ScrollView

A scrollable view that provides integration with the pointer-locking responder system.

ScrollView must have a bounded height: either set the height of the view directly (discouraged) or make sure all parent views have bounded height (e.g., apply `{ flex: 1}` down the view stack).

```
import { ScrollView } from 'react-native';

<ScrollView {...props}>{children}</ScrollView>;
```

API

Props

> **...ViewProps**: ?ViewProps
> All the props supported by [View](https://necolas.github.io/react-native-web/docs/view/).

> **centerContent**: ?boolean
> When `true`, the scroll view automatically centers the content when the content is smaller than the scroll view bounds; when the content is larger than the scroll view, this property has no effect.

> **contentContainerStyle**: ?Style
> These styles will be applied to the scroll view content container which wraps all of the child views.

> **disableScrollViewPanResponder**: ?boolean = false
> When `true`, the default `PanResponder` on the `ScrollView` is disabled, and full control over pointers inside the `ScrollView` is left to its child components. This is meant to be used when native “snap-to” scrolling behavior is needed.

> **horizontal**: ?boolean = false

> When `true`, the scroll view's children are arranged horizontally in a row instead of vertically in a column.

> **keyboardDismissMode**: ?("none" | "on-drag")
> Determines whether the keyboard gets dismissed in response to a scroll drag.

> **onContentSizeChange**: ?(width: number, height: number) => void
> Called when scrollable content view of the ScrollView changes.

> **onScroll**: ?(e: ScrollEvent) => void
> Called during scrolling. The frequency of the events can be controlled using the `scrollEventThrottle` prop.

> **pagingEnabled**: ?boolean = false
> When `true`, the scroll view snaps to individual items in the list when scrolling.

> **scrollEnabled**: ?boolean = true
> When `false`, the content does not scroll.

> **scrollEventThrottle**: ?number = 0
> This controls how often the scroll event will be fired while scrolling (as a time interval in ms). A lower number yields better accuracy for code that is tracking the scroll position, but can lead to scroll performance problems. The default value is `0`, which means the scroll event will be sent only once each time the view is scrolled.

> **stickyHeaderIndices**: ?Array<number>
> An array of child indices determining which children get docked to the top of the screen when scrolling. For example, passing `stickyHeaderIndices={0}` will cause the first child to be fixed to the top of the scroll view. This property is not supported in conjunction with the `horizontal` prop.

ScrollEvent

The `nativeEvent` on the event passed to `onScroll` is a custom object of information related to the layout of the ScrollView.

> **contentOffset**: { x: number, y: number}
> How far the scroll view is scrolled along each axis.

> **contentSize**: { height: number, width: number}
>
> The size of the scrollable content area.

> **layoutMeasurement**: { height: number, width: number}
>
> The `border-box` height and width of the scroll view.

Instant methods

> **getInnerViewNode**: () => void
>
> Returns a reference to the underlying content container DOM node within the `ScrollView`.

> **getScrollableNode**: () => void
>
> Returns a reference to the underlying scrollable DOM node.

> **getScrollResponder**: () => void
>
> Returns a reference to the underlying scroll responder, which supports operations like `scrollTo()`. All `ScrollView`-like components should implement this method so that they can be composed while providing access to the underlying scroll responder's methods.

> **scrollTo**: (options?: { x: number, y: number, animated: boolean }) => void
>
> Scrolls to a given `x`, `y` offset (animation depends on browser support for scroll-behavior).

> **scrollToEnd**: (options?: { animated: boolean }) => void
>
> Scrolls to the end of the scroll view.

Examples

Updated July 20, 2023 Edit