TextInput

Accessible single- and multi-line text input via a keyboard.

Supports features such as auto-complete, auto-focus, placeholder text, and event callbacks. Note: some props are exclusive to or excluded from `multiline`.

```
import { TextInput } from 'react-native-web';

<TextInput {...props} />;
```

API

Props

...AccessibilityProps

The accessibility props.

...PointerProps

The PointerEvent props.

...FocusProps

The FocusEvent props.

...KeyboardProps

The KeyboardEvent props.

autoCapitalize: ?string

Equivalent to HTMLElement.autocapitalize.

autoComplete: ?string

Equivalent to HTMLElement.autocomplete.

> **autoCorrect**: ?boolean = true
>
> A string indicating whether or not auto-correct behavior is on or off. Safari only.

> **autoFocus**: ?boolean = false
>
> If `true`, focuses the input on mount. Only the first form element in a document with auto-focus is focused.

> **blurOnSubmit**: ?boolean
>
> If `true`, the text field will blur when submitted. The default value is `true` for single-line fields and `false` for multiline fields. Note, for multiline fields setting `blurOnSubmit` to `true` means that pressing return will blur the field and trigger the `onSubmitEditing` event instead of inserting a newline into the field.

> **clearTextOnFocus**: ?boolean = false
>
> If `true`, clears the text field automatically when focused.

> **dataSet**: ?Object
>
> Equivalent to HTMLElement.dataset.

> **defaultValue**: ?string
>
> The initial value of the input. Useful for simple use-cases where you don’t want to deal with listening to events and updating the value prop to keep the controlled state in sync.

> **dir**: ?("auto" | "ltr" | "rtl") = "auto"
>
> Equivalent to HTMLElement.dir.

> **disabled**: ?boolean = false
>
> Equivalent to HTMLElement.disabled

> **enterKeyHint**: ?string
>
> Specifies what action label (or icon) to present for the enter key on virtual keyboards. Equivalent to HTMLElement.enterkeyhint

> **id**: ?string
>
> Equivalent to HTMLElement.id.

> **inputMode**: ?string
>
> Hints at the type of data that might be entered by the user while editing the element or its contents. Equivalent to HTMLElement.inputMode. Safari iOS requires an ancestral `<form action>`

> element to display the search keyboard. (Not available when multiline is true.)

> **lang**: ?string
>
> Equivalent to HTMLElement.lang. This prop is used to infer writing direction if no `dir` is set.

> **maxLength**: ?string
>
> Equivalent to HTMLElement.maxlength.

> **multiline**: ?boolean = false
>
> If `true`, the text input can be multiple lines.

> **onChange**: ?(e: ChangeEvent) => void
>
> Equivalent to the React DOM `ChangeEvent`.

> **onChangeText**: ?(text: string) => void
>
> Called when the text content of the input changes.

> **onContentSizeChange**: ?({ nativeEvent: { contentSize: { width, height } } }) => void
>
> Callback that is called when the text input's content size changes.

> **onKeyPress**: ?(e: KeyboardEvent) => void
>
> Equivalent to the onKeyDown prop.

> **onLayout**: ?({ nativeEvent: { layout: { x, y, width, height } } }) => void
>
> This is invoked when a component is mounted and when its layout changes. `x` and `y` are the offsets from the parent node.

> **onScroll**: ?(e: ScrollEvent) => void
>
> Called during scrolling.

> **onSelectionChange**: ?({ nativeEvent: { selection: { start, end } } }) => void
>
> Callback that is called when the text input's selection changes.

> **onSubmitEditing**: ?() => void
>
> Callback that is called when the keyboard's submit button is pressed. When `multiline={true}`, this is only called if `blurOnSubmit={true}`.

> **placeholder**: ?boolean
>
> Text that appears in the form control when it has no value set.

> **placeholderTextColor**: ?string
>
> Equivalent to defining `::placeholder { color }` via a CSS property.

> **readOnly**: ?boolean = false
>
> Equivalent to HTMLElement.readonly

> **rows**: ?number = 1
>
> Sets the number of lines for a multiline input. (Requires `multiline` to be `true`.)

> **secureTextEntry**: ?boolean = false
>
> Set to `true` for passwords and other sensitive data. Equivalent to HTMLInputElement "password" `type`. (Not available when `multiline` is `true`.)

> **selectTextOnFocus**: ?boolean = false
>
> If `true`, all text will automatically be selected on focus.

> **spellCheck**: ?boolean
>
> Equivalent to HTMLElement.spellcheck

> **style**: ?Style
>
> Set the styles of the view. `TextInput` supports typographic styles in addition to those of `View`.

> **testID**: ?string
>
> Set the test selector label (via `data-testid`).

> **value**: ?string
>
> The value of the input when using it as a controlled component.

Instance methods

> **blur**: () => void
>
> Blur the underlying DOM input.

> **clear**: () => void
>
> Clear the text from the underlying DOM input.

> **focus**: () => void
>
> Focus the underlying DOM input.

> **isFocused**: () => boolean
>
> Returns `true` if the input is currently focused; `false` otherwise.

Examples

Updated July 20, 2023 Edit