iOS ACCESSIBILITY HANDBOOK

iOS Accessibility Handbook

A clear, concise and complete reference.

Luis Abreu

This book is for sale at http://leanpub.com/iosaccessibility

This version was published on 2016-03-11

Tweet This Book!

Please help Luis Abreu by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just bought ”iOS Accessibility Handbook: A clear, concise and complete reference” and supported @watsi

The suggested hashtag for this book is #iosa11y.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#iosa11y

Contents

Introduction

Thank you for showing an interest in making your product available to all users, regardless of their disabilities.

The 2011 WHO World Report on Disability estimates more than 15 percent (1 Billion) of the world's population (7 Billion) live with some kind of disability. Contrary to common perception, not all disabilities are severe (e.g. blindness, paraplegia, etc).

The 2010 WHO Global Data on Visual Impairments estimates that an average 30 percent of the population in developed countries suffer from some kind of visual impairment. This includes even light Astigmatism (short/long-sightedness), which in combination with low text contrast may cause uncomfortable headaches.

Insufficient text contrast, bad ergonomics, or high cognitive load impact everyone, it's just that people with severe impairments feel it most, as for them, these issues don't just slow them down, they stop them from achieving their goals.

Everyone can experience External Impairments, they introduce or aggravate existing impairments.

Examples of External Impairments include:

- Bright sunlight may impair color contrast, resulting in temporary color blindness and sight reduction;
- Driving or even walking impair cognitive and motor abilities, resulting in a temporary learning or physical impairment;
- Noisy environments impair hearing, resulting in temporary hearing loss;
- Flashing lights impair vision, causing distraction or epileptic seizures.

Accessibility adds robustness, improving experience for both temporarily and permanently impaired users.

> "Disability need not be an obstacle to success.", (Professor Stephen W. Hawking, 2011 World Report on Disability)

Stephen Hawking is proof that given the right conditions and Assistive Technology, even severe disabilities pose no barrier to living a happy, successful life.

I've put myself in the shoes of someone who relies on Assistive Technology, it didn't take long to encounter a barrier I could not surpass.

We have the moral duty to remove the barriers to participation, to invest time and effort into making tools appropriately accessible to everyone, unlocking their full potential.

“Unfortunately, unseen customers are considered nonexistent.”

The UN Factsheet on Persons with Disabilities estimates that 75% of the FTSE 100 companies in the UK do not meet basic levels of accessibility, thus missing out on more than £96 ($147) million in revenue. Unfortunately, unseen customers are considered nonexistent.

If not for moral values, do it for financial values, either way everyone benefits.

iOS Assistive Technologies

iOS provides free, built-in Assistive Technologies to make apps and websites accessible for people with impairments in the following groups: **Vision**; **Hearing**; **Physical and Motor Skills**; **Learning and Literacy**.

The most popular Assistive Technology is VoiceOver. It enables understanding of, and interaction with on-screen content for vision-impaired users — here's how to enable VoiceOver and give it a try.

Input and Output peripherals (e.g. Switches and Braille Displays respectively), are automatically handled by iOS.

For detailed information, visit Apple's Accessibility Website.

Low Effort—High Reward

iOS Accessibility has been designed to be Low Effort—High Reward.

Low Effort

By design, iOS apps have enough Accessibility Semantics to be **80% accessible**. Reaching **90% is easy** (mostly labeling elements), and **100% is usually trivial**, even for custom elements and gestures.

> ⚠ **Accessibility isn't a checkbox.** Reaching 100% is greatly appreciated, however, if you can, I'd recommend you create something delightfully accessible (more on that later).

High Reward

Basic Accessibility Semantics enable a wide variety of aids, from Visual (e.g. VoiceOver) to Physical and Motor (e.g. Switch Control). Accessibility Semantics are also the cornerstone of development best practices such as Automated Testing, making Accessibility Conformance a by-product of a robust application.

Creativity

Accessibility on iOS does not limit design possibilities or control over the final interface.

Custom elements and interactions can painlessly be made as accessible as built-in Buttons/Labels/etc (more on that later).

While users control the status of features such as Invert Colors [1] or Mono Audio [2], app creators can and should[3] define how their app behaves when Assistive Technologies are enabled (see documentation and examples for: Accessibility Accommodations, and other Technical Reference topics).

