Web Accessibility Evaluierung

Mag. Maria Putzhuber
www.wienfluss.net

Mag. Wolfram Huber
www.web-tech.at

ATAG 2016, Wien

wienfluss
INFORMATION.DESIGN.SOLUTIONS

WEB-TECH
COACHING

Web Accessibility Evaluierung

- Allgemeines
- Ablauf einer Evaluierung
- Aufbau eines Berichts
- Verwendete Test Tools

- Fokus Frontend - typische Accessibility Fehler
- Test Demo (falls noch Zeit bleibt)

Allgemeines

- In der Regel eher aufwändigerer manueller Prozess
- Manuelle und automatische Überprüfung von allen WCAG Checkpunkten (A, AA und nach Absprache AAA) mittels verschiedener Hilfsmittel.
- Je nach Templateanzahl ca. 20 bis 60 seitiger Bericht / ein umfassendes Gutachten über den Accessibilitystand der Website inkl. Optimierungsvorschlägen
- Preis richtet sich nach Komplexität, Anzahl und Unterschied der zu testenden Seiten, Anzahl der Testpersonen etc.

Ablauf

- Templates
  Aus dem Webangebot werden Seitenvorlagen (Templates) ausgesucht, die wichtig sind oder für möglichst viele andere Seiten sprechen
  z.b. Startseite, Detailseite, Listenseite, Formularseite, Suchergebnis, Warenkorb, Kassa, Userlogin etc.
- Anzahl
  Je nach Komplexitätsgrad des Webangebots werden i.d.R. zwischen 5 und 25 Seitenvorlagen bestimmt
- Evaluierungsprozess
  Der Evaluierungsprozess läuft häufig in 3 Schritten ab
  - Automatisierte Evaluierung
  - Experten Evaluierung
  - Accessible Usertests

Automatisierte Evaluierung

Mittels automatisierten Testtools werden erste technische Optimierungsmöglichkeiten aufgezeigt.

Website Checker

» Wave (http://wave.webaim.org)
Automatisierte Analyse einer URL, Wave kann WCAG Checkpunkte überprüfen, welche automatisch überprüfbar sind (z.b. haben alle Bilder ein Alternativattribut, ob das Alternativattribut aber sinnvoll ist, muss manuell angeschaut werden)
und/oder

» Eiii (http://checkers.eiii.eu)
Alternative zu Wave, ebenfalls automatisiertes Testtool, analysiert auch PDF Dokumente

HTML Validatoren

» validator.w3.org (https://validator.w3.org)
Analysiert den HTML Code auf Fehler
und/oder

» html-validator (https://addons.mozilla.org/en-US/firefox/addon/html-validator/)
Firefox Addon bzw. in Webdeveloper Toolbar integrierter W3C Validator

Experten Evaluierung

Die Experten Evaluierung ist der Hauptteil im Evaluierungsprozess. Alle Punkte der WCAG 2.0 werden hier von ExpertInnen unter die Lupe genommen. Nützliche Hilfsmittel:

Browser Toolbar

» Web Developer Toolbar (https://addons.mozilla.org/en-US/firefox/addon/web-developer)
Evaluierungs Addon für Firefox

» Web Accessibility Toolbar (https://www.paciellogroup.com/resources/wat)
Evaluierungs Addon für Internet Explorer (Version 9/10/11)

Code Analyse

» Firebug (https://addons.mozilla.org/en-US/firefox/addon/firebug)
Code Analyse Addon für Firefox

» Chrome Browser
Integrierte Code Analyse (Inspector)

Screenreader

» Jaws, NVDA, VoiceOver

Accessible Usertest

Ein weiterer zentraler Kernpunkt sind User Tests. Bei diesen werden in erster Linie NutzerInnen von Screenreadern diverse Aufgaben im Webangebot gestellt.

