H. Fischer, A. Endmann & M. Krökel (Hrsg.): Mensch und Computer 2015 – Usability Professionals, Berlin: De Gruyter, S. 184- 191.

Ein Accessibility (a11y)-Konzept für Google Calendar

Schritt für Schritt zum *Inclusive Design*

Michael „Mitch“ Hatscher
Google Switzerland GmbH
Brandschenkestr. 110
8002 Zürich / Schweiz
mitchhatscher@google.com

Astrid Weber
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043 / USA
aweber@google.com

Abstract

Wie entwickelt man ein *Inclusive Design*-Konzept für ein Produkt mit so hoher Informationsdichte, starker Visualität und diverser NutzerInnenpopulation wie Google Calendar? Neben der Beschreibung der Vorgehensweise erklären die Autoren, welche besonderen Herausforderungen sie meistern mussten und welche Erkenntnisse sie hierdurch gewannen. Der Artikel schließt mit *Best Practice*-Empfehlungen und Design-Implikationen, die über Google-Produkte hinaus relevant und interessant sein dürften.

Keywords

Accessibility, Barrierefreiheit, Mobile, Android, iPhone, Screen Reader

Ausgangssituation

UX-Produkt-Team: Google Calendar

Im Herbst 2013 erhöhte sich innerhalb von Google die Aufmerksamkeit in Sachen Accessibility; damit wurden Engineering- und Produktmanagement-Kapazitäten für diese wichtige und herausfordernde Aufgabe frei. Zu diesem Zeitpunkt bestand im gesamten User Experience-Team für Google Calendar kaum Vorwissen über Accessibility, abgesehen von einigen basalen Grundlagen, geschweige denn von Implikationen für die Implementierung.

Zentrales Accessibility-Engineering-Team

Das zentrale Accessibility-Engineering-Team unterstützt Produktteams darin, ihre Accessibility-Strategie auszuarbeiten. Zentrale Fragen lauten beispielsweise: Was ist für einen blinden Menschen ein tatsächlich nützlicher Kalender? Welche Farb- und Form-Prinzipien ermöglichen es einem farbenblinden Nutzer, Googles Produkte mühelos zu nutzen?

Mit Hilfe von User Research strebt die Accessibility Community innerhalb von Google an, in all ihren Design-Entscheidungen und -Definitionen einer Philosophie des *Inclusive Design* zu folgen.

Vorgehen

Wir hatten den Auftrag, innerhalb sehr kurzer Zeit Barrierefreiheit als zentralen Designaspekt in die App zu integrieren. Aus einem früheren Projekt bestand bereits ein enger und vertrauensvoller Kontakt zwischen dem Calendar UX-Team und Astrid, die als User Researcherin im zentralen Accessiblity-Team arbeitet. Über mehrere Wochen intensiver Zusammenarbeit am Standort Zürich erstellten wir gemeinsam ein Grundkonzept der Accessibility für Google Calendar. Wir begannen mit einer praktischen und eindrücklichen Einführung ins Thema Accessibility: Wir luden mehrere blinde NutzerInnen von außerhalb zu uns ins Büro ein, um uns zeigen zu lassen, wie Menschen mit visuellen Einschränkungen digitale Produkte nutzen. Anschließend durchliefen wir die folgenden Schritte:

- Auswahl und Priorisierung zentraler Use Cases, die wir barrierefrei gestalten wollten, beispielsweise: Einen Überblick über die anstehenden Ereignisse vermitteln; einen neuen Kalendereintrag erstellen; einen Zahnarzttermin finden, der für die nächsten Wochen vereinbart wurde; zu einem Datum navigieren
- Brainstorming über die ideale Accessible User Experience – z.B.: wie viel Sprachausgabe ist angemessen? Wie kann man Navigationsvorgänge beschleunigen, beispielsweise mit Hilfe von Sprungmarken?
- Definition von Storylines und User Journeys, die es uns ermöglichten, die Nutzungserfahrung besser nachzuvollziehen und zu optimieren
- Entwicklung eines Ansatzes für Low-Fidelity Prototyping für Accessibility, um erste Konzepte ohne Engineering-Vorarbeit testen zu können: Nachdem wir für alle zentralen Use Cases eine Vorstellung von einer geeigneten barrierefreien UX entwickelt hatten, definierten wir diese Flows in Form von Geschichten. Mit verteilten Rollen lasen wir dann vor, was der Nutzer tut und was die Assistive Technology (AT, meist: der Screen Reader) daraufhin ausgibt. So konnten wir ein AT-gestütztes Konzept simulieren, ohne dass Code geschrieben werden musste, und lernten in kurzer Zeit sehr viel, v.a. was die Menge des ausgegebenen Materials und die System- und Ausgabe-Steuerbarkeit angeht. Das Konzept funktioniert sogar für Remote User Testing via Telefon und Videokonferenz