[1]Invert Colors: inverting screen colors helps people who are sensitive to brightness, color blind, or have low vision.

[2]Mono Audio: Stereo audio sometimes contains distinct tracks for each ear. When using headphones, people who are deaf or hard to hear on one ear would miss the audio track for the impaired ear. Mono Audio delivers both tracks to the non-impaired ear so that no information is lost.

[3]Assistive Technologies such as “Reduced Motion“ or “Reduced Transparency” help people with certain impairments, it is recommended to tone down animations and transparency when these settings have been enabled by the user.

Guidelines

Summary

“There was no such document for iOS.”

For the Web, WCAG (Web Content Accessibility Guidelines) is the industry-standard accessibility reference.

WCAG results from the effort of more than 100 accessibility professionals over multiple years, offering solid recommendations, straightforward guideline classification ranging from Single-A (essential) to Triple-A (ideal), and is the adopted legislation by most governments around the world.

There’s no such document for iOS.

Apple offers the best and cheapest (free) assistive technologies, but documentation is either purely technical (#1), in hour-long video form (#2, #3), or consumer-oriented (#4).

This book aims to fill that gap, the guidelines here outlined provide an **actionable** and **easy-to-understand reference** for designers, developers or anyone wanting to understand more about iOS Accessibility.

The four main WCAG principles have been kept (listed below), while references to technologies (e.g. HTML, User-Agent) or input methods (e.g. Keyboard, Mouse) have been removed/replaced with iOS equivalents to ensure validity and familiarity.

The WCAG Accessibility Conformance Levels have been kept, supporting all Single-A Guidelines is required for claiming your app is accessible.

It is understood that fully satisfying higher conformance levels (Double/Triple-A) may not always be possible due to the nature of certain apps (i.e. a drawing app cannot be made fully keyboard-accessible, or a live-streaming app cannot offer accurate real-time captions).

The Accessibility Conformance chapter provides more information on this topic.

Principles

1. Perceivable
2. Operable
3. Understandable
4. Robust

1. Perceivable

"Information can't be invisible to all of their senses."

1.1 Text Alternatives

Provide text alternatives for any non-text content.

1.1.1 Non-text Content (Level A)

Guideline

Describe non-text content using Accessibility Semantics so Assistive Technologies can help users perceive (and interact with) content.

Understanding

Assistive Technologies inform and help users interact with your app. For this to happen, your app must be described to Assistive Technologies through Accessibility Semantics.

By understanding your app, Assistive Technologies provide features that allow users to skim content without even seeing the screen, navigate by tilting their heads, reading and writing with a Braille keyboard, and much more without developer having explicitly add support for these and future features.

Examples of non-text content include: Image, Video, Audio, etc; controls such as Sliders, Buttons, Switches, etc; CAPTCHAS, Decoration, Visual Formatting, and Invisible Elements.

Text is an elementary Accessibility Semantic, assigned to elements using an Accessibility Label, it helps the Visual-impaired to at least perceive screen contents through Text-to-Speech.

Solving

Consider and plan how non-text content will be described to users (e.g. Speech) and understood by Assistive Technologies. All non-text content should be described using one or more of the following Accessibility Semantics: Label, Traits, Hint, Value, and Visibility.

Examples

Example #1: An application where the description for a Video element fails to convey its nature or metadata such as duration, is seen as a failure to meet this guideline.

Example #2: An application, such as iOS Photos, where the description for a Video element successfully conveys the element's nature (Video), metadata useful for element distinction (e.g. duration, creation time), and appropriate Accessibility Semantics such as a Trait identifying the element as interactive (e.g. Button), and a Hint informing the video will be played back upon button activation (double-tapping it). In simpler terms, this application displays videos as buttons with a thumbnail for background, sets their Label to the video's duration and creation date, and Hint with a simple string such as "double-tap to play".

1.2 Time-based Media

Provide alternatives for time-based media.

1.2.1 Audio-only and Video-only (Prerecorded) (Level A)

Guideline

Provide text alternatives containing equivalent information to original prerecorded audio-only and video-only media, so users can still perceive content despite their impairments.

Understanding

Audio-only, video-only, or any content form presented over time, require Hearing, Vision, or Learning abilities that users may not possess.

Assistive Technologies allow users to select alternatives that convey equivalent information through other senses, at a more comfortable pace, or through reading aids.

Solving

Provide Audio Descriptions for the Vision-impaired, Closed Captions for Hearing-impaired, and Transcripts for Learning or Vision & Hearing-impaired.

Refer to Apple's HTTP Live Streaming Documentation for more information, and the HTTP Live Streaming overview for HLS streaming examples and more resources.

Examples

Example #1: An application, where a series of steps are demonstrated in prerecorded video-only media, should provide an audio or text representation of the same content.

Example #2: An application containing prerecorded audio-only media (e.g. Podcasts), should provide a text representation (i.e. transcript).

1.2.2 Captions (Prerecorded) (Level A)

Guideline

Provide Closed Captions for all prerecorded Audio-only, or Audio media synchronized with other formats such as Video, or time-based interactive components. Unless the media is clearly labeled as an alternative for existing text content.

Understanding

Audio media requires Hearing, or Learning abilities that users may not possess.

Solving

Provide Closed Captions so users can perceive content despite their impairments. Alternatively, solutions to Guideline 1.2.3, namely Transcripts, will also ensure content can be perceived despite Hearing or Learning impairments.

Refer to Apple's HTTP Live Streaming Documentation for more information, and the HTTP Live Streaming overview (HLS) for HLS streaming examples and more resources.

Examples

Example #1: An app serving prerecorded audio content (e.g. Podcast) which includes Closed Captions for the selected media.

Example #2: An app serving prerecorded Video content synchronized with an Audio track (e.g. Netflix), which includes Closed Captions or the selected media.

1.2.3 Audio Description or Media Alternative (Prerecorded) (Level A)

Guideline

Provide an audio Description of prerecorded video-only media, an alternative for video-synchronized media (i.e. video + audio) or time-based interactive components (i.e. animation). Unless the media is already a text alternative.

Understanding

Video-only, or any content form presented over time (i.e. animated elements), can only be perceived if Vision is available, or if an individual's Learning & Literacy abilities are sufficient to keep up with the rate of information.

Solving

Provide Audio Descriptions for the Vision-impaired, Closed Captions for Hearing-impaired, and Transcripts for Learning or Vision & Hearing-impaired.

Refer to Apple's HTTP Live Streaming Documentation for more information on how to add audio and video alternatives to media, and the HTTP Live Streaming overview (HLS) for HLS streaming examples and more resources.

No work is necessary if the same information is already conveyed through text, as this provides the use with at least one way to access the information.

Examples

Example #1: An application where the evolution of a cell over a period of time is represented exclusively as video, should also provide an audio description for Vision-impaired users.

Example #2: An application such as Netflix containing prerecorded video media (I.e. Movies, Series, Documentaries), where in addition to Closed Captions, an alternative Audio Description track containing not only the original audio, but additional voiceover describing information contained in the video (e.g. “a woman sits by the window”).

Example #3: An educational app containing an interactive animation of a working engine, should also convey the lifecycle through text, which not only can be consumed by Vision-impaired users, but at a pace that’s comfortable to the user.

Conformance

For the purpose of Accessibility Conformance, Web Accessibility Guidelines 2.0 (WCAG 2.0) have been adopted or adapted by the vast majority of governments, including Europe, Australia, New Zealand, or Israel.

In the United States, Section 508 of the Rehabilitation Act of 1973, a subset of WCAG 2.0, is used as the official legislation.

Accessibility Conformance makes business sense, as it increases your user base, and is mandatory for products used by Government entities, NGOs, Education, and industries such as Travel. This will change depending on country, and have in mind lawsuits against inaccessible products are frequent.

The guidelines included in this book are based on WCAG 2.0, while references to web and desktop have been replaced with their iOS equivalents, the original goals have been respected to ensure Accessibility Conformance.

Accessibility Conformance validation is a manual process as there are currently no automated tools for this purpose.

To aid you with Accessibility Conformance validation I've included in this book a section titled "Accessibility Semantics Audit" (ASA).

Additionally, if you're targeting the US market, the Voluntary Product Accessibility Template® (VPAT®) will help you document your product's conformance with Section 508. Within VPAT, the section you're looking is "Section 1194.21 Software Applications and Operating Systems".

IMPORTANT: No tool or process will provide a complete guarantee your product is Accessible to everyone, even if it conforms to all Accessibility Guidelines, due to the unpredictable nature of impairments, users may still encounter challenges when using your product. Conversely, a product that does not comply to some Accessibility Guidelines may still be Accessible as the supported guidelines may account for all the needs of a specific user.

Accessibility Semantics Audit

The Accessibility Semantics Audit (ASA) helps quickly determine a basic level of Accessibility Conformance, below Single-A, for individual user interface elements.

While Single-A is required to conform with Government legislation, ASA-level conformance still provides significant value at a very low cost.

The value of ASA lies in the effort to reward ratio. First, the simple querying of every user interface element with 6 straightforward questions (e.g. Where am I?), results in a preliminary Accessibility Compliance status.

Second, applications created with built-in user interface components leave few questions unanswered, filling in the gaps is not only easy, but also goes a long way to meeting many of the Accessibility Guidelines which only require the presence of Accessibility Semantics, and facilitates Automated Testing.

The following is a list of questions, along with their description and answer information.

- Is it relevant for Accessibility?
- Is it succinctly and accurately identified?
- Are Behavior, Impact, Type and State perceivable?
- Is its value accurate at any point in time?
- Are outcome and interaction understood?
- Where is it?

Technical Reference

Intro

This section includes main and less known accessibility-related methods, constants, notifications and helper functions from the iOS SDK (9.0).

This documentation has been made easier to understand by reducing the amount of abstraction, jargon, verbosity, while still including declaration information, usage tips and examples.

Main takeaways:

- All UIViews conform to the `UIAccessibility` protocol
- Built-in iOS technologies do most of the work for you
- Accessibility APIs provide granular control, down to speech language, pitch, and interpretation.

The `UIAccessibility` prefix in methods, variables or classes has been collapsed to a single period character (".") to speed up text skimming. E.g. `.IsBoldTextEnabled` equates to `UIAccessibilityIsBoldTextEnabled`.

Table of Contents

  - .IsSwitchControlRunning
  - .IsVoiceOverRunning
- Accessibility Notifications
  - Assistive Technology Identifiers
    * .NotificationSwitchControlIdentifier
    * .NotificationVoiceOverIdentifier
  - Post
    * .AnnouncementNotification
    * .PauseAssistiveTechnologyNotification
    * .ResumeAssistiveTechnologyNotification
    * .RegisterGestureConflictWithZoom
    * .RequestGuidedAccessSession
    * .ZoomFocusChanged
  - Observe
  - Assistive Technology State Changes
    * .AnnouncementDidFinishNotification
    * .GuidedAccessStatusDidChangeNotification
    * .ShakeToUndoDidChangeNotification
    * .SpeakScreenStatusDidChangeNotification
    * .SpeakSelectionStatusDidChangeNotification
    * .SwitchControlStatusDidChangeNotification
    * .VoiceOverStatusChanged
  - Accessibility Accommodations State Changes
    * .BoldTextStatusDidChangeNotification
    * .InvertColorsStatusDidChangeNotification
    * .DarkerSystemColorsStatusDidChangeNotification
    * .GrayscaleStatusDidChangeNotification
    * .MonoAudioStatusDidChangeNotification
    * .ClosedCaptioningStatusDidChangeNotification
    * .ReduceMotionStatusDidChangeNotification
    * .ReduceTransparencyStatusDidChangeNotification
    * UIContentSizeCategoryDidChangeNotification
  - Content Changes
    * .LayoutChangedNotification
    * .ScreenChangedNotification
    * .PageScrolledNotification
- Control Events
  - .AllEvents
  - .AllTouchEvents
  - .AllEditingEvents
  - .EditingDidBegin
  - .EditingChanged
  - .EditingDidEnd
  - .EditingDidEndOnExit
  - .PrimaryActionTriggered
  - .TouchDown

  - .TouchDownRepeat
  - .TouchDragInside
  - .TouchDragOutside
  - .TouchDragEnter
  - .TouchDragExit
  - .TouchUpInside
  - .TouchUpOutside
  - .TouchCancel
  - .ValueChanged
- Customizing Speech
  - .Pitch
  - .Punctuation
  - .Language
- Helpers
  - UIAccessibilityConvertFrameToScreenCoordinates
  - UIAccessibilityConvertPathToScreenCoordinates
  - UIGuidedAccessRestrictionStateForIdentifier