Beispiel Vorgaben für Screenreader TestuserInnen

- Frage 1: Versuchen Sie sich kurz einen Überblick über die Seite zu machen und beschreiben Sie Ihre Eindrücke?
- Frage 2: Bedienen Sie einen Menüpunkt. Können Sie schnell erfassen auf welcher Subseite Sie sich befinden?
- Frage 3: Wie geht es Ihnen bei der Navigation innerhalb einer Seite? Sind die Bereiche abgetrennt erkennbar?
- Frage 4: Sonstige Verbesserungsvorschläge für Screenreaderuser?
- Frage 5: z.b. Finden Sie die E-Mail Adresse des Unternehmens und beschreiben Sie die Vorgangsweise (mit potentiellen Barrieren)
- Frage 6: z.b. bei Shop: Suchen und bestellen Sie folgendes Produkt…. (mit zur Verfügung gestellten Anmeldedaten)

Accessibility Prüfbericht

Ergebnisse der 3 Bereiche werden in einem Bericht zusammengefasst.

Der Bericht wird i.d.R. nach den Vorlagetemplates strukturiert

- T1: Analyse WCAG Punkte A, AA, (AAA)
  Fehlerauflistung, Einschätzung, Korrektur- /Optimierungs-Empfehlungen, Erklärung der Einzelpunkte
- T2 bis TX (falls ähnliches Rahmenlayout)
  nur Punkte, welche nicht erfüllt sind, werden gelistet
- Fazit und Zusammenfassung

Web Accessibility Evaluierung – Teil 2

Die häufigsten Accessibilityfehler in der Frontendentwicklung

WIENFLUSS
INFORMATION.DESIGN.SOLUTIONS

WEB-TECH
COACHING

WCAG Kriterien aufräumen

WCAG 2.0 A + AA Erfolgskriterien (Success Criteria, Sc)	38
Minus Sc für Multimedia (im weiteren Sinn)	- 7
Minus Sc, die schon Designentscheidungen betreffen	- 4
Minus Sc, die in moderner, **professioneller** Frontend Entwicklung weniger Bedeutung haben	-12
Minus leicht zu behebender Peanuts Probleme	- 3
Ergibt übersichtliche	12

WCAG Kriterien, die sich Frontend Developer genauer ansehen sollten.

	Erfolgskriterium	Erfüllt
Wahrnehmbar		
Richtlinie 1.1 Textalternativen		
A	1.1.1 Nicht-Text Inhalt	Nein
Richtlinie 1.3 Anpassbar		
A	1.3.1 Info und Beziehungen	Nein
A	1.3.3 Sensorische Eigenschaften	Nein
Bedienbar		
Richtlinie 2.1 Per Tastatur zugänglich		
A	2.1.1 Tastatur	Nein
Richtlinie 2.2 Ausreichend Zeit		
A	2.2.2 Pausieren, beenden, ausblenden	Nein
Richtlinie 2.4 Navigierbar		
A	2.4.3 Fokus-Reihenfolge	Nein
A	2.4.4 Linkzweck (im Kontext)	Nein
AA	2.4.7 Fokus sichtbar	Nein
Verständlich		
Richtlinie 3.3 Hilfestellung bei der Formulareingabe		
A	3.3.1 Fehlererkennung	Nein
A	3.3.2 Beschriftungen (Labels) oder Anweisungen	Nein
AA	3.3.3 Fehlerempfehlung	Nein
Robust		
Richtlinie 4.1 Kompatibel		
A	4.1.2 Name, Rolle, Wert	Nein

