

Wie mit den Anforderungen an Barrierefreiheit in Softwareprojekten umgehen

07.01.2020

Werner Hänggi

Accessibility Specialist

Lina Witzel

Studentin People-Oriented Computing

ADNOVUM



Agenda

1. Über uns
2. Demo
3. 3 Mythen - Busted!
4. WCAG, P028 und ARIA?
5. Frameworks
6. Minimierung des Aufwands
7. Take Aways
8. Q&A
9. Ressourcen

Über uns

Werner Hänggi



Lina Witzel

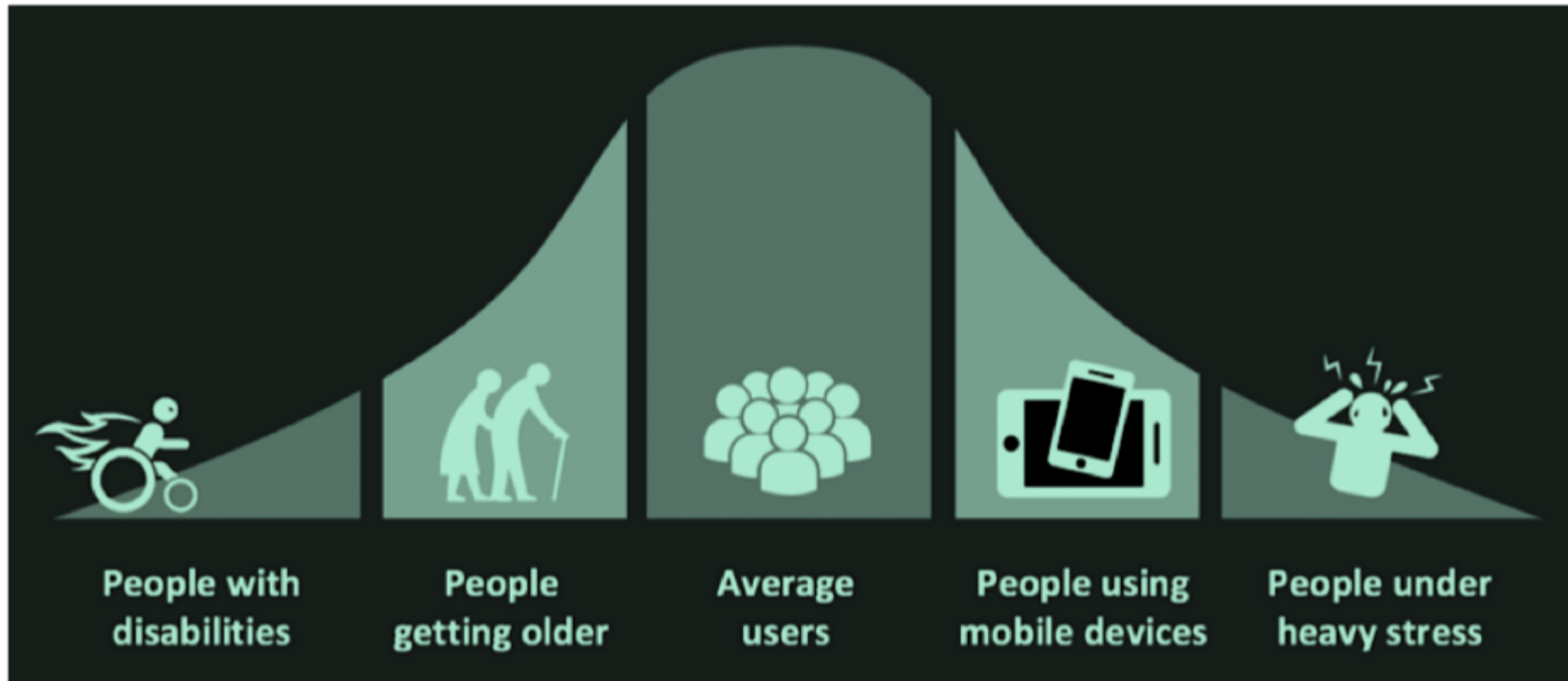


Demo

3 Mythen - Busted!

Mythos I

“Nur eine **kleine Nutzergruppe profitiert** von barrierefreien Applikationen und Webseiten.”



