Libraries Elements

Version: 6.x

Elements Library

A component library containing the UI elements and helpers used in React Navigation. It can be useful if you're building your own navigator, or want to reuse a default functionality in your app.

Installation

To use this package, ensure that you have `@react-navigation/native` and its dependencies (follow this guide), then install `@react-navigation/elements`:

npm Yarn

```
npm install @react-navigation/elements
```

Components

`Header`

A component that can be used as a header. It accepts the following props:

`headerTitle`

String or a function that returns a React Element to be used by the header. Defaults to scene `title`. When a function is specified, it receives an object containing `allowFontScaling`, `tintColor`, `style` and `children` properties. The `children` property contains the title string.

`headerTitleAlign`

How to align the header title. Possible values:

- `left`

- `center`

Defaults to `center` on iOS and `left` on Android.

`headerTitleAllowFontScaling`

Whether header title font should scale to respect Text Size accessibility settings. Defaults to false.

`headerLeft`

Function which returns a React Element to display on the left side of the header. You can use it to implement your custom left button, for example:

```
<Stack.Screen
  name="Home"
  component={HomeScreen}
  options={{
    headerLeft: (props) => (
      <MyButton
        {...props}
        onPress={() => {
          // Do something
        }}
      />
    ),
  }}
/>
```

`headerRight`

Function which returns a React Element to display on the right side of the header.

`headerShadowVisible`

Whether to hide the elevation shadow (Android) or the bottom border (iOS) on the header.

This is a short-hand for the following styles:

```
{
  elevation: 0,
  shadowOpacity: 0,
  borderBottomWidth: 0,
}
```

If the above styles are specified in `headerStyle` along with `headerShadowVisible: false`, then `headerShadowVisible: false` will take precedence.

`headerStyle`

Style object for the header. You can specify a custom background color here, for example.

`headerTitleStyle`

Style object for the title component

`headerLeftContainerStyle`

Customize the style for the container of the `headerLeft` component, for example to add padding.

`headerRightContainerStyle`

Customize the style for the container of the `headerRight` component, for example to add padding.

`headerTitleContainerStyle`

Customize the style for the container of the `headerTitle` component, for example to add padding.

By default, `headerTitleContainerStyle` is with an absolute position style and offsets both `left` and `right`. This may lead to white space or overlap between `headerLeft` and `headerTitle` if a customized `headerLeft` is used. It can be solved by adjusting `left` and `right` style in `headerTitleContainerStyle` and `marginHorizontal` in `headerTitleStyle`.

`headerBackgroundContainerStyle`

Style object for the container of the `headerBackground` element.

`headerTintColor`

Tint color for the header

`headerPressColor`

Color for material ripple (Android >= 5.0 only)

`headerPressOpacity`

Press opacity for the buttons in header (Android < 5.0, and iOS)

`headerTransparent`

Defaults to `false`. If `true`, the header will not have a background unless you explicitly provide it with `headerBackground`. The header will also float over the screen so that it overlaps the content underneath.

This is useful if you want to render a semi-transparent header or a blurred background.

Note that if you don't want your content to appear under the header, you need to manually add a top margin to your content. React Navigation won't do it automatically.

To get the height of the header, you can use `HeaderHeightContext` with React's Context API or `useHeaderHeight`.

`headerBackground`

Function which returns a React Element to render as the background of the header. This is useful for using backgrounds such as an image or a gradient.

For example, you can use this with `headerTransparent` to render a blur view to create a translucent header.

```
import { BlurView } from 'expo-blur';

// ...

<Stack.Screen
  name="Home"
  component={HomeScreen}
  options={{
    headerTransparent: true,
    headerBackground: () => (
      <BlurView tint="light" intensity={100} style={StyleSheet.absoluteFill} />
    ),
  }}
/>;
```

Try this example on Snack

`headerStatusBarHeight`

Extra padding to add at the top of header to account for translucent status bar. By default, it uses the top value from the safe area insets of the device. Pass 0 or a custom value to disable the default behavior, and customize the height.

`HeaderBackground`

A component containing the styles used in the background of the header, such as the background color and shadow. It's the default for `headerBackground`. It accepts the same props as a `View`.

Usage:

```
<HeaderBackground style={{ backgroundColor: 'tomato' }} />
```

`HeaderTitle`

A component used to show the title text in header. It's the default for `headerTitle`. It accepts the same props as a `Text`.

The color of title defaults to the theme text color. You can override it by passing a `tintColor` prop.

Usage:

```
<HeaderTitle>Hello</HeaderTitle>
```

`HeaderBackButton`

A component used to show the back button header. It's the default for `headerLeft` in the stack navigator. It accepts the following props:

- `disabled` - Boolean which controls Whether the button is disabled.
- `onPress` - Callback to call when the button is pressed.
- `pressColor` - Color for material ripple (Android >= 5.0 only).
- `backImage` - Function which returns a React Element to display custom image in header's back button.
- `tintColor` - Tint color for the header.
- `label` - Label text for the button. Usually the title of the previous screen. By default, this is only shown on iOS.
- `truncatedLabel` - Label text to show when there isn't enough space for the full label.
- `labelVisible` - Whether the label text is visible. Defaults to `true` on iOS and `false` on Android.
- `labelStyle` - Style object for the label.
- `allowFontScaling` - Whether label font should scale to respect Text Size accessibility settings.
- `onLabelLayout` - Callback to trigger when the size of the label changes.
- `screenLayout` - Layout of the screen.
- `titleLayout` - Layout of the title element in the header.
- `canGoBack` - Boolean to indicate whether it's possible to navigate back in stack.
- `accessibilityLabel` - Accessibility label for the button for screen readers.
- `testID` - ID to locate this button in tests.
- `style` - Style object for the button.

Usage:

```
<HeaderBackButton label="Hello" onPress={() => console.log('back pressed')} />
```

`MissingIcon`

A component that renders a missing icon symbol. It can be used as a fallback for icons to show that there's a missing icon. It accepts the following props:

- `color` - Color of the icon.
- `size` - Size of the icon.
- `style` - Additional styles for the icon.

`PlatformPressable`

A component which provides an abstraction on top of `Pressable` to handle platform differences. In addition to `Pressable`'s props, it accepts following additional props:

- `pressColor` - Color of material ripple on Android when it's pressed
- `pressOpacity` - Opacity when it's pressed if material ripple isn't supported by the platform

`ResourceSavingView`

A component which aids in improving performance for inactive screens by utilizing `removeClippedSubviews`. It accepts a `visible` prop to indicate whether a screen should be clipped.

Usage:

```
<ResourceSavingView visible={0}>{/* Content */}</ResourceSavingView>
```

Utilities

`SafeAreaProviderCompat`

A wrapper over the `SafeAreaProvider` component from `react-native-safe-area-context which includes initial values.

Usage:

```
<SafeAreaProviderCompat>{/* Your components */}</SafeAreaProviderCompat>
```

`HeaderBackContext`

React context that can be used to get the back title of the parent screen.

```
import { HeaderBackContext } from '@react-navigation/elements';
```

```
// ...

<HeaderBackContext.Consumer>
  {(headerBack) => {
    if (headerBack) {
      const backTitle = headerBack.title;

      /* render something */
    }

    /* render something */
  }}
</HeaderBackContext.Consumer>;
```

`HeaderShownContext`

React context that can be used to check if a header is visible in a parent screen.

```
import { HeaderShownContext } from '@react-navigation/elements';

// ...

<HeaderShownContext.Consumer>
  {(headerShown) => {
    /* render something */
  }}
</HeaderShownContext.Consumer>;
```

`HeaderHeightContext`

React context that can be used to get the height of the nearest visible header in a parent screen.

```
import { HeaderHeightContext } from '@react-navigation/elements';

// ...

<HeaderHeightContext.Consumer>
  {(headerHeight) => {
    /* render something */
  }}
</HeaderHeightContext.Consumer>;
```

`useHeaderHeight`

Hook that returns the height of the nearest visible header in the parent screen.

```
import { useHeaderHeight } from '@react-navigation/elements';

// ...

const headerHeight = useHeaderHeight();
```

`getDefaultHeaderHeight`

Helper that returns the default header height. It takes the following parameters:

- `layout` - Layout of the screen, i.e. an object containing `height` and `width` properties.
- `statusBarHeight` - height of the statusbar

`getHeaderTitle`

Helper that returns the title text to use in header. It takes the following parameters:

- `options` - The options object of the screen.
- `fallback` - Fallback title string if no title was found in options.

Edit this page