Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Report

Best Practices for Creating Accessible Mobile Applications

Report no 1031
Authors Trenton Schulz
Frederik Gladhorn
Jan Arve Sæther
Date January 26, 2015
ISBN 978-82-539-0541-9

Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

The Authors

Mr. Trenton Schulz (MSc) is a senior research scientist at the Norwegian Computing Center. Trenton has years of experience with prototyping, software development and software engineering on a variety of platforms and has developed applications and libraries for both desktop and mobile environments. His master's thesis was about testing various forms of user evaluation that were designed for desktop environments on mobile phones and creating an automated tool for generating GOMS keystroke-level models. His interests include human-computer interaction, information security, open source software and development and other areas of computer science. As a research scientist at the Norwegian Computing Center, Trenton has worked on EU projects in security (GEMOM), projects from the Norwegian Research Council dealing with semantic web technologies and design of mobile phones for the elderly. He currently is leading the Norwegian Computing Center's contributions to the EU-funded uTRUSTit project that combines both e-inclusion, security, and trust in the Internet of Things.

Jan Arve Sæther and **Frederik Gladhorn** are working at The Qt Company. Together they improve Qt's accessibility offering. Their mission is to make accessibility effortless for Qt app developers. They make sure it's supported on all platforms, including iOS, Android, Windows, OS X, and Linux. In addition, they have experience in many areas, for example, input handling, localization, and user interfaces in general.

Norwegian Computing Center

Norsk Regnesentral (Norwegian Computing Center, NR) is a private, independent, non-profit foundation established in 1952. NR carries out contract research and development projects in the areas of information and communication technology and applied statistical modeling. The clients include a broad range of industrial, commercial and public service organizations in the national and international market. Our scientific and technical capabilities are further developed in cooperation with The Research Council of Norway and key customers. The results of our projects may take the form of reports, software, prototypes, and short courses. A proof of the confidence and appreciation our clients have for us is given by the fact that most of our new contracts are signed with previous customers.

Norsk Regnesentral
Norwegian Computing Center
Postboks 114, Blindern
NO-0314 Oslo, Norway

Besøksadresse
Office address
Gaustadalléen 23
NO-0373 Oslo, Norway

Telefon · telephone
(+47) 22 85 25 00
Telefaks · telefax
(+47) 22 69 76 60

Internett · internet
www.nr.no
E-post · e-mail
nr@nr.no

Title	**Best Practices for Creating Accessible Mobile Applications**
Authors	**Trenton Schulz , Frederik Gladhorn , Jan Arve Sæther**
Quality assurance	Kristin Skeide Fuglerud
Date	January 26, 2015
ISBN	978-82-539-0541-9
Publication number	1031

Contents

Abstract

This report is a list of best practices for creating accessible mobile apps. The primary focus is on the Android and iOS platforms, but most items should be applicable to other platforms that come into existence. Besides listing the best practice, we try to give some reasoning about why it is good to follow this practice, sometimes followed by examples. We also provide some information about the types of assistive technology that currently is available for mobile phones. Besides the references in the report, we include a list of references that also include information on accessible mobile apps

Keywords	mobile apps, usability, accessibility, universal design, software development, user-interface design
Target group	Everyone
Availability	Open
Project	BestApps
Project number	320530
Research field	Universal Design of ICT
Number of pages	19

Acknowledgement

This work was done as a part of the BestApps project. We are grateful for funding to the project from UNIKT, a program run by the Delta Center. The Delta Center is the Norwegian Government's National Resource Center for Participation and Accessibility, and it is a part of the Norwegian Directorate for Children, Youth and Family Affairs (BUFDIR).

1 About the BestApps Project

The goal of the BestApps project was to find the best practices for creating accessible applications, incorporate some of these items into the Qt framework so that developers can more easily make accessible apps. The other goal was to document the items that cannot be made generic, so that developers could easily look up information when they run into the problems.

The project was run in cooperation with **the Qt Company**. The Qt Company is a wholly owned subsidiary of Digia Plc., and they are responsible for all Qt activities including product development, commercial and open source licensing together with the Qt Project under the open governance model. Together with its licensing, support and services capabilities, the Qt Company operates with the mission to work closely with developers to ensure that their Qt projects are deployed on time, within budget and with a competitive advantage.

1.1 Target audience

The target audience for this report are developers and designers of mobile applications. At the time of the writing, the biggest platforms are iOS and Android, but the information here should also be applicable for applications on other platforms or websites. When specifics are necessary, they are specified using code in Qt, but there should be similar analogues for these items in all systems.

1.2 Limitations

Given the resource limitations in the project, we were only able to focus on users with vision impairment. We fully recognize that creating accessible apps does not mean only targeting people with vision impairment, there are many other people with disabilities that can benefit from using apps. We discussed with developers who worked with different types of disabilities, and looked at issue with using switches to interact with apps, but our guidelines may not cover all the issues that are encountered by people with other disabilities. Yet, following these guidelines should help in creating apps that are accessible to people with other disabilities as well.

As mentioned in the previous section, this report focuses on Android and iOS as those are the most popular mobile operating systems at the moment. We mention Windows Phone 8 in assistive technology, but this is more for completeness. We encourage developers and designers to become familiar with the different platforms to provide the best experience on them. Besides, there may be some day in the future where other platforms are dominant.

2 Know your assistive technology

This chapter explores the different types of assistive technology that exists. We will focus on the main types of assistive technology that is available for Android and iOS phones,

but there are other bits that are specific to each operating system. For example, iOS allows inverting the colors and, now, grayscale. Those interested in keeping up-to-date on changes in assistive technology are encouraged to look at the accessibility settings on each platform and become familiar with how they work.

Part of the goal of being familiar with the types of assistive technology is that it helps to know how to test the accessibility of the app. Beyond what is covered here, the Global Accessibility Reporting Initiative (2014) has a web page that links to different demonstrations of assistive technology on the different phones.

2.1 Screen readers

Screen readers are complex and change the way you interact with the device. It's a good idea to know something about them. Luckily, the screen readers are readily available for both Android and iOS instead of having to buy them. This makes it easy for developers to spend some time getting to know how to use them.

On iOS, this story is simple, it's VoiceOver. This was functionality that was introduced as part of the system on the iPhone 3GS in 2009 and has been further enhanced with each version of iOS. You still use touch to access the phone, but VoiceOver offers a different mode of interaction. It creates a cursor for navigating the different elements on the screen. By moving you finger around the screen, VoiceOver will read aloud the element the cursor is on. You can also more systematically control the cursor using specific gestures or by using a Bluetooth keyboard. Actually, VoiceOver is the starting point for many of the other aasistive technologies on iOS.

Though not part of VoiceOver, iOS also includes a feature called *Assistive Touch* (Apple, 2014). Assistive Touch allows users alternate ways to interact with the screen or press buttons. For example, if a gesture is difficult for a user (e.g., pinch or a four-finger triple tap), the user can have the gesture recorded and assistive touch can play it back when desired or controlled via one finger. At the same time, if a hardware button cannot be accessed, Assistive Touch provides another interface to them.

Android offers a built-in screen reader called TalkBack. It has evolved over time from when it was initially added in 2009 with the introduction of Android 1.6. Back at this time, it was assumed that TalkBack would be used in conjunction with physical direction keys. This could have been in the form of a trackball or physical keyboard keys. As time went on, Google removed the requirement for physical direction keys, which brought up issues for using TalkBack on pure touchscreen devices. First, there was a virtual keyboard that could be used to move the TalkBack cursor, which was eventually replaced with *Explore by Touch* in Android 4.0. Explore by Touch is a method where a user could drag a finger across the screen to hear the different elements, similar to iOS. In Android 4.1, the ability to move by using flicking gestures was added. Finally, with Android 4.2, the addition of a separate *accessibility focus* made it possible to fully remove TalkBack from the keyboard control it initially had. This version (and later versions) offer their own set of gestures as shortcuts for accessing different items.