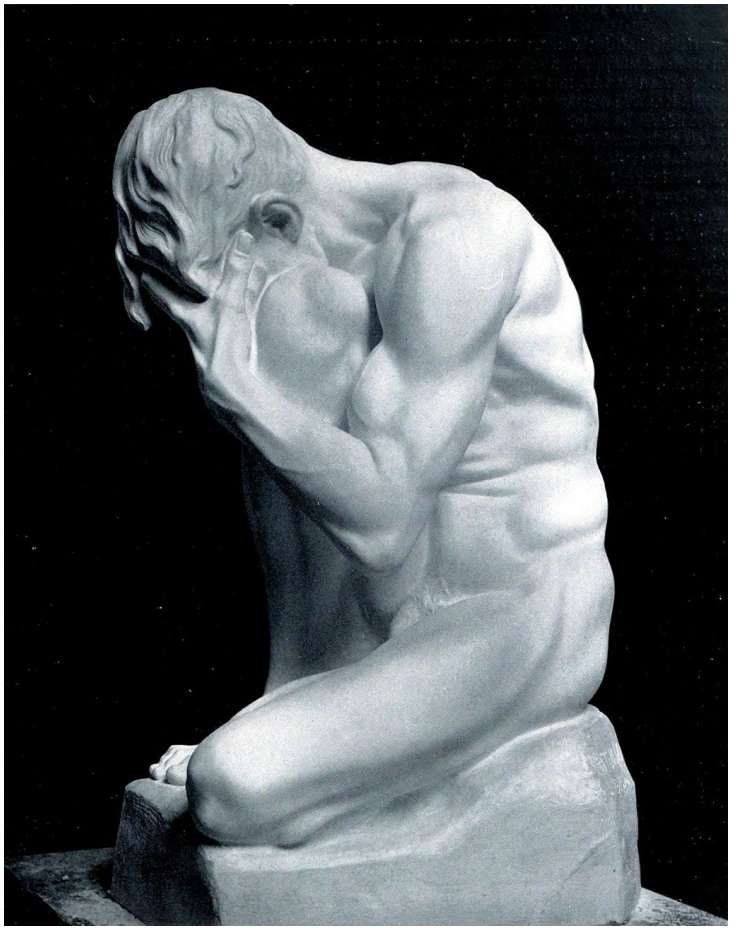
Accessibility Hauptprobleme

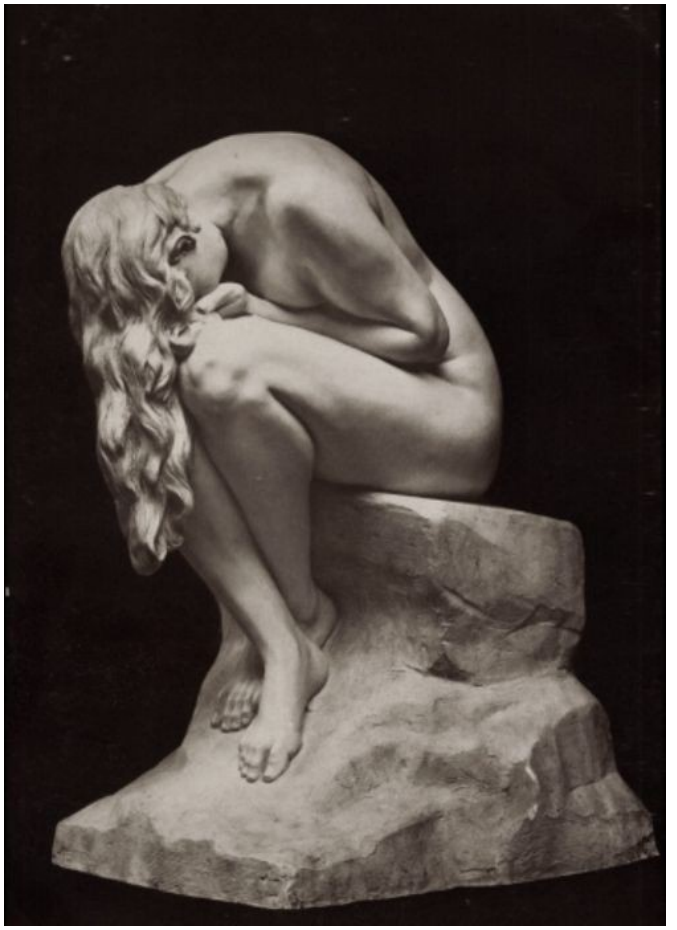
Fehlender
Textersatz für
visuelle Inhalte