- Iterative Definition des Basic Accessibility-Konzepts mithilfe des beschriebenen Prototyping-Ansatzes in zahlreichen Interviews mit blinden KollegInnen
- Erstellen einer Functional Spec für die barrierefreie UX für iPhone und intensive Diskussion mit dem zentralen Accessibility Team. Diese Functional Spec ergänzte die vorhandene Produktdefinition für die allgemeine Nutzungserfahrung
- (Teilweise) Implementierung des optimierten Accessibility-Konzept im nächsten iPhone-Prototypen und iteratives Testen mit externen NutzerInnen mit Einschränkungen
- Aktualisieren der Spec für iPhone, Erstellen der Spec für Android auf Grundlage der Spec für iPhone und Übergabe ans Engineering

In späteren Phasen des Projektes fand die Weiterentwicklung der barrierefreien UX für Google Calendar in Buganizer, unserem Bug-Tracking-Tool, in einzelnen Bugreports und Lösungen statt. Dies geschah in Form von spezifischen Designlösungen für konkrete Probleme und Accessibility Bugs, die unser Quality Assurance (QA)-Team aufspürte.

Herausforderungen und Erkenntnisse

Design-Prozess und -Deliverables

Der Gestaltungsprozess der barrierefreien Nutzungserfahrung für Calendar funktionierte grundsätzlich sehr ähnlich wie der Gestaltungsprozess für die allgemeine UX: Ausgehend von Nutzer-Personas und -szenarien definierten wir Design-Skizzen in Form kurzer Dialog-Abschnitte, erstellten Prototypen in Form ausgearbeiteter Skripte, testeten die Prototypen mit echten NutzerInnen und entwickelten die Design-Ansätze über mehrere Iterationen mit zunehmend höherer Fidelity weiter, bis wir abschließend die Design-Lösung in einer Spezifikation dokumentierten, die Ingenieure bei der Implementierung unterstützten und auf Design-Bugs der Qualitätssicherung bzw. des Testing Teams reagierten oder unsererseits Bugs öffneten.

Allerdings waren die Problemlösungen i.d.R. eben keine vorrangig visuellen Lösungen, und als Deliverables kamen nicht Wireframes oder annotierte Mock-ups zum Einsatz. Stattdessen lagen die Lösungen in Kombinationen von visuellen, auditiven, haptischen Reizen und zusätzlichen Definitionen des Systemverhaltens (z.B. wenn das System, wie bei iOS, für die NutzerInnen den Fokus setzt). Die Lösungen mussten in ausführlichen Spezifikationsdokumenten niedergelegt werden und wiesen oft eine Kleinteiligkeit auf, wie wir sie im UX-Design vorher nicht angetroffen hatten. Auch war der Übergang zwischen Spezifikations- und Build-Phase sehr fließend: Wohl auch, weil die Functional Specs anfangs nicht detailliert genug waren, fand letztlich ein guter Teil der Design-Arbeit im Zusammenspiel mit dem Engineering und der QA auf den fast fertigen Builds statt.

Eine weitere Herausforderung bestand darin, dass niemand im Product Management- oder auch im restlichen Calendar UX-Team Erfahrung mit Accessibility hatte und somit UX

Reviews, wie wir sie für alle anderen Teile des Produkts durchführten, schwer zu realisieren waren. Hier waren wir oft auf Rat und Empfehlungen der Google Accessibility Community angewiesen.

Dokumentation und Best Practice

Es liegt eine Fülle an technischer Dokumentation vor, wie sich Barrierefreiheit aus Engineering-Sicht realisieren lässt; aber es gibt sehr wenig *best practices* oder Beispiele für wirklich gelungenes barrierefreies Design. Auch Google stand bei diesem Thema am Anfang. Von Nutzern mit Einschränkungen hörten wir mehrfach Aussagen wie diese: „[Wir] sind für alles dankbar, was die Tools überhaupt nutzbar macht“ (siehe auch Fast Company, 2015).

Sowohl für Android als auch für iOS hatten wir Mühe, Beschreibungen der eigentlichen barrierefreien User Experience oder des angestrebten Verhaltens zu finden. Es liegt sehr viel technische Dokumentation zur Implementierung vor (Apple, 2015; Google, 2015a). Daher mussten wir sehr viel *reverse engineering* durchführen: Stundenlang testeten wir mit geschlossenen Augen und Kopfhörer eine Vielzahl von Apps auf ihr Accessibility-Verhalten hin und versuchten, Verhaltensmuster und -standards zu identifizieren, die auch für Google Calendar sinnvoll sein könnten.

