarer. Diese beschreiben den Content noch einmal genauer als andere Elemente dies tun könnten, wie z.B. ein div. Ein div erhält lediglich den Namen GenericContainer was keine Informationen beinhaltet außer, dass der Inhalt in irgendeiner Form zusammen gehört.

ARIA hat dabei keinen Einfluss auf die Darstellung im DOM aber ändert den Accessibility Tree wodurch dieser semantisch erweitert werden kann.

Eine genaue Erklärung kann hier nachgelesen werden: Baum für Barrierefreiheit | Articles | web.dev



ARIA FE BE

Alle dynamischen Inhalte welche sich anders verhalten als dies mit HTML normalerweise möglich wäre, müssen Inhaltlich erweitert werden um richtig interpretiert werden zu können. Dazu kann ARIA verwendet werden. Bei fertigen Bootstrap Komponenten wie z.B. Collapsibles sind diese bereits richtig vergeben (Collapse). Bei Unklarheiten über aria-expanded usw. am besten direkt in der Bootstrap Doku durchlesen, hier gibt es zu jeder Komponente einen Bereich zum Thema Accessibility.
Genaue Erklärungen und Anwendungsfällen von ARIA werden hier aufgelistet: ARIA Authoring Practices Guide

Inhalte die nachgeladen werden, müssen mit aria-live=”polite” gekennzeichnet werden. Details: ARIA live regions - Accessibility | MDN

Ein kurzes Intro zu ARIA zeigt dieses ca 9 Minütige Video: