Semantic Structure

Building meaningful and well organized content hierarchies



Content Hierarchy

Proper headings and landmarks help users understand content organization.

```
// Good Example
<View accessibilityRole="header">
  <Text accessibilityRole="heading">
    Main Title
 </Text>
</View>
<View accessibilityRole="main">
  <Text accessibilityRole="heading">
    Section Title
  </Text>
</Views
```



Navigation Order

Logical tab order that matches visual layout improves navigation.

- Use natural reading order
- Group related elements
- Maintain consistent structure.



Accessibility Gestures

Learn and practice common accessibility gestures. used with screen readers

Basic Screen Reader Gestures



Single Tap

Select and announce items under your finger.



Double Tap

Activate the selected item

Navigation Gestures



Swipe Right/Left

Move to next/previous item in the interface

Advanced Gestures



Rotor Control

■ Mobile Accessibility Best Practices

Essential guidelines for optimizing your app for VoiceOver and TalkBack



Platform-Specific Features

Key considerations for iOS VoiceOver and Android Talk Back

- Proper heading and landmark roles
- Custom action support
- Focus management



Essential Gestures

Common screen reader gestures and interactions:

- Single tap to select
- ... Double tap to activate
 - Three-finger scroll.



Announcements

Best practices for screen reader announcements

- Clear and concise descriptions
- State changes and updates
 - Error messages and alerts

Mobile Accessibility Best Practices

Guidelines for implementing effective keyboard and focus navigation



Focus Flow

Managing the order and flow of focus navigation

- Logical tab order
- Clear focus indicators
- Skip navigation patterns



Focus Management

Handling focus during interface changes

- Modal and dialog focus
- · Focus restoration
- Dynamic content updates

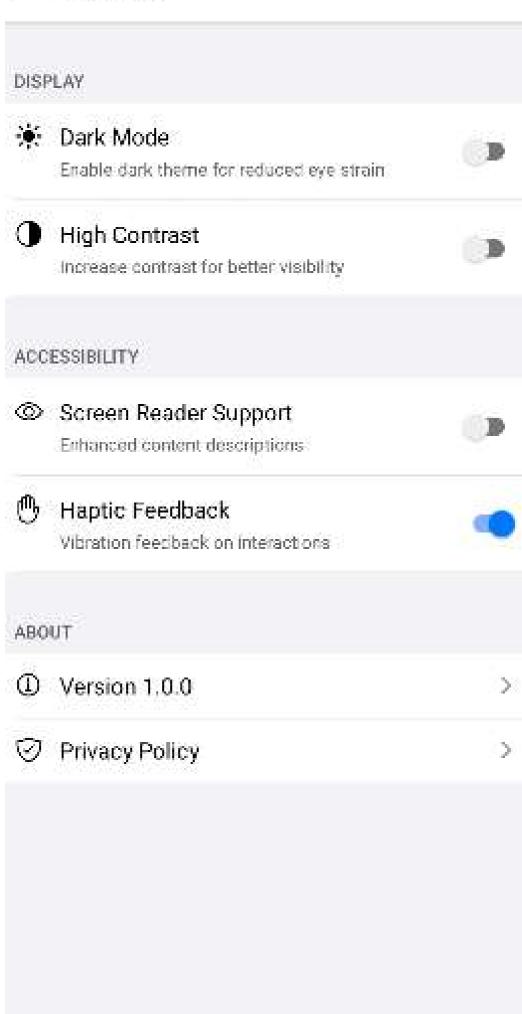


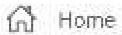
Keyboard Navigation

Supporting keyboard-only navigation

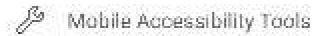
- Keyboard shortcuts
- Focus trapping.
- Custom key handlers