Mängel in
Tastatur
Bedienbarkeit

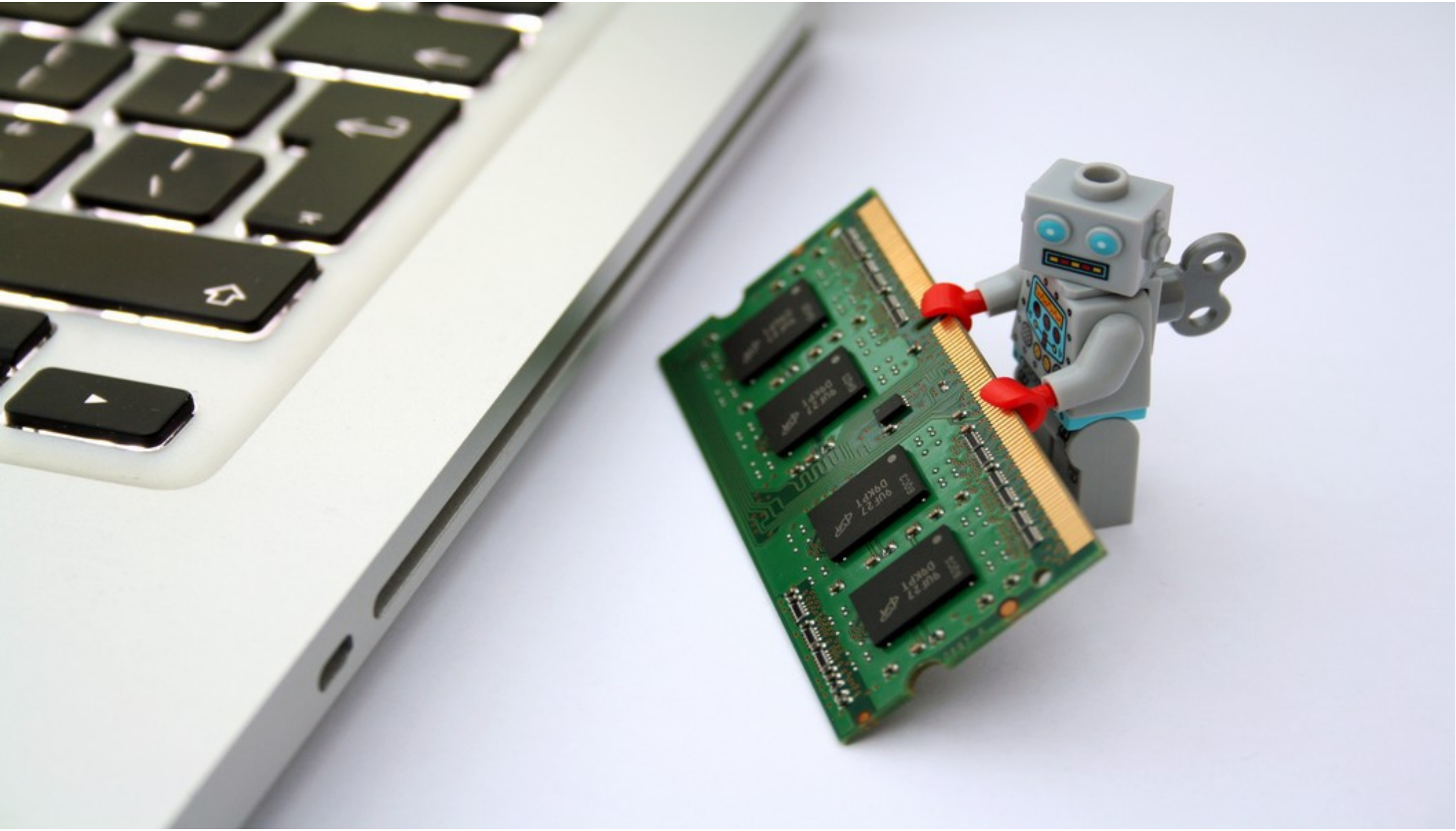
Fehlende
Semantik

Alle haben hier schon Fehler gemacht!





1. Fehlender Textersatz für visuelle Inhalte



1. Fehlender Textersatz für visuelle Inhalte

» fehlende Maschinenlesbarkeit

» fehlende Information für Screenreader NutzerInnen

» und für Zoomsoftware - Sprachausgabe

Richtlinie 1.1 Textalternativen	
A	1.1.1 Nicht-Text Inhalt
Richtlinie 1.3 Anpassbar	
A	1.3.3 Sensorische Eigenschaften
Richtlinie 2.4 Navigierbar	
A	2.4.4 Linkzweck (im Kontext)
Richtlinie 4.1 Kompatibel	
A	4.1.2 Name, Rolle, Wert

Die typischsten Fehler bei Textalternativen 1

Falsche / verwirrende / unklare Information

- Bilder mit Dateinamen im alt-Attribut
- Bilder ohne alt-Attribut (VoiceOver liest den Dateinamen)
- Verlinkte Bilder mit leerem alt-Attribut (SR lesen Teil des Linkpfads)
- Nicht sinnvolle Info im alt-Attribut (z.B. ausführliche Bidlbeschreibung statt dem Linkziel bei verlinkten Bildern
- Unnötige Info bei Platzhalter- / Dekobildern, z.b. alt="Abstandhalter" statt alt=""
- Leerzeichen im leeren alt-Attribut, alt=" " statt alt="" (SR lesen „Bild" o. „Grafik")
- Englische Beschriftungen von Bedienelementen („arrow down icon", „previous / next", „close"…)

Die typischsten Fehler bei Textalternativen 2

Leere (Bild)-Links oder Buttons

SR lesen Teil des Linkpfads, javascript void(); oder einfach nur Link / Button

- Nicht oder mangelhaft beschriftete Bedienelemente bei Menüs, Carousels (Bildslidern) und Lightboxen
- Bedienelemente mit Hintergrundbildern ohne SR-lesbaren Textersatz
- Font-Icons ohne Textersatz
- Falsch belegte Font-Icons (VoiceOver liest z.B. „Kirche“ statt „Telefon“) , fehlendes aria-hidden=“true“ bei Font-Icon, fehlender Textersatz
- SVG Icons ohne title
- Auch in der Web APP Programmierung werden die vorgegebenen Beschriftungsmöglichkeiten nicht genutzt

Die typischsten Fehler bei Textalternativen 3

Nicht 100% ausreichende Information

- Textinformation bei leeren Links nur über das Link title-Attribut (nicht sicher für alle Screenreader bzw. SR Konfigurationen)
- nur placeholder Attribut bei Formular Inputfeldern
- Toggle Links ohne aria-expanded Attribut (die Toggle-Funktion ist unklar)
- Textinformation für Screenreader über aria-label, die andere wichtige sichtbare Textinformation überschreibt

Die typischsten Fehler bei Textalternativen 4

Redundante Information

- Gleiche Information in Bild alt-Attribut und Link title-Attribut
- wiederholte Information bei gemeinsam verlinkten Bild+Text Teasern: im Bild alt-Attribut, Überschrift, Link title-Attribut, Teasertext (statt leerem alt-Attribut)
- doppelte Links: Bildlink, Textlink mit gleicher Information

Fehler bei versteckter Textinformation für Screenreader

- mit display:none; versteckte Textinformation
- aus dem Viewport geschoben mit top:-99999px (statt top:auto, Scrolleffekt in manchen Browsern)

2. Mangelnde Tastaturbedienbarkeit

2. *Mangelnde Tastaturbedienbarkeit*

» Websites / Applikationen sind nur mit der Maus / Touchgesten bedienbar

» Hilfsmittel wie Kopfstab, Mundstab, Switch, auch Screenreader bauen auf Tastaturbedienbarkeit

Richtlinie 2.1 Per Tastatur zugänglich		
A	2.1.1 Tastatur	Nein
Richtlinie 2.2 Ausreichend Zeit		
A	2.2.2 Pausieren, beenden, ausblenden	Nein
Richtlinie 2.4 Navigierbar		
A	2.4.3 Fokus-Reihenfolge	Nein
AA	2.4.7 Fokus sichtbar	Nein
Richtlinie 4.1 Kompatibel		
A	4.1.2 Name, Rolle, Wert	Nein