The later versions of Android do allow you to redefine gestures for TalkBack. Though not a direct analog to iOS's Assistive Touch, it does allow users to redefine difficult gestures to something that is easier.

There are a couple of other screen reading solutions on Android, but TalkBack is the only one that gives you full access to the system and is still actively developed.

2.2 Braille displays

A Braille display usually works in concert with a screen reader and displays the text that has focus in Braille. Instead of having the text read aloud, a user can read the text in Braille at their own pace.

Bluetooth Braille displays have been supported on iOS since the introduction of VoiceOver on the platform. Braille displays have been supported since Android 4.1 by installing BrailleBack on the device.

2.3 Screen magnifiers

Android (since 4.2), iOS (since 3.0), and Windows Phone (since 8) have built-in screen magnifiers. They must first be enabled in settings, but work in similar ways. One uses a gesture (triple-tap with one finger on Android, double-tap with three fingers on iOS) to enable the zoom. One can then drag around the screen to see items different parts of the enlarged screen. It's also possible to change the zoom level to something that works better for the user.

Zooming is done by the system. Developer don't need to do anything special for zooming to work other than not using the zoom gestures for other actions in the app.

2.4 Switches

Android and iOS have support for switches, but the level of support depends on version of the operating system.

Before iOS 7, all the different switches used VoiceOver. The switches used tricks to implement scanning by steering VoiceOver. Most of the switches would register as a keyboard and send keys, but since the iOS home screen is not accessible, it was not possible to launch apps without turning on VoiceOver.

In iOS 7, Apple introduced Switch Control, which adds a lot of functionality and configuration for switches. Most of this is supported via VoiceOver. iOS adds some features for grouping controls together allowing a switch user to target a group and then drill down to a specific control. It also adds a way of scanning the screen point by point via moving horizontal and vertical bars, and delivering a gesture at the point of intersection by activating the switch. This can make something that does not have accessibility built in a bit more accessible (for example games).

While the switches are using VoiceOver information, VoiceOver is not turned on by default, so it is still possible to use standard interaction at the same time (e.g., you can use swipes and other motions). You can also simulate gestures or actions using Assistive Touch.

iOS 8 added some changes to how the items are shown on the pop-up menu and a slower speed for the horizontal and vertical bars (Ablenet Technology, 2014).

For versions of Android prior to 5.0, switch support was handled by the different switch manufacturers. For example, one manufacturer created a custom keyboard that could be used with the switch. However, since the early versions of Android encouraged keyboard control, using a switch was straight forward once the developer understood how keyboard control worked. It was sufficient that an app was accessible via the keyboard, but it required that each switch manufacturer create their own extension.

Android 5.0 introduces additional switch support to remove the need of custom solutions. It should now be possible to pair Bluetooth switches with the phone and configure how they work in Android's Switch Access Preferences. The current autoscan implementation seems to use the information that is available to TalkBack. Unfortunately, the current implementation does not group controls nor have the scanning bar function (Avila, 2014).

2.5 Text size

Later version of Android, iOS, and Windows Phone 8 allow some control of text size. For Android, there is a simple "Large Text" checkbox that will change the default text size to a larger size. From iOS 7, Apple offers a sliding control to choose how large or small standard text will be. See the next section for best practices regarding this. Windows Phone 8 follows a similar method as iOS 7.

3 Best practices for creating accessible apps

When creating accessible applications, there are many things that can be done. In this chapter we try to flag up things that can be helpful in the process. As for all best practices, there are times and places for when following a best practice causes an issue. We try to provide a good reason for each practice, so that you can make a good choice and break rules while being fully informed.

3.1 Include people with disabilities in your user research

To help highlight accessibility needs, consider including people with disabilities in your research and investigations. Do you wish to gather input from users to help inform the design of your app? Be sure to include people with disabilities in your sample. Are you planning on developing personas? Try to create some personas with disabilities (Schulz and Fuglerud, 2012).

