Accessibility

Familiar web accessibility APIs in a platform-agnostic form.

Accessibility in React Native for Web combines several separate web APIs into a cohesive system. Assistive technologies (e.g., VoiceOver, TalkBack screen readers) derive useful information about the structure, purpose, and interactivity of web apps from their HTML elements, attributes, and ARIA in HTML.

Accessibility Props API

React Native for Web includes APIs for making accessible apps. (Note that the React Native-specific **`accessibility*`** props are deprecated in favor of **`aria-*`** props).

> **aria-activedescendant**: ?string
> Equivalent to aria-activedescendant.

> **aria-atomic**: ?boolean
> Equivalent to aria-atomic.

> **aria-autocomplete**: ?string
> Equivalent to aria-autocomplete.

> **aria-busy**: ?boolean
> Equivalent to aria-busy.

> **aria-checked**: ?(boolean | "mixed")
> Equivalent to aria-checked.

> **aria-colcount**: ?number
> Equivalent to aria-colcount.

> **aria-colindex**: ?number
> Equivalent to aria-colindex.

> **aria-colspan**: ?number
> Equivalent to aria-colspan.

> **aria-controls**: ?string
> Equivalent to aria-controls.

> **aria-current**: ?(boolean | "page" | "step" | "location" | "date" | "time")
> Equivalent to aria-current.

> **aria-describedby**: ?string
> Equivalent to aria-describedby.

> **aria-details**: ?string
> Equivalent to aria-details.

> **aria-disabled**: ?boolean
> Equivalent to aria-disabled.

> **aria-errormessage**: ?string
> Equivalent to aria-errormessage.

> **aria-expanded**: ?boolean
> Equivalent to aria-expanded.

> **aria-flowto**: ?string
> Equivalent to aria-flowto.

> **aria-haspopup**: ?string
> Equivalent to aria-haspopup.

> **aria-hidden**: ?boolean
> Equivalent to aria-hidden.

> **aria-invalid**: ?boolean
> Equivalent to aria-invalid.

> **aria-keyshortcuts**: ?string
> Equivalent to aria-keyshortcuts.

> **aria-label**: ?string

> Equivalent to aria-label.

> **aria-labelledby**: ?string
> Equivalent to aria-labelledby.

> **aria-level**: ?number
> Equivalent to aria-level.

> **aria-live**: ?("assertive" | "off" | "polite")
> Equivalent to aria-live.

> **aria-modal**: ?boolean
> Equivalent to aria-modal.

> **aria-multiline**: ?boolean
> Equivalent to aria-multiline.

> **aria-multiselectable**: ?boolean
> Equivalent to aria-multiselectable.

> **aria-orientation**: ?("horizontal" | "vertical")
> Equivalent to aria-orientation.

> **aria-owns**: ?string
> Equivalent to aria-owns.

> **aria-placeholder**: ?string
> Equivalent to aria-placeholder.

> **aria-posinset**: ?number
> Equivalent to aria-posinset.

> **aria-pressed**: ?boolean
> Equivalent to aria-pressed.

> **aria-readonly**: ?boolean
> Equivalent to aria-readonly.

> **aria-required**: ?boolean
> Equivalent to aria-required.

> **role**: ?string
> Equivalent to role.

> **aria-roledescription**: ?string
> Equivalent to aria-roledescription.

> **aria-rowcount**: ?number
> Equivalent to aria-rowcount.

> **aria-rowindex**: ?number
> Equivalent to aria-rowindex.

> **aria-rowspan**: ?number
> Equivalent to aria-rowspan.

> **aria-selected**: ?boolean
> Equivalent to aria-selected.

> **aria-setsize**: ?number
> Equivalent to aria-setsize.

> **aria-sort**: ?("ascending" | "descending" | "none" | "other")
> Equivalent to aria-sort.

> **aria-valuemax**: ?number
> Equivalent to aria-valuemax.

> **aria-valuemin**: ?number
> Equivalent to aria-valuemin.

> **aria-valuenow**: ?number
> Equivalent to aria-valuenow.

> **aria-valuetext**: ?string
> Equivalent to aria-valuetext.

Accessibility patterns

Links

The **`Text`** and **`View`** components can be rendered as links. If the **`href`** prop is set, the element will render **`<a>`** tags without altering the presentation of the element.

```
<Text href="/" />
// <a href="/" ></a>
```

The **`hrefAttrs`** prop sets link-related attributes.

```
const hrefAttrs = { download: true, rel: "nofollow", target: "blank" };

<Text
  href="/document.pdf"
  hrefAttrs={hrefAttrs}
/>
// <a download href="/document.pdf" rel="nofollow" target="_blank"></a>
```

Keyboard focus

The **`tabIndex`** prop determines whether a component is user-focusable and appears in the keyboard tab flow. This prop should be used instead of the **`accessible`** (or **`focusable`** prop) found in React Native for Android/iOS, which is not implemented by React Native for Web/Windows/macOS.

```
<View tabindex={0} />
// <div tabindex="0"></div>

<Text tabindex={-1} href="/" />
// <a href="/" tabindex="-1"></a>
```

> **Did you know?** Any element (including elements not in the keyboard tab flow) can be programmatically focused via **`UIManager.focus(viewRef.current)`**.

Accessible HTML

React Native for Web components express semantics exclusively via the **`aria-*`** props.

```
<View
  aria-label="..."
  aria-pressed={false}
  id="abc"
  role="menuitem"
/>
/*
<div
  aria-label="..."
  aria-pressed="false"
  id="abc"
  role="menuitem"
/>
*/
```

Semantic HTML

The value of the **`role`** prop is used to infer an analogous HTML element where appropriate. This is done to rely on well-supported native mechanisms for encoding semantics and accessibility information.

```
<View role="article">
  <Text role="paragraph">This is an article</Text>
</View>
/*
<article>
  <p>This is an article</p>
</article>
*/
```

If the **`"heading"`** role is combined with an **`aria-level`**, the equivalent HTML heading element is rendered. Otherwise, it is rendered as **`<h1>`**.

```
<Text role="heading" /> /* <h1> */
<Text role="heading" aria-level={2} /> /* <h2> */
```

Note: Avoid changing **`role`** values over time or after user actions. Generally, accessibility APIs do not provide a means of notifying assistive technologies if a **`role`** changes.

Updated July 20, 2023 Edit