Spezifikation

In unseren Functional Specs definierten wir die Grundzüge der barrierefreien Nutzungserfahrung. Viele Detailprobleme erkannten wir allerdings erst später, als die Apps bereits gebaut wurden oder waren und wir Bugs vom QA-Team oder auch von NutzerInnen gemeldet bekamen. Viele der Lösungen zu diesen Problemen sind aktuell in Buganizer in einzelnen Bugreports beschrieben. Ebenso haben jegliche Feature- und Design-Änderungen und -Neuentwicklungen Implikationen für die Accessible User Experience. Es ist wichtig, diese im Blick zu behalten und jede Design-Änderung ebenfalls in der Definition der barrierefreien Nutzungserfahrung nachzuziehen.

Organisation

Starke Unterstützung und ein klares Mandat vom Top-Management zu bekommen, war sehr hilfreich, um Engineering- und Produktmanagement-Ressourcen zu erhalten und Accessibility gegenüber anderen Anforderungen zu priorisieren. Allerdings erlebten wir auch als Herausforderung, die gerade für Barrierefreiheit dringend erforderliche Konsistenz über Produkte hinweg zu erreichen, wenn viele Produktteams gleichzeitig auf ein ehrgeiziges Ziel losstürmen.

Kontinuierliches Testen ist unerlässlich. Von großem Vorteil für das Calendar Team war es, dass einige Mitglieder des QA-Teams sich für einige Monate komplett auf das Testen von Accessibility konzentrieren konnten. Ein internes Google Accessibility Rating (GAR) half bei der Priorisierung der gefundenen Probleme für das Produktmanagement. Jetzt, wo ein

gewisser (hoher) Standard erreicht worden ist, stellt sich allerdings die Frage, wie Accessibility Bugs gegenüber den anderen Bugs oder gar der Entwicklung von neuen Features zu gewichten sind.

Es stellte sich ebenfalls als sehr hilfreich heraus, ein Team von Accessibility-Experten innerhalb der Organisation zu haben und auf einen Pool von internen und externen *trusted testers* mit Einschränkungen zurückgreifen zu können, um zeitnah Ideen und Konzepte diskutieren und testen zu können.

Spezifische vs. generische Lösung und Implementation

Unsere internen Accessibility-Experten rieten uns davon ab, Accessibility-spezifisches Design zu implementieren. Der Code werde oft später nicht gewartet und könne „spröde" werden, so dass die ursprünglich optimierte Erfahrung dann häufig nicht mehr funktioniere. Für Calendar sind wir allerdings an mehreren Stellen von dieser Empfehlung abgewichen und haben für einzelne Elemente eine besondere Experience für Barrierefreiheit implementiert. Beispielsweise bewegen wir den „Floating Action Button" auf dem iPhone von der rechten unteren Bildschirmecke in die obere horizontale Action Bar, wenn VoiceOver eingeschaltet ist – der eingebaute Screen Reader in iOS, der auch zusätzliche Gesten zur Steuerung freischaltet. Durch Designentscheidungen wie diese versuchen wir, jeder Gruppe von NutzerInnen die bestmögliche Produktinteraktion zu ermöglichen und alle unsere NutzerInnen dabei zu unterstützen, zu jedem Zeitpunkt den Überblick über ihre täglichen Termine und Aufgaben zu haben.

Idealerweise bringen UI-Komponenten in der Plattform ihre Accessibility-Eigenschaften bereits mit, so dass ein hohes Maß an Konsistenz über das gesamte Design-System hergestellt werden kann und Änderungen über das System hinweg ausgerollt werden können. In der Entwicklung von Google Calendar war dies nur eingeschränkt möglich: Plattformübergreifend befanden sich die UI-Komponenten von *Material Design*, Googles neuer Design-Sprache im damals gerade angekündigten Android *Lollipop*-Release, zu diesem Zeitpunkt noch in der Konzeptionsphase (Google, 2015b).