Die typischsten Fehler bei Tastaturbedienbarkeit 1

Fehlendes Tastatur Fokus Styling

- outline:0;
- keine :focus, :active Styleangaben, nur :hover Styles
- keine Orientierung, wo sich der Cursor gerade befindet

Die typischsten Fehler bei Tastaturbedienbarkeit 2

Nicht fokussierbare, nicht tastaturtaugliche Bedienelemente

- Links ohne href
- nur mit der Maus bedienbare Accordions, Carousels, Dialogboxen, Lightboxen, Menüs, Tooltipps, Tabreiter
- Durch Verwendung von Elementen ohne Link/Button/Formularfeld Funktionalität wie , , Überschriften und nur mausbedienbare Eventhandler im JavaScript, kein tabfocus="0", keine ARIA Auszeichnung für Screenreader
- Keine Definition der zu verwendenden Tasten im JavaScript (Tab-, Enter-, Pfeil-Leertaste)
- Nicht mit Tastatur fokussierbare grafische Formular Elemente, vorallem Checkboxen und Radiobuttons
- Carousels, die sich nicht mit der Tastatur und Maus anhalten lassen

Die typischsten Fehler bei Tastaturbedienbarkeit 3

Fehlendes oder fehlerhaftes Fokus Management

- bei Dialogboxen (z.B. modalen Dialogen) oder Lightboxen wird der Fokus nicht in das Overlay Element gesetzt (es ist nur theoretisch tastaturerreichbar, da es sich in der Fokusreihenfolge vielleicht erst am Ende der Seite befindet oder per JavaScript vor dem gerade fokussierten Toggle Button eingefügt wird.)
- Overlay Elemente werden nicht geschlossen, wenn man aus ihnen hinaustabbt
- Aufklappmenüs werden beim Weitertabben nicht geschlossen
- Bei Tabreitern oder Menüs kommt man nicht in der logischen Tabreihenfolge in die geöffneten Reiter oder Submenüs, sondern erst nach Durchtabben durch alle Tabreiter oder alle Hauptmenüpunkte.
- Tabfokus bei Formularen zwingt eine Tabreihenfolge auf, man kann wichtige Inhalte nicht oder nur mehr durch Rückwärts Tabben erreichen.
- Die visuelle Struktur orientiert sich an mobilen Applikationen, aber es gibt kein Tastatur Fokushandling für z.B. einen Fertig Button, der sich oben rechts befindet, statt am Ende der Seite.

Die typischsten Fehler bei Tastaturbedienbarkeit 4

Fehlende Alternative bei fehlender Tastaturbedienbarkeit

- Aufklappmenüs klappen nur bei mouseOver oder Mausklick auf (was bei umfangreichen Menüs auch von Vorteil sein kann), es gibt aber keine statischen Submenüs auf Unterseiten als Ersatzlösung

Unübliche Tastatur Lösungen im Web

- Aufklappmenüs und Tabreiter, die nicht mit Tabtaste und Enter, sondern nur mit Pfeiltasten oder anderen nicht vertrauten Tasten bzw. Tastenkombinationen bedienbar sind
- Lightboxen / Overlays lassen sich nicht mit ESCAPE Taste schließen

3. Fehlende Semantik

3. Fehlende Semantik

» Inhalte ohne Hierarchie, ohne Zusammenhang

» Keine Navigationsmöglichkeiten für Screenreader

Richtlinie 1.3 Anpassbar		
A	1.3.1 Info und Beziehungen	Nein
Richtlinie 2.4 Navigierbar		
A	2.4.4 Linkzweck (im Kontext)	Nein
Richtlinie 3.3 Hilfestellung bei der Formulareingabe		
A	3.3.1 Fehlererkennung	Nein
A	3.3.2 Beschriftungen (Labels) oder Anweisungen	Nein
AA	3.3.3 Fehlerempfehlung	Nein
Richtlinie 4.1 Kompatibel		
A	4.1.2 Name, Rolle, Wert	Nein

Die typischsten Semantik Fehler 1

Keine oder zuwenig Überschriftenstruktur

- keine semantischen Überschriften
- vorallem keine <h1> als Inhaltsüberschrift
- oder mit Styling oder statt Zwischenüberschriften
- Falsche Überschriftenhierarchie (bei zusammengehörigen Teilen, z.B. Teaser mit <h4>Datum und <h3>Newsheadline statt <p>Datum, <h3>Newsheadline)
- leere Überschriftenelemente
- keine Zwischenüberschriften oder fieldset+legend bei komplexeren Formularen
- keine Überschriften, die zusammengehörige Elementgruppen, wie Teasern mit „Weiter" Links oder Shopartikeln mit Zusatzinfo, semantische Zusammengehörigkeit geben.

