Accessibility

Introduction

The primary focus of this topic is on how to design a Titanium application to use the spoken feedback accessibility features of Android and iOS devices.

Designing universal accessible applications

An application is universally accessible if it can be used by all users, especially ones with visual, hearing and motor disabilities. Android and iOS provide many built-in OS features to aid in accessibility, such as a screen magnifier or spoken feedback for people with low vision or blindness. However, these utilities do not aid with all disabilities. Your application should be designed so it can be easily navigable by all people. For instance, people with motor disabilities require larger buttons to navigate the application, people with color-blindness need information communicated to them by other means if color is used, and people with hearing disabilities require other cues if sounds are used.

Review the following external guides for information about designing accessible applications:

- Accessibility Programming Guide for iOS
- Android Developers: Accessibility API guide
- Android Developers: Accessibility Design guide
- W3C Accessibility
- Web Accessibility Initiative

Spoken feedback accessibility features

Android TalkBack and iOS VoiceOver are accessibility features for the visually impaired for Android and iOS devices, respectively. Both of these features provide spoken feedback to the user to describe what they are touching, selecting, activating, etc.

On the Android platform, TalkBack varies by Android version. Both the supported view elements and user interface varies between Android versions. The user may need to download TalkBack from Google Play for earlier Android devices. TalkBack for Android uses gestures similar to iOS VoiceOver.

To support both spoken feedback assistants, all Titanium view elements support a set of accessibility properties (see table below), which can be used to specify the text spoken back for the UI element. Many of these view elements have a default value based on the control type and may already be accessible to visually-impaired users. Setting these properties may not be necessary for your application. If you have non-textual elements in your application, that is, you are providing information in a graphical way, such as a star-rating system, you need to explicitly set these properties with helpful hints, labels, and values, so they can convey the same information and be accessible.

Property	Default Value	Description
accessibilityHidden	false	Whether the view should be "hidden" from (i.e., ignored by) the accessibility service.
accessibilityHint	-	Briefly describes what performing an action, such as a click, on the view will do.
accessibilityLabel	Title or label of the control.	A succinct label identifying the view for the device's accessibility service.
accessibilityValue	State or value of the control.	A string describing the value (if any) of the view for the device's accessibility service.

For the Android platform, a custom string is created based on these properties, in the order, `accessibilityLabel`, `accessibilityValue`, then `accessibilityHint`, which maps to the android:contentDescription property. Do not specify the `accessibilityLabel` property for view elements that display textual items, such as the Label text or Button title, unless you want to override the default behavior. If this property is defined, the text on these view elements will not be spoken.

For the iOS platform, these properties map directly to the UIAccessibility Protocol. Do not specify accessibility properties for container views, that is, views that contain other view elements. If you define accessibility properties for container views, it will either block interaction with or override the accessibility properties of the view elements contained within it.

The application can monitor the accessibilityChanged event to check when the system's accessibility mode changes. The application can also fire a system event to ask the system to make a voice announcement. On iOS the application can also fire events to notify the accessibility system that the layout or current screen has changed. For details on these events, see the *System-Level Accessibility Events* section on the Ti.App reference page.

Android TalkBack vs. iOS VoiceOver behavior

Android TalkBack and iOS VoiceOver behave differently with navigation and spoken feedback. Take the following code as an example:

```
var win = Ti.UI.createWindow({
    title: 'Welcome',
    layout: 'vertical',
    backgroundColor: 'white'
});

var win2 = Ti.UI.createWindow({
    layout: 'vertical',
    left: '25dp',
    top: '25dp',
    width: '100dp',
    height: '100dp',
    backgroundColor: 'red',
    zIndex: 1,
});

var button2 = Ti.UI.createButton({
    title: 'Red',
    accessibilityLabel: 'Double-click me to close the red window'
});
button2.addEventListener('click', function(){
 win2.close();
});
var label2 = Ti.UI.createLabel({
    text: 'Salut, Monde!',
    font: {fontSize: '24dp'},
    accessibilityHint: 'I do not speak French.'
});


win2.add(label2);
win2.add(button2);

var win3 = Ti.UI.createWindow({
    layout: 'vertical',
    bottom: '25dp',
    right: '25dp',
    width: '100dp',
    height: '100dp',
    backgroundColor: 'blue',
    zIndex: 3,
    accessibilityHint: 'I am a blue window'
});


var button3 = Ti.UI.createButton({
    title: 'Blue',
    accessibilityHint: 'Close the blue window.'
});
button3.addEventListener('click', function(){
```

```
        win3.close();
});
var label3 = Ti.UI.createLabel({
        text: 'Hello, World!',
        font: {fontSize: '24dp'},
        accessibilityLabel: 'You pushed me',
        accessibilityValue: 'Nominal',
        accessibilityHint: 'I am a label'
});

win3.add(label3);
win3.add(button3);

var slider = Ti.UI.createSlider({
        min: 0,
        max: 100,
        width: '100%',
        value: 50
});
var button = Ti.UI.createButton({
        title: 'Open'
});
button.addEventListener('click', function() {
        win2.open();
        win3.open();
});
var label = Ti.UI.createLabel({
        text: 'I pity the foo.',
        font: {fontSize: '24dp'}
})

win.add(button);
```

```
win.add(slider);
win.add(label);
win.open();
```

Initially, the application displays a white window with a button, slider and label. At this point in the user experience, both TalkBack and VoiceOver behave similarly. Since none of the accessibility properties are defined, both spoken feedback assistants provide default feedback. See the table below for the spoken responses of each element.

When you double-click to use the open button, the application displays a red window in the top-left and a blue window in the bottom-right. At this point in the user experience, TalkBack and VoiceOver behave differently.

Since `win3`, the blue window, is the top-level view container in the UI hierarchy, VoiceOver only provides feedback for this view container. All other view elements in the background are disabled by VoiceOver. The user cannot interact with the red window or white parent window. However, TalkBack still provides feedback and does not disable the user interaction with the any of the visible elements.

Since an accessibilityHint is defined for `win3`, the blue window, VoiceOver does not provide spoken feedback for its children view elements, only the window, but TalkBack provides feedback for all elements as well as the window. Additionally, all children view elements are disabled or inaccessible by VoiceOver, so the user cannot double-click the button to close the window and are stuck in this application state.

> ⚠ For the iOS platform, do not specify accessibility properties for container views, that is, views that contain other view elements, and design your UI so the user can return to the parent view if you are implementing a layered UI. If you define accessibility properties for container views, it will either block interaction with or override the accessibility properties of the view elements contained within it.

The following table describes the responses based on accessibility property settings (if the `win3.accessibilityHint` is removed, which blocks most of the view elements from being spoken by VoiceOver).

Element	TalkBack Response	VoiceOver Response
button	"Open. Button."	"Open. Button."
slider	"Seek control. 50 percent."	"50 percent. Adjustable. Slide up and down to adjust the value."
label	"I pity the foo."	"I pity the foo."
button2	"Double-click me to close the red window. Button."	"Double-click me to close the red window. Button."
label2	"I do not speak French."	"Salut, Monde. I do not speak French."
button3	"Close the blue window. Button."	"Blue. Button. Close the blue window."
label3	"You pushed me. Nominal. I am a label."	"You pushed me. Nominal. I am a label."

Testing spoken feedback accessibility features

Android emulator

The Android Emulator does not have a method to test spoken feedback accessibility features. Test on device.

Android device

TalkBack varies by the Android version.

To enable TalkBack:

1. Open **Settings** on the device.
2. Click **Accessibility**, then **TalkBack**.
3. Click the **TalkBack** switch to on to enable TalkBack.
4. A dialog box appears asking you to confirm your selection. Click the **OK** button.

To use TalkBack, tap on a view element to select it, then double-tap to activate the element. Swipe left or right to select the previous or next item, respectively. Swipe right then left or left then right in one motion to scroll vertically or horizontally through views.

To disable TalkBack:

1. Tap the App icon to select it, then double-tap to open it.
2. Swipe right then left in one motion until you see the **Settings** icon.
3. Tap the **Settings** icon to select it, then double-tap to open it.

4. Swipe right then left in one motion until you see **Accessibility**.
5. Tap **Accessibility** to select it, then double-tap to open it.
6. Tap **TalkBack** to select it, then double-tap to open it.
7. Tap the **TalkBack** switch to select it, then double-tap top disable it.
8. A dialog box appears asking you to confirm your selection.
9. Tap the **OK** button to select it, then double-tap to confirm.

For more information about using Talkback, see the Nexus Accessibility Guide.

iOS simulator

The iOS Simulator has a utility called Accessibility Inspector that displays accessibility information for each view element. This is not a substitute for the actual VoiceOver assistant but may help test your application during development.

To enable Accessibility Inspector:

1. Open **Settings** on the simulator.
2. Click **General**, then **Accessibility**.
3. Switch **Accessibility Inspector** to on.

A rainbow colored bar appears in the simulator. To temporary enable or disable the utility during testing, click the 'x' icon in the top-left corner to uncollapse and collapse the accessibility information preview, respectively. While the utility is enabled, in your application, click on a view element to select it to preview the information in the Accessibility Inspector. To navigate in your application while the utility is enabled, for most view elements, tap the view element to select it, then double-tap or triple-tap to activate it.

To disable Accessibility Inspector:

1. Click the 'x' icon in the top-left corner of Accessibility Inspector to collapse the utility to disable it.
2. Repeat the steps to enable it except switch Accessibility Inspector to off.

For more information about using Accessibility Inspector, refer to Accessibility Programming Guide for iOS: Testing the Accessibility of Your iPhone Application.

iOS device

To enable VoiceOver:

1. Open **Settings** on the iOS device.
2. Navigate to **General** > **Accessibility** > **VoiceOver**.
3. Click the **VoiceOver** switch to on to enable VoiceOver.

To use VoiceOver, tap on a view element to select it, then double- or triple-tap to activate the element. Swipe left or right to select the previous or next item, respectively. Swipe with three fingers to scroll vertically or horizontally. If your screen is locked, double-tap it to unlock it.

To disable VoiceOver:

1. Tap **Settings** to select it, then double-tap to open it.
2. Tap **General** to select it, then double-tap to open it.
3. Swipe right until you select **Accessibility**, then double-tap to open it.
4. Tap **VoiceOver** to select it, then double-tap to open it.
5. Tap **VoiceOver** switch to select it, then double-tap to disable it.

For more information about using VoiceOver, refer to Accessibility Programming Guide for iOS: Testing the Accessibility of Your iPhone Application.