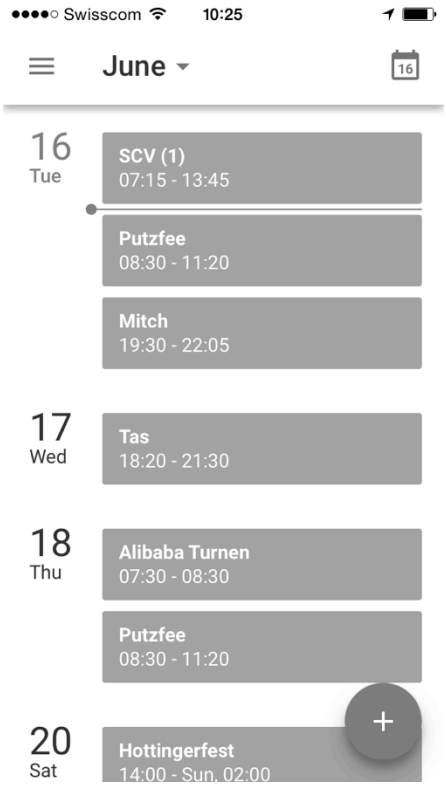

Abbildung 1: Google Calendar iPhone App ohne VoiceOver: der „Floating Action Button" sitzt in der rechten unteren Bildschirmecke

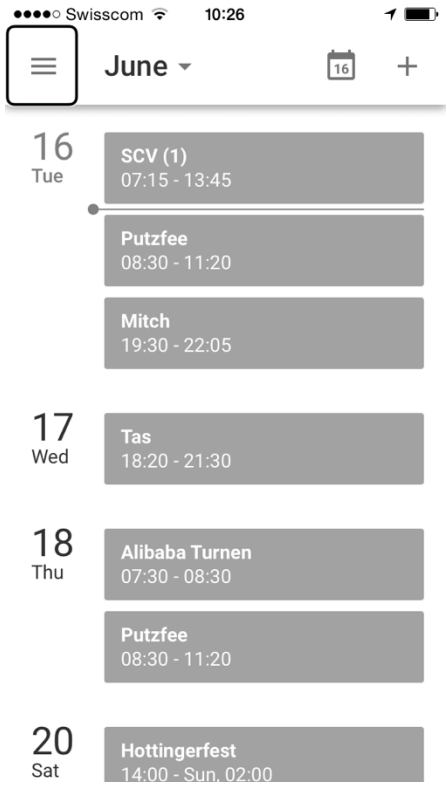

Abbildung 2: Die iPhone App mit VoiceOver – der „+"-Button wird in die Action Bar verschoben (rechte obere Ecke)

Unterschiedliche Plattformen weisen unterschiedliche Accessibility-Standards auf. Diese Unterschiede wirken sich auf die Lösungen aus und erfordern die Erstellung spezifischer Spezifikationen für Plattform (Android vs. iOS vs. Web) und Gerätetyp (Handy vs. Tablet vs. Desktop) (Apple, 2015; Google, 2015a; O Connor, 2012). Einige Beispiele für diese Unterschiede:

- Rotor (iOS) vs. Local / Global Context Menu (Android): iOS-NutzerInnen können mittels einer Zwei-Finger-Drehbewegung ein Menü aufrufen, in dem sie beispielsweise die Granularität der VoiceOver-Ausgabe (Buchstabe vs. Wort vs. Absatz) oder Fokus-Sprungweiten definieren. In Android erlauben Local und Global Context Menu ähnliche, aber nicht identische Aktionen
- Fokus-Behandlung: iOS setzt den Fokus automatisch auf das erste Element einer jeden Bildschirmseite, so dass immer ein Element den Fokus hat. Android kennt kein automatisches Setzen des Fokus

- Gesten: iOS verwendet drei-Finger-Wischbewegungen, Android zwei-Finger-Wischbewegungen für weite Sprünge
- Tastaturkürzel: Dies betrifft v.a. die unterschiedlichen Screen Reader-Systeme für den Desktop, die jeweils unterschiedliche Tastaturkürzel verwenden; aber auch Tastaturunterstützung für Tablets

Barrierefreiheit für native Apps ist einfacher zu handhaben als fürs Web, weil die Accessibility-Technologien ins jeweilige mobile OS integriert sind und eine Anwendung daher weiß, ob die NutzerInnen eine unterstützende Technologie verwenden. Touch-Interaktion stellt allerdings eine besondere Herausforderung dar. Dagegen "weiß" die Web-Anwendung nicht, dass sie z.B. mit einem Screen Reader genutzt wird. In diesem Fall können wir keine spezifischen Accessibility Experiences definieren, sondern müssen die barrierefreie Nutzungserfahrung grundsätzlich in das System hineingestalten (O Connor, 2012; Pickering, 2014; Rucker et al., 2006).