Die typischsten Semantik Fehler 2

Keine semantische Seitenstruktur

- Keine Listenstruktur bei Menüs
- Shopartikel nur in Konstrukten statt als Liste oder/und mit Überschriften
- keine HTML5 Strukturelemente ODER keine ARIA landmark roles, besonders fehlendes <main> Element ODER fehlende role="main"
- Fehlende semantische Labels bei Formularfeldern (kein <label> mit for und ID oder kein title-Attribut oder kein aria-label)
- Formularfehlermeldungen wie E-Mail Adresse bitte ausfüllen oder Hilfetexte sind semantisch nicht zugeordnet (z.b. als Teil des <labels> oder mit aria-describedby)
- Formular User Notifications werden eingefügt ohne semantischen Hinweis für Screenreader – keine ARIA role=alert oder kein aria-live="polite"
- Vermeiden von Tabellenstrukturen, obwohl eine Datentabelle vielleicht sinnvoll wäre (z.b. in einem Warenkorb)

Die typischsten Semantik Fehler 3

Keine korrekten semantischen Elemente

- Links statt Buttons bei Funktionen, die auf der Seite selbst ausgelöst werden
- , , Überschriften als Bedienelemente statt Links oder Buttons, ohne ARIA Auszeichnung für die nötige Semantik, ohne Tastatur- und Fokushandling
- Fehlende ARIA Auszeichnung bei Toggle Elementen (aria-expanded)
- Doppelte Zeilenumbrüche

 oder statt Absatz <p>
- <b> statt
- Line-Styling, dessen Interpretation Screenreader per default deaktiviert haben, z.B. <del> oder <strike>~~€ 160,-~~ für nicht mehr gültige Preise ohne zusätzliche Information („Statt ~~€ 160,-~~ € 140,-)

Die typischsten Semantik Fehler 4

Zuviel Überschriftenstruktur

- mehrere <h1>, viel zu viele Überschriften
- Zu viele Strukturüberschriften, die die Suchmaschinenoptimierung stören

Zuviel Semantik generell

- zuviele Listen, z.B. eine eigene UL bei jedem Hauptmenüpunkt
- Unnötig viele HTML5 Strukturelemente (<header> bei jeder Überschrift, zuviele <nav> /<aside> Elemente oder zuviele landmark roles
- HTML5 Elemente UND landmark roles (redundante Information)
- ARIA Amok – unnötig und falsch eingesetzte ARIA Attribute und Rollen
- Semantische Überfrachtung bei sehr einfachen Elementen wie z.B. einem Suchfeld mit landmark role=search, versteckter Überschrift „Suche", Label „Suche" und Placeholder „Suchbegriff eingeben" und Suche Button.

Bildnachweise:

Siegerpodest: Google Bildsuche, helantec.de

Reue: von Theodor Stundl [Public domain], via Wikimedia Commons
https://upload.wikimedia.org/wikipedia/commons/f/fe/Theodor_Stundl_-_Reue.jpg

Reue: Google Bildsuche, von Oskar Garvens

Mr Robot has some RAM: von Chris Isherwood auf flickr.com (creative commons-Lizenz, bestimmte Rechte vorbehalten: CC BY-SA 2.0)

IBM Tastatur: von http://www.flickr.com/people/moparx - http://www.flickr.com/photos/moparx/3887360487, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=30448504

Marokko Wüste: von
https://upload.wikimedia.org/wikipedia/commons/2/2d/Marokko_W%C3%BCste_02.JPG