Implementing Accessibility

Practical steps toward making your iOS apps Accessible

Steve Kohls
Mostly Zeros Software
Jack Henry & Associates

What is Accessibility?
Why Should You Make Your App Accessible?
iOS Accessibility Features
Accessibility APIs
Implementation Tips
Demo
Testing
Summary
Resources

What is Accessibility?

Accessibility is a collection of features and technologies that assist users in interacting with iOS devices.

These technologies support users with the following disabilities:

- blindness or low vision
- deafness or hearing impairment
- physical or motor challenges
- autism or attention and sensory challenges
- speech impairments

Apple has made a significant investment in supporting Accessibility in iOS since the beginning.

Why Should You Make Your App Accessible?

You should make your iPhone application accessible to VoiceOver users because:

- It increases your user base. You've worked hard to create a great application; don’t miss the opportunity to make it available to even more users.
- It allows people to use your application without seeing the screen. Users with visual impairments can use your application with the help of VoiceOver.
- It helps you address accessibility guidelines. Various governing bodies create guidelines for accessibility and making your iPhone application accessible to VoiceOver users can help you meet them.
- It's the right thing to do.

* From the *Accessibility Programming Guide for iOS*

iOS Accessibility Features

Vision

- VoiceOver
- Speak Screen
- Zoom
- Font Adjustment
- Invert Colors and Grayscale
- Braille Displays

Hearing

- Closed Captions
- Mono Audio
- Visible and Vibrating Alerts
- Made for iPhone Hearing Aids

Physical and Motor Skills

- Assistive Touch
- Switch Control
- Touch Accommodations
- Third-party Keyboards
- Hardware Keyboard Support

Learning and Literacy

- Guided Access
- Speak Screen
- Safari Reader

Speech

- Speak Selection
- Speech Synthesis

Accessibility Features Demo

Accessibility Implementation

While implementing Accessibility is relatively straightforward, there are a lot of nuances to be aware of. I will address some of these.

I recommend implementing VoiceOver support first before addressing other Accessibility features.

Properly implementing VoiceOver gives you support for a lot of other Accessibility features for free. Including Switch Control and some of external hardware keyboard support.

Accessibility Implementation Checklist

Because there is a lot to pay attention to, I created a checklist I use on every screen to help ensure I've covered all the bases.

The checklist is available at the GitHub repository linked to at the end of these slides.

Accessibility APIs

There are iOS APIs which support the following features:

- VoiceOver
- Dynamic Type
- Switch Control
- External Keyboard Support
- Speech Synthesis
- Captioning and Audio Descriptions
- Guided Access

A large number of API additions were made in iOS 8.

Accessibility APIs

UIAccessibility protocol
UIAccessibilityContainer protocol
UIAccessibilityElement
UIGuidedAccessRestrictionDelegate protocol
UIKit Function Reference
AVSpeechSynthesizer
AVSpeechSynthesizerDelegate protocol
AVSpeechSynthesisVoice

Dynamic Type is not part of the Accessibility libraries, but is equally important. Supporting Dynamic Type is outside the scope of this talk, however.

Captioning and Audio Descriptions are supported in AVFoundation in iOS 8 and above.

UIAccessibility

This API enables you to support VoiceOver and other features. These methods listed below are the most commonly used. See the API Guide for more.

```
@property (nonatomic) BOOL isAccessibilityElement;
@property (nullable, nonatomic, copy) NSString *accessibilityLabel;
@property (nullable, nonatomic, copy) NSString *accessibilityHint;
@property (nullable, nonatomic, copy) NSString *accessibilityValue;
@property (nonatomic) UIAccessibilityTraits accessibilityTraits;
@property (nonatomic) BOOL accessibilityElementsHidden;
@property (nonatomic) BOOL accessibilityViewIsModal;
```

UIAccessibilityContainer

`UIAccessibilityContainer` allows you to group elements in a custom order.

```
- (NSInteger)accessibilityElementCount;
- (nullable id)accessibilityElementAtIndex:(NSInteger)index;
- (NSInteger)indexOfAccessibilityElement:(id)element;
@property (nullable, nonatomic, strong) NSArray *accessibilityElements;
```

UIAccessibilityElement

`UIAccessibilityElement` allows you to make custom non-UIKit elements accessible.

```
@property (nullable, nonatomic, assign) id accessibilityContainer;
@property (nonatomic, assign) BOOL isAccessibilityElement;
@property (nullable, nonatomic, strong) NSString *accessibilityLabel;
@property (nullable, nonatomic, strong) NSString *accessibilityHint;
@property (nullable, nonatomic, strong) NSString *accessibilityValue;
@property (nonatomic, assign) CGRect accessibilityFrame;
@property (nonatomic, assign) UIAccessibilityTraits accessibilityTraits;
```

UIKit (Accessibility)

Some examples of what's available in the API.

Accessibility Notifications

```
NSString *const UIAccessibilityVoiceOverStatusChanged;
NSString *const UIAccessibilitySpeakSelectionStatusDidChangeNotification;
```

Posting Accessibility Notifications

```
void UIAccessibilityPostNotification(UIAccessibilityNotifications notification, argument);

UIAccessibilityNotifications UIAccessibilityScreenChangedNotification;
UIAccessibilityNotifications UIAccessibilityLayoutChangedNotification;
UIAccessibilityNotifications UIAccessibilityAnnouncementNotification;
```

Accessibility state functions

```
BOOL UIAccessibilityIsVoiceOverRunning();
BOOL UIAccessibilityIsInvertColorsEnabled();
BOOL UIAccessibilityIsBoldTextEnabled();
```

AVSpeechSynthesizer

`AVSpeechSynthesizer` allows you to generate synthesized speech.

`AVSpeechSynthesizerDelegate` methods allow you to respond to speech events.

`AVSpeechSynthesisVoice` allows you to change the voice.

```
import AVFoundation

let string = "Hello, World!"
let utterance = AVSpeechUtterance(string: string)
utterance.voice = AVSpeechSynthesisVoice(language: "en-US")

let synthesizer = AVSpeechSynthesizer()
synthesizer.speakUtterance(utterance)
```

`UIGuidedAccessRestrictionDelegate`

Guided Access is a feature that restricts iOS to running only one app, while disabling the use of hardware buttons.

This protocol allows an app to specify additional features that can be disabled by users when in Guided Access mode.

See **<UIKit/UIGuidedAccessRestrictions.h>**

* From *<UIKit/UIGuidedAccessRestrictions.h>*

Implementation Tips

- Use the Accessibility Inspector in the Simulator
- Use the Accessibility shortcut
- Use a checklist so you don't forget anything
- Be aware not all UIKit elements fully or properly implement the `UIAccessibility` protocol! `UIBarButtonItem` is one.
- Use the power of the Internet if you're stuck

Accessibility Implementation Demo

Testing Tips

- Audit your app using VoiceOver
- Close your eyes
- Swipe fast. Blind users I have met do this.
- Use an external keyboard. Make sure the standard keyboard controls work with your app.
- Find someone with the disabilities you're building for

Summary

- iOS supports a wide range of Accessibility features
- Implementing Accessibility support in your apps has many benefits, both for you and your users
- Basic Accessibility implementation is straightforward, but there are a lot of gotchas
- Testing is very important to ensuring your Accessibility implementation is usable

Resources

Start here for all iOS implementation guidelines, API documentation, and WWDC videos:

http://www.apple.com/accessibility/ios/

More Accessibility resources are available here:

http://www.apple.com/accessibility/resources/

Demo project and Implementation Checklist:

https://github.com/stevekohls/AccessibilityDemo

Questions?

Steve Kohls

@stevekohls

steve@mostlyzeros.com

skohls@jackhenry.com