Diejenigen Nutzer, von denen wir wissen, dass sie eine spezifische unterstützende Technologie verwenden, können leichter bedient werden, da wir ihre Bedürfnisse kennen und so das Design anpassen können (z.B. mit speziellen Modi, High Contrast etc.). Herausfordernder ist es, für Nutzer zu gestalten, die eine Einschränkung mitbringen, aber keine unterstützende Technologie verwenden: z.B. Farbfehlsichtigkeit oder eine mildere Form einer Sehbehinderung. Hierzu zählen NutzerInnen mit nicht korrigierbaren Hornhautveränderungen und ältere NutzerInnen. Die unterschiedlichen Ansprüche und Präferenzen in Bezug auf Kontrast und Farbkonzept unterschiedlicher Gruppen von NutzerInnen stellen eine kontinuierliche Herausforderung ans visuelle Design dar, da sich die Frage stellt: Was ist ein universell funktionierendes, attraktives Design, das die Usability- und Accessibility-Bedürfnisse aller Nutzer berücksichtigt?

Fazit

Die folgenden zentralen Erkenntnisse ergaben sich für uns aus diesem Projekt:

- Accessibility muss – wie Usability – von Beginn an Teil der Produktdefinition sein, weil Accessibility einige spezifische Anforderungen beinhaltet und bei der Schätzung des Entwicklungsaufwandes berücksichtigt werden muss
- Accessibility und Usability schließen sich nicht aus, sie ergänzen sich – gute Usability ist eine wichtige Grundlage für Accessibility. Als Designer muss man allerdings jeweils entscheiden, ob man einen Accessibility Layer auf eine bestehende Nutzungserfahrung aufsetzt oder eine neue, alternative barrierefreie Erfahrung zusätzlich zur allgemeinen UX definiert.
- Es gibt interessante Spannungsfelder zwischen Accessibility-Anforderungen und visuellem Design, z.B. bei Kontrastverhältnissen zwischen Schrift- und Hintergrundfarbe

- So, wie man die allgemeine Nutzungserfahrung in Testverfahren einschätzen kann, kann man auch die barrierefreie Nutzungserfahrung simulieren, bereits früh Nutzerfeedback sammeln und über das Design iterieren
- Die Gestaltung der barrierefreien Nutzungserfahrung erwies sich als extrem kleinteilig und erforderte ein hohes Maß an Abstraktionsvermögen. Während der Gestaltungsprozess ähnlich ablief wie bei herkömmlicher UX-Arbeit, waren die Problemlösungen i.d.R. eben keine vorrangig visuellen Lösungen, sondern eine Kombination aus visuellen, auditiven, haptischen Reizen und zusätzlichen Definitionen des Systemverhaltens (z.B. wenn das System, wie bei iOS, für die NutzerInnen den Fokus setzt)

Abschließend möchten wir noch festhalten: *Inclusive Design* bedeutet nicht nur: Design für Menschen mit Einschränkungen in der visuellen Wahrnehmung. Mehrfachbehinderungen (wie z.B. blind / gehörlos) stellen besondere Herausforderungen dar; starke motorische oder gar kognitive Einschränkungen noch deutlich mehr. Hier stehen wir noch ganz am Anfang.

Literatur

Apple Inc. (2015). *Accessibility for Developers: iOS. https*://developer.apple.com/accessibility/ios/ Letzter Zugriff: 22.6.2015

Cunningham, K. (2012). *Accessibility Handbook*. Sebastopol, CA: O'Reilly Media.

FastCompany (2015): *Google's Guide To Designing With Empathy.* http://www.fastcodesign.com/3047545/googles-guide-to-designing-with-empathy Letzter Zugriff: 22.6.2015

Google Inc. (2015a). *Android Developers: Accessibility.* https://developer.android.com/guide/topics/ui/accessibility/index.html Letzter Zugriff: 22.6.2015

Google Inc. (2015b). *Material Design: Accessibility.* https://www.google.com/design/spec/usability/accessibility.html# Letzter Zugriff: 22.6.2015

Horton, S. & Quesenbery, W. (2014). *A Web for Everyone: Designing Accessible User Experiences*. Brooklyn, NY: Rosenfeld Media.

O Connor, J. (2012). *Pro HTML5 Accessibility*. New York, NY: Apress Media.

Parker, T., Jehl, S., Wachs, M.C., & Toland, P. (2010). *Designing with Progressive Enhancement: Building the Web that Works for Everyone*. San Francisco, CA: New Riders.

Pickering, H. (2014). *Apps For All: Coding Accessible Web Applications*. Freiburg: Smashing Magazine GmbH.

Rutter, R., Lauke, P.H., Waddell, C., et al. (2006). *Web Accessibility: Web Standards and Regulatory Compliance.* New York, NY: Apress Media.