Making iOS Applications Accessible

Adrian Kosmaczewski

2016-09-15

This is the talk I gave at the Mobile Developer Summit, Bangalore, India, September 15th, 2016

- Disabilities
- Convention on the Rights of Persons with Disabilities
  - Guiding Principles
- What is Accessibility?
- Making Accessible Apps
  - Design
    - User Experience
    - Typography
    - Layouts
    - Color
    - Iconography
    - Simple guidelines:
  - Development
  - Testing
  - Demo
    - Introduction
    - Overview of iOS Accessibility Features
    - The App
    - VoiceOver in Detail
    - Accessibility Inspector
    - Auditing the App
    - The Form Screen
    - Finally
- Conclusion
- References
- Slides
- Video

Thanks everyone for attending my talk, and thanks for the organizers of the Mobile Developer Summit to invite me during all of these years since 2011. I'm very glad to be here.

I will start this talk with a story.

In 2010 I was part of an itinerating group of people trying to get developers to pay attention to the iPhone and the recently released iPad. We gave talks about the subject in Denmark, Zurich, Geneva and finally in London, more or less during the time of the football World Cup in South Africa – I know, not the best moment for a developer conference, but still, that's how things work sometimes.

In that conference I gave a talk that I also gave here in MODS 2011, the one called "Ten Commandments for iOS Development;" maybe some of you might remember from my last time here.

I gave my talk, and at the end, as I was wrapping my cables and packing my laptop, a man approached me and told me, "you forgot one commandment." Still distracted by the jungle of cables in my bag I did not immediately raise my head, so I asked him directly as I was still storing items.

"Really? What is the commandment that I forgot?" I asked.

"That of making apps accessible."

I raised my eyes and I saw a tall blind man next to me, with his cane and an iPhone in his hand. I stopped packing my items and I got up, really curious.

The man, whose name I have unfortunately forgotten since then, told me his story. He was a software developer himself; he worked in London in an agency, and, understandably enough, specialized in making apps for users with viewing difficulties and disabilities.

He proceeded them to quickly demo his app to me, and to my absolute amazement he was able to use his iPhone without any kind of assistance, just by using a feature called VoiceOver, which I will explain in detail later in this talk.

Needless to say, I was ecstatic. Talking to this man was one of the most enlightening moments in my life, a real eye-opener, and since that day I have tried to bring the topic of accessibility to all of my fellow developers, in every project that I have worked on since then.

Disabilities

The World Health Organization defines disabilities as follows:

> Disabilities is an umbrella term, covering impairments, activity limitations, and participation restrictions. An impairment is a problem in body function or structure; an activity limitation is a difficulty encountered by an individual in executing a task or action; while a participation restriction is a problem experienced by an individual in involvement in life situations.

There are three major kinds of disabilities:

- Impairments
- Activity limitations
- Participation restrictions

It turns out that 1 billion people in the world have a disability of some kind, of which around 300 million people are either blind or have some kind of vision impairment. 20% of Americans, for example, have a disability of some kind. One out of every 200 women and one out of every 12 men has color blindness.

Amazing, right? It turns out that by ignoring the problem, we make it worse.

A very important concept to keep in mind that disabilities are not a barrier: the barriers are something created by society, when we deny basic rights of movement, use, and enjoyment to some fellow humans. It is our duty to change our mindsets, and to make our applications as accessible as possible. We must not make assumptions about the capabilities of the users of our applications!

Actually, it turns out that this is a legal duty.

Convention on the Rights of Persons with Disabilities

The Convention on the Rights of Persons with Disabilities is an international human rights treaty of the United Nations intended to protect the rights and dignity of persons with disabilities. Parties to the Convention are required to promote, protect, and ensure the full enjoyment of human rights by persons with disabilities and ensure that they enjoy full equality under the law.

Article 1 says that the purpose of the convention is

> to promote, protect and ensure the full and equal enjoyment of all human rights and fundamental freedoms by all persons with disabilities, and to promote respect for their inherent dignity

Guiding Principles

There are eight guiding principles that underlie the Convention:

1. Respect for inherent dignity, individual autonomy including the freedom to make one's own choices, and independence of persons.
2. Non-discrimination.
3. Full and effective participation and inclusion in society.
4. Respect for difference and acceptance of persons with disabilities as part of human diversity and humanity.
5. Equality of opportunity.
6. Accessibility.
7. Equality between men and women.
8. Respect for the evolving capacities of children with disabilities and respect for the right of children with disabilities to preserve their identities.