One issue to remember is that not all disabilities are the same. Even looking at vision impairment, some users rely on screen readers, where others may not use a screen reader at all and only use screen enlargement, yet another group may eschew using built-in screen enlargement and only use a real magnifying glass. So, a solution for one person may not be helpful for another. Part of the point of including people with disabilities is not to be representative, but to create awareness for the different accessibility issues.

3.2 Use standard controls when possible

The built-in controls (for example, buttons, tables, sliders, labels, scroll areas, etc.) have a lot of hooks for accessibility. They are identified by assistive technology correctly, have built-in hints in how one should interact with it, and perform proper notification when things change. They should also be tested and work well with assistive technology.

As an app developer, You can use these controls and get basic accessibility with little effort. You should prefer these instead of creating your own controls that mimic a control since it requires not just mimicking the visual behavior, but also the accessibility behavior.

This doesn't mean that you should restrict yourself to *only* standard controls. It's possible that some interactions or way of presenting information can't be expressed using only standard controls. Create the interface that the app needs, but accessibility work will also be needed for new controls, so plan accordingly. Thankfully, all the platforms have documentation on how to make a new control accessible.

3.3 Follow the platforms' interaction patterns

Standard interaction patterns make it easier for people using assistive technology to orient themselves. They have used these patterns on other apps, so if you follow these patterns in your app, they can quickly discover the basic layout and build a logical structure of the app. This is important in finding their way in the app and what they can do.

3.4 Choose good color contrast

Choosing a good contrast for controls and labels is something that makes it easy for everyone to find a control, read text, and reduce confusion in an app. Some may argue that a text or control should have lower contrast so that something will be less visible, but if the information or control is meant to be there, you will be making it difficult or impossible for someone who has low vision.

For choosing contrast, look at the Web Content Accessibility Guidelines (WCAG). The plethora of available tools for checking contrast on web pages make it an easy place to start checking colors. The AAA guideline of 7:1 for regular text and 4.5:1 for large text is a good starting point, though one can make the contrast even higher than what is suggested there. Also consider specifying colors in hue, saturation, and lightness (HSL) instead of values of red, green, and blue (RGB); it may make it easier to lighten or darken colors.

Items such as incidental text or logos are exempt from this practice.

3.5 Pick a good typeface

When choosing a typeface for your interface, pick one that is standard with the system or in common use for system interfaces. Several (for example Fira Sans, Roboto, or San Francisco) have been designed for specifically for user interfaces. Other fonts are designed for reading large groups of text. Most modern phones have a high resolution screen along with font smoothing techniques that make choosing between a serif or sans serif font irrelevant. As the Royal National Institute for the Blind state, "Bizarre and indistinct typefaces should be avoided" (Gill, 1998).

3.6 Make sure that your views are large enough

Make sure that your touch targets are at least the size recommended in the user interface guidelines for the system, but you should likely go even larger. This helps people with motor impairments, in a moving vehicle, etc.

Hoober (2013) suggests that visual cues for a touch target, e.g., text or icons, should be at least 2.1 mm and 2.8 mm respectively. The preferred size should be 8 mm, while the spacing between targets should be 10 mm from center to center. Yet, he does point out that a 20-mm sized button may be a better to deal with the "inaccuracy due to motor-function issues." Hoober notes that these numbers may be bigger or smaller for other devices because one normally holds larger devices further away and smaller devices closer.

Additionally, if the operating system offers the ability to adjust the size of its text, your app should respect those settings and adjust its text size accordingly (with corresponding changes in layout, spacing etc.).

3.7 Have the app speak the language of the users

Providing a translation to a user's native language makes it easier for that user to understand what is happening in the app and use it more efficiently and effectively. This is especially true for apps that may be used by visitors. Avoiding the use of jargon can help reach users that are not as familiar about an area or theme.

3.8 Provide alternate text for icons, pictures, and other visual information

This is normally the first item developers think of when creating an accessible app, but it is just one part in a whole process. If a control or label is represented by an icon, it should have text that describes what it does. If there is information that is important and only represented graphically, it should have a description. All frameworks provide a mechanism for describing its content for assistive technology. Providing this description can mean that a user may actually have an idea of what a button does.

On the opposite, side, there can be decorative items or other bits of information that should *not* have an extra description. There are techniques to ensure that these items are invisible to assistive technology. Removing these items makes it easier for someone to navigate and use an app.

Sometimes writing alternative for a control is *different* than what text is provided. Examples are provided later (§ 3.11). Also, keep in mind that alternative texts also need be translated to the languages that you support.

3.9 Use multiple cues to present important information

Important information, for example, warnings and alerts, should not be presented with just one cue, such as not *only* color, not *only* sound, not *only* text. Use additional cues, like position, sound or animation, to help convey the information. Using the additional cues makes it possible to be aware of information when the user cannot perceive the change. For example, sound is turned off, the user is red-green colorblind, or the user is occupied and cannot focus on the screen.

3.10 Consider providing your media in alternate formats

When providing your own visual and audio content, consider providing an alternate format (for example, captions or a script) to make it the information available to a larger audience. This is especially true if listening or viewing the media is important to accomplish a task. This also makes the media more robust (for example, a user is able to follow along in a noisy environment).

3.11 Write good labels for your items

Many times a mobile phone app presents information in the form of a list of items. Each item may contain text labels, icons, and additional controls. A naive approach to presenting this information is to make sure that the individual parts are accessible. Yet, the overall layout of the item may convey additional information that will be absent if a label is read on its own.

To understand this better, it helps to look at an example. The list item in the Quick Forecast app (Figure 1) shows the day of the week, the date, an icon of the current conditions, the high and low temperature, the wind direction and speed.

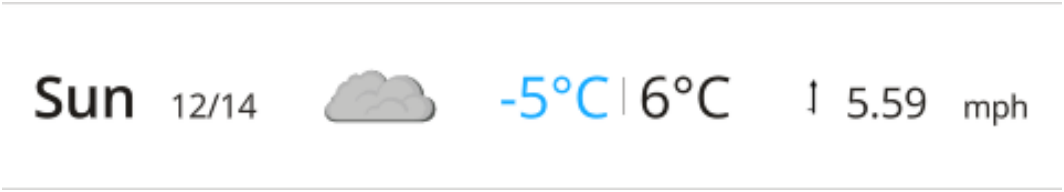

Figure 1. List item from 10-day forecast in the Quick Forecast app.

This is a lot of information. The icons for the current conditions and the arrow for wind direction tell us a lot in a little space, but the layout of the item also provides lots of visual clues to compress the information. For example, the low temperature on the left and the high temperature on the right separated by a vertical bar helps give an indication about the relationship of the information, the use of common abbreviations ('C' for Celsius, 'º' for degrees, and "mph" for miles per hour) helps convey units in a small place. The date also uses abbreviations both for the day of the week and the date via the month number followed by a slash followed by the day in the month.

Without doing anything, the screenreader delivers a soup of letters and numbers as the user goes over each label:

"Sun 12 14", *blank*, "minus 5 C 6 C" *blank* "5.59 m p h"

If this was all you had to go on the information would not be very helpful. One would be disappointed if this was how the weather forecast was read on the radio. It requires the user of the screenreader to infer a lot of information. Someone hearing this might say, "I guess the first two numbers are the date (or time)? I guess the numbers ending in 'C' are temperature…"

Yet, we can control the label that is spoken back by the screen reader. It is much better to disable accessibility for the individual parts and instead set the label for the entire item:

"Sunday, December 14, Cloudy, Temperatures: Low minus 5 degrees Celsius, High 6 Degrees Celsius, Wind from the South at 6 miles per hour"

This label could probably still use some work, but at least the information is understandable without having to know the underlying layout of the table item. It also provides information that was lost (the condition and wind direction). Notice also that we rounded up the wind speed; in this context, the 0.41 difference is not significant and can be safely ignored. More precision may be necessary in other situations (for example, when dealing with currency).

Many times a group of text and pictures are laid out to present information that is more than the sum of their text. Don't be afraid to only return one label for the entire group. Alternatively, there may be ways to alert the system that one view provides information for another (for example, the `<label>` element in HTML); this may reduce the need for combining elements.

3.12 Use the accessibility testing tools on your platform to check accessibility information

Beyond using assitive technology, developers can make use of several accessibility tools to check the accessibility information of their app.

The iOS Simulator has a feature called the *Accessibility Inspector* (Figure 2), this is a floating window on the simulator that displays accessibility information for a selected item. This includes the Label, its traits, its hint if it has one, its size and position on the simulator, and any notifications that have recently happened. The Accessibility Inspector may be quicker to use in situations where you want to check the labels and traits instead of using something like VoiceOver.

The Android emulators do not have any built in facilities, but there are a couple of nice features in TalkBack that developers can use. One is QueryBack. When QueryBack and Talkback's Explore by Touch is enabled, the contents of the accessibility information is shown on the screen. Another option is to go into the Developer Options in TalkBack itself and choose the option to Display text that is read aloud by TalkBack.

Additionally, You can use UI automation, uiautomatorview, (The Android Open Source Project, 2014) in Android to test out accessibility. The tool is designed for UI testing, but it depends on accessibility in order to extract the information about the UI, and shows most of the accessibility properties.

3.13 Try out your app with assistive technology

As mentioned in the previous section, it is a good idea for developers and designers to have familiarity with the different kinds of assistive technology that is available on devices. Part of the reason is that it allows early testing and the possibility of addressing the most glaring accessibility issues. Extensive knowledge is not needed. For example, a

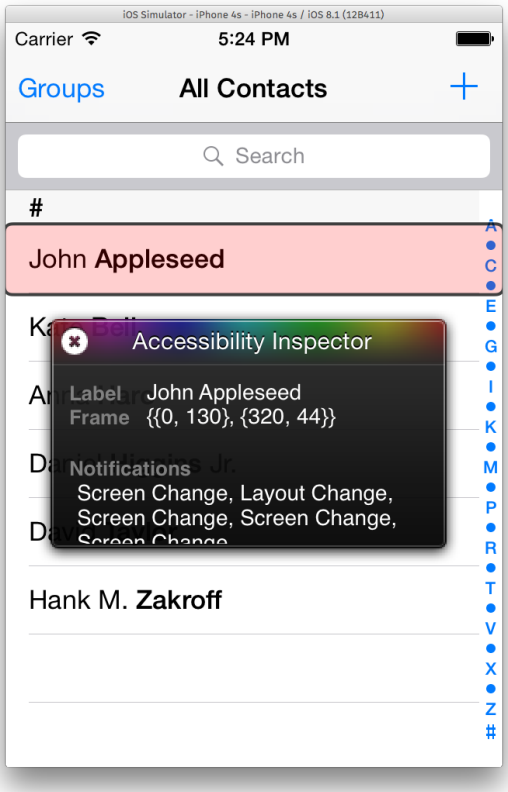

Figure 2. The Accessibility Inspector on iOS.

designer or developer knowing enabling the VoiceOver or TalkBack and use gestures to be able to navigate the app in a logical way can uncover many problems. For VoiceOver and TalkBack you can remove the temptation to look at the screen by toggling the Screen Curtain on iOS (triple tap with three fingers) or lowering the screen brightness on an Android phone.

In comparison to the large combination of web browsers and screen readers on desktop platforms, the barrier to using VoiceOver or TalkBack is much lower. But, remember that knowing how to use an assitive technology does not mean being an expert. This is sometimes easy to conflate, especially with VoiceOver and TalkBack's simplicity when starting to use them. Typically, people who have no choice but to use an assitive technology develops many techniques and strategies to accomplish tasks and keep themselves oriented. They likely also have good ideas about how information may be better presented for a particular AT.

3.14 Perform user evaluations with people with disabilities

It is important to test your app with assistive technology. But, it is also important to perform user evaluations that include people with disabilities. These are people that are experts with assistive technology and know how to interact with apps. As mentioned above, they have techniques and strategies to accomplish tasks and keep themselves oriented. Just as usability testing uncovers issues with a design that wouldn't be found

by the team alone, testing with people with disabilities will also uncover accessibility *and* usability issues that cannot be found without being used by people who depend on assistive technologies.

You can run the accessibility tests separately from usability tests. That is, you run tests only looking for accessibility issues. But, if much of the accessibility work has been done up front, then the evaluations can be included as part of the normal usability testing.

Part of the reason to include people with disabilities is that following guidelines alone is not enough to ensure a solution is accessible (Rømen and Svanæs, 2011; Power, Freire, Petrie, and Swallow, 2012). Testing with people with disabilities can help cover the gaps in the guidelines.

4 Other resources for creating accessible guidelines

There are multiple places that one can turn to for creating accessible apps. Some items have already been referred to in the best practices above. We repeat some here and include new ones as a place to gather resources.

Global Accessibility Reporting Initiative This organization, part of the Mobile Manufacturing Forum, has a website (`https://www.gari.info/index.cfm`) that provides information about mobile devices and their accessibility features. It offers special sections for governments, manufacturers, and developers. The Initiative encourages developers to submit their accessible apps to help build a searchable database.

The Web Content Accessibility Guidelines (WCAG) 2.0 The WCAG 2.0 by the W3C, while initially targeted on the world wide web are applicable to content in multiple digital arenas. They have already been mentioned for color contrast, but the guidelines have usable information text layout, sound, and video. Considering that information in mobile apps can often be presented as a page of HTML, knowing these guidelines can be helpful.

Guidelines for the development of accessible mobile interfaces This short report from Funka Nu AB (2012) provides a set of guidelines for creating mobile interfaces, with a primary focus on mobile websites, but items can be applicable to apps as well. The guidelines are divided into section for choosing a solution, design, layout, interaction, content, and user settings.

Tilgjengelighetsevaluering av sosiale medie app'er på smarttelefon This report by Dale, Drivenes, Schulz, and Tollefsen (2012), only available in Norwegian, describes an accessibility evaluation of three different social media apps. While the evaluation itself is now dated, the report includes advice for developers that want to create accessible apps and a list of resources for doing evaluations.

Tips for Taking Full Advantage of VoiceOver in Your App AppleVis user Mehgcap (2014) presents a developer guide that gives a good breakdown of how a developer can do different things with VoiceOver. The guide lists different tasks that a developer may

want to do with VoiceOver (for example, focus on a specific control, check VoiceOver is running, or react to specific gestures). The AppleVis website is a great place to see reviews of apps and discover pitfalls users can run into when using apps on iOS.

Results from User Testing in BestApps The first report from Schulz, Sæther, and Gladhorn (2014) is a companion report from the BestApps project and describes findings from a user test that was carried out to check the accessibility of an app. It includes a sample interview guide (in Norwegian) and a possible set-up for doing an evaluation. It also discusses the issues that were uncovered while testing. A possible resource for performing an evaluation.

D6.3 Design Iteration II – Evaluation Report This report by Busch, Wolkerstorfer, Hochleitner, Schulz, Fuglerud, Tjøstheim, Leister, Solheim, Pürzel, Lorenz, Wittstock, Dumortier, Vandezande, and Petro (2013) includes a summary of issues of working with TalkBack during that time period. It also highlights interesting accessibility issues when working with phones and other smart objects in the Internet of Things.

References

Global Accessibility Reporting Initiative. Learn About Mobile Accessibility, 2014. URL `https://www.gari.info/learn-mobile-accessibility.cfm`.

Apple. Using AssistiveTouch on your iOS device, 2014. URL `http://support.apple.com/en-us/HT202658`.

Ablenet Technology. iOS 8 New Accessibility Features, 2014. URL `http://www.ablenetinc.com/emails/Technology/iOS8-FollowUp-Sept2014.html`.

Jonathan Avila. Android 5 Lollipop – Switch Access has Arrived on Android!, 2014. URL `https://www.ssbbartgroup.com/blog/2014/11/19/android-5-lollipop-switch-access-has-arrived-on-android/`.

Trenton Schulz and Kristin Skeide Fuglerud. Creating Personas with Disabilities. In Klaus Miesenberger, Arthur Karshmer, Petr Penaz, and Wolfgang Zagler, editors, *Computers Helping People with Special Needs*, volume 7383 of *Lecture Notes in Computer Science*, pages 145–152. Springer Berlin / Heidelberg, Linz, Austria, 2012. ISBN 978-3-642-31533-6. URL `http://dx.doi.org/10.1007/978-3-642-31534-3_22`.

John Gill. Access Prohibited? Information for Designers of Public Access Terminals. Technical report, Royal National Institute for the Blind, 1998. URL `http://tiresias.org/about/publications/pats/index.htm`.

Steven Hoober. Common Misconceptions About Touch. *UXMatters*, March 2013. URL `http://www.uxmatters.com/mt/archives/2013/03/common-misconceptions-about-touch.php`.

The Android Open Source Project. UI Testing, 2014. URL http://developer.android.com/tools/testing/testing_ui.html.

Dagfinn Rømen and Dag Svanæs. Validating WCAG versions 1.0 and 2.0 through usability testing with disabled users. *Universal Access in the Information Society*, September 2011. ISSN 1615-5289. doi: 10.1007/s10209-011-0259-3. URL http://www.springerlink.com/index/10.1007/s10209-011-0259-3.

Christopher Power, André Freire, Helen Petrie, and David Swallow. Guidelines are only half of the story. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, pages 433–442, New York, New York, USA, 2012. ACM Press. ISBN 9781450310154. doi: 10.1145/2207676.2207736. URL http://dl.acm.org/citation.cfm?id=2207736http://dl.acm.org/citation.cfm?doid=2207676.2207736.

Funka Nu AB. Guidelines for the development of accessible mobile interfaces. Technical report, Funka Nu AB, Stockholm, 2012. URL http://www.funkanu.com/PageFiles/9429/Guidelines_for_the_development_of_accessible_mobile_interfaces.pdf.

Oystein Dale, Therese Drivenes, Trenton Schulz, and Morten Tollefsen. Tilgjengelighetsevaluering av sosiale medie app'er på smarttelefon. Technical report, MediaLT, Oslo, 2012. URL http://www.medialt.no/tilgjengelighetsevaluering-av-sosiale-medie-apper-paa-smarttelefon/1137.aspx.

Mehgcap. Tips for Taking Full Advantage of VoiceOver in Your App, 2014. URL http://applevis.com/guides/ios/tips-taking-full-advantage-voiceover-your-app.

Trenton Schulz, Jan Arve Sæther, and Fredrik Gladhorn. Results from User Testing in BestApps. Technical Report December, Norsk Regnesentral, Oslo, Norway, 2014. URL http://publications.nr.no/1418390459/DART-13-2014-BestApps-UserEval-Results-Schulz.pdf.

Marc Busch, Peter Wolkerstorfer, Christina Hochleitner, Trenton Schulz, Kristin Skeide Fuglerud, Ingvar Tjøstheim, Wolfgang Leister, Ivar Solheim, Franziska Pürzel, Mario Lorenz, Eckhart Wittstock, Jos Dumortier, Niels Vandezande, and Daniel Petro. D6.3 Design Iteration II – Evaluation Report. Technical report, CURE, Vienna, Austria, 2013. URL http://www.utrustit.eu/uploads/media/utrustit/uTRUSTit_D6_3_Design_Iteration_2-Evaluation_Report_FINAL.pdf.