Accessibility

react-modal aims to be fully accessible, using the WAI-ARIA guidelines to support users of assistive technologies. This page describes some of react-modal's accessibility-oriented features, along with their configuration options.

The app element

It is important for users of screenreaders that other page content be hidden (via the `aria-hidden` attribute) while the modal is open. To allow react-modal to do this, you should call `Modal.setAppElement` with a query selector identifying the root of your app. For example, if your app content is located inside an element with the ID `root`, you could place the following call somewhere in your code before any modals are opened:

```
Modal.setAppElement('#root');
```

You can also pass a DOM element directly, so that the above example could be rewritten:

```
Modal.setAppElement(document.getElementById('root'));
```

If you are already applying the `aria-hidden` attribute to your app content through other means, you can pass the `ariaHideApp={false}` prop to your modal to avoid getting a warning that your app element is not specified.

Using `Modal.setAppElement` will not embed react-modal into your react app as a descendent component. It will just help boost up the app accessiblity.

Keyboard navigation

When the modal is opened, it restricts keyboard navigation using the tab key to elements within the modal content. This ensures that elements outside the modal (which are not visible while the modal is open) do not receive focus unexpectedly.

By default, when the modal is closed, focus will be restored to the element that was focused before the modal was opened. To disable this behavior, you can pass the `shouldReturnFocusAfterClose={false}` prop to your modal.

The modal can be closed using the escape key, unless the `shouldCloseOnEsc={false}` prop is passed. Disabling this behavior may cause accessibility issues for keyboard users, however, so it is not recommended.

ARIA attributes

Besides the `aria-hidden` attribute which is applied to the app element when the modal is shown, there are many other ARIA attributes which you can use to make your app more accessible. A complete list of ARIA attributes can be found in the ARIA specification.

One ARIA attribute is given a dedicated prop by react-modal: you should use the `contentLabel` prop to provide a label for the modal content (via `aria-label`) if there is no visible label on the screen. If the modal is already labeled with visible text, you should specify the element including the label with the `aria-labelledby` attribute using the `aria` prop described below.

To pass other ARIA attributes to your modal, you can use the `aria` prop, which accepts an object whose keys are the attributes you want to set (without the leading `aria-` prefix). For example, you could have an alert modal with a title as well as a longer description:

```
<Modal
  isOpen={modalIsOpen}
  aria={{
    labelledby: "heading",
    describedby: "full_description"
  }}>
  <h1 id="heading">Alert</h1>
  <div id="full_description">
    <p>Description goes here.</p>
  </div>
</Modal>
```