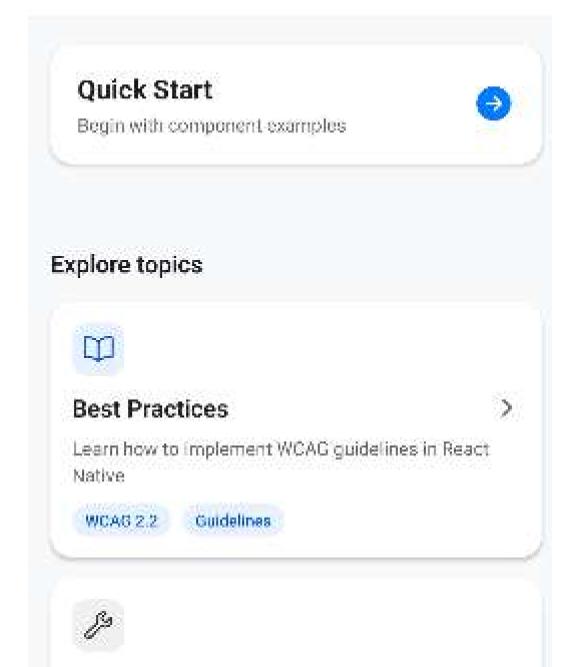


Settings

An accessibility testing manual for developers

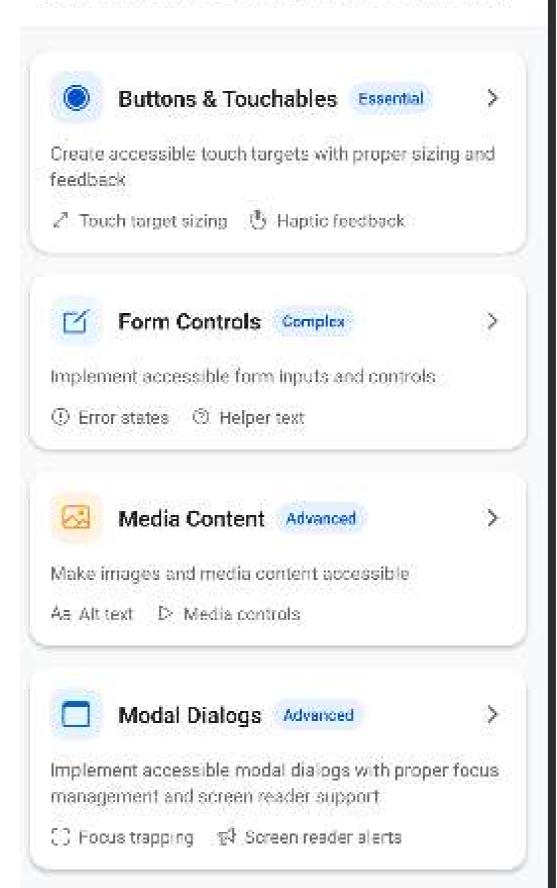
A developer's guide to creating inclusive applications

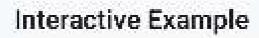
15+ WCAG 100% Components 2.2 Ready Accessible



Accessibility Components

Interactive examples of accessible React Native components with code samples and best practices.







Try this button with VoiceOver/TalkBack enabled

Implementation

```
45X
                                     Copy
<TouchableOpacity
   accessibilityRole="button"
  accessibilityLabel="Submit form"
   accessibilityHint-"Activates form
submission"
   style-{{
    minHeight: 44,
    paddingHorizontal: 16,
    backgroundColor: '#00/AFF',
    borderRadius: 8,
    justifyContent: 'center',
    alignItems; 'center',
  \mathbb{P}_k
   <Text style={{ color: '#fff' }}>
     Submit
   </TouchableOpacity>
```

Accessibility Features

Components Accessible Code

```
JSX.
                                     T Cupy
<TouchableOpacity
  accessibilityRole="button"
  accessibilityLabel="Submit form"
  accessibilityHint="Activates form
submission"
  style={{
     minHeight: 44,
     paddingHorizontal: 16,
     backgroundColor: '#007AFF',
    borderRadius: 8,
    justifyContent; 'center',
    alignItems: 'center',
  }}
3
  <Text style={{ color: '#fff' }}>
     Submit
  </Text>
</TouchableOpacity>
```

Accessibility Features

Minimum Touch Target

44x44 points minimum size ensures the button is easy to tap.