Quelle: <https://www.deque.com/blog/5-digital-accessibility-myths-busted/>

Mythos II

"Es benötigt **zu viel Zeit, Aufwand und Geld** eine Applikation oder Webseite barrierefrei zu entwickeln."

Start in Phase

Design

Coding

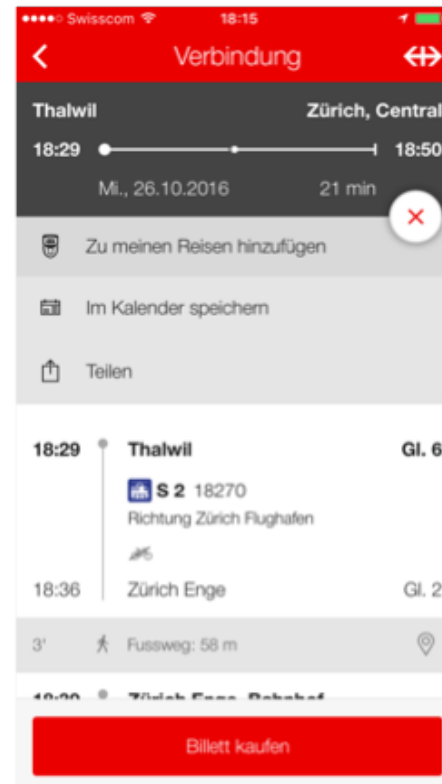
Testing

Aufwand um Fehler zu beheben



Mythos III

"Barrierefreie Webseiten und Applikationen sind
zu schlicht und hässlich."



WCAG, P028 und ARIA?

Web Content Accessibility Guidelines

4 Prinzipien

1. Wahrnehmbar
2. Bedienbar
3. Verständlich
4. Robust

12 Guidelines mit Erfolgskriterien und empfohlenen Techniken zur Erfüllung

3 Konformitätsstufen - A, AA, AAA

P028

- **Richtlinien** des Bundes
- Basierend auf dem WCAG 2.0
- **Checkliste** online verfügbar

Richtlinien & Checkliste:

https://www.isb.admin.ch/isb/de/home/ikt-vorgaben/prozesse-methoden/p028-richtlinien_bund_gestaltung_barrierefreie_internetangebote.html

Barrierefreies HTML Markup

Umfasst u.a. folgendes:

- Erfüllung des **WCAG** 2.0/2.1 o. P028
- **Validierung** des HTML Markup
- **Bedienbarkeit** der Applikation **ohne Maus** (nur Tastatur)
- Verwendung der **HTML Attribute** entsprechend ihres **Zweckes**
- Klare **Struktur** (Headings und Landmarks verwenden)



Accessible Rich Internet Applications

Darum geht es:

- Seit März 2014 ein W3C Standard
- Set von **zusätzlichen** HTML Attributen
- Im Zuge des Web 2.0 entwickelt
- Vom Browser ignoriert
- Von Screenreadern unterstützt
- Liefert zusätzliche **semantische Informationen**



Anwendung von ARIA

Die Verwendung von ARIA macht z.B. in folgenden Situationen Sinn:

- Verknüpfung von **Fehlermeldungen** mit Eingabefeldern
- **Dynamische** Änderung des Inhalts je nach Interaktion
- Auf Seiten, die sich automatisch **aktualisieren**

Mehr Anwendungsbeispiele im ADG:

<https://www.accessibility-developer-guide.com>

Frameworks

Accessibility Support in JavaScript Frameworks



Folgende Frameworks unterstützen ARIA:

- **Angular**
 - ab Version 2.0
 - nicht alle Komponenten sind barrierefrei, z.B. Modale o. Togglertips
- **React**
 - verschiedene Tools/Plugins zum Testen der Barrierefreiheit vorhanden
 - Fragmente können Semantik des HTML stören
 - title-Attribut bleibt auf allen Seiten gleich

- Vue

- sehr einfache Anwendung von ARIA möglich
- nicht alle Komponenten sind barrierefrei, z.B. Modale
- Bootstrap-Vue Library liefert umfangreiches ARIA Markup

Mobile Accessibility



Für iOS:

- **Accessibility API** vorhanden
- Accessibility Inspector in XCode nutzen
- UIKit bietet zahlreiche Werkzeuge:
 - **Labels** - ähnlich wie Alt-Attribute
 - **Traits** - beschreiben Art der Interaktion
 - **Announcement Notification** - geben Auskunft über abgeschlossene Interaktionen
 - etc.

Mobile Accessibility



Für **Android**:

- **Accessibility API** vorhanden
- Hilfreiche Tools:
 - Linter
 - Accessibility Scanner
- API bietet folgende Werkzeuge:
 - **Accessibility Attributes** - Label für UI Elemente
 - **Content Description** - zur Beschreibung von Buttons und Bildern
 - **Grouping** - um Fokusreihenfolge für den Screenreader festzulegen

Minimierung des **Aufwands**

Planung und Design



Hilfreiche Deliverables:

- Anforderungen hinsichtlich Barrierefreiheit definieren: A, AA, AAA? Zertifizierung?
- Accessibility Spezifikation erstellen

Weitere Tipps:

- Framework wählen, welches Barrierefreiheit unterstützt
- Designer sollte vertraut mit dem WCAG sein

Coding

Theoretisch **kein zusätzlicher Aufwand!**

- Workshop, falls kein Vorwissen vorhanden ist
- Accessibility Spezifikation von Anfang an berücksichtigen
- Vorsicht bei interaktiven Widgets!

Auf Hilfsmittel zurückgreifen:

<https://www.accessibility-developer-guide.com>



Testing

Automatisiertes Testing

Tools:

- Google Lighthouse
<https://developers.google.com/web/tools/lighthouse/>
- ARC Toolkit
<https://www.paciellogroup.com/toolkit/>
- etc.

Regelmässig testen und Barrieren früh beheben!



Manuelles Testing

Tools:

- Screenreader!
- User Testings
- Checklisten

Weitere Tipps:

- User Testings decken auch generelle Usability Probleme auf
- Die wichtigsten Use Cases abdecken

Take Aways

- Barrierefreiheit - Kleiner Aufwand, grosser Gewinn
- Je früher Barrierefreiheit berücksichtigt wird, desto geringer ist der Aufwand
- Den WCAG kennenlernen und in allen Phasen berücksichtigen
- Gebrauch von Hilfsmitteln machen - Frameworks, ADG, Testing Tools, etc.



Q&A

Ressourcen

ADG:

<https://www.accessibility-developer-guide.com>

Leonie Watson - Developer's Guide to Accessibility Mechanics:

<https://www.youtube.com/watch?v=qi0tY60Hd6M>

Introduction to WAI ARIA:

<https://dev.opera.com/articles/introduction-to-wai-aria/>

How to Use ARIA Effectively with HTML5:

<https://www.sitepoint.com/how-to-use-aria-effectively-with-html5/>

Angular, Accessibility, and You:

<https://blog.dinolytics.com/2018/10/25/angular-accessibility-and-you/>

IOS Accessibility Tutorial:

<https://www.scaledrone.com/blog/ios-accessibility-tutorial-tips-on-making-your-app-more-accessible/>

Android Accessibility Tutorial:

<https://www.raywenderlich.com/240-android-accessibility-tutorial-getting-started>

Kontakt

Werner Hänggi

Accessibility Specialist

werner.hänggi@adnovum.ch

Lina Witzel

Studentin People-Oriented Computing

lina.witzel@uzh.ch