The Convention stresses that persons with disabilities should be able to live independently and participate fully in all aspects of life. To this end, States Parties should take appropriate measures to ensure that persons with disabilities have access, to the physical environment, to transportation, to information and communications technology, and to other facilities and services open or provided to the public.

What is Accessibility?

What is accessibility, then?

Accessibility is the degree to which a product, device, service, or environment is available to as many people as possible.

Accessibility is not to be confused with usability, which is the extent to which a product (such as a device, service, or environment) can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

Accessibility is not only about disabled people, and it is not only about apps or websites. Accessibility is for everyone. Accessibility is about everything and everyone, to ensure a fair degree of access to goods and services and places and situations, to as many people as possible, at all times.

For example, you can ensure that your conference is accessible by making it friendly to people with disabilities, but also by making it friendly to people coming from far away, to people speaking other languages, to people from other cultures or from any other social group you might think of.

You can also design your cities to be accessible, of course, but there are still far too many places in the world where walkways, public transportation, buildings and homes are just not accessible enough, like New York City, for example.

But most of you are developers, and as such the first place where you can change the world is through your applications. And your applications are born in your teams, and Apple recognized the fact that including people with disabilities in their teams made their teams produce better apps.

Not only that, but you can even shop accessibility accessories in the Apple Store as well; this is an active message to a large community, telling them that they care. Of course this is a great business, as well, why denying it? The opportunities are endless in this space, and the time is right for us to step in.

All of this is happening, right now, and this talk will be a call to action, to all of you, to become better software developers through accessibility.

Making Accessible Apps

As I told you at the beginning of this session, the accessibility features of macOS, iOS, watchOS and tvOS are unparalleled and extremely powerful. Apple has

historically had a strong commitment to inclusiveness, and many features are available to developers to increase the reach of their applications:

- In all the operating systems:
  - Internationalization
  - VoiceOver
  - Color adjustments (Grayscale, etc)
  - Dynamic Type
- In iOS:
  - Magnifier
  - Typing feedback
- In watchOS:
  - Taptic time
- In tvOS:
  - Switch control

Applications can provide accessibility:

- Motor: for example, the dwell control in macOS.
- Vision: for example, the new iOS 10 magnifier, or VoiceOver.
- Hearing: for example, visual cues instead of sounds.
- Learning: for example, for users with autism and dyslexia.

When designing and developing a new application, accessibility has to be considered seriously from the start. I am going to talk about the different aspects to take into account in the different phases of the creation of an application, that is, during design, development and testing.

We must engage with people with disabilities during these phases, in order to learn more about their needs, and to modify our designs and our code in order to make the final result accessible and easier to use.

Design

When designing an application with accessibility in mind, there are five areas to consider:

- User Experience
- Typography
- Layouts
- Color
- Iconography

Let's talk about each of these in detail.

User Experience At the very beginning of a project, when the idea of an application starts to take shape, the design team should consider the experience of users with disabilities as a requirement. This includes creating the required user

personas and scenarios for them, the required mockups and their corresponding user testing sessions, so that the needs of all types of users is considered thoroughly.

In some countries, particularly for government projects, there are written guidelines regarding accessibility, and in many cases they are enforced by authorities. Pay attention to those guidelines, as well as those provided by Apple for each of its platforms, so that you begin your projects by taking the right decisions. In those cases you not only might risk a rejection from the App Store or a bad experience for your end users, but flatly the rejection of the final product by the government authorities, and you do not want that to happen.

Typography Typography is a very important element in application design. Through hierarchy, designers are able to highlight elements and guide through the various steps required to perform a task.

There are several techniques to create hierarchies in text:

- Spaces
- Size
- Bold
- Italic
- Color
- Upper- and lowercase

Whenever you design an application, keep in mind the final languages in which this application should be available. This information is very important, because not all writing systems support uppercase or italic. In general, the best strategy to manage hierarchies is to use larger font sizes.

Needless to say, some languages like German have very long words, and they take more space on the screen.

Also, please stick to standard fonts whenever possible, or at least make sure that the custom fonts you use support bold styles, to use those as a basis for your hierarchies.

Finally, and here's a tip for both designers and developers: the `clipsToBounds` property should be disabled in Interface Builder, because some alphabets have diacritics below the descender or above the ascender – this means that using bigger line heights is a must in your designs.

Layouts Dynamic layouts are very, very important, for the simple reason that there is an accessibility feature called "Dynamic Text." With this feature, that all application should be supporting, users can request the text of all applications to be bigger – and sometimes very big! – or smaller than the standard size.