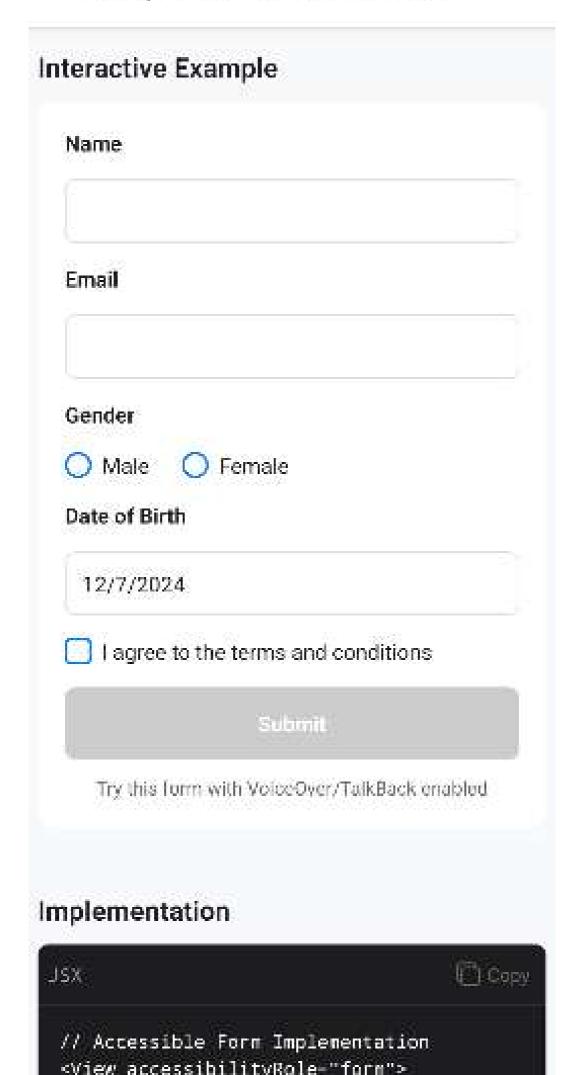
Aa Screen Reader Label

Clear description announces the button's purpose

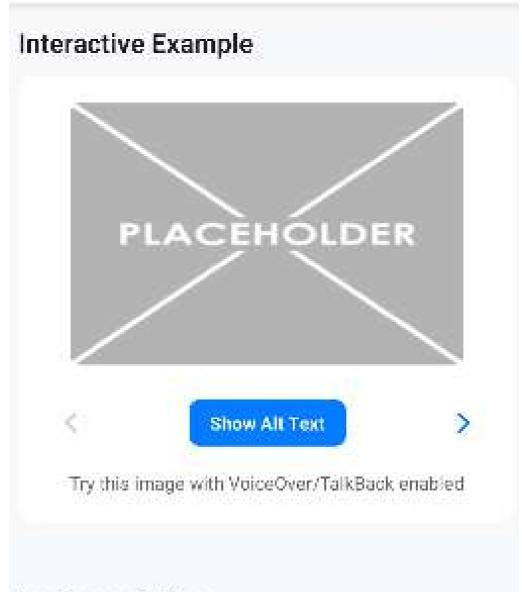
Action Hint

Additional context about what happens on activation:

■ Components Accessible Code



Components Accessible Code



Implementation

```
Inage
    source-(require(',/path/to/image,png'))
    accessibilityLabel="Detailed
    description of the image content"
    accessible={true}
    accessibilityRole="image"
    style-{{
        width: 300,
        height: 200,
        borderRadius: 8,
    }}
/>
```

■ Components Accessible Code

Interactive Example

Open Dialog

Try this dialog with MoiceOver/TalkBack enabled

Implementation

```
Copy
\mathsf{JSX}
// Accessible Dialog Implementation
const AccessibleDialog = ({ visible,
onClose, title, children }) => {
   const closeRef = useRef(null);
   const contentRef = useRef(null);
  useEffect(() => (
     if (visible) {
       // Focus first element when dialog
opens.
      contentRef.current?.focus();
  }, [visible]);
   return (
     ≺Modal⊦
       visible={visible}
      transparent
       animationType="fade"
```

Accessibility Features

Mobile Accessibility Best Practices

Essential guidelines for creating accessible React Native applications



WCAG Guidelines

2.2 Documentation

Understanding and implementing WCAC 2.2 guidelines in mobile apps

Success Criteria

⟨> Examples



Semantic Structure

Codo Examples

Creating meaningful and well-organized content hierarchies



Gesture Tutorial

Interactive Guide

Learn and test common accessibility gestures:



Screen Reader Support

Buidelines

Optimizing your app for VoiceOver and

WCAG 2.2 Guidelines

Essential guidelines for mobile accessibility



Perceivable

Information must be presentable to users in ways they can perceive.

- Text alternatives for non-text content
- Sufficient color contrast ratios.
- Clear content structure and relationships



Operable

Interface components must be operable by all users.

- . All functionality available via keyboard
- Sufficient time to read and use content.
 - No centent that could cause seizures.



Understandable

Information and interface operation must be understandable.

- · Readable and understandable text content
- Predictable functionality
- Input assistance and error prevention