This means that your designers should not only give you the font specifications for the application, but rather a table where the following styles are available

in all the following heights:

Styles:

- Title 1
- Title 2
- Title 3
- Headline
- Body
- Callout
- Subhead
- Footnote
- Caption1
- Caption2

Sizes:

- xSmall
- Small
- Medium
- Large (Default)
- xLarge
- xxLarge
- xxxLarge

Using this table, you should be able to test your application in all the sizes, and this will be a great help for the end users of your application.

Pay attention to the possibility for users to select the "Bold Text" option for text in all operating systems, to increase the legibility of text.

Finally, remember the rules for right-to-left languages; there are two of them, Hebrew and Arabic, and when your application displays those languages, all the UI should be reversed. However, pay attention to the fact that the following elements must appear in left-to-right mode, even in the case of Hebrew and Arabic:

- Numerals
- Phone numbers
- Clocks
- Music notes
- Graphs
- Video playback controls
- Images

Color Color is a great tool for conveying emotions to the user. The most common choices can be quickly summarized as follows:

- Blue is the most popular color, it has a short wavelength and represents peace, quietness and loneliness.

- Green represents nature, good luck (in Western cultures) and safety.
- Red has a long wavelength, and usually conveys the ideas of revolution, conflict, passion and love. It also represents good luck in asian cultures, which is shown in the Stocks application for example.

There is a lot of people with color blindness; one out of every twelve men, and one out of 200 women; that is, around 700 million humans have this condition!

For your designers, then, the most important thing to keep in mind is to always work in their designs using black and white screens! This is a very simple trick but it always work.

How do you create contrast with grayscale? Well, keep in mind that the ratio of contrast between black and white is 21:1, while between gray and black is only 4.4:1 and only 1.9:1 between dark gray and black. This means that you should not use dark gray for disabled elements! Use it only for decoration. For disabled elements, use a lighter shade of gray, or a color that translates as easily into that shade.

It is very important that your designs always have higher contrast ratios, so that they work good in grayscale environments. The contrast between the elements on the UI should not rely only on color alone. Use geometry, contrast, typography for that.

Remember when you are designing your user interfaces, to work on grayscale screens. You can change that setting very easily in macOS, iOS, tvOS and watchOS. If your UI works well in grayscale, then it will work even better with color. And, if possible, test the application with users with viewing disabilities, to make sure that your designs are usable by everyone.

Iconography Iconography is as much an art as a science, most of which is derived from Semiotics, that is, the science of meaning. Given the differences in the cultures of this planet, you should not assume a given meaning for an icon, as different people will interpret them in different ways. Cultural, religious, technical and moral reasons might cause your users to feel offended by the iconography of your application.

So, here's a simple tip: when in doubt, test with real users, or if all else fails, just use text. Speak to your users and tell them your story! Sometimes a good word might be all you need.

If your UIs have to work on Arabic or Hebrew, use non-directional icons that work in both left-to-right and right-to-left orientations.

Simple guidelines: First and foremost, read the Apple Guidelines for the platforms you are working on; they contain a wealth of information regarding accessibility.

Here go some other simple tips and tricks to design accessible applications:

1. Add descriptive text in UI controls
    - Why? Users cannot see everything, devices can "read" aloud
        - For whom? For blind users and when assistive devices cannot help
        - How? ALT attribute, android:contentDescription or accessibilityLabel + accessibilityTrait in iOS
2. Minimize text input in UIs
    - Why? To avoid mistakes and frustration
    - For whom? Everyone, actually; particularly for motion disabilities, also for shaky situations (bus, train)
    - How? Avoid free text; choose from lists; default entry modes for different types of text; keyboard shortcuts if available.
3. Avoid scrolling
    - Why? Small screens
    - For whom? Users with restricted vision or using screen zooming devices
    - How? Clear language; limit scrolling to one axis (usually vertical); avoid large images; position important elements on top and visible at first; large clickable areas.
4. Do not use gestures alone; complement with buttons or links
    - Why? Not intuitive, unknown to most users
    - For whom? Blind users, temporary disabilities (plaster), users in the street.
    - How? Add links and buttons; add text to explain.
5. Allow zooming of UI elements
    - Why? It can be blocked, particularly in web pages
    - For whom? Users with bad sight
    - How? HTML5 meta tags controlling the viewport
6. Add captioning to video content
    - Why? Because this can be read by screen readers for blind users, but it is also useful if for some reason the user cannot increase the volume of a video, and wants to know its contents.
    - For whom? For everyone, but mainly for users with hearing problems.
    - How? Subtitle files and adding chapter information in long video files.

Development

Developers can increase the accessibility of their applications using different mechanisms. I am going to explain them in this section.

First of all, you should get acquainted with the `UIAccessibility` protocol (and, if you are creating a macOS app, you should check out the analog `NSAccessibility` protocol instead.)

Just like any protocol in Cocoa, and as the word "protocol" suggests, the `UIAccessibility` protocol represents the phases of a conversation between the

accessibility subsystem of the operating system and your user interface. Each of the methods in `UIAccessibility` represents a question to which your UI will give an appropriate answer in time.

The main four questions that are asked through `UIAccessibility` are the following:

1. Are you accessible?
2. If yes, what are you?
3. And who are you?
4. And where are you?

The first question is trivial but fundamental, and it is answered by the `isAccessibilityElement` boolean property. If you answer `true` to this question, then you should provide suitable answers for the following three questions.

The second question is answered by the `accessibilityTraits:` property, which holds a value of the `UIAccessibilityTraits` enumeration. This enumeration is a bitmask, which means that any `UIView` can be many of these.

The answer to the third question is provided by the `accessibilityLabel` property, which returns a string.

Finally, the answer to the fourth question is given by the `accessibilityFrame` property, itself returning, predictably enough, a `CGRect` value.

If your application uses only standard iOS `UIView` subclasses, then you do not need to do any of this; all of the standard iOS components already provide this information for your automatically. But of course, if you create your own `UIView` or `UIControl` subclasses, then you must provide the information required by the `UIAccessibility` protocol in your code.

Developers can also use Interface Builder to provide the information required by the `UIAccessibility` protocol, instead of using code. The "Accessibility" panel in the Identity Inspector at the right of Xcode provides a UI that allows you to provide the required information by the accessibility subsystem of your target operating system.

And finally, we come to the possibilities of internationalization of your applications. First, using the `NSLocalizedString()` family of functions, developers can separate localizable strings from their application into individual `.strings` files, usually saved with the UTF-16 encoding, and which can be handled to translators so that your application can be used by users in different cultures and languages.

The second important feature of internationalization is the possibility to localize storyboards and XIBs, which is particularly important when supporting right-to-left languages such as Hebrew or Arabic. In those languages, your UI should be reversed as well, so localizing your UI files helps you achieve that.

I am not going to say much more about internationalization, because Laura Savino is going to talk about that tomorrow at 10:50 in the SD Hall, so please make sure to attend her talk called "iOS Localization with Modern Xcode Tools."

Testing

To test applications for their accessibility, previously one had to manually check applications (code *and* Interface Builder files) or to have people manually testing the applications, to make sure that the user experience matches the expectations.

There are good news from Xcode 8, however; the brand new version of Xcode for iOS 10 and Swift 3 brings a new tool with it: the Accessibility Inspector, available as usual in the Xcode menu, under the Tools submenu. Using this inspector, developers can get a complete report of the current state of the accessibility features of applications, and using this information, they can add the missing parts.

As complete and as awesome as the Accessibility Inspector is, I cannot stress enough the importance of having actual users, preferably with disabilities, testing the applications and providing feedback to the design and development teams.

There is, however, one major advantage to adding accessibility information to applications; and that is, Xcode UI Testing. It turns out that the UI testing capabilities of Xcode make use of the accessibility features of applications, in order to find and interact with UI controls. By adding accessibility information to your iOS applications, you are actually enabling a whole new range of automated testing, inside of Xcode itself.

Demo

Before starting the demo:

- Prepare Mouseposé
- Open a QuickTime document recording the screen of the iPhone connected by cable, so that one does not have to unplug the computer.
- Use the iPhone 6S simulator for the app demo

Introduction In this demo I am going to give you a small, very small overview of the different accessibility features available to iOS developers, many of which are very similar to those available in the other operating systems, like macOS, tvOS and watchOS.

Overview of iOS Accessibility Features First of all, let's dive into iOS and let's take a look at the possibilities. The Settings application allows users to enable and disable a plethora of options about accessibility; let's take a look at some of them.

First and foremost, VoiceOver, which is a technology available since iPhone OS 3 and the iPhone 3GS, which allows users with vision impairments to have the operating system "read" whatever is on the screen. We are going to use this feature in the demo.

Another interesting accessibility feature in iOS 10 is the AssistiveTouch control, which provides on-screen functionality that is otherwise available through gestures. This helps users with mobility or learning disabilities to use iOS more effectively. It is also a great feature if your hardware buttons are broken; I used this feature extensively with an old iPhone 4S I had a couple of years ago, whose wake button was broken.

iOS 10 also brings a new accessibility feature, the Magnifier; this feature was mentioned during the last WWDC keynote, and it is very simple to use. Just enable the feature and then triple-tap on the home button to launch it.

The App But let us go to the heart of this demo.

I have created a small iOS application that loads the contents of a JSON file contained within the bundle of the application, and displays the data on a `UITableView`. The application in that sense could not be simpler. As you can see in the storyboard, it is a classic master/detail application, only for iPhone.

By the way, if you see me zooming in and out during this demo, this is because I have enabled yet another accessibility feature on macOS: Go to the System Preferences, Accessibility, Zoom and select "Use scroll gesture with modifier keys to zoom" and then I just press Control and zoom with a gesture on my trackpad. Very useful for teaching!

Let me launch the application so I can show you what it does. First it shows the data on a standard table view, and when I tap on any of these entries, a small form appears for the user to modify the information. The form features a very rudimentary form of validation; if any of the fields is empty, the name of the field appears in red to indicate that it should not be empty.

Nothing else but enough for this demo.

All of the components in this application are standard, except for the small "favourite" star control, which is an extremely simple subclass of `UIControl`.

VoiceOver in Detail I will now activate VoiceOver, and will try to navigate back to the application just by using the voice feedback, and without looking at the screen. As you can guess I have rehearsed this a few times, but it is still challenging. We take for granted our sight so much.

As you can see, VoiceOver does quite a good job highlighting the currently selected item in the application, and reading back aloud whatever its value may be. But still, in the application itself the feedback could be better. In particular, when I select an item on the list by double-tapping, and then I dive on the form,

the favourite custom control says “star” and “white star” which are the names of the Unicode characters that I have used in my control to show both states. Not very helpful, is it?

Also, you can imagine that developing a complex application and checking it through this procedure might become quite tedious, so again we can do better than that.

Accessibility Inspector Enter the Accessibility Inspector, a new feature of Xcode 8, to increase the accessibility of an application. We can access it by selecting the “Xcode / Open Developer Tool / Accessibility Inspector” menu.

The Accessibility Inspector is a very simple application that has three main screens:

- The Inspection screen
- The Audit screen
- The Settings screen

The toolbar also features a menu, to inspect any application running either in the current Mac or in any connected iOS device, and a selector, allowing you to point to a specific control or view on the screen and find out its accessibility settings.

The Inspection screen shows the state of the currently selected item, exactly as the accessibility subsystem can see it and interact with it.

The Audit screen performs a quick scan of your application, and gives you a report about the major problems found with the application.

Finally, the Settings screen allows you to quickly set your app in inverse colour mode or to change the Dynamic Type settings, which triggers the required `NSNotifications` as expected. Speaking about which, you can open a separate window in the “Window / Show Notifications” where all the notifications sent are displayed.

Auditing the App So, let us run an audit on the master screen, and very quickly we see that there are absolutely no warnings. This is because most standard UIKit controls already provide accessibility information by default.

However good this is, we can make it better. Remember when I hovered my finger over the individual cells? VoiceOver would read to me the whole contents of the cell, which is a bit ridiculous, because just the name would be better in this context. For that we are going to edit the code of the application a little bit, and we are going to specify a default value for the `cell.accessibilityLabel` property.

I am going to go now to the settings panel to check if Dynamic Type works well; as you can see we can change the sizes of the text, but unfortunately the height of the cell stays the same. By making some small changes in the `viewDidLoad`

method of the master controller, we can tell it to automatically resize those objects as well.

Let us try VoiceOver now; yes, now the system only says the first name of the customer, and it sounds much better and useful.

The Form Screen Let us pay attention now to the detail / form screen. First of all, a simple usability fix that actually helps everyone is to use the "numeric" keyboard for the "age" field; this will benefit the application greatly, because it minimises the effort required to enter data. Accessibility is about everyone, remember.

In the Accessibility Inspector (whose window can stay permanently on top of the others, as you can see) I run an audit of the form screen and I can see that there are a few problems related to Dynamic Type. Let us solve these problems in the code.

First of all, to react to Dynamic Type changes, we have to listen for the `UIContentSizeCategoryDidChange` notification, and in the notification handler we call the `UIFont.preferredFont(forTextStyle:)` method, passing the name of the desired style as a parameter.

Now, having done that, we can see that the application reacts correctly to the font changes. However, pay attention to the fact that instances of the `UITextField` class do not change their height; you might want to use other components such as a `UITextView` if required.

You can also see how AutoLayout handles automatically the resizing of your controls, and how the fields keep their baseline aligned to that of the labels next to them.

Validation errors are another interesting problem; if I set my Mac to display in grayscale, you can see that the contrast of the error message is not great; so what we're going to do is to use an error label at the bottom of the form, and we are going to display it. Also, the labels of the fields that do not pass validation will display an asterisk, to make them more prominent and visible. This way we avoid using colour as the unique way to convey information, and we increase the contrast.

I am also going to add a vibration effect, to make it even more clear to all users that the application has a problem. But the truth is that blind users would not know exactly what is happening. We are going to add now a text-to-speech help voice that will explain the problem to the user, but this has to be done only if VoiceOver is activated.

Thankfully, the `UIAccessibilityVoiceOverStatusChanged` notification and the `UIAccessibilityIsVoiceOverRunning()` function tell us exactly that. So, now, if VoiceOver is running, when there is an error, we are going to ask the iOS speech synthesiser subsystem to read the error message out loud.

Another important change consists in enabling proper accessibility to our "favourite" control. We can do that directly in the source code of the control, by answering the three questions that the `UIAccessibility` protocol will ask it every time it is required.

Of course this can also be done on an instance-basis, instead of directly at a class basis; you can do this in code, or in Interface Builder. Actually, we are going to use now Interface Builder to add the required information for the text fields, which by themselves are not very useful to VoiceOver. For that, we are going to use the "Accessibility Value" field on the Interface Builder inspector of our fields.

Finally Running VoiceOver on our application now yields much better results. The cells on the table only say as much as needed; the screen also reacts to Dynamic Type properly. The form is much better, too: it has better contrast for colour blind users, provides better information to VoiceOver users, shows a better keyboard to enter numeric values, reacts properly to Dynamic Type and it even speaks out the error message if needed.

The Accessibility warning now displays no warnings at all, and our testing shows that the user experience is good.

Not only that, but our application is now ready for UI Testing in Xcode, because, precisely, those kind of tests use accessibility information to access the different items on the screen. In this case our test selects the first item in the list, and then taps the favourite control to test its value and behaviour.

Conclusion

Accessibility is a fancy word for empathy.

Making our apps accessible is the way of the app maker to make the world a better place. It is a way to apply our religious sensibilities, whatever your beliefs may be, into helping the other. Going to the other. Projecting yourself with empathy, not by saying "I am sorry for your situation," whatever that may be, but actively trying to understand the other, and offering ways to overcome those situations.

Because accessibility is empathy, it is also internationalisation. It is all about telling the other "I want you to understand me," which of course requires me understanding you.

Because accessibility is empathy, it is also inclusion. Understanding other ethnographic groups, other religions, other nations, other cultures, other sexual orientations makes us better people. Including the other in the development team will invariably yield better applications, and yes, if that is what you want to know, that also translates in more sales and more money.

So, in the end, through accessibility we all win; both materially and non-materially speaking. The "side effects" of accessibility, those which economists often talk about in their papers, are tremendous. The overall balance for society is incredibly positive.

And in my humble opinion, India is in a particularly good position to lead this change, because of your diversity. I see the richness of your country, I see so many people fluent in lots of different languages, I see your culture, your music, your food, your natural kindness, and I see a country that is naturally biased to inclusiveness and accessibility.

I leave you with a quote by Helen Keller, writer, political activist, and the first deaf-blind person to earn a bachelor degree:

> Science may have found a cure for most evils; but it has found no remedy for the worst of them all - the apathy of human beings. Helen Keller, My Religion, 1927.

We as developers can embrace empathy, and fight apathy, through accessibility. Please make sure that your apps are accessible. This is very important.

My name is Adrian Kosmaczewski, and this was Making iOS Applications Accessible.

Thank you so much.

References

The following WWDC sessions have helped me to create this presentation and I strongly recommend you watch them too:

- WWDC 2016:
  - 104: Disability & Innovation: The Universal Benefits of Accessible Design
  - 201: Internationalization Best Practices
  - 202: What's New in Accessibility
  - 232: What's New in International User Interfaces
  - 407: Auditing Your Apps for Accessibility
  - 801: Inclusive App Design
- WWDC 2015:
  - 406: UI Testing in Xcode